Cyber Security Division Advisory 2015-09-10

Cross Domain applications in Embedded IFRAME
Original release date: September 9, 2015
Source: Cyber Security Division

Systems affected:
All browser based applications using IFRAME (eg. playing video, maps etc…)

Overview:

Most of the current browsers support HTML5 and could playback video available on the website. These browsers also offer user the ability to embed and view its videos on web pages outside their domain. Each video is generally accompanied by a piece of HTML that can be used to embed it using IFRAME on any page on the Web.

In case the video is embedded via IFRAME within the same domain it is usually safe and trustful and does not impose any security implications. Also, in view of any third-party code added directly to web application which does not use IFRAMEs, same access rights are passed on to the embedded code.

However, if a web application is using IFRAME from other domain then browsers effectively maintain the context of the IFRAME and web application separate by default. In this state the Web application and IFRAME do not have access to each other's DOM, CSS styles, JavaScript code. Also, if the IFRAME comes from a different domain, a browser's cross-domain policy prevents the IFRAME from accessing cookies, local storage, or the DOM from its embedding document.

The cross-domain IFRAMEs generally have the ability to trigger alerts, run plugins (malicious or otherwise), auto play videos, and present submittable forms in an attempt to phish users' information. Thus, if the requirement of an IFRAME is to perform a certain task (like playing a video or map navigation etc) then, there should be some secure mechanism in place to restrict it to the specific task only.

Description:
Few of the mechanisms that are incorporated by default in restricting IFRAMEs is using of *sandbox* attribute for playing a video:

e.g. *<IFRAME sandbox src="http://abc.com"></IFRAME>*

Hardcoded IFRAME
IFRAME should be hardcoded and only allow specific domain to be rendered as an iframe in the application. Application should not allow any type of modification in defined IFRAME.

Disable IFRAME inside IFRAME
IFRAME should not contain another IFRAME inside it. IFRAME inside IFRAME options should be disabled.

Disable Javascript inside an IFRAME
Disabling javascript inside and IFRAME is also another type of security measure. By disabling javascript inside IFRAME we can restrict any malicious/harmful activity via an IFRAME.

Disable IFRAME to Browser Communication
IFRAME should not be able to communicate or have any access on browser properties. IFRAME should not have access to parent domain resources, cookies and files. If the communication is required then it should be done in secure manner using Cross-document messaging.

Risk:

- Large ,medium and Small government entities: High

Recommendations:

Sandbox attribute helps to enable several tasks that are essential for the embedded IFRAME by allowing us to specify a space-separated list of permissions, with one or more of the following choices. These need to be used only on requirement basis.

Add *sandbox* attribute in IFRAME so that desired security level can be achieved to a certain level.

e.g. *<IFRAME sandbox src="..."></IFRAME>*

With sandbox attribute set, by default the document inside the IFRAME cannot do the following:

- Run any JavaScript, even if it would only affect contents of the IFRAME.
- Change the parent's URL
- Open pop-ups, new windows, or new tabs
- Submit forms
- Run plug-ins
- Use pointer lock
- Read cookies or local storage from the parent, even if it's from the same origin

The tags which are used along with IFRAME with their features are explained below:-

- *allow-same-origin* - allows the IFRAME to access cookies and local storage from the parent, as if it came from the same domain.
- *allow-top-navigation* - allows the IFRAME to navigate the parent to a different URL.
- *allow-forms* - allows form submission
- *allow-scripts* - allows JavaScript execution
- *allow-popups* - allows the IFRAME to open new windows or tabs
- *allow-pointer-lock* - allows pointer lock
- *src* = URL potentially surrounded by spaces
  Gives the address of a page that the nested browsing context is to contain.
- *srcdoc* = HTML contents
  Gives the content of the page that the nested browsing context is to contain.
- *name* = browsing-context name
  Represents the browsing-context name. When the browsing context is created, if the attribute is present, the browsing context name must be set to the value of this attribute; otherwise, the browsing context name must be set to the empty string.
- *sandbox* = allow-same-origin/ allow-top-navigation/ allow-forms/ allow-scripts
- *seamless* = boolean
  Indicates that the iframe element's browsing context is to be rendered in a manner that makes it appear to be part of the containing document (seamlessly included in the parent document).
- *width* = non-negative integer
  Give the width of the visual content of the element, in CSS pixels.
- *height* = non-negative integer
  Give the height of the visual content of the element, in CSS pixels.

References:

https://html.spec.whatwg.org/multipage/comms.html#web-messaging
http://www.w3.org/TR/html-markup/iframe.html
https://www.w3.org/wiki/HTML/Elements/iframe
http://www.w3schools.com/tags/att_iframe_sandbox.asp