Browser Security Model

John Mitchell

Acknowledgments: Lecture slides are from the Computer Security course taught by Dan Boneh and John Mitchell at Stanford University. When slides are obtained from other sources, a reference will be noted on the bottom of that slide. A full list of references is provided on the last slide.
Sampling of 2015 security incidents by attack type, time and impact

Size of circle estimates relative impact of incident in terms of cost to business, based on publicly disclosed information regarding leaked records and financial losses.
Reported Web Vulnerabilities "In the Wild"

Data from aggregator and validator of NVD-reported vulnerabilities.
Current vulnerabilities

https://geekflare.com/online-scan-website-security-vulnerabilities/
Web vs System vulnerabilities

Decline in % web vulns since 2009

- 49% in 2010 -> 37% in 2011.
- Big decline in SQL Injection vulnerabilities
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
- **HTTPS: goals and pitfalls**
  - Network issues and browser protocol handling
- **Content security policies**
  - Additional mechanisms for sandboxing and security

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?
  - WebView?

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
  - Site A cannot compromise session at Site B

Support secure web applications

- Applications delivered over the web should be able to achieve the same security properties as stand-alone applications
Web security threat model

Alice

Set up malicious site visited by victim; no control of network
Network security threat model

Network Attacker
Intercepts and controls network communication

System

Alice
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

Special characters are encoded as hex:
- \%0A = newline
- \%20 or + = space, \%2B = +
## HTTP Request

<table>
<thead>
<tr>
<th>Method</th>
<th>File</th>
<th>HTTP version</th>
<th>Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/index.html</td>
<td>HTTP/1.1</td>
<td>Accept: image/gif, image/x-bitmap, image/jpeg, /<em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Accept-Language: en</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Connection: Keep-Alive</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host: <a href="http://www.example.com">www.example.com</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Referer: <a href="http://www.google.com?q=dingbats">http://www.google.com?q=dingbats</a></td>
</tr>
</tbody>
</table>

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

HTTP/1.0 200 OK
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

<HTML> Some data... blah, blah, blah </HTML>
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
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<button onclick="document.write(5 + 6)">Try it</button>
</body>
</html>
```

Source: http://www.w3schools.com/js/js_output.asp
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)
The HTML DOM Tree of Objects

Document

Root element: <html>

Element: <head>

Element: <title>
  Text: "My title"

Element: <body>

Element: <a>
  Text: "My link"

Element: <h1>
  Text: "My header"
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My First Paragraph</p>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>
</body>
</html>

Source: http://www.w3schools.com/js/js_output.asp
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
<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

Q: what threat model are we talking about here?
Basic web functionality

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
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

Operating system

Primitives
- System calls
- Processes
- Disk

Principals: Users
- Discretionary access control

Vulnerabilities
- Buffer overflow
- Root exploit

Web browser

Primitives
- Document object model
- Frames
- Cookies / localStorage

Principals: “Origins”
- Mandatory access control

Vulnerabilities
- Cross-site scripting
- Cross-site request forgery
- Cache history attacks
- ...
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/Part1)  
[https://code.google.com/p/browsersec/wiki/Part2](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
Domain Relaxation

- Origin: scheme, host, (port), hasSetDomain
- Try `document.domain = document.domain`
COMMUNICATION
window.postMessage

API for inter-frame communication

- Supported in standard browsers

- A network-like channel between frames
postMessage syntax

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

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

![Diagram showing message flow between A.com and B.com](image-url)
Why include “targetOrigin”?

What goes wrong?

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

Messages sent to frames, not principals

- When would this happen?
A Guninski Attack

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

```
> top.frames[1].location = "http://www.attacker.com/...";
> top.frames[2].location = "http://www.attacker.com/...";
```
## 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?
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
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

Will talk about this later....
Mixed Content: HTTP and HTTPS
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 Header:
Set-cookie: NAME=VALUE ;
domain = (who can read) ;
expires = (when expires) ;
secure = (only over SSL)

If expires=NULL:
this session only

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

Get restricted.html
Cookie: auth=val

If YES,

restricted.html

Check val

YES/NO
Cookie Security Policy

**Uses:**

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

**Origin is the tuple** \( <\text{domain}, \text{path}> \)

- Can set cookies valid across a domain suffix
Secure Cookies

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

GET ...

HTTP Header:
Set-cookie: NAME=VALUE ; Secure=true
FRAMES AND FRAME BUSTING
Frames

Embed HTML documents in other documents

```html
<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