

Implementing Secure Login in AJAX

Allen I. Holub
Holub Associates
www.holub.com
allen@holub.com

©2010, Allen I. Holub www.holub.com 1

The Basic Principle

- **Never send (or store) a password in the clear.**
 - And that means never.
 - No exceptions.
 - Ever.
 - Period.
 - I mean it!


©2010, Allen I. Holub www.holub.com 2

The Laws of Physics (1)

I. Same-Origin Policy (SOP)

- A. JavaScript code may access only that code (or data) that was loaded from the *same* domain. ("Same-origin policy").
 1. Protocol (http/https) and domain name must match.
 2. Subdomains are *not* in the same domain as parent.
- B. Code from the same domain may access all code from that domain, even if it's in another window (iFrame or top-level window).

©2010, Allen I. Holub www.holub.com 3




The Laws of Physics (2)

II. HTTPS (HTTP over SSL) follows SOP

- A. Since https: and http: are different protocols, code loaded with one protocol cannot access code loaded with the other.
 - 1. Even if they come from the same domain.
- B. Pages loaded using https cannot include content loaded via http, and vice versa.
 - 1. Use relative links. Pages served via https shouldn't contain ``
 - 2. A page served via http: cannot make an https: AJAX request (to protect a password during login, for example)
 - 3. Some browsers (Firefox) don't enforce this rule, but they all should.
 - 4. Some browsers (IE) enforce this rule too strictly.
 - IE won't let you use a different protocol to access an image that's in a *different* domain, even though that's harmless.

©2010, Allen I. Holub www.holub.com 4




HTTPS Caching

- **HTTPS CONTENT MAY NOT BE CACHED!**
 - by Firefox, Opera, or Chrome.
 - Is cached by IE.
- In HTTP response header

```
HTTP/1.1 200 OK
...
Cache-Control: max-age=3600, must-revalidate
Expires: Fri, 30 Oct 1998 14:19:41 GMT
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT
...
```

 - **public** always cache
 - **must-revalidate** pay attention to my rules
 - **no-cache** server must revalidate before releasing cached version
 - **no-store** never cache
- Pragma: no-cache not reliable!
- *Caching Tutorial* at: http://www.mnot.net/cache_docs/


©2010, Allen I. Holub www.holub.com 5



Cookies

- Cookies are client-side data associated with a domain.
 - Subdomains are considered to be *different* domains than parent
 - Protocol doesn't matter.
 - *This is bad* (see Laws of Physics II.B): A session ID sent during login under https should not be sent with an image request (under http) to the same domain.
- Browser sends all of the domains' cookies to server with every request for content from that domain.
 - You can't suppress this behavior.
- A page can access only those cookies that are associated with that page's origin domain.
 - You can tell the browser to make main-domain cookies available to subdomains, (using a "domain" attribute) but you shouldn't.


©2010, Allen I. Holub www.holub.com 6



Cookies and Passwords

- Never store an unencrypted password in a cookie...
 - Cookies are passed with all HTTP requests, including <img...>.
- ... unless the cookie is attached to a domain (or subdomain) that is always accessed using SSL.
 - Your server must reject all http (no 's') requests that access the secure (sub)domain...
 - ... or redirect all insecure access using http


©2010, Allen I. Holub www.holub.com 7



Passwords on the Server

- Never store an unencrypted password on the server.
 - You don't want a hacker who manages to get into your database to get your user's password.
 - Users might use the same password for their bank as they do for your site.
 - You're probably legally liable if a hacker steals a password from your site and then uses it to drain your customer's bank account.


©2010, Allen I. Holub www.holub.com 8



Not Your Dad's Hash Algorithm

- A "cryptographically secure one-way hash" is central to password management.
 - Distills arbitrary input into a largish number (e.g. 128 bits).
 - Think of it as a "fingerprint."
 - Unique for a given input.
 - Cannot be converted back to original input.
 - Can be recreated, given original input.
 - MD-4, MD-5, SHA-1, BCrypt, etc.
- Store hashes, not passwords.


©2010, Allen I. Holub www.holub.com 9



Salt

- Passwords should be “salted” before hashing.
 - Salt == a random number mixed into the hash.
- Without salt, two identical passwords will hash to the same value.
- Protects against “Dictionary” and “Rainbow Table” attacks.
 - Create a list of common (dictionary) or all possible (Rainbow) passwords.
 - Hash them, and put the plain-text/hashed-text pairs in a table.
 - To attack, get a hashed value from the database and look it up in your table.
 - Attack won't work if passwords are randomized using salt.

©2010, Allen I. Holub www.holub.com 10



Hashing with Java APIs


- Over Complicated

```
byte[] salt = new byte[] { (byte)0x3a, (byte)0x44,
    ... , (byte)0x31 };

int iterations = 1000; // should be >= 1000

char[] password = getPasswordFromUser();
PBKDFKeySpec spec = new PBKDFKeySpec(
    password, salt, iterations );
SecretKeyFactory factory =
    SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
SecretKey key = factory.generateSecret( spec );
Cipher c = Cipher.getInstance("PBKDF2WithHmacSHA1");
byte[] hashedValue = c.doFinal( plaintext );
```

©2010, Allen I. Holub www.holub.com 11



BCrypt

- “Blowfish” based password encryption.
- <http://www.mindrot.org/projects/jBCrypt/>

```
String hashed = BCrypt.hashpw(
    password, BCrypt.gensalt());
//...
if( BCrypt.checkpw(candidate, hashed) )
    // candidate matches

public static boolean checkpw(
    String plaintext, String hashed)
{
    return hashed.compareTo(
        hashpw(plaintext, hashed)) == 0;

    // note: salt is stored in the hashed value
    // produced by earlier hashpw() call.
}
```

©2010, Allen I. Holub www.holub.com 12

Digression: Java Security Note

- Storing a password in a String is dangerous.
 - The String remains in memory until it's garbage collected.
 - The String might remain in the virtual-memory swap file for months.
- It's better to use a byte[], and then explicitly load the array with zeros when you're done with it.
- Doesn't solve the problem entirely.

©2010, Allen I. Holub www.holub.com 13

Create A Password

- Store the *hashed* password in the database, indexed by username.

The diagram shows a user icon on the right sending a box labeled 'password' to a server icon on the left. Above the server, a box contains 'username, password (ssl connection)'. Below the server, an arrow points to a database cylinder icon, with a box labeled 'password' above the arrow.

©2010, Allen I. Holub www.holub.com 14

Verify With The Hashed Password

- Hash the password you get from the browser.
- Compare the *hashed* password from the browser with the *hashed* password in the database.

The diagram shows a user icon on the right sending a box labeled 'password' to a server icon on the left. Above the server, a box contains 'Username, password (ssl connection)'. Below the server, an arrow points to a database cylinder icon. From the database, an arrow points back to the server with a box labeled 'password'. An equals sign '==' is placed between the two 'password' boxes.

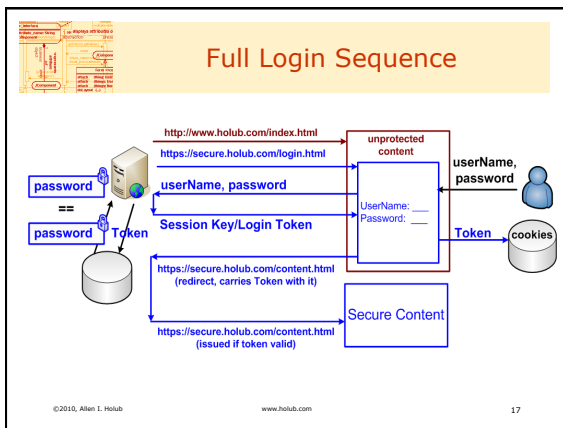
©2010, Allen I. Holub www.holub.com 15

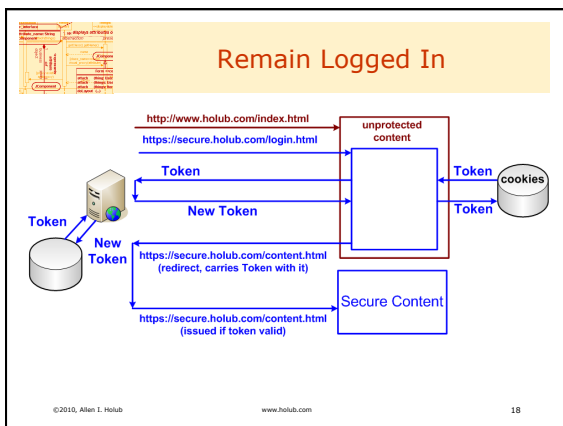
Page Structure:


- Insecure page holds a secure iframe.
 - Iframe has a different cookie namespace than the main page

```
<html> <!-- http://www.holub.com/index.html -->
...
<body>
  <iframe src= 'https://secure.holub.com/loginPanel.html' >
  </iframe> ...
  Unprotected content goes here.
</body>
```

©2010, Allen I. Holub www.holub.com 16








Passing the Token

- Beware “Cross site request forgery:”
 - If all your AJAX requests are encoded in the URL,
 - Bad guy sends you email containing:

```

```
 - If that person has logged into your system, and has a login token in a cookie, that cookie is sent with the AJAX request.
- Put the login tokens into the body of a POST
 - Simply using a POST, without token in message body, *doesn't* fix the problem.


©2010, Allen I. Holub www.holub.com 19



Other Issues


- The token should be both long enough and random enough to repel a brute-force attack.
- The login token can be used as a session key (or not).
 - Don't use the hashed password as the token.
 - A hacker who had compromised the database would effectively have your password, in that case.
- Replacing the token with each login is probably unnecessary, but it makes the page safer.
- Destroy the token on both the client and server when you log off.
- Never store the token in a cookie in the insecure domain.

©2010, Allen I. Holub www.holub.com 20



Inter-frame Communication


©2010, Allen I. Holub www.holub.com 21



The Problem

- According to the Laws Of Physics:
 - Code in a frame that comes from one domain (e.g. the login panel loaded from the secure domain)
 - Cannot talk to code that is loaded from a different domain (e.g. the main page that holds the login panel).


©2010, Allen I. Holub www.holub.com 22



What Doesn't Work

- `postMessage()`
 - Okay in Firefox.
 - Goes only from parent to child in Safari.
 - Doesn't work at all in IE, which has nonstandard call.
- One frame modifies the "hash" in the URL, and the other frame polls to see when the hash changes.
 - Modifying the hash doesn't cause a reload, but
 - Polling is very inefficient, and
 - Doesn't work with AJAX systems that use the hash to manage the back button (e.g. GWT).

©2010, Allen I. Holub www.holub.com 23




What Does Work

- Pages can only talk to other pages from the same domain,
- But all pages from a given domain can talk to each other,
- And pages can create iframes that load from any domain.


```
http://mydomain.com/mainPage.html  
  
<iframe src=  
"https://secure.mydomain.com/loginPanel.html">  
  
function doLogin(user, passwd)  
{  
  verifyWithServer(...);  
  someElement.innerHTML =  
    "<iframe src='...'>";  
}  
  
<iframe src="http://mydomain.html/okay.html">  
  
function onLoad()  
{  
  loginOkay();  
}  
  
function loginOkay(){ ... }
```

©2010, Allen I. Holub www.holub.com 24



The Direct Login Pattern


©2010, Allen I. Holub www.holub.com 25



History

- Written up in Michael Mahemoff's *Ajax Design Patterns* (O'Reilly, 2006).
 - http://ajaxpatterns.org/Direct_Login
- The "pattern" was "invented" by James Dam
 - Example at:
http://www.jamesdam.com/ajax_login/login.html
- Direct Login is just the very first step of the Kerberos Protocol.

©2010, Allen I. Holub www.holub.com 26



When (Not) To Use It

- The pattern does nothing but get the password across the network in a way that it can't be seen.
- Good for personalization, etc. E.g.
 - page layout preferences,
 - search preferences,
 - the last issue of a newsletter/blog that you read,
 - etc.
- **Do not use** if any sensitive information is on the web page that you serve after login completes!

©2010, Allen I. Holub www.holub.com 27

Pluses and Minuses

- ✓ Does not require https/ssl
- ✓ Fast
- ✓ Convenient
- ✗ Protects password, but nothing else.
 - Susceptible to man-in-the-middle for page access (but password is not revealed)
- ✗ Susceptible to brute-force attacks.
 - Add delay after N login attempts.
 - Email admin when too many logins attempted.
 - Use a long and random seed.

©2010, Allen I. Holub www.holub.com 28


Requires Secure "Hash" on Client

- SHA-1 and MD-5 are vulnerable!
- Use SHA-2 (SHA-256).
- Many JavaScript versions:
 - <http://jssha.sourceforge.net/>
 - <http://www.movable-type.co.uk/scripts/sha256.html>
 - <http://code.google.com/p/crypto-js/>
 - Etc.
- SHA-384, SHA-512, more secure than needed.
- Java versions easy come by

©2010, Allen I. Holub www.holub.com 29

Direct Login Protocol


©2010, Allen I. Holub www.holub.com 30



Issues

- Seeds (S) must be used only once!
- Verify any seeds that are sent by the browser to the server.
- Seed must be reasonably unique.
- Seed could be stored in the server-side session rather than being passed back.
 - But it's not worth the trouble: it's already been passed in plain text.


©2010, Allen I. Holub www.holub.com 31



Vulnerabilities

- Eve has:
 - The seed
 - The double-hashed password
- Eve can't:
 - Figure out the password
 - Look up the hashed password if the database is compromised
- Eve can:
 - Stage a man-in-the-middle attack and log in as you exactly once.
 - Threat is minimized if seed has a short lifetime (issued by AJAX request as part of login process)
 - Can't do much damage: you should only be using Direct Login to remember page layout, etc.

©2010, Allen I. Holub www.holub.com 32



Example

- For an example, see:
http://www.jamesdam.com/ajax_login/login.html
- Example uses MySQL and PHP
- He doesn't seed the password hash stored in the database.
 - So his code is vulnerable to dictionary/rainbow attack.

©2010, Allen I. Holub www.holub.com 33

