httpbis
Internet-Draft
Intended status: Standards Track
Expires: June 23, 2019

N. Riffat

December 20, 2018

Request Header Originated With
draft-request-header-originated-with-00

Abstract

This document proposes a new Request Header that must be initiated every time a user-agent sends XMLHttpRequest. The aim of this header is to limit the possibilities of XSS to RCE and preventing Javascript from stealing CSRF tokens on other URLs of same domain. This will allow developers to block request if it wasn't supposed to be sent via XMLHttpRequest.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 23, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
  1.1. Limitations 2
  1.2. Goals 2
  1.3. Examples 3
2. Acknowledgements 5
3. IANA Considerations 5
4. Security Considerations 5
5. References 5
  5.1. Normative References 5
  5.2. Informative References 5
Author's Address 5

1. Introduction

Request header is the base on which the server determines what data should be delivered to the user agent. Any request header can be initiated or manipulated by Javascript except the forbidden ones i.e. Origin, Cookie, Access-Control-Request-Headers etc.

The proposed header is similar to "X-Requested-With" header which is initiated every time during an Ajax call by jQuery but it can be controlled and tampered using Javascript. While the proposed header should be a forbidden header just like Origin, Cookie etc so it doesn't get manipulated using Javascript.

1.1. Limitations

This idea to limit the impact of XSS will not be effective if the request is naturally supposed to be sent via XMLHttpRequest i.e. JSON APIs

1.2. Goals

The proposed request header can provide an extra defensive measure to limit the impact of XSS including followings.

1. Will limit the impact of XSS on the vulnerable URL only.

2. Kill or reduce the possibility of XSS that can lead to RCE in some cases i.e. Wordpress.

3. It will not impact any of current user-agent or server side functionality and will be totally be dependent upon developers if they want to implement this new technique.

4. Allow developers to check either the request was originated via XMHttpRequest or standard HTTP.

1.3. Examples

Since Wordpress is the most popular CMS around so it has been impacted a lot in terms of XSS to RCE. So following example is based on recent Wordpress XSS to RCE attacks and how the proposed header could prevent this. Wordpress is a secured CMS by itself but it is incomplete without custom scripts i.e. plugins and themes and it is very common for those custom scripts to be prone to XSS attacks. Supposedly a Wordpress site is vulnerable to any XSS i.e. Reflected, Stored or DOM. The following Javascript code given perfect conditions i.e. having administrator session will inject a new Administrator account on the CMS which can then be used to execute arbitrary server side code.

```
var ajaxRequest = new XMLHttpRequest();

var requestURL = "/wp-admin/user-new.php";

var wp_nonceRegex = /ser" value="([^"]*?)"/g;

ajaxRequest.open("GET", requestURL, false);

ajaxRequest.send();

var nonceMatch = wp_nonceRegex.exec(ajaxRequest.responseText);

var nonce = nonceMatch[1];

var params = "action=createuser&_wpnonce_create-user="+nonce+"&user_l
ogin=config&email=w3bdrill3r@gmail.com&pass1=attackpass&pass2=attackp
ass&role=administrator";

ajaxRequest = new XMLHttpRequest();

ajaxRequest.open("POST", requestURL, true);

ajaxRequest.setRequestHeader("Content-Type", "application/x-www-form-
urlencoded");

ajaxRequest.send(params);
```

Here the Javascript is making 2 calls. 1st call is to get the CSRF Token which is known as "nonce" in Wordpress and the 2nd call is using that token to inject the Administrator account.

Following is how the proposed header could have prevented this.

Proposed request headers

```
GET /wp-admin/user-new.php HTTP/1.1

Host: local.tld

User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:64.0)
Gecko/20100101 Firefox/64.0

Accept: */*

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Referer: http://local.tld/sample-page/

Connection: close

Cookie: xxx;

Pragma: no-cache

Cache-Control: no-cache

Originated-With: XMLHttpRequest
```

The following server side PHP code would have terminated the request. As a result the 1st XMLHttpRequest call would have failed to grab the CSRF token and thus also have prevented the malicious attempt of injecting new user.

```
if(isset($_SERVER['HTTP_ORIGINATED_WITH'])){wp_die();}
```

The above code is initially added in functions.php of current theme file of Wordpress for demonstration purposes. Documents opened using window.open() in Javascript can also be controlled by parent window so this header should also be sent in window.open() and similar requests.

2. Acknowledgements

3. IANA Considerations

4. Security Considerations

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

5.2. Informative References

[wordpress-xss-to-rce]
Sucuri, "https://blog.sucuri.net/2017/12/javascript-injection-creates-rogue-wordpress-admin-user.html", 2017.

Author's Address

Noman Riffat

Email: w3bdrill3r@gmail.com