When Good Standards Go Bad

IETF 83 | March 25, 2012
Chris Weber, Casaba Security
Problem Statement

The implementation of new browser features can counter-intuitively open applications and their users up to attacks that were not possible before...
Shouldn’t security be getting easier?

- Do you need to be a rocket scientist to build a Web application today?
- With security, you can think you’ve covered everything, but that one misplaced switch out of 1000 could short-circuit the entire operation.
- It seems that getting security right is becoming harder, or at least more confusing.
A mixed bag of mitigations

- Same-origin policy
- Content-Security-Policy
- iframe sandbox
- postMessage
- CORS
- toStaticHtml
- anti-CSRF
- anti-Clickjacking
- Cryptography

- X-Frame-Options
- X-Content-Type-Options
- HttpOnly, Secure cookies
- Cache-Control
- Strict-Transport-Security
- Access-Control-Allow-Origin
- Content-Type
- Content-Disposition
Web application, meet Web browser

• You don’t own your primary interface but you still have to balance allowing it and protecting against it...

• But you have to support lots of clients:
  – PC, Mac, Linux
  – IE 7/8/9, Firefox 5-10, Chrome, and Safari
  – Mobile smartphones and tablets

• And transition pains when Web browsers change.
Example: Facebook compromised using CORS

• A certain feature on Facebook would take a URL like:
  http://touch.facebook.com/#profile.php

• Then make an XmlHttpRequest (XHR) to “profile.php” and load the response content into the main document.
Example: Facebook compromised using CORS

• Before Cross-Origin Resource Sharing (CORS), an attacker couldn’t do this:
  http://touch.facebook.com/#http://evil.example.org/foo

• Because it would naturally be prohibited by the XHR same-origin policy but post-CORS, the attack works...
Example: Bypassing HTML sanitizers with HTML5

```
javascript:alert(1)
```

Is now equivalent to

```
javascript&colon;alert(1)
```
Example: Inline SVG support opens up XSS

Mozilla Foundation Security Advisory 2011-27

Title: XSS encoding hazard with inline SVG
Impact: Moderate
Announced: June 21, 2011
Reporter: Mario Heiderich
Products: Firefox, SeaMonkey

Fixed in: Firefox 5
SeaMonkey 2.2

Description
Security researcher Mario Heiderich reported that HTML-encoded entities were being improperly decoded when displayed inside SVG elements. This could lead to XSS attacks on sites relying on HTML encoding of user-supplied content.
Some root causes

• Implementation quirks are not well-known.
• API security considerations may be documented but are not widely understood by application developers.
• Interoperability
  – “We can’t implement protection X because browser Y doesn’t support it yet, so we need to do Z for now”
• Transition pains while churning to new standards.
References

• The Tangled Web, by Michal Zalewski

• http://m-austin.com/blog/?p=19

• http://heideri.ch/jso/#html5