Avoiding "Mixed Content" Issues In Your Client-Side Scripts

Author	josh
Date	2017-10-10 00:05:56

Overview

Web applications are particularly vulnerable to security issues, given that the internet is one giant, unsecured network that *anyone* can access. One of these major vulnerabilities is man-in-the middle attacks, where a malicious user accesses content sent between the client (the machine receiving web content) and the server (the machine providing web content).

Fortunately, the widespread adoption of SSL/TLS has improved the security of web pages by encrypting content between client and server and making it difficult for man-in-the-middle attacks to occur. It is important, however, that web pages and web applications are properly marked up and coded to avoid having content sent unsecured.

Avoiding mixed content for greater security

What is mixed content?

Mixed content refers to *unsecured* (HTTP) content loaded within a secure page (HTTPS). For example, this content could be an image, video, or audio file requested by the page with an unsecure URL. This could also be a script loaded by the page over HTTP. Scripts loaded unsecurely can potentially be more dangerous than media files like images.

Why is mixed content a problem?

Privacy concerns

Mixed content can create several problems for an end user of your website. The first issue is that of privacy. When a user loads a page over SSL/TLS (HTTPS), the only thing someone snooping on the network (a "man-in-the-middle") can see is the domain of the page being requested. For example, a if user visiting `https://somesite.com/sensitive-article`, the only thing a snooper would see is that the user is requesting content from `somesite.com`. They can't see the particular page the user is requesting. However, what if the page `sensitive-article` requests an image over HTTP? Then the snooper could see the unencrypted image, possibly giving them clues about what the end user is viewing!

When browsing with HTTPS, the user has some expectation of privacy between his or her self and the server. Mixed content breaks that by exposing some parts of the page in plaintext as they travel across the wire, potentially giving some clues as to what the user is viewing.

Code security concerns

A second concern with mixed content is that a script sent in plain text could potentially be modified en-route to the client. If a script is sent over HTTP, a malicious user could potentially intercept that script and inject some malicious code of their own. This code could be used to collect sensitive information or trick the user into visiting an illegitimate website (very common with phishing scams). The end user expects a legitimate script sent securely from the server (they requested a page over HTTPS, after all), but in a mixed content scenario this code could be intercepted and modified by an attacker.

How to avoid mixed content

Use relative links as much as possible

One of the easiest ways to avoid mixed content in web pages is to use relative links. With a relative link, the browser will use the protocol the page was requested with by default, no hard-coding of a protocol required. For example, loading your images from a subfolder called `photos` can be done with a relative path like so: `src="photos/myimage.jpg"`. If the user requests your page over HTTPS, the image will be loaded over HTTPS as well.

Use // to auto-detect the protocol for outside links, or always use HTTPS

Sometimes you may want to load content from outside your own domain and your own web server. Many developers like to load scripts like jQuery from Content Delivery Networks (CDN's) rather than store those scripts on their own server. In this case, you can use `//` in front of the URL instead of a hard coded protocol like `http://`. This will match the protocol the page was requested from. For example, `src="//cdn.jquery.com/some-jquery-version.js"`. Better yet, if you know the site provides content over HTTPS, just use that by default!

Avoiding mixed content is fairly straightforward!

While mixed content can pose several issues for user privacy and application security, it is thankfully trivial to mitigate. Avoiding mixed content problems is as easy as fixing the links in any pages your site provides. If you're not sure your pages avoid this problem, the developer console in a modern browser like Firefox will tell you if you're trying to load insecure content. Many browsers (including Firefox) even block that content from loading. Be sure to be mindful of how media and scripts are requested in your web applications, and your users will be more secure for it!