DISADVANTAGES PRESENTED BY HTML INLINE FRAMES IN INTEGRATION OF 3rd PARTY CONTENT

Joița Alin-Cristian[1]

Abstract

As new virtual structures emerge, new applications, new widgets, new services become available to embed in websites. One of the preferred solutions for embedding 3rd party content is the HTML Inline Frame or iFrame. In this context, the introduction establishes the importance of the matter: major market players like Facebook, Google and Microsoft decided to include this tag in their solutions, but is it safe? Also, what other problems might webmasters face by implementing it? The results of the research, problems and security threats, are classified in five categories: cross-domain communication, reflection attacks with XSS (cross-site scripting), CSS Overlay, URL Redirection and Host Content Dependence. For each group, examples and code samples are provided, where applicable.

Keywords: iFrame, security, cross-site scripting(XSS), cross domain, malware

JEL Classification: L86

Introduction

The raise in popularity of web services, application clouds, software-as-a-service, search engines, social networks and all other new virtual structures brought upon the Internet not only more functionality, but also more vulnerability. The main causes for the added risks are not the concepts themselves but their improper implementation. Thus, inter-domain data transfer has not seldom proven to be the root for data loss, inoperability and malicious activities that produced heavy damages. Malware is at a peak for underground hacking communities began obtaining financial benefits by creating and maintaining exploitation frameworks and packs such as ZeuS, Fiesta and MPack (Erasmus 2009) (Stubley 2007).

It is important to discuss iFrame disadvantages in an attempt to establish what types of data would be recommended to embed and display and also determine best practices. As of 2004, Karspersky Labs statistics show that more than 75 percent of computer viruses are using IFRAME in their propagation technique (Nikishin 2004). Attacks vary from displaying alerts and provoking epilepsy seizures (Gordon 2008) to spamming Google's My Search History (Richardson 2005), stealing bank account information (Bradbury 2010) and more (Watson 2007).

One of the HTML elements that imply inter-domain data transfer is the iFrame. W3 Consortium defines the element as a nested browsing context. It is often chosen as a form

[1] Teaching assistant, Facultatea de Informatică Managerială, Universitatea Româno-Americană; Ph.D. student, Academy of Economic Studie, Bucharest, Romania; E-mail: alincristianjoita@yahoo.com

of implementation and embedding external applications written by 3rd parties more or less trusted or responsible.

What could go wrong with IFRAMEs?

Since 1996, when it was introduced by Microsoft along with Internet Explorer 3+, iFRAME presented itself not only as an asynchronously content loading element, but also as a preferred mechanism for mischief. In order to balance the situation, starting with HTML5, a new MIME type has emerged: text/html-sandboxed. This allows the embedded content to be treated as hostile and protect the user from receiving malware infections. Also, the iFRAME tag acquired more attributes, one of which levels-up security: sandbox. When this attribute is set and with an empty value, the following restrictions apply:

- frame content is prevented from accessing data from the same server
- plug-ins are disabled inside iframe
- scripts are disabled
- links are forbidden to target other frames
- forms are disabled

Even with these restrictions, HTML code is still prone to vulnerability errors generated by poor practices in programming or misuse of the IFRAME (Consortium 2011). Moreover, delayed browser implementation of the new specifications renders the new technology useless. In addition, L. David Baron, developer advocate for Mozilla, asserts that standards and specifications might be deliberately flawed and delayed implemented as to still sustain the market for offline software. No evidence was brought to sustain his belief (Baron 2009).

In the following paragraphs, problems and threats have been classified and accompanied by examples, where applicable.

Browser cross-domain communication

The embedded page is loaded and then the user can react upon the embedded content as being the site that embedded the page. This can lead to a flawed representation of the source of the attack. Also, the content in the child frame could be modified by script in parent window and vice-versa. This is only possible if the Same-Origin Policy is overridden or if a server side script is used to acquire and manipulate the embedded content.

The disadvantage is that communication between legitimate pages through iframes has become more difficult.

Reflection attacks with XSS

Reflection attacks are also called non-persistent and their target is loading malicious hyperlinks on clients or stealing browser cookies. Embedded iFrames on a compromised

site are used to reflect attacks on different servers as to conceal the focal point of the threat.

Persistent XSS attacks consist in placing a JavaScript code or an iFrame on a webpage. The iFrame embeds code that tracks user activity or steals user cookies to obtain his identity. Moreover, if a malware scanner reaches this infected site, it will register it on a shared blacklist in a security cloud as a harmful destination for users. This could attract a permanent ban from all the services owned by the partners in the security cloud. For example, if WebSense ThreatSeeker detects a malware site, it is banned from Google and Twitter as both are part of the Websense Security Cloud (Bradbury 2010).

A sample of the code for stealing cookies is presented below.:

```
<script type="text/javascript">
 var ifram = document.createElement("iframe");
  ifram.src = "http://www.siteHacker.ro/?cookie=" + document.cookie;
  ifram.style.visibility = "hidden";
  document.body.appendChild(ifram);
 </script>
```

CSS overlay

Because iFrames can be embedded inside or over each other, the user can be confused as to what site is the content really on. In this manner, one can be determined to interact with malicious script, having trust the site that was overlaid. Also, CSS Parser can be used alone to obtain and transmit private information from the victim(e.g. Twitter account, Hotmail) to the attacker's server, exploiting a flaw in Internet Explorer as stated in a study by Google and Carnegie Mellon University (Lin-Shung, et al. 2010). Other examples are available (Heyes 2007).

URL redirection

Javascript on embedded page can perform a redirect, thus having access to pages that were not intended to appear on the site. As an example, an unsecured e-mail can be loaded in an iFrame and after a login operation, the iFrame redirects to user private e-mail inbox. Moreover, because the iFrame can establish the location for the parent window, the iFrame could redirect the user to a phishing site. This represents a threat mostly if the host site is malicious or compromised.

To replace the window parent with content from iframe, the following JavaScript has to exist in iFrame:

```
<script type="text/javascript">
  if (top.location.href != self.location.href)
     top.location.href = self.location.href;
</script>
```

As a Proof of Concept, URL redirection has been experienced on major e-mail systems using Microsoft Internet Explorer 9, Mozilla Firefox 3.6 and Google Chrome 12.

For the experiment, two web pages were created, index1 and index2. Index2 contained and iFrame that included external content set by the “src” attribute. In this form it was embedded in an iFrame in index1.

The conclusions were mostly satisfactory: Hotmail and Yahoo refused to load the Sign-in page if embedded in an iFrame, Gmail redirected without confirmation the entire site to https://www.gmail.com thus avoiding problems and Facebook displayed a logo and a redirect link to the main site. This behavior displays appreciated concern for the safety of their users by reducing the number of fraud schemes taking place on the “bullet-proof hosting” servers (Watson 2007).

Host Content Dependence

Content in an iFrame from a different domain, port or protocol is embedded as it is, unless intermediation through a server side script occurs. When server-side scripts are not used, a series of impediments arise:

- if the host of the website being imported is hacked, the content being displayed will be hacked;
- the parent window has to rely on the site hosting the child frame content to provide error free code, as it cannot intervene due to same-origin policy;
- content cannot be filtered on the parent window and the appearance cannot be changed.

Conclusion

Although it seems to be the preferred HTML tag for hackers in their attacks, iFrames proves stable and trusty, with few real disadvantages. Most of the problems appear from exceptions generated by bad browser versions, by invalidated user input, uncertified sources for embedded content or bad intentions. Moreover, iFrame is a very popular choice among methods of exporting widgets and web applications supported and created by the major players on the market like Facebook, Google, Microsoft and Yahoo.

All these positive aspects will assure iFrame a place in line for continuity and improvement in and beyond the new era of HTML 5.

Acknowledgements

This article is a result of the project POSDRU/88/1.5./S/55287 „Doctoral Programme in Economics at European Knowledge Standards (DOESEC)". This project is cofunded by the European Social Fund through The Sectorial Operational Programme for Human Resources Development 20072013, coordinated by The Bucharest Academy of Economic Studies in partnership with West University of Timisoara.

Refrences

- Baron, L. David. "ex-HTML." *David Baron's Weblog*. July 2009. http://dbaron.org/log/20090707-ex-html (accessed May 2011).

- Bradbury, Danny. "Avoiding URL hell." *Network Security* 2010, no. 11 (11 2010): 4-6.
- Consortium, W3C - World Wide Web. "HTML5 - A vocabulary and associated APIs for HTML and XHTML - W3C Working Draft ." *W3.ORG.* May 2011. http://dev.w3.org/html5/spec/Overview.html#the-iframe-element (accessed May 2001).
- Erasmus, Jacques. "Anatomy of a malware attack." *Network Security* 2009, no. 1 (2009): 4-7.
- Gordon, Sarah. "Attackers target epilepsy site." *Network Security* (Elsevier Ltd), 4 2008: 2.
- Heyes, Gareth. "OpenID security CSS overlays." *TheSpanner.* 09 28, 2007. http://www.thespanner.co.uk/2007/09/28/openid-security-css-overlays/ (accessed 05 2011).
- Lin-Shung, Huang, Weinberg Zack, Evans Chris, and Jackson Collin. "Protecting browsers from cross-origin CSS attacks." *CCS '10 Proceedings of the 17th ACM conference on Computer and communications security.* New York: ACM New York, 2010. 619-629.
- Nikishin, Andrey. "Malicious software – past, present and future." *Information Security Technical Report.* 9, no. 2 (2004): 6-18.
- Richardson, Chris. *WebProNews.* May 05, 2005. http://www.webpronews.com/iframe-tag-allows-google-my-search-history-spam-2005-05 (accessed April 20, 2011).
- Stubley, Alan. "MPack packs a nasty punch." *Network Security* 2007, no. 7 (2007): 2.
- Watson, David. "The evolution of web application attacks." *Network Security* 2007, no. 11 (11 2007): 7-12.