HTML5 Security Cheat Sheet

From OWASP

Contents

Introduction

HTML 5

Browser Securability Chart

There are a few sites charting browser capabilities as they related to the HTML 5 / CSS 3 standard. I have not seen any that mention security. There may not be a need for it, but e.g. 'sandbox' will be ignored in down browsers, but which HTML 5 compliant browsers support it. If there are differences in implementations, my assumption is that there will be differences in security configuration / settings.

Cross Origin Resource Sharing

- Validate URLs passed to XMLHttpRequest.open, current browsers allow these URLS to be cross domain.
- Ensure that URLs responding with Access-Control-Allow-Origin: * do not include any sensitive content or information that might aid attacker in further attacks. Use Access-Control-Allow-Origin header only on chosen URLs that need to be accessed cross-domain. Don't use that header for the whole domain.
- Take special care when using Access-Control-Allow-Credentials: true response header. Whitelist the allowed Origins and never echo back the Origin request header in Access-Control-Allow-Origin.
- Allow only selected, trusted domains in Access-Control-Allow-Origin header. Prefer whitelisting domains over blacklisting or allowing any domain (either through * wildcard or echoing the Origin header content).

Input Validation

Local Storage (a.k.a. Offline Storage, Web Storage)

WebDatabase

WebSockets

- Drop backward compatibility in implemented client/servers and use only protocol versions above hybi-00. Popular Hixie-76 version and olders are outdated and insecure.
- While it is relatively easy to tunnel TCP services through WebSockets (e.g. VNC, FTP), doing so enables access to these tunneled services for the in-browser attacker in case of a Cross-Site-Scripting attack. These services might also be called directly from a malicious page or program.
- The protocol doesn't handle authorisation and/or authentication. Application-level protocols should handle that separately in case sensitive data is being transferred.
- Endpoints exposed through ws:/ protocol are easily reversible to plaintext. Only wss:// (WebSockets over SSH) should be used for protection against Man-In-The-Middle attacks
- Spoofing the client is possible outside browser, so WebSockets server should be able to handle incorrect/malicious input. Always validate input coming from the remote site, as it might have been altered.
- When implementing servers, check the Origin: header in Websockets handshake. Though it might be spoofed outside browser, browsers always add the Origin of the page which initiated Websockets connection.
- As WebSockets client in browser is accessible through Javascript calls, all

Websockets communication can be spoofed or hijacked through Cross-Site-Scripting. Always validate data coming through WebSockets connection.

Geolocation

Use the "sandbox" attribute for untrusted content (iFrame)

[[1] (http://blog.whatwg.org/whats-next-in-html-episode-2-sandbox)]

Content Deliverability

CDN or src links to foreign domains = know your content

Progressive Enhancements and Graceful Degradation Risks

The best practice now is to determine the capabilities that a browser supports and augment with some type of substitute for capabilities that are not directly supported. This may mean an onion-like element, e.g. falling through to a Flash Player if the <video> tag is unsupported, or it may mean additional scripting code from various sources that should be code reviewed.

CSS 3

I haven't seen any specific to CSS 3 and it's been a while since I worried about url / !import. I think privacy leaks are the most well know - e.g. querying global history using :visited (https://bugzilla.mozilla.org/show_bug.cgi?id=147777)

Javascript and Javascript Frameworks

Do we have cheatsheets for Javascript (e.g. use closures, protect the global namespace) or any of the frameworks like JQuery, script.aculo.us, Prototype, Mootools

Related Cheat Sheets

Other Articles in the OWASP Cheat Sheet Series

- Authentication Cheat Sheet
- Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet
- Cryptographic Storage Cheat Sheet
- Input Validation Cheat Sheet
- DOM based XSS Prevention Cheat Sheet
- Forgot Password Cheat Sheet
- **HTML5 Security Cheat Sheet**
- SQL Injection Prevention Cheat Sheet
- Security Architecture Cheat Sheet
- Session Management Cheat Sheet
- Transport Layer Protection Cheat Sheet
- XSS (Cross Site Scripting) Prevention Cheat Sheet
- Web Service Security Cheat Sheet

Authors and Primary Editors

Mark Roxbury - mark.roxberry [at] owasp.org

Krzysztof Kotowicz - krzysztof [at] kotowicz.net