Web Security: Basic Web Security Model [continued]

Spring 2016

Franziska (Franzi) Roesner
franzi@cs.washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...
Reminder: 2 Sides of Web Security

• Web browser
  – Responsible for securely confining Web content presented by visited websites

• Web applications
  – Online merchants, banks, blogs, Google Apps …
  – Mix of server-side and client-side code
    • Server-side code written in PHP, Ruby, ASP, JSP… runs on the Web server
    • Client-side code written in JavaScript… runs in the Web browser
  – Many potential bugs: XSS, XSRF, SQL injection
Reminder: Browser Sandbox

• Goal: safely execute JavaScript code provided by a website
  – No direct file access, limited access to OS, network, browser data, content that came from other websites

• Same origin policy
  – Can only access properties of documents and windows from the same domain, protocol, and port
Recap: Same-Origin Policy

• Goal: ensure that sites from different origins can’t interfere with each other:
  – DOM manipulation
  – Window navigation
  – Cookies (reading and writing)
  – Cross-site content

• Implemented in various places by the browser – some inconsistencies!
Cross-Origin Communication?

• Websites can embed scripts, images, etc. from other origins.

• **But:** AJAX requests (aka XMLHttpRequests) are not allowed across origins.

On example.com:

```html
<script>
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = handleStateChange; // Elsewhere
xhr.open("GET", "https://bank.com/account_info", true);
xhr.send();
</script>
```
Cross-Origin Communication?

• Websites can embed scripts, images, etc. from other origins.

• **But:** AJAX requests (aka XMLHttpRequests) are not allowed across origins.

• Why not?
  • Browser automatically includes cookies with requests (i.e., user credentials are sent)
  • Caller can read returned data (clear SOP violation)
Allowing Cross-Origin Communication

• Domain relaxation
  – If two frames each set document.domain to the same value, then they can communicate
    • E.g. www.facebook.com, facebook.com, and chat.facebook.com
    • Must be a suffix of the actual domain

• Access-Control-Allow-Origin: <list of domains>
  – Specifies one or more domains that may access DOM
  – Typical usage: Access-Control-Allow-Origin: *

• HTML5 postMessage
  – Lets frames send messages to each other in controlled fashion
  – Unfortunately, many bugs in how frames check sender’s origin
What about Browser Plugins?

- **Examples:** Flash, Silverlight, Java, PDF reader
- **Goal:** enable functionality that requires transcending the browser sandbox
- **Increases browser’s attack surface

---

Java and Flash both vulnerable—again—to new 0-day attacks

Java bug is actively exploited. Flash flaws will likely be targeted soon.

by Dan Goodin (US) - Jul 13, 2015 9:11am PDT

- **Good news:** plugin sandboxing improving, and need for plugins decreasing (due to HTML5 and extensions)
What about Browser Extensions?

- Most things you use today are probably extensions
- **Examples**: AdBlock, Ghostery, Mailvelope
- **Goal**: Extend the functionality of the browser

- (Chrome:) Carefully designed security model to protect from malicious websites
  - *Privilege separation*: extensions consist of multiple components with well-defined communication
  - *Least privilege*: extensions request permissions
What about Browser Extensions?

• But be wary of malicious extensions: not subject to the same-origin policy – can inject code into any webpage!