Client-Side Link Selection Validation

Michael McNally
Tuna Toksoz
Alex Jacobson

Follow this and additional works at: http://www.tdcommons.org/dpubs_series

Recommended Citation
McNally, Michael; Toksoz, Tuna; and Jacobson, Alex, "Client-Side Link Selection Validation", Technical Disclosure Commons, (November 01, 2016)
http://www.tdcommons.org/dpubs_series/309

This work is licensed under a Creative Commons Attribution 4.0 License.
This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.
Client-Side Link Selection Validation

ABSTRACT

A method for reducing unintentional user clicks due to reflow and clickjacking in mobile apps. Links in electronic documents such as web pages may be marked as special or protected. When a protected link is present, the document presentation environment prevents selection of links that exhibit untrustworthy behavior, which might have been clicked by accident in an attempt to select the protected link. Examples of untrustworthy behavior include repositioning the link just prior to the link selection or presenting the link in a transparent manner.

DETAIL

Personal computers, tablets, mobile “smart” phones, and other such computing devices execute applications that can present electronic documents to a user of the computing device. The electronic documents include content, typically received from a server, that may also include interactive elements that, when activated, cause the device to request or retrieve additional content. The user device may retrieve such content from a server, resulting in a client/server paradigm where the user device is a client device. The interactive elements in the content may include selectable elements such as webpage hyperlinks and actionable buttons. Examples of selectable elements include webpage hyperlinks, form submission buttons, social media “like” or “share” type buttons, e-commerce “purchase” or “buy” type buttons, and so forth. These
applications may present content from multiple sources, which may include third-party content (e.g., advertisements, shared content, social content, etc.). In some instances, the application, the electronic document, or the interactive content may include functional elements that can present third-party content supplied by a server. In some instances, the server may select what third-party content to present. In some instances, the application, the electronic document, or the interactive content may allow, intentionally or unintentionally, for third-party content to be presented in place over a selectable element of the interactive content. As a result, an attempt by a user to select the covered element of the interactive content may be intercepted or redirected to a third-party link. This is referred to as “clickjacking.”

Applications that present content with interactive elements include web-browsers, e-book readers, news-readers, electronic mail (e-mail) managers, media players, games, utilities, and so forth. An example form of interactive content is a web page, e.g., an electronic document formatted in accordance with the hypertext markup language (HTML) that may include dynamic elements (e.g., JavaScript) and actionable or selectable elements such as hyperlinks to uniform resource locators (URLs). The application may present interactive content that is local to the client-side device or may receive the interactive content from a remote server via a network. In some instances, the interactive content may include elements that dynamically update or retrieve additional content once the interactive content is activated or loaded for presentation. In some instances, the interactive content includes multi-media elements such as video, audio, or haptic data. Presentation of the interactive content may include displaying text, image, or video content on a screen or display. Presentation of the interactive content may include playing audio through speakers. Presentation of the interactive content may include vibrating haptic feedback devices.
In general, the concern with clickjacking is presentation of interactive content in which an interaction, e.g., selection of a hyperlink, can be surreptitiously redirected by a third-party, for example, by causing a user to inadvertently select a link different from one the user intended to select. There are many ways in which a third-party link can be inadvertently selected when a user attempts to select a link. For example, the third-party link can be placed in a re-locatable “floating” presentation layer that moves into a display or presentation position layered above the intended link (or selectable element encompassing the intended link). As another example, in some applications, the third-party link can be presented in a transparent manner, e.g., within a graphic that is fully transparent. In some applications, the third-party link can be very small, effectively invisible to the user, e.g., by using a small number of display pixels.

Allowing third-party content to intercept or redirect user selections is not desirable. It can erode user trust in the application or content source, frustrate users, and possibly place the user or the client device at risk. For example, if the redirected link may cause the client device to retrieve or download executable data from a third-party that could include a computer virus or other malicious software (“malware”). As another example, the redirected link selection could present information or data collection forms where the source is unclear, potentially confusing a user into providing personal or confidential information to a third-party (this is a form of attack known as “phishing”). Accordingly, it is desirable to prevent such inadvertent element selections.
An example of clickjacking is shown in FIG. 1. Illustrated in FIG. 1 are screenshot 100 and screenshot 104, each presenting content 125 in a display 120. The content 125 includes selectable elements, e.g., up and down rating buttons and a comment button 130. The screenshot 100 includes expanding advertisement 140 and banner advertisement 142. The screenshot 104 includes an expanded advertisement 144 and the banner advertisement 142. The expanded advertisement covers the selectable comment elements, e.g., causing the comment button 130 to be obscured 134. The expanding advertisement 140 might, for example, expand after a predetermined period of time, e.g., 5 seconds after the page loads, or after an event, e.g., a mouse-over event, or after some other triggering condition, e.g., any movement of the mouse or detection of a finger or stylus in close proximity to a screen, or after any other trigger event.
An application developer could prevent clickjacking by avoiding third-party elements, or by pre-screening all third-party elements. However, these “walled garden” approaches can be onerous for the application developer and restrict what the developer can do with the application. In some instances, for example, a developer may wish to sell space on a webpage to an advertising network, where the advertising network chooses what ads to present on the webpage. Some developers work in a collaborative environment, where some portions of an application or electronic document are authored by different developers with different objectives or concerns. For example, a content author may work with a webpage designer to present content, where the author might not have editorial control over the final page design. Accordingly, it is desirable to prevent clickjacking using a minimum of burden on application developers, webpage designers, and other types of interactive content authors and architects.

The client device provides a host execution environment for the application, which is responsible for processing network link selections. The host execution environment may include, for example, the client device operating system and any dynamically loadable application programming libraries used by the application. The host execution environment may be an encapsulating application. That is, the document presentation application may be executing within another hosting application. For example, the hosting application may be a web browser and the document presentation application may be a dynamic page loaded by the web browser. The host execution environment is well positioned to block link selections when a set of suspect conditions exist. For example, a selectable element of an interactive content may be marked as special and, when a link is selected that meets conditions indicating interference with the special element, the host execution environment can determine that the link selection is
suspect. The host execution environment can then either block the suspect link selection or take a predefined action such as redirecting the suspect link selection to the special element.

**FIG. 2** is a flowchart of a method of determining whether to allow or reject a link selection. In broad overview, at stage 210, an application executing in a host execution environment on a client device generates a model of an electronic document. At stage 220, the application receives or detects a link selection, which it then passes to the host execution environment for processing. At stage 230, the host execution environment determines, from the document model, whether the document includes a protected link. If not, the host execution environment allows the link selection at stage 250. Otherwise, at stage 240, the host execution environment determines whether the link selection satisfies additional “elevated” criteria. The host execution environment then only allows the link selection at stage 250 if the link selection
satisfies the elevated criteria at stage 240. Otherwise, at stage 260, the host execution environment rejects the link selection.

Referring to the flowchart in FIG. 2 in more detail, at stage 210, an application executing in a host execution environment on a client device generates a model of an electronic document. For example, the application (e.g., a web browser application) may generate a document object model (DOM) for the electronic document. Any document model can be used as long as it can indicate which links (if any) are protected. In some implementations, a link is indicated as protected by including a special tag. For example, the tag might be a “PROTECT” tag and protected links (e.g., an HTML anchor) might read:

```
<A HREF="LINK_URL" PROTECT="1">Link Text</A>
```

At stage 220, the application receives or detects a link selection, which it then passes to the host execution environment for processing. A link might be selected, for example, by a user tapping on a touch screen or clicking on a link using a mouse. In some instances, a link might be selected using another interface. When the application detects the link selection, it passes the selection to the execution environment for completion.

At stage 230, the host execution environment determines, from the document model, whether the document includes a protected link. If not, the host execution environment allows the link selection at stage 250. Otherwise, at stage 240, the host execution environment determines whether the link selection satisfies additional “elevated” criteria.

At stage 240, the host execution environment determines whether the link selection satisfies additional “elevated” criteria. The elevated criteria are configured to identify links that exhibit untrustworthy behavior. For example, if a link is in a presentation element that moves
or expands, this may suggest that the link is being repositioned to intercept a selection attempt. Accordingly, the elevated criteria may identify links that have been repositioned or moved, links that have been repositioned or moved within a specified time window prior to the selection event, or links that have been repositioned or moved to within a specified presentation proximity to the protected link. As another example, a link may be included with a visual presentation element such as an image file. The image may be very small, e.g., a single pixel. Accordingly, the elevated criteria may identify links associated with image files smaller than a specified threshold size. The link may be presented transparently, e.g., in association with a transparent image file. Accordingly, the elevated criteria may identify links associated with image files having a transparency property. In some implementations, the elevated criteria may identify links associated with image files having a transparency property that is more transparent than a partial transparency threshold (e.g., more than 50% transparent). In some implementations, if the selection event points to a destination (a URL) other than the destination indicated in the link marked special, the selection event is blocked. Other behaviors may also suggest an attempt to redirect user interactions and the criteria may test for these behaviors as well.

If the link selection satisfies the elevated criteria at stage 240, the host execution environment then allows the link selection at stage 250. Otherwise, at stage 260, the host execution environment rejects the link selection. In some implementations, when a link selection is rejected at stage 260, the application provides feedback to the user indicating that the link selection was rejected. For example, the application may pop a warning window stating that the selection was rejected. In some such instances, the warning window might include an opportunity to confirm the user’s intent to select the link (e.g., showing the actual destination of the link selected). In some implementations, the application rejects the link silently such that,
from the user perspective, it is as though no link was selected at all. In some implementations, if the protected link is presented in close proximity to the rejected link selection, the protected link is selected in lieu of the rejected link (unless the rejected link is, itself, marked for the special protected status).

In the above description of FIG. 2, a protected link anywhere in an electronic document would trigger the elevated criteria analysis in stage 240. In some implementations, when a selectable element marked for the special protected status is presented by the client device, only selection events proximate to the protected element are subjected to the screening process.

In this manner, if the selection event is directed to a link in a visual element that had been recently moved (e.g., within a preset number of seconds prior to the selection), action on the selection is blocked. If the selection event is directed to a link in a visual element that is marked transparent, action on the selection is blocked.