Internet Engineering Task Force (IETF)
Request for Comments: 7034
Category: Informational
ISSN: 2070-1721

D. Ross
Microsoft
T. Gondrom
Thames Stanley

October 2013

HTTP Header Field X-Frame-Options

Abstract

To improve the protection of web applications against clickjacking, this document describes the X-Frame-Options HTTP header field, which declares a policy, communicated from the server to the client browser, regarding whether the browser may display the transmitted content in frames that are part of other web pages.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7034.

Table of Contents

1. Introduction 3
1.1. Requirements Language 3
2. X-Frame-Options Header 4
2.1. Syntax 4
2.2. Augmented Backus-Naur Form (ABNF) 5
2.2.1. Examples of X-Frame-Options 6
2.3. Design Issues 6
2.3.1. Enable HTML Content from Other Domains 6
2.3.2. Browser Behavior and Processing 6
2.3.2.1. Violation of X-Frame-Options 6
2.3.2.2. Variation in Current Browser Behavior 7
2.3.2.3. Usage Design Pattern and Example Scenario for the ALLOW-FROM Parameter 8
2.3.2.4. No Caching of the X-Frame-Options Header 8
3. IANA Considerations 9
3.1. Registration Template 9
4. Security Considerations 9
4.1. Privacy Considerations 10
5. References 10
5.1. Normative References 10
5.2. Informative References 11
Appendix A. Browsers That Support X-Frame-Options 13
Appendix B. Description of a Clickjacking Attack 13
B.1. Shop 13
B.2. Online Shop Confirm Purchase Page 13
B.3. Flash Configuration 13
Appendix C. Acknowledgements 13

1. Introduction

In 2009 and 2010, many browser vendors ([Microsoft-X-Frame-Options], [CLICK-DEFENSE-BLOG], and [Mozilla-X-Frame-Options]) introduced the use of a non-standard HTTP [RFC2616] header field "X-Frame-Options" to protect against clickjacking [Clickjacking]. HTML-based web applications can embed or "frame" other web pages. Clickjacking is a type of attack that occurs when an attacker uses multiple transparent or opaque layers in the user interface to trick a user into clicking on a button or link on another page from server B when they were intending to click on the same place of the overlaying page from server A. Thus, the attacker is "hijacking" clicks meant for page A and routing them to page B. The attacker is tricking the user (who sees the overlaying user interface content from page A) into clicking specific locations on the underlying page from server B, triggering some actions on server B and potentially using an existing session context in that step. This is an attack on both the user and on server B. In addition, server A may or may not be the attacker.

This specification provides informational documentation about the current use and definition of the X-Frame-Options HTTP header field. As described in Section 2.3.2.2, not all browsers implement X-Frame-Options in exactly the same way, which can lead to unintended results. And, given that the "X-" construction is deprecated [RFC6648], the X-Frame-Options header field will be replaced in the future by the Frame-Options directive in the Content Security Policy (CSP) version 1.1 [CSP-1-1].

A study [FRAME-BUSTING] demonstrated that existing anti-clickjacking measures, e.g., frame-breaking JavaScript, have weaknesses that allow their protection to be circumvented.

Short of configuring the browser to disable frames and scripts entirely, which massively impairs browser utility, browser users are vulnerable to this type of attack.

The use of "X-Frame-Options" allows a web page from host B to declare that its content (for example, a button, links, text, etc.) must not be displayed in a frame (<frame> or <iframe>) of another page (e.g., from host A). This is done by a policy declared in the HTTP header and enforced by browser implementations as documented here.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. X-Frame-Options Header

The X-Frame-Options HTTP header field indicates a policy that specifies whether the browser should render the transmitted resource within a <frame> or an <iframe>. Servers can declare this policy in the header of their HTTP responses to prevent clickjacking attacks, which ensures that their content is not embedded into other pages or frames.

2.1. Syntax

The header field name is:

X-Frame-Options

There are three different values for the header field. These values are mutually exclusive; that is, the header field MUST be set to exactly one of the three values.

DENY
: A browser receiving content with this header field MUST NOT display this content in any frame.

SAMEORIGIN
: A browser receiving content with this header field MUST NOT display this content in any frame from a page of different origin than the content itself.

  If a browser or plugin cannot reliably determine whether or not the origin of the content and the frame are the same, this MUST be treated as "DENY".

  Please note that current implementations vary on the interpretation of this criteria. In some, it only allows a page to be framed if the origin of the top-level browsing context is identical to the origin of the content using the X-Frame-Options directive; in others, it may consider the origin of the framing page instead. Also see Section 2.3.2.2 for more details on the nesting of frames and variations in the handling of this header field by different browsers. In addition, refer to Section 4, paragraph 2 for the resulting potential security problems.

ALLOW-FROM (followed by a serialized-origin [RFC6454])
: A browser receiving content with this header MUST NOT display this content in a frame from any page with a top-level browsing context of different origin than the specified origin. While this can

expose the page to risks by the trusted origin, in some cases, it may be necessary to allow the framing by content from other domains.

The meaning of the term "serialized-origin" is given in [RFC6454]. If the ALLOW-FROM value is used, it MUST be followed by a valid origin [RFC6454] (as a subset of the URI [RFC3986]).

Any data beyond the domain address (i.e., any data after the "/" separator) is to be ignored. The algorithm to compare origins from [RFC6454] SHOULD be used to verify that a referring page is of the same origin as the content (in the case of SAMEORIGIN) or that the referring page's origin is identical with the ALLOW-FROM serialized-origin (in the case of ALLOW-FROM). Though in conflict with [RFC6454], current implementations do not consider the port as a defining component of the origin; i.e., existing implementations differ with [RFC6454] in that origins with the same protocol but different port values are considered equivalent.

Wildcards or lists to declare multiple domains in one ALLOW-FROM statement are not permitted (see Section 2.3.2.3).

2.2. Augmented Backus-Naur Form (ABNF)

The RFC 5234 [RFC5234] ABNF of the X-Frame-Options header field value is the following:

```
X-Frame-Options = "DENY"
          / "SAMEORIGIN"
          / ( "ALLOW-FROM" RWS SERIALIZED-ORIGIN )

RWS             = 1*( SP / HTAB )
                ; required whitespace
```

with serialized-origin as defined in [RFC6454] and required whitespace (RWS) as defined in [HTTPbis-P1].

RWS is used when at least one linear whitespace octet is required to separate field tokens. RWS SHOULD be generated as a single space (SP). Multiple RWS octets that occur within field-content SHOULD either be replaced with a SP or transformed to all SP octets before interpreting the field value or forwarding the message downstream.

SP and horizontal tab (HTAB) are as defined in Appendix B.1 of RFC 5234 [RFC5234].

The values are specified as ABNF strings; therefore, they are case-insensitive.

2.2.1. Examples of X-Frame-Options

```
X-Frame-Options: DENY

X-Frame-Options: SAMEORIGIN

X-Frame-Options: ALLOW-FROM https://example.com/
```

2.3. Design Issues

2.3.1. Enable HTML Content from Other Domains

There are a number of main direct vectors that enable HTML content from other domains, and browser implementations of X-Frame-Options cover all of them:

- IFRAME tag
- Frame tag
- Object tag (requires a redirect)
- Applet tag
- Embed tag

Besides these, other ways to host HTML content can be possible. For example, some plugins may host HTML views directly. If these plugins appear essentially as frames (as opposed to top-level windows), the plugins must conform to the X-Frame-Options policy as specified in this document as well.

2.3.2. Browser Behavior and Processing

To allow secure implementations, browsers must behave in a consistent and reliable way.

If an X-Frame-Options HTTP header field prohibits framing, the user agent of the browser MAY immediately abort downloading or parsing of the document.

2.3.2.1. Violation of X-Frame-Options

When a browser discovers that loaded content with the X-Frame-Options header field would be displayed in a frame against the specified orders of the header, the browser SHOULD redirect to a "NOFRAME" page as soon as possible. For example, this can be a noframe.html page that also states the full URL and hostname of the protected page.

The NOFRAME page could provide the user with an option to open the target URL in a new window.

Implementations of this vary: some browsers will show a message that allows the user to safely open the target page in a new window, whereas other implementations will simply render an empty frame.

2.3.2.2. Variation in Current Browser Behavior

There are currently variations in the implementation of the X-Frame-Options header. For example, not all browsers support the "ALLOW-FROM" option. "ALLOW-FROM" was initially an Internet Explorer extension and, at the time of writing, has not been uniformly implemented by other user agents.

Furthermore, the criteria for the SAMEORIGIN (and ALLOW-FROM) directive may not be evaluated unanimously either: the known implementations in Appendix A evaluate the SAMEORIGIN directive based on the origin of the framed page and the top-level browsing context, while other implementations might evaluate it based on the framed page and the framing page, or the whole chain of nested frames in between.

To illustrate the difference between the comparison of the "framing page" and the "top-level browsing context", consider the following scenario: web pages may embed frames with other pages that, in turn, embed frames with other pages as well, and so on. In theory, this can result in an infinite nesting of framed pages. For example, web page A may contain web page B in a frame, and web page B may contain web page C in a frame.

```
Web page A
<html>
....
<frame src="https://URI_of_web_page_B" />
</html>

Web page B
<html>
....
<frame src="https://URI_of_web_page_C" />
</html>
```

and so forth.

In this example, for the nested frames with the inner-framed web page C, the most outer web page A would be the "top-level browsing context", and web page B would be the "framing page".

These potential variations in the evaluation of the header by different implementations impair the usage and reliability of this HTTP header and have security implications as described in Section 4. A revised version of X-Frame-Options in the form of a Frame-Options directive in CSP 1.1 [CSP-1-1] will unify the behavior, and it is expected that newer implementations will use it rather than the mechanisms documented here.

2.3.2.3. Usage Design Pattern and Example Scenario for the ALLOW-FROM Parameter

As the "ALLOW-FROM" field only supports one serialized-origin, in cases when the server wishes to allow more than one resource to frame its content, the following design pattern can fulfill that need:

1. A page that wants to render the requested content in a frame supplies its own origin information to the server providing the content to be framed via a query string parameter.

2. The server verifies that the hostname meets its criteria, so that the page is allowed to be framed by the target resource. This may, for example, happen via a lookup of a whitelist of trusted domain names that are allowed to frame the page. For example, for a Facebook "Like" button, the server can check to see that the supplied hostname matches the hostname(s) expected for that "Like" button.

3. The server returns the hostname in "X-Frame-Options: ALLOW-FROM" if the proper criteria was met in step #2.

4. The browser enforces the "X-Frame-Options: ALLOW-FROM" header.

2.3.2.4. No Caching of the X-Frame-Options Header

Caching the X-Frame-Options header for a resource is not recommended. Caching the X-Frame-Options response could result in problems because:

1. For every http-request of the resource, the browser has to check whether the X-Frame-Options header has been set and then act accordingly, as a resource itself might be created dynamically and the header could change with it, too.

2. Also, as outlined in Section 2.3.2.3, servers may generate X-Frame-Options header responses depending on the request. Example case: Considering that we have only one serialized-origin in the ALLOW-FROM directive, imagine a user has multiple pages open in his browser tabs with web page 1 from domain A and web

page 2 from domain B, and both frame the same page from domain C with the ALLOW-FROM directive. In that case, the page needs to reply to both requests with different X-Frame-Options headers, with the first pointing to origin A and the second pointing to origin B.

However, we found that none of the major browsers listed in Appendix A cache the responses.

3. IANA Considerations

IANA has included the specified HTTP header in the "Permanent Message Header Field Name" registry as outlined in "Registration Procedures for Message Header Fields" [RFC3864].

3.1. Registration Template

Permanent Message Header Field Names Template:

Header field name: X-Frame-Options

Applicable protocol: http [RFC2616]

Status: Informational

Author/change controller: IETF

Specification document(s): RFC 7034

Related information: None

4. Security Considerations

The introduction of the X-Frame-Options HTTP header field improves the protection against clickjacking. However, it is not self-sufficient enough to protect against all kinds of these attack vectors. It must be used in conjunction with other security measures like secure coding (e.g., input validation, output encoding, etc.) and the Content Security Policy version 1.0 [CSP].

It is important to note that current implementations do not check the origins of the framing resources' entire ancestor tree of frames, and this may expose the resource to attack in multiple-nested scenarios.

The browser implementations evaluate based on the origin of the framed page and the top-level browsing context (i.e., the most outer frame):

If a resource from origin A embeds untrusted content from origin B, that untrusted content can embed another resource from origin A with an "X-Frame-Options: SAMEORIGIN" policy, and that check would pass when the user agent only verifies the top-level browsing context. Therefore, web developers should be aware that embedding content from other sites can leave their web pages vulnerable to clickjacking even if the X-Frame-Options header is used.

Furthermore, X-Frame-Options must be sent as an HTTP header field and is explicitly ignored by user agents when declared with a meta http-equiv tag.

4.1. Privacy Considerations

There are two kinds of potential data leakage to consider:

1. Using X-Frame-Options with the parameter ALLOW-FROM allows a page to guess or infer information about who is framing it. A web server may answer requests with the "X-Frame-Options: ALLOW-FROM" header and thus determine which other page is framing it. This is inherent by design, but it may lead to data-leakage or data-protection concerns.

2. The web server using the ALLOW-FROM directive effectively discloses the origin specified in the header. If a web server wishes to reduce this leakage, it is recommended to generate the ALLOW-FROM header for each request based on the design pattern as described in Section 2.3.2.3.

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, December 2011.

5.2. Informative References

[CLICK-DEFENSE-BLOG]
Lawrence, E., "IE8 Security Part VII: Clickjacking Defenses", Microsoft Developer Network Blogs, January 2009, <http://blogs.msdn.com/b/ie/archive/2009/01/27/ie8-security-part-vii-clickjacking-defenses.aspx>.

[CSP] Sterne, B. and A. Barth, "Content Security Policy 1.0", W3C Candidate Recommendation CR-CSP-20121115, November 2012, <http://www.w3.org/TR/2012/CR-CSP-20121115/>.

[CSP-1-1] Barth, A. and M. West, "Content Security Policy 1.1", W3C Working Draft WD-CSP11-20130604, June 2013, <http://www.w3.org/TR/2013/WD-CSP11-20130604/>.

[CSRF] OWASP (Open Web Application Security Project), "Top-10 2013-A8-Cross-Site Request Forgery (CSRF)", June 2013, <https://www.owasp.org/index.php/Top_10_2013-A8-Cross-Site_Request_Forgery_%28CSRF%29>.

[Clickjacking]
OWASP (Open Web Application Security Project), "Clickjacking", April 2013, <http://www.owasp.org/index.php/Clickjacking>.

[FRAME-BUSTING]
Stanford Web Security Research, "Busting frame busting: a study of clickjacking vulnerabilities at popular sites", July 2010, <http://seclab.stanford.edu/websec/framebusting/>.

[HTTPbis-P1]
Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", Work in Progress, September 2013.

[Microsoft-X-Frame-Options]
Lawrence, E., "Combating ClickJacking With X-Frame-Options", Microsoft Developer Network Blogs, March 2010, <http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.aspx>.

[Mozilla-X-Frame-Options]
Mozilla Developer Network, "The X-Frame-Options response header", August 2013, <https://developer.mozilla.org/en-US/docs/The_X-FRAME-OPTIONS_response_header>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC6648] Saint-Andre, P., Crocker, D., and M. Nottingham, "Deprecating the "X-" Prefix and Similar Constructs in Application Protocols", BCP 178, RFC 6648, June 2012.

Appendix A. Browsers That Support X-Frame-Options

- Internet Explorer 8+
- Firefox 3.6.9+
- Opera 10.5+
- Safari 4+
- Chrome 4.1+

Appendix B. Description of a Clickjacking Attack

A more detailed explanation of clickjacking scenarios follows.

B.1. Shop

An Internet marketplace/shop offering a feature with a link/button to "Buy this" gadget wants their affiliates (who could be malicious attackers) to be able to stick the "Buy such and such from XYZ" IFRAMES into their pages. There is a possible clickjacking threat here, which is why the marketplace/online shop needs to then immediately navigate the main browsing context (or a new window) to a confirmation page that is protected by anti-clickjacking protections.

B.2. Online Shop Confirm Purchase Page

The "Confirm Purchase" page of an online shop must be shown to the end-user without the risk of an overlay or misuse by an attacker. For that reason, the confirmation page uses a combination of anti-CSRF (Cross Site Request Forgery [CSRF]) tokens and the X-Frame-Options HTTP header field, mitigating clickjacking attacks.

B.3. Flash Configuration

Macromedia Flash configuration settings are set by a Flash object that can run only from a specific configuration page on Macromedia’s site. The object runs inside the page and thus can be subject to a clickjacking attack. In order to prevent clickjacking attacks against the security settings, the configuration page uses the X-Frame-Options directive.

Appendix C. Acknowledgements

This document was derived from input from specifications published by various browser vendors such as Microsoft (Eric Lawrence and David Ross), Mozilla, Google, Opera, and Apple.

Authors' Addresses

David Ross
Microsoft

EMail: dross@microsoft.com

Tobias Gondrom
Thames Stanley

EMail: tobias.gondrom@gondrom.org