Stream:	Internet Engineering Task Force (IETF)	
RFC:	9207	
Category:	Standards Track	
Published:	March 2022	
ISSN:	2070-1721	
Authors:	K. Meyer zu Selhausen	D. Fett
	Hackmanit	*yes.com*

RFC 9207
OAuth 2.0 Authorization Server Issuer Identification

Abstract

This document specifies a new parameter called `iss`. This parameter is used to explicitly include the issuer identifier of the authorization server in the authorization response of an OAuth authorization flow. The `iss` parameter serves as an effective countermeasure to "mix-up attacks".

Status of This Memo

This is an Internet Standards Track document.

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). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9207.

Table of Contents

1. Introduction

The OAuth 2.0 Authorization Framework [RFC6749] allows clients to interact with multiple independent authorization servers under the control of separate entities. Some OAuth grant types utilize the resource owner's user agent to deliver the authorization server's response to the OAuth client. One example of this pattern is the authorization response of the authorization code grant.

The authorization response as specified in Section 4.1.2 of [RFC6749] does not contain any information about the identity of the authorization server that issued the response. Therefore, clients receiving a response from the resource owner's user agent cannot be sure who initially issued the response and the secrets contained therein. The lack of certainty about the origin of the response enables a class of attacks called "mix-up attacks".

Mix-up attacks are a potential threat to all OAuth clients that interact with multiple authorization servers. When at least one of these authorization servers is under an attacker's control, the attacker can launch a mix-up attack to acquire authorization codes or access tokens issued by any one of the other authorization servers. There are multiple ways in which an attacker can gain control over an authorization server supported by the client; for instance, an authorization server could become compromised, or the attacker could register their own authorization server, for example, using dynamic client registration [RFC7591].

OAuth clients that interact with only one authorization server are not vulnerable to mix-up attacks. However, when such clients decide to add support for a second authorization server in the future, they become vulnerable and need to apply countermeasures to mix-up attacks.

Mix-up attacks aim to steal an authorization code or access token by tricking the client into sending the authorization code or access token to the attacker instead of the honest authorization or resource server. This marks a severe threat to the confidentiality and integrity of resources whose access is managed with OAuth. A detailed description and different variants of the mix-up attack class can be found in Section 4.4 of "OAuth 2.0 Security Best Current Practice" [OAUTH-SECURITY-TOPICS] as well as in the original research first highlighting this attack class, "On the security of modern Single Sign-On Protocols: Second-Order Vulnerabilities in OpenID Connect" [arXiv.1508.04324] and "A Comprehensive Formal Security Analysis of OAuth 2.0" [arXiv.1601.01229].

This document defines a new parameter in the authorization response called `iss`. The `iss` parameter allows the authorization server to include its identity in the authorization response explicitly. The client can compare the value of the `iss` parameter to the issuer identifier of the authorization server (e.g., retrieved from its metadata) it believes it is interacting with. The `iss` parameter gives the client certainty about the authorization server's identity and enables it to send credentials such as authorization codes and access tokens only to the intended recipients.

The effectiveness of the `iss` parameter against mix-up attacks was analyzed and formally proven in "A Comprehensive Formal Security Analysis of OAuth 2.0" [arXiv.1601.01229].

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", "authorization code", "authorization code grant", "authorization server", "resource server", "authorization response", "grant type", and "client" defined by the OAuth 2.0 Authorization Framework [RFC6749]. The term "issuer identifier" is defined by OAuth 2.0 Authorization Server Metadata [RFC8414].

2. Response Parameter `iss`

In authorization responses to the client, including error responses, an authorization server supporting this specification **MUST** indicate its identity by including the `iss` parameter in the response.

The `iss` parameter value is the issuer identifier of the authorization server that created the authorization response, as defined in [RFC8414]. Its value **MUST** be a URL that uses the "https" scheme without any query or fragment components.

2.1. Example Authorization Response

The following example shows an authorization response from the authorization server whose issuer identifier is `https://honest.as.example` (extra line breaks and indentation are for display purposes only):

```
HTTP/1.1 302 Found
Location: https://client.example/cb?
  code=x1848ZT64p4IirMPT0R-X3141MFPTuBX-VFL_cvaplMH58
  &state=ZWVlNDBlYzA1NjdkMDNhYjg3ZjUxZjAyNGQzMTM2NzI
  &iss=https%3A%2F%2Fhonest.as.example
```

2.2. Example Error Response

The following example shows an error response from the same authorization server (extra line breaks and indentation are for display purposes only):

```
HTTP/1.1 302 Found
Location: https://client.example/cb?
  error=access_denied
  &state=N2JjNGJhY2JiZjRhYzA3MGJkMzNmMDE5OWJhZmJhZjA
  &iss=https%3A%2F%2Fhonest.as.example
```

2.3. Providing the Issuer Identifier

Authorization servers supporting this specification **MUST** provide their issuer identifier to enable clients to validate the `iss` parameter effectively.

For authorization servers publishing metadata according to [RFC8414], the following rules apply:

- The issuer identifier included in the server's metadata value `issuer` **MUST** be identical to the `iss` parameter's value.
- The server **MUST** indicate its support for the `iss` parameter by setting the metadata parameter `authorization_response_iss_parameter_supported`, defined in Section 3, to `true`.

Authorization servers **MAY** additionally provide the issuer identifier to clients by any other mechanism, which is outside of the scope of this specification.

2.4. Validating the Issuer Identifier

Clients that support this specification **MUST** extract the value of the `iss` parameter from authorization responses they receive if the parameter is present. Clients **MUST** then decode the value from its "application/x-www-form-urlencoded" form according to Appendix B of [RFC6749] and compare the result to the issuer identifier of the authorization server where the authorization request was sent to. This comparison **MUST** use simple string comparison as defined in Section 6.2.1 of [RFC3986]. If the value does not match the expected issuer identifier, clients **MUST** reject the authorization response and **MUST NOT** proceed with the authorization grant. For error responses, clients **MUST NOT** assume that the error originates from the intended authorization server.

More precisely, clients that interact with authorization servers supporting OAuth metadata [RFC8414] **MUST** compare the `iss` parameter value to the `issuer` value in the server's metadata document. If OAuth metadata is not used, clients **MUST** use deployment-specific ways (for example, a static configuration) to decide if the returned `iss` value is the expected value in the current flow (see also Section 4).

If clients interact with both authorization servers supporting this specification and authorization servers not supporting this specification, clients **MUST** retain state about whether each authorization server supports the `iss` parameter. Clients **MUST** reject authorization responses without the `iss` parameter from authorization servers that do support the parameter according to the client's configuration. Clients **SHOULD** discard authorization responses with the `iss` parameter from authorization servers that do not indicate their support for the parameter. However, there might be legitimate authorization servers that provide the `iss` parameter without indicating their support in their metadata. Local policy or configuration can determine whether to accept such responses, and specific guidance is out of scope for this specification.

In general, clients that support this specification **MAY** accept authorization responses that do not contain the `iss` parameter or reject them and exclusively support authorization servers that provide the `iss` parameter in the authorization response. Local policy or configuration can determine when to accept such responses, and specific guidance is out of scope for this specification.

In OpenID Connect [OIDC.Core] flows where an ID Token is returned from the authorization endpoint, the value in the `iss` parameter **MUST** always be identical to the `iss` claim in the ID Token.

Section 4.1.2 of [RFC6749] already mandates that clients that do not support this specification **MUST** ignore the unrecognized `iss` parameter.

> Note: The "JWT Secured Authorization Response Mode for OAuth 2.0 (JARM)" [JARM] defines a mechanism that conveys all authorization response parameters in a JSON Web Token (JWT). This JWT contains an `iss` claim that provides the same protection if it is validated as described in Section 2.4. Therefore, an additional `iss` parameter outside the JWT is not necessary when JARM is used.

3. Authorization Server Metadata

The following parameter for the authorization server metadata [RFC8414] is introduced to signal the authorization server's support for this specification:

`authorization_response_iss_parameter_supported`: Boolean parameter indicating whether the authorization server provides the `iss` parameter in the authorization response as defined in Section 2. If omitted, the default value is false.

4. Security Considerations

Clients **MUST** validate the `iss` parameter precisely as described in Section 2.4 and **MUST NOT** allow multiple authorization servers to use the same issuer identifier. In particular, when authorization server details can be manually configured in the client, the client **MUST** ensure that the accepted `iss` values are unique for each authorization server.

The `iss` parameter enables a client to decide if an authorization server "expects" to be used in an OAuth flow together with a certain token endpoint and potentially other endpoints, like the userinfo endpoint [OIDC.Core]. When OAuth metadata is used, the `iss` parameter identifies the issuer and therefore the respective OAuth metadata document that points to the other endpoints. When OAuth metadata is not used, the client can use, for example, a statically configured expected `iss` value for each configured authorization server.

The issuer identifier contained in the authorization response is not cryptographically protected against tampering. In general, mechanisms such as JWTs (as specified in [JARM]) could be used to protect the integrity of the authorization response. However, in mix-up attacks, the client generally receives the authorization response from an uncompromised authorization server. If an attacker can tamper with this authorization response before it is received by the client, the attacker would also have direct access to the authorization code. The attacker does not need to execute a mix-up attack to steal the authorization code. Therefore, integrity protection for the authorization response is not necessary to defend against mix-up attacks.

There are also alternative countermeasures to mix-up attacks. When an authorization response already includes an authorization server's issuer identifier by other means and this identifier is checked as laid out in Section 2.4, the use and verification of the `iss` parameter is not necessary and **MAY** be omitted. For example, this is the case when OpenID Connect response types that return an ID Token from the authorization endpoint (e.g., `response_type=code id_token`) or

[JARM] are used. However, if a client receives an authorization response that contains multiple issuer identifiers, the client **MUST** reject the response if these issuer identifiers do not match. The details of alternative countermeasures are outside of the scope of this specification.

Mix-up attacks are only relevant to clients that interact with multiple authorization servers. However, clients interacting with only one authorization server might add support for a second authorization server in the future. By supporting multiple authorization servers, they become vulnerable to mix-up attacks and need to apply countermeasures.

5. IANA Considerations

5.1. OAuth Authorization Server Metadata

IANA has registered the following value in the "OAuth Authorization Server Metadata" registry of [IANA.OAuth.Parameters] established by [RFC8414].

Metadata Name: `authorization_response_iss_parameter_supported`
Metadata Description: Boolean value indicating whether the authorization server provides the `iss` parameter in the authorization response.
Change Controller: IETF
Specification Document(s): Section 3 of RFC 9207

5.2. OAuth Parameters Registration

IANA has updated the `iss` entry to appear as follows in the "OAuth Parameters" registry of [IANA.OAuth.Parameters] established by [RFC6749].

Parameter name: `iss`
Parameter usage location: authorization request, authorization response
Change Controller: IETF
Specification Document(s): Section 2 of RFC 9207, [RFC9101], and Section 4.1.1 of [RFC7519].

6. References

6.1. 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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[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>.

[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>.

[RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <https://www.rfc-editor.org/info/rfc8414>.

6.2. Informative References

[arXiv.1508.04324] Mainka, C., Mladenov, V., and J. Schwenk, "On the security of modern Single Sign-On Protocols: Second-Order Vulnerabilities in OpenID Connect", August 2015, <https://arxiv.org/abs/1508.04324>.

[arXiv.1601.01229] Fett, D., Kuesters, R., and G. Schmitz, "A Comprehensive Formal Security Analysis of OAuth 2.0", DOI 10.1145/2976749.2978385, January 2016, <https://arxiv.org/abs/1601.01229>.

[IANA.OAuth.Parameters] IANA, "OAuth Parameters", <https://www.iana.org/assignments/oauth-parameters>.

[JARM] Lodderstedt, T. and B. Campbell, "Financial-grade API: JWT Secured Authorization Response Mode for OAuth 2.0 (JARM)", October 2018, <https://openid.net/specs/openid-financial-api-jarm.html>.

[OAUTH-SECURITY-TOPICS] Lodderstedt, T., Bradley, J., Labunets, A., and D. Fett, "OAuth 2.0 Security Best Current Practice", Work in Progress, Internet-Draft, draft-ietf-oauth-security-topics-19, 16 December 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics-19>.

[OIDC.Core] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 1", November 2014, <https://openid.net/specs/openid-connect-core-1_0.html>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[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>.

[RFC9101] Sakimura, N., Bradley, J., and M. Jones, "The OAuth 2.0 Authorization Framework: JWT-Secured Authorization Request (JAR)", RFC 9101, DOI 10.17487/RFC9101, August 2021, <https://www.rfc-editor.org/info/rfc9101>.

Acknowledgements

We would like to thank Brian Campbell, Roman Danyliw, Vladimir Dzhuvinov, Joseph Heenan, Takahiko Kawasaki, Torsten Lodderstedt, Christian Mainka, Vladislav Mladenov, Warren Parad, Aaron Parecki, and Rifaat Shekh-Yusef for their valuable feedback on this document.

Authors' Addresses

Karsten Meyer zu Selhausen
Hackmanit
Email: karsten.meyerzuselhausen@hackmanit.de

Daniel Fett
yes.com
Email: mail@danielfett.de