Safety against Clickjacking / UI Redressing Attacks

Brad Hill <bhill {at} paypal-inc.com>
Jeff Hodges <jeff.hodges {at} kingsmountain.com>

Clickjacking / UI Redressing

A web application can induce the web user agent to include, frame or embed another application from a different security domain.

In so doing, it may be able to convince the user to interact with the nested application out-of-context, by obscuring or modifying the nested application's presentation to the user.

Two types of cross-origin mixing:

–Transclusion: *OUT OF SCOPE*

inlined content which becomes part of the same web security principal (img, font, etc.)

–Framing / Embedding: *IN SCOPE*

distinct browsing contexts, with different security principals (origins) and enforced security boundaries (iframe, plugins)

Attacks arise due to incomplete isolation at the *User Interface* (presentation) layer

Difficult problem to solve

- User Interface context mixing is *by design* and a *desirable* property of the web user agent
  - Except when it isn't
- No unambiguous fixes possible at the protocol or browsing context security model
- Diversity of user agent / user interface features:
  - Modal vs. multi-window, mouse vs. touch, voice or assistive technologies

(legacy) X-Frame-Options Header

- DENY, SAMEORIGIN, [ALLOW-FROM]
  - All-or-nothing
    - Use cases which *require framing* cannot use this policy (e.g. like, +1)
- But..
  - Application authors need more granularity:
    - Allow, and apply protections if possible
    - Only allow if possible to apply protections
    - Report, don't block, if things look suspicious

“UI Safety” spec in WebAppSec WG

http://dvcs.w3.org/hg/user-interface-safety/raw-file/tip/user-interface-safety.html

- Use Content Security Policy header to convey *UI policy* and *tuning hints* to the user agent

- Non-normative recommendations on how to apply such recommendations at the user agent
  - Screenshot comparisons to detect overlays, repositioned content
  - Click timing measurements
  - Etc.

“UI Safety” spec @ W3C WebAppSec WG

From the web application provider's perspective, “frame options” and UI Safety are part of a single risk management policy around how the web user agent manages the application’s user interface.

Going forward, it may make the most sense to define both policy pieces in the same spec.

Advantages to moving “frame options” features to CSP UI Safety specification:

Simplify policy combination

If “frame options” going forward is moved to CSP UI Safety, app authors can write one policy to express all framing/embedding requirements, and optionally supply an XFO policy for legacy user agents.

If moved to a “FRAME-OPTIONS” header, app authors must always explicitly think about policy combination logic across two different specifications, defined in slightly different terms.

Single conveyance mechanism may give broader adoption

- "frame options" policies really are associated with the *web application user interface*, not the underlying network protocol

- Chrome extensions "application manifest" already has a way to set a Content Security Policy
  - App manifest would require enhancement to do same for a separately expressed "frame options" notion

- Widgets, app cache, etc.
  - All could have a mechanism to attach or persist "frame options" *and* CSP, but easier to just do one

“Frame options” features can take advantage of CSP features

- CSP specifies a reporting channel and is developing a DOM API
  - –Application authors may wish to use these for risk management with the “frame options” features
- Re-use CSP expression of origin
  - –Likely source of error to require authors to continue to use legacy XFO origin syntax and CSP's, as well as two headers to express one intention