Browser Security Model

John Mitchell
Reported Web Vulnerabilities "In the Wild"

Evolution of the web vulnerabilities over the years by types

Data from aggregator and validator of NVD-reported vulnerabilities
Web vs System vulnerabilities

Decline in % web vulns since 2009
- 49% in 2010 -> 37% in 2011.
- Big decline in SQL Injection vulnerabilities
Web application vulnerabilities

**Web Application Vulnerabilities**
as a Percentage of All Disclosures in 2010

- **Web Applications:** 49%
- **Others:** 51%

**Web Application Vulnerabilities**
as a Percentage of All Disclosures in 2011 H1

- **Web Applications:** 37 percent
- **Others:** 63 percent
Five lectures on Web security

- Browser security model
  - The browser as an OS and execution platform
  - Protocols, isolation, communication, ...

- Web application security
  - Application pitfalls and defenses

- Authentication and session management
  - How users authenticate to web sites
  - Browser-server mechanisms for managing state

- Content security policies
  - Additional mechanisms for sandboxing and security

- HTTPS: goals and pitfalls
  - Network issues and browser protocol handling

This two-week section could fill an entire course.
Web programming poll

- Familiar with basic html?
- Developed a web application using:
  - Apache?
  - PHP?
  - Ruby?
  - Python?
  - SQL?
  - JavaScript?
  - CSS?
  - JSON?
- Know about:
  - postMessage?
  - NaCL?
  - Webworkers?
  - CSP?

Resource: http://www.w3schools.com/
Goals of web security

- **Safely browse the web**
  - Users should be able to visit a variety of web sites, without incurring harm:
    - No stolen information (without user’s permission)
    - Site A cannot compromise session at Site B

- **Support secure web applications**
  - Applications delivered over the web should have the same security properties we require for stand-alone applications
Web security

Web Attacker
Sets up malicious site visited by victim; no control of network

Alice

System
Network security

Intercepts and controls network communication
Web Threat Models

**Web attacker**
- Control attacker.com
- Can obtain SSL/TLS certificate for attacker.com
- User visits attacker.com
  - Or: runs attacker’s Facebook app, etc.

**Network attacker**
- Passive: Wireless eavesdropper
- Active: Evil router, DNS poisoning

**Malware attacker**
- Attacker escapes browser isolation mechanisms and run separately under control of OS
Malware attacker

- Browsers may contain exploitable bugs
  - Often enable remote code execution by web sites
  - Google study: [the ghost in the browser 2007]
    - Found Trojans on 300,000 web pages (URLs)
    - Found adware on 18,000 web pages (URLs)

- Even if browsers were bug-free, still lots of vulnerabilities on the web
  - *All* of the vulnerabilities on previous graph: XSS, SQLi, CSRF, ...
Outline

- Http
- Rendering content
- Isolation
- Communication
- Navigation
- Security User Interface
- Cookies
- Frames and frame busting
HTTP
URLs

Global identifiers of network-retrievable documents

Example:

http://stanford.edu:81/class?name=cs155#homework

- Protocol
- Hostname
- Port
- Path
- Query
- Fragment

Special characters are encoded as hex:
- "%0A" = newline
- "%20" or + = space, "%2B" = + (special exception)
## HTTP Request

**Method** | **File** | **HTTP version** | **Headers**
--- | --- | --- | ---
GET | /index.html | HTTP/1.1 | Accept: image/gif, image/x-bitmap, image/jpeg, */*  
|  |  |  | Accept-Language: en  
|  |  |  | Connection: Keep-Alive  
|  |  |  | User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)  
|  |  |  | Host: www.example.com  
|  |  |  | Referer: http://www.google.com?q=dingbats

**Blank line**

**Data** – none for GET

**GET**: no side effect  
**POST**: possible side effect
HTTP Response

HTTP version: HTTP/1.0
Status code: 200 OK
Reason phrase: OK

Headers:
- Date: Sun, 21 Apr 1996 02:20:42 GMT
- Server: Microsoft-Internet-Information-Server/5.0
- Connection: keep-alive
- Content-Type: text/html
- Last-Modified: Thu, 18 Apr 1996 17:39:05 GMT
- Set-Cookie: ...
- Content-Length: 2543

Data:
<HTML> Some data... blah, blah, blah </HTML>

Cookies:
RENDERING CONTENT
Rendering and events

Basic browser execution model

- Each browser window or frame
  - Loads content
  - Renders it
    - Processes HTML and scripts to display page
    - May involve images, subframes, etc.
  - Responds to events

Events can be

- User actions: OnClick, OnMouseover
- Rendering: OnLoad, OnBeforeUnload
- Timing: setTimeout(), clearTimeout()
Example

```
<html>
  <body>
    <div style="-webkit-transform: rotateY(30deg) rotateX(-30deg); width: 200px;"> 
      I am a strange root.
    </div>
  </body>
</html>
```

Document Object Model (DOM)

- Object-oriented interface used to read and write docs
  - Web page in HTML is structured data
  - DOM provides representation of this hierarchy

Examples

- **Properties**: document.alinkColor, document.URL, document.forms[], document.links[], document.anchors[]
- **Methods**: document.write(document.referrer)

Includes Browser Object Model (BOM)

- window, document, frames[], history, location, navigator (type and version of browser)
Changing HTML using Script, DOM

Some possibilities

- `createElement(elementName)`
- `createTextNode(text)`
- `appendChild(newChild)`
- `removeChild(node)`

Example: Add a new list item:

```javascript
var list = document.getElementById('t1');
var newitem = document.createElement('li');
var newtext = document.createTextNode(text);
list.appendChild(newitem);
newitem.appendChild(newtext);
```

HTML

```html
<ul id="t1">
  <li>Item 1</li>
</ul>
```
Basic web functionality

HTML Image Tags

```html
<html>
  ...
  <p> ... </p>
  ...
  <img src="http://example.com/sunset.gif" height="50" width="100">
</html>
```

Displays this nice picture ➔ Security issues?
Image tag security issues

- Communicate with other sites
- Hide resulting image
  - `<img src="..." height="1" width="1">`
- Spoof other sites
  - Add logos that fool a user

Important Point: A web page can send information to any site
JavaScript onError

Basic function
- Triggered when error occurs loading a document or an image

Example

```html
<img src="image.gif" onerror="alert('The image could not be loaded.')"
>
```
- Runs onError handler if image does not exist and cannot load

http://www.w3schools.com/jsref/jsref onError.asp
When response header indicates that page is not an image, the browser stops and notifies JavaScript via the onerror handler.

```html
<html>
<body>
<img id="test" style="display: none">
<script>
    var test = document.getElementById('test');
    var start = new Date();
    test.onerror = function() {
        var end = new Date();
        alert("Total time: " + (end - start));
    }
    test.src = "http://www.example.com/page.html";
</script>
</body>
</html>
```
Port scanning behind firewall

JavaScript can:

- Request images from internal IP addresses
  - Example: `<img src="192.168.0.4:8080"/>
- Use timeout/onError to determine success/failure
- Fingerprint webapps using known image names
Remote scripting

Goal
- Exchange data between a client-side app running in a browser and server-side app, without reloading page

Methods
- Java Applet/ActiveX control/Flash
  - Can make HTTP requests and interact with client-side JavaScript code, but requires LiveConnect (not available on all browsers)
- XML-RPC
  - Open, standards-based technology that requires XML-RPC libraries on server and in your client-side code.
- Simple HTTP via a hidden IFRAME
  - IFRAME with a script on your web server (or database of static HTML files) is by far the easiest of the three remote scripting options

Important Point: A page can maintain bi-directional communication with browser (until user closes/quits)

Simple remote scripting example

client.html: RPC by passing arguments to server.html in query string

```html
<script type="text/javascript">
function handleResponse() {
    alert('this function is called from server.html') }
</script>
<iframe id="RSIFrame" name="RSIFrame"
style="width:0px; height:0px; border: 0px"
src="blank.html">
</iframe>
<a href="server.html" target="RSIFrame">make RPC call</a>
```

server.html: another page on same server, could be server.php, etc

```html
<script type="text/javascript">
    window.parent.handleResponse()
</script>
```

RPC can be done silently in JavaScript, passing and receiving arguments
ISOLATION
Frame and iFrame

- **Window may contain frames from different sources**
  - Frame: rigid division as part of frameset
  - iFrame: floating inline frame
- **iFrame example**

```html
<iframe src="hello.html" width=450 height=100>
If you can see this, your browser doesn't understand IFRAME.
</iframe>
```

- **Why use frames?**
  - Delegate screen area to content from another source
  - Browser provides isolation based on frames
  - Parent may work even if frame is broken
Windows Interact
## Analogy

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Web browser</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Primitives</strong></td>
<td><strong>Primitives</strong></td>
</tr>
<tr>
<td>- System calls</td>
<td>- Document object model</td>
</tr>
<tr>
<td>- Processes</td>
<td>- Frames</td>
</tr>
<tr>
<td>- Disk</td>
<td>- Cookies / localStorage</td>
</tr>
<tr>
<td><strong>Principals: Users</strong></td>
<td><strong>Principals: “Origins”</strong></td>
</tr>
<tr>
<td>- Discretionary access control</td>
<td>- Mandatory access control</td>
</tr>
<tr>
<td><strong>Vulnerabilities</strong></td>
<td><strong>Vulnerabilities</strong></td>
</tr>
<tr>
<td>- Buffer overflow</td>
<td>- Cross-site scripting</td>
</tr>
<tr>
<td>- Root exploit</td>
<td>- Cross-site request forgery</td>
</tr>
<tr>
<td></td>
<td>- Cache history attacks</td>
</tr>
<tr>
<td></td>
<td>- ...</td>
</tr>
</tbody>
</table>
Policy Goals

- Safe to visit an evil web site
- Safe to visit two pages at the same time
  - Address bar distinguishes them
- Allow safe delegation
Browser security mechanism

Each frame of a page has an origin
- Origin = protocol://host:port

Frame can access its own origin
- Network access, Read/write DOM, Storage (cookies)

Frame cannot access data associated with a different origin
Components of browser security policy

Frame-Frame relationships
- canScript(A,B)
  - Can Frame A execute a script that manipulates arbitrary/nontrivial DOM elements of Frame B?
- canNavigate(A,B)
  - Can Frame A change the origin of content for Frame B?

Frame-principal relationships
- readCookie(A,S), writeCookie(A,S)
  - Can Frame A read/write cookies from site S?

See https://code.google.com/p/browsersec/wiki/Part1
https://code.google.com/p/browsersec/wiki/Part2
Library import excluded from SOP

```html
<script src=https://seal.verisign.com/getseal?
host_name=a.com></script>
```

- Script has privileges of imported page, NOT source server.
- Can script other pages in this origin, load more scripts
- Other forms of importing
Origin: scheme, host, (port), hasSetDomain

Try `document.domain = document.domain`
Additional mechanisms

- Cross-origin network requests
  - Access-Control-Allow-Origin: <list of domains>
  - Access-Control-Allow-Origin: *

- Cross-origin client side communication
  - Client-side messaging via navigation (old browsers)
  - postMessage (modern browsers)
COMMUNICATION
window.postMessage

- New API for inter-frame communication
  - Supported in latest betas of many browsers
  - A network-like channel between frames

![Contact Sharing](image)
postMessage syntax

frames[0].postMessage("Attack at dawn!", "http://b.com/");

window.addEventListener("message", function (e) {
  if (e.origin == "http://a.com") {
    ... e.data ... }
}, false);

Anecdote

Attack at dawn!
Why include “targetOrigin”? 

What goes wrong?

frames[0].postMessage("Attack at dawn!");

Messages sent to *frames*, not principals
- When would this happen?
NAVIGATION
A Guninski Attack

window.open("https://attacker.com/", "awglogin");
What should the policy be?
# Legacy Browser Behavior

<table>
<thead>
<tr>
<th>Browser</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE 6 (default)</td>
<td>Permissive</td>
</tr>
<tr>
<td>IE 6 (option)</td>
<td>Child</td>
</tr>
<tr>
<td>IE7 (no Flash)</td>
<td>Descendant</td>
</tr>
<tr>
<td>IE7 (with Flash)</td>
<td>Permissive</td>
</tr>
<tr>
<td>Firefox 2</td>
<td>Window</td>
</tr>
<tr>
<td>Safari 3</td>
<td>Permissive</td>
</tr>
<tr>
<td>Opera 9</td>
<td>Window</td>
</tr>
<tr>
<td>HTML 5</td>
<td>Child</td>
</tr>
</tbody>
</table>
Window Policy Anomaly
# Legacy Browser Behavior

<table>
<thead>
<tr>
<th>Browser</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE 6 (default)</td>
<td>Permissive</td>
</tr>
<tr>
<td>IE 6 (option)</td>
<td>Child</td>
</tr>
<tr>
<td>IE7 (no Flash)</td>
<td>Descendant</td>
</tr>
<tr>
<td>IE7 (with Flash)</td>
<td>Permissive</td>
</tr>
<tr>
<td>Firefox 2</td>
<td>Window</td>
</tr>
<tr>
<td>Safari 3</td>
<td>Permissive</td>
</tr>
<tr>
<td>Opera 9</td>
<td>Window</td>
</tr>
<tr>
<td>HTML 5</td>
<td>Child</td>
</tr>
</tbody>
</table>
## Adoption of Descendant Policy

<table>
<thead>
<tr>
<th>Browser</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE7 (no Flash)</td>
<td>Descendant</td>
</tr>
<tr>
<td>IE7 (with Flash)</td>
<td>Descendant</td>
</tr>
<tr>
<td>Firefox 3</td>
<td>Descendant</td>
</tr>
<tr>
<td>Safari 3</td>
<td>Descendant</td>
</tr>
<tr>
<td>Opera 9</td>
<td>(many policies)</td>
</tr>
<tr>
<td>HTML 5</td>
<td>Descendant</td>
</tr>
</tbody>
</table>
When is it safe to type my password?

SECURITY USER INTERFACE
Safe to type your password?
Safe to type your password?
Safe to type your password?

https://www.bankofthevest.com/OW/home

thevvest.com/

thewest.com/
Safe to type your password?

BANK OF THE WEST
Portfolio Online

Welcome to Portfolio Online!

Please enter your access ID and click “Continue.”

Terms and Conditions
please read our Terms & Conditions.

Access ID:

Done

portfolioonline.metavante.com
Safe to type your password?
Mixed Content: HTTP and HTTPS

Problem
- Page loads over HTTPS, but has HTTP content
- Network attacker can control page

IE: displays mixed-content dialog to user
- Flash files over HTTP loaded with no warning (!)
- Note: Flash can script the embedding page

Firefox: red slash over lock icon (no dialog)
- Flash files over HTTP do not trigger the slash

Safari: does not detect mixed content

Still current? Dan will talk about this later....
Mixed Content: HTTP and HTTPS

silly dialogs
Mixed content and network attacks

- banks: after login all content over HTTPS
  - Developer error: Somewhere on bank site write
    `<script src=http://www.site.com/script.js> </script>`
  - Active network attacker can now hijack any session

- Better way to include content:
  `<script src=//www.site.com/script.js> </script>`
  - served over the same protocol as embedding page
Lock Icon 2.0

- **Extended validation (EV) certs**

- Prominent security indicator for EV certificates

- **note:** EV site loading content from non-EV site does not trigger mixed content warning
Finally: the status Bar

Trivially spoofable

<a href="http://www.paypal.com/">
  onclick="this.href = 'http://www.evil.com/';">
  PayPal</a>
COOKIES: CLIENT STATE
Cookies

Used to store state on user’s machine

HTTP is stateless protocol; cookies add state
Cookie authentication

**Browser**
- POST login.cgi
  - Username & pwd
  - Set-cookie: auth=val

**Web Server**
- Validate user
  - auth=val

**Auth server**
- Store val
- Check val
  - restricted.html
  - YES/NO
  - If YES,
    - restricted.html
Cookie Security Policy

- **Uses:**
  - User authentication
  - Personalization
  - User tracking: e.g. Doubleclick (3rd party cookies)

- **Browser will store:**
  - At most 20 cookies/site, 3 KB / cookie

- **Origin is the tuple** `<domain, path>`
  - Can set cookies valid across a domain suffix
Secure Cookies

- Provides confidentiality against network attacker
  - Browser will only send cookie back over HTTPS

- ... but no integrity
  - Can rewrite secure cookies over HTTP
    ⇒ network attacker can rewrite secure cookies
    ⇒ can log user into attacker’s account
httpOnly Cookies

- Cookie sent over HTTP(s), but not accessible to scripts
  - cannot be read via `document.cookie`
  - Helps prevent cookie theft via XSS

... but does not stop most other risks of XSS bugs
FRAMES AND FRAME BUSTING
Frames

Embed HTML documents in other documents

<iframe name="myframe"
src="http://www.google.com/">
  This text is ignored by most browsers.
</iframe>
Frame Busting

Goal: prevent web page from loading in a frame
- example: opening login page in a frame will display correct passmark image

Frame busting:

```javascript
if (top != self)
top.location.href = location.href
```
Better Frame Busting

Problem: Javascript OnUnload event

Try this instead:

```html
<body onUnload="javascript: cause_an_abort;"> if (top != self)  top.location.href = location.href else { ... code of page here ...}
```
Summary

- Http
- Rendering content
- Isolation
- Communication
- Navigation
- Security User Interface
- Cookies
- Frames and frame busting