Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: January 24, 2020

D. Fett
yes.com
J. Bradley
Yubico

July 23, 2019

OAuth 2.0 Integrity Verification for Authorization Requests (IVAR) draft-fett-oauth-ivar-00

Abstract

This document describes a mechanism for the integrity protection of OAuth 2.0 authorization requests.

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 January 24, 2020.

Table of Contents

1. Introduction

A number of attacks on OAuth 2.0 are based on the fact that the contents of the OAuth authorization request lack integrity and authenticity protection. To launch an attack, an attacker might, for example, start an OAuth flow in his browser, use the authorization request URI created by the client, and send it to its victim (with or without manipulations). The victim might then complete the authorization. Since the attacker knows or has manipulated parts of the authorization request URI, certain security mechanisms in OAuth might then not work as expected --- undermining the security of OAuth or protocols based on OAuth, like OpenID Connect.

Among others, the following attacks are facilitated by the lack of integrity and authenticity of the authorization request:

- Attacks on the redirection URI, in which an attacker manipulates the redirection URI and let it point either to a server controlled by the attacker or an endpoint at the client which discloses contents of the authorization response to the attacker.

- The PKCE Chosen Challenge Attack, described in [arXiv.1901.11520], wherein an attacker uses his access to the authorization response (see attacker model A3 in [I-D.ietf-oauth-security-topics]) to gain access to the user's resources.

- A variant of the AS Mix-Up attack in which a malicious AS redirects the user to an honest AS, re-using request parameters. (See [arXiv.1601.01229] for details.)

While TLS protects the integrity of the authorization request, these attacks leverage the fact that an attacker can make a victim's browser visit arbitrary URIs, including those manipulated by the attacker or obtained by the attacker from one of his own interactions with the client.

This document describes IVAR, a mechanism for the verification of the integrity and origin of the contents of the authorization request.

1.1. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification uses the terms "access token", "refresh token", "authorization server", "resource server", "authorization endpoint", "authorization request", "authorization response", "token endpoint", "grant type", "access token request", "access token response", and "client" defined by The OAuth 2.0 Authorization Framework [RFC6749].

2. Concept

On a high level, IVAR works as follows: When the client starts a new OAuth authorization flow, it stores the whole authorization URI (or a hash thereof) in the web storage of the resource owner's browser under the client's origin. When the AS received the authorization request, it opens an iframe from a URI the client registered with the AS beforehand. To this "checking" iframe, the AS sends a cross-document message (postMessage) containing the whole authorization request URI as received by the AS. If the URI matches the one stored by the client earlier, the client responds with a message "ok". If not, the AS aborts the transaction.

3. Client Metadata for IVAR

Clients that support IVAR register the following metadata parameter in the OAuth 2.0 Dynamic Client Registration Protocol [RFC7591]:

"ivar_uri". The content MUST be an https URI from which the checking iframe is loaded by the AS.

4. Protocol

The steps of the IVAR protocol are defined in the following.

4.1. Storing the Authorization Request

Before the client redirects the resource owner's browser to the authorization server, the client stores information about the full redirection URI (including query parameters) in the web storage [WebStorage] of the resource owner's browser. It is at the client's discretion to store either the full URI or a hash value of the URI.

The data MUST be stored such that it is only accessible to the client's origin. Its contents MUST NOT be modifiable or readable by any other origin.

Since multiple OAuth flows may happen in the same browser at the same time, the storage mechanism MUST be able to store multiple entries in parallel.

4.2. IVAR Verification

After receiving the authorization request, the AS opens the client's "ivar_uri" in an iframe in its web site. (The client is identified using the "client_id" parameter.) The IVAR iframe script from the client sends a postMessage with the content "ready" to its parent iframe.

The AS then sends a postMessage containing the full authorization request URI to the iframe. It is important that the AS limits the intended receiver of this message to the origin of the "ivar_uri" to avoid leaking contents of the authorization request URI to an attacker.

The client's script in the iframe then checks if an exact match of the authorization request URI sent by the AS can be found in the list of stored authorization request URIs. If so, it checks that the postMessage containing the URI was received from the origin of the authorization request URI. It then sends the string "ok" in a postMessage to its parent window. It is again important that the intended receiver of this postMessage is set to the authorization request's origin.

The AS ensures that the string "ok" is received in a postMessage originating from the IVAR iframe and the correct origin (from the "ivar_uri"). Only then it continues with the authorization flow.

If any of these steps fail, the AS MUST abort the authorization flow and redirect the browser back to the client with an error value of "ivar_fail".

5. Fallback if JavaScript is unavailable

If the resource owner's browser does not support JavaScript, or JavaScript is disabled, the client cannot store the redirection URI. Likewise, the AS cannot run its part of the IVAR protocol. The AS may skip the IVAR checks if, and only if, it detects that the resource owner's browser does not have JavaScript enabled (which is required for IVAR). The respective check for JavaScript support MUST NOT be open to influence by an attacker.

An attacker cannot usually disable JavaScript in a user's browser for an origin other than his own. The attacker might, however, trick the uesr's browser into treating the IVAR checking script on the AS's origin as part of a cross-site scripting attack and thus disabling the affected JavaScript. To achieve this, the attacker can add the text representation of the respective JavaScript in a (new) URI parameter. Some browser's cross-site scripting auditing engines string match URI parameters inputs with contents in the source code of the web page. If a match is found, the respective JavaScript is disabled.

To avoid disabling of the IVAR checking script by an attacker, AS MUST disable browser-based detection of cross-site scripting using the non-standardized header "X-XSS-Protection: 0" or by sufficiently randomizing the source code of the IVAR checking script.

6. Security Considerations

7. IANA Considerations

8. References

8.1. Normative References

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC7591] Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", RFC 7591, DOI 10.17487/RFC7591, July 2015, <https://www.rfc-editor.org/info/rfc7591>.

8.2. Informative References

[arXiv.1601.01229]
Fett, D., Kuesters, R., and G. Schmitz, "A Comprehensive Formal Security Analysis of OAuth 2.0", January 2016, <http://arxiv.org/abs/1601.01229/>.

[arXiv.1901.11520]
Fett, D., Hosseyni, P., and R. Kuesters, "An Extensive Formal Security Analysis of the OpenID Financial-grade API", January 2019, <http://arxiv.org/abs/1901.11520/>.

[I-D.ietf-oauth-security-topics]
Lodderstedt, T., Bradley, J., Labunets, A., and D. Fett, "OAuth 2.0 Security Best Current Practice", draft-ietf-oauth-security-topics-13 (work in progress), July 2019.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[WebStorage]
Hickson, I., "Web Storage (Second Edition) - W3C Recommendation 19 April 2016", Apr 2016, <https://www.w3.org/TR/webstorage/>.

Appendix A. Document History

[[To be removed from the final specification]]

-00

o first draft

Authors' Addresses

Daniel Fett
yes.com

Email: mail@danielfett.de

John Bradley
Yubico

Email: ve7jtb@ve7jtb.com