

The diagram shows a class hierarchy where JComponent is an abstract class, and Form is a concrete class that inherits from it. JComponent has methods like getClass() and getName(). Form has methods like attach(), dotLayout(), and add_field(). There are also associations with UserInterface and Iterator.

Web 2.0/Ajax Security

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

©2008, Allen I. Holub Web 2.0/Ajax Security www.holub.com 1

Some Facts

- Hackers attack bugs.
- The more complex the system, the more bugs it will have.
- The entire ecosystem matters.
 - A bug in EJB or Hibernate will make your app vulnerable.
- Perimeter defenses cannot protect weak software.
 - If the castle walls are unbreachable, then subvert the economy.
- See Security-101 slides from www.holub.com/publications/notes_and_slides

©2008, Allen I. Holub Web 2.0/Ajax Security www.holub.com 2

SOA/SOAP


- Most legacy apps were designed to run entirely inside the data center.
- A SOAP interface to that app allows a hacker to access a fundamentally insecure application through the firewall.
- There is no way to make that application secure short of a rewrite.

©2008, Allen I. Holub Web 2.0/Ajax Security www.holub.com 3

Your code is public

- The source code for half of your application is in the hands of the hacker.
- Try not to give away too much.
- All client-side protections can be subverted.
 - E.g. figure the order total on both the client and server side.

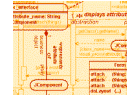
©2008, Allen I. Holub Web 2.0/Ajax Security www.holub.com 4



Man-in-the-middle

- AJAX apps typically talk to the server using RPC or equivalent.
- The communication path between the client and server can be both monitored and modified by a "man in the middle."
- Do not trust any function argument that arrives over the internet.
 - The first thing a hacker is going to try is to give your application bogus data.

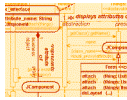
©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 5



HTTPS

- HTTPS protects from the man in the middle, but doesn't protect from a hacker who is pretending to be your AJAX client.

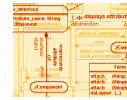
©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 6



Same-Origin Policy

- Code loaded from site A cannot (in theory) access data or network resources from site B.
 - That way a hacker can't inject code into a page that site A provides that sends protected data (passwords, etc.) to site B.
- SOP prevents an Ajax app from making an XMLHttpRequest (or RPC call) to a URL that's not in domain from which the page was loaded.


©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 7



HTTPS and SOP

- Not only must the request be to the same sever, but it must also use the same protocol.
- You cannot make an RPC-over-HTTPS call from a normal HTTP page.
- Your login page must be https:, and can serve the application on success.
- Do secure login panel by embedding an iframe with a src="https://..."


©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 8



SOA can be subverted

- A web page owns its own data and can submit it back to its server.
- Javascript can do anything that you could do in HTML.
- A web page can get data from anywhere.
 - Eg. Insert an `` tag into the DOM.
- But it can also send data in a "read-only" operation:
 - ``
 - or
 - ``


©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 9



Cross-Site Scripting (XSS)

- Hacker's code can:
 - Create a hidden iframe containing a `<form>` who's action is to submit to hacker's site.
 - Create a hidden iframe, put your data into URL and then set frame's "src" to hacker's site.
 - Create a `<script>` tag that does pretty much anything.
 - Etc.

©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 10




How does the evil code get into your page?

Consider a user name of
"Fred `<script>evil code</script>`"

```
protected void doGet(HttpServletRequest req,
    HttpServletResponse resp)...
{
    PrintWriter out = resp.getWriter();
    out.println( "<html><body>Hello " +
        getUsernameFromDatabase() +
        "</body></html>" );
}
```

©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 11




Or Alternatively

- Consider a page that echos back one of the URL arguments, when passed this URL:


```
http://myDomain.com?
    name=Dan%3Cscript%20%3Ealert%28
```
- Note that an evil URL like this can exploit your page without any active involvement on your part.

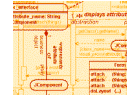
©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 12



Don't trust user input

- Solve the problem by performing white-list testing on all user input.
 - Accept only input that contains safe characters or sequences.
 - E.g. Accept only alphanumeric.
- Do not do black-list testing
 - Reject input that contains "evil" characters.
 - It's too easy to miss something.
 - E.g. Reject < > and % characters

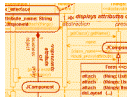
©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 13



The evils of cookies

- Consider a web site that uses cookies to hold login credentials so that a user can "remain logged in."
- In theory, the cookie will be sent only to you—the server that issued the page that set the cookie.
- The cookie expires after a set period of time, NOT when the user leaves your site.
- The cookie is sent every time the browser accesses your site, whether or not you need it.

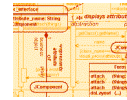
©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 14



Cross-site request forging (XSRF)

- A user logs into your ebay-like trading site (setting a cookie).
- A listing for an item has a "more info" link that goes to hackerheaven.com.
- Your user clicks on it.
- Hackerheaven.com the request came from your site. It returns a page to the browser that contains a <script> triggers some action on your site.
- Your site checks the cookie, sees that it's valid, and performs the action.


©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 15



This is an AJAX problem.

- Note that in a traditional web site, the hacker code won't be able to issue a request and then read the result (because of the SOA policy).
- In an AJAX world
 - the server response can go anywhere (encoded in a URL, for example).
 - An AJAX "RPC" call may not issue a response. The hacker can invoke the service without caring about the result.
- XSRF is a problem any time an operation is performed as a result of a single HTTP request, with no user-verification required.


©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 16



Solving XSRF

- Don't use cookies for session info.
 - Session ID's, login tokens, etc., should be passed as arguments to RPC calls (in the body of an HTTP Post, for example).
 - That is, if the session ID is managed inside the JavaScript application, there's no way for the bad guy to get it.
 - You can also compare a passed sessionID with one that comes from a cookie and make sure they match.


©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 17



JSON

- Is a convenient way to pass structured data around.
- If data is represented as the source code for a JavaScript object, then you can let JavaScript do all the parsing.
- JSON data returned from an RPC call is easy to interpret by the calling function.

©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 18



For Example:


```

    • var result= ['value1', 'value2'];

    • handle(
      {
        'sessionID' : '12345',
        'userName'  : 'Fred',
        'data'      : [ '1','2' ],
      }
    );

    • eval ( someJSON );
    
```


©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 19



<script> insertion

- JavaScript can add <script> tags to a document.
- Those <script> tags can load code from other sites.
 - Useful for mashups.
 - Doesn't violate the SOP because the code that's injecting the <script> tag is trusted.
 - Yeah, right.
- These scripts often generate JSON results.

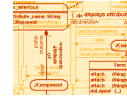
©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 20



JavaScript "Array Hack"

- Redefine the Array constructor:
 - function Array(){ alert("Hello"); }
- Verify that your constructor is called:
 - var a = [43];
- Use this feature:
 - function Array(){ this[1] = 50; }
var a = [40];
alert(a[0] + a[1]); // yields 90

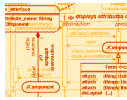
©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 21



Getting the data

- ```
<script type='text/javascript'>
function Array() {
var obj = this;
var ind = 0;
var getNext = function(x) {
obj[ind++] setter = getNext;
if (x) alert(Data stolen from array: " +
x.toString());
};
this[ind++] setter = getNext;
}
</script>
<script type='text/javascript'
src='http://bank.com/jsonservice'> </script>
```

©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 22

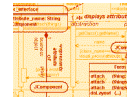


## Protecting the JSON data

- Wrap JSON data in a coment:
 

```
/*['foo', 'bar']*/
```
- This practice prevents the JSON data from being interpreted by a <script> tag.
- You'll have to strip the comments off before you can use the data yourself.


©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 23



## To Read Further

- [http://getahead.org/blog/joe/2007/03/05/json\\_is\\_not\\_as\\_safe\\_as\\_people\\_think\\_it\\_is.html](http://getahead.org/blog/joe/2007/03/05/json_is_not_as_safe_as_people_think_it_is.html)
- <http://robubu.com/?p=24>


©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 24



## Problems with innerHTML

```
<html><head>
 <script language="JavaScript">
 function fillMyDiv(newContent) {
 document.getElementById('mydiv').
 innerHTML = newContent;
 }
 </script>
</head>
<body>
 ... <div id="mydiv"></div> ...
</body>
</html>
```

©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 25

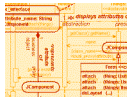


## The Hack

- The hacker manages to get a user to pass in the following as newContent:
 

```
<div
 onmousemove="alert('Hi!');">
 Some text</div>
```
- An alert appears every time the user "mouses" over the div.

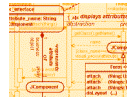
©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 26



## Don't trust strings

- Just because it came from your sever, doesn't mean it's safe.
  - Though it's better to test for this stuff on the server than the client.
- Look for embedded `<script>` tags
- Look for embedded "javascript:" in URLs


©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 27



## SQL Injection

- Consider a simple login screen, with a forgotten-password link.
  - Prompt for an email login
  - Email a password
  - `SELECT someField FROM someTable WHERE someField= '$EMAIL'`

©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 28




### Test for Vulnerability

- Enter `foo@bar.com'` as an email, yielding:
 

```
SELECT someField
FROM someTable
WHERE someField='foo@bar.com'
```
- Will create a SQL error. If the error message isn't "email address unknown," then the site is probably vulnerable.
- Don't ever print the SQL error message!

©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 29

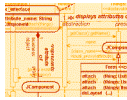


### Prove that it's vulnerable

- Enter `junk' OR 'x'='x` as an email, yielding:
 

```
SELECT someField
FROM someTable
WHERE someField='junk'
OR 'x'='x'
```
- Selects everything from the table!
- Result is:
  - "Login information sent to foo@bar.com"
- Probably the first email in the table.


©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 30



### Guess a few field names

- ```
SELECT someField
FROM someTable
WHERE someField='x'
AND email is NULL;--`
```
- Fails if "email" is not a field.
- Keep trying other obvious column names until it works!

©2008, Allen I. Holub Web 2.0/Ajax Security
www.holub.com 31




Find the Table Name

- ```
SELECT someField
FROM someTable
WHERE someField='x'
AND 1=(SELECT COUNT(*)
FROM tablename);--`
```
- Fails if "tablename" is not the table name.
- Keep trying other obvious table names until it works!

©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 32

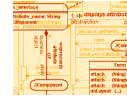




## Etc. You can put any SQL in there!

- **DROP tablename**
- **UPDATE tablename**
  - **Add yourself!**
  - **Change a password!**
- **xp\_cmdshell**
  - In SQLServer, executes an arbitrary OS-level command!

©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 33




## Prepared Statements

- Use "Prepared statements"
  - SQL is precompiled, user input is added later and is not treated as SQL
- Bad:
 

```
Statement s = connection.createStatement();
ResultSet = s.executeQuery(
 "Select email from table where name="
 + formField);
```

©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 34

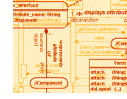


## The Solution

- Good
 

```
PreparedStatement s =
connection.prepareStatement(
 "select email from table where name=?");
ps.setString(1, formField);
ResultSet = s.executeQuery();
```

©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 35




## Use Prepared Statements (2)

- Might not work if prepared statements are simulated in the driver.
- Don't do this:
 

```
Statement s = connection.createStatement();
ResultSet = s.executeQuery(
 "Select email from table where name="
 + formField);
```

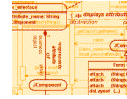
©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 36



## Other Precautions

- Verify user input as safe.
  - Use white-list testing (approve only valid characters as compared to rejecting invalid ones).
- Limit database permissions
  - Login has read-only permission on table
- Assume that the bad guy can get full administrator access to machine!
- Limit information in error reports
  - Do not show output from database server!

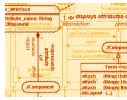
©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 37



## Some References

- Much of the material in this talk is discussed at:
  - <http://groups.google.com/group/Google-Web-Toolkit/web/security-for-gwt-applications>

©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 38



## Q&A

Allen Holub  
[www.holub.com](http://www.holub.com)

Slides at [http://www.holub.com/publications/notes\\_and\\_slides](http://www.holub.com/publications/notes_and_slides)

©2008, Allen I. Holub Web 2.0/Ajax Security  
www.holub.com 39