Web Browser Security

Collin Jackson

Running Remote Code is Risky
- Integrity
  - Compromise your machine
  - Install malware rootkit
  - Transact on your accounts
- Confidentiality
  - Read your information
  - Steal passwords
  - Read your email

Browser Sandbox
- Goal
  - Run remote web applications safely
  - Limited access to OS, network, and browser data
- Approach
  - Isolate sites in different security contexts
  - Browser manages resources, like an OS
  - Access control: same-origin policy
    - Pages from the “same site” can interact
    - Pages from “different sites” separated

Why study browser security?
- if you’re not Microsoft, Mozilla, Apple, Google or Opera?
  - Build better browsers
  - Contribute to open source browsers (Firefox, Safari)
  - Embed a renderer in your program (Gecko, WebKit)
  - Build better web applications
  - Servers and firewalls can mitigate browser limitations
  - Take advantage of opt-in browser security features
  - Be a safer surfer
    - Make informed security decisions
    - Distinguish harmless warnings from attacks

Threat Models
- Web attacker
  - Controls attacker.com
  - Has HTTPS certificate for attacker.com ($0)
  - User visits attacker.com
- Network attacker
  - Passive: Wireless eavesdropper
  - Active: Evil router, DNS poisoning
- Malware attacker
  - Escaped from browser sandbox

Security User Interface
When is it safe to type my password?
URLs

Global identifiers of network-retrievable documents

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

Protocol
Username
Password
Host
Port
Path
Query
Fragment

Safe to type your password?
Same-Origin Policy
How does the browser isolate different sites?

Are all interactions good?

Browser Same-Origin Policy
- Different origins have limited interaction
- Origin is the tuple <domain, port, protocol>
  - http://www.example.com:80/whoami  - Full access
  - http://www.example.com:80/hello
  - https://www.example.com:443/hello  - Limited access
  - http://www.example.com:443/hello

Same-Origin Policy Examples
- Example HTML at http://www.site.com/
  <iframe src="http://othersite.com"></iframe>
  <img src="http://othersite.com/logo.gif">

- Disallowed:
  alert(frames[0].document.body.innerHTML)
  alert(frames[0].location)

- Allowed:
  alert(images[0].height)
  frames[0].location = "http://othersite.com/foo";

Mixed Content
A Guninski Attack

What should the policy be?

Legacy Browser Behavior

Window Policy Anomaly

Principle: Pixel Delegation

What should the policy be?
Why Frame Busting Matters

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>

Intermission

Ask me about:
- 1000 lines of regression tests
- Frame busting and Yahoo
- PR for “extended validation” user study

Mashups

How can different sites communicate?

HousingMaps.com

Windows Live Contacts

Integrator

Add a contact
Share contacts
window.postMessage

- Secure channel between frames
  frames[0].postMessage("Attack at dawn!", "http://gadget.com/");
  window.addEventListener(function (e) {
    if (e.origin == "http://integrator.com") {
      e.data ...
    }, false);
- Supported in brand-new browsers

Why include “targetOrigin”?

- What goes wrong?
  frames[0].postMessage("Attack at dawn!");

- Messages sent to frames; not principals
- When would this happen?

Thanks!

You've been a great audience

Network Access Policy

- Send anywhere
  (Some ports are inaccessible)
- Read only from your origin
  (Some formats executable across origins)

Goals: Prevent Bot-like Activity

- Spam
- Reading documents behind a firewall
- Clicking advertisements
- Denial of service?

Same Origin Requests

```javascript
<script>
var xhr = new XMLHttpRequest();
xhr.open("POST", "http://www.example.com:81/foo/example.cgi", true); // asynchronous
xhr.send("Hello world!");
xhr.onload = function() {
  if (xhr.status == 200) {
    alert(xhr.responseText);
  }
}
</script>
```
Sending a Cross-Domain GET

- Data must be URL encoded

```html
<img src="http://othersite.com/file.cgi?foo=1&bar=x\%28y">
```

- Browser sends:

```
GET file.cgi?foo=1\&bar=x HTTP/1.1
Host: othersite.com
```

- Can't send to some restricted ports, like 25 (SMTP)

Sending a Cross-Domain POST

- Can use any encoding

```html
<form method="POST" action="http://othersite.com/file.cgi"
encoding="text/plain">
  <input type="hidden" name="Hello\nworld!\n2\+2\=" value="4">
</form>
```

- Browser sends:

```
POST file.cgi HTTP/1.1
Host: othersite.com
```

```
Hello world!
2+2=4
```

- Can target a hidden iframe to do this in background
- Can't send to some restricted ports, like 25 (SMTP)

Cross-Domain Network Reading

- Executable data formats:

  ```html
  <script src="http://othersite.com/file.js"></script>
  <link rel="stylesheet" href="http://othersite.com/file.css">
  <img src="http://othersite.com/file.jpg">
  <applet code="http://othersite.com/File.class">
  ```

  - Used extensively in mashups
  - Not ideal for mutual distrust scenarios
  - Server opt-in
    - Access-Control-Allow-Origin (W3C)
    - Flash Player's crossdomain.xml

Brief Review of DNS

- www.evil.com?
  - ns.evil.com
  - .com

- 171.64.7.115
  - TTL = "24 hours"

- 192.168.0.100

DNS Rebinding Attack

- DNS-SEC cannot stop this attack

- Read permitted: it's the "same origin"

DNS Rebinding Defenses

- Server-side defenses:
  - Check Host header for unrecognized domains
  - Authenticate users with something other than IP
- Firewall defenses:
  - External names can't resolve to internal addresses
  - Protects browsers inside the organization