OAuth Working Group M. Jones
Internet-Draft Microsoft
Intended status: Standards Track J. Bradley
Expires: July 23, 2016 Ping Identity
January 20, 2016

OAuth 2.0 Mix-Up Mitigation
draft-jones-oauth-mix-up-mitigation-01

Abstract

This specification defines an extension to The OAuth 2.0 Authorization Framework that enables the authorization server to dynamically provide the client using it with additional information about the current protocol interaction that can be validated by the client and that enables the client to dynamically provide the authorization server with additional information about the current protocol interaction that can be validated by the authorization server. This additional information can be used by the client and the authorization server to prevent classes of attacks in which the client might otherwise be tricked into using inconsistent sets of metadata from multiple authorization servers, including potentially using a token endpoint that does not belong to the same authorization server as the authorization endpoint used. Recent research publications refer to these as "IdP Mix-Up" and "Malicious Endpoint" attacks.

Table of Contents

1. Introduction

OAuth 2.0 [RFC6749] clients use multiple authorization server endpoints when using some OAuth response types. For instance, when using the "code" response type, the client uses both the authorization endpoint and the token endpoint. It is important that endpoints belonging to the same authorization server always be used together. Otherwise, information produced by one authorization server could mistakenly be sent by the client to different authorization server, resulting in some of the attacks described in Section 7. Recent research publications refer to these specific attacks as "IdP Mix-Up" [arXiv.1601.01229v2] and "Malicious Endpoint" [arXiv.1508.04324v2] attacks.

The client obviously cannot be confused into using endpoints from multiple authorization servers in an authorization flow if the client is configured to use only a single authorization server. However, the client can potentially be tricked into mixing endpoints if it is configured to use more than one authorization server, whether the configuration is dynamic or static. The client may be confused if it has no way to determine whether the set of endpoints belongs to the same authorization server. Or, a client may be confused simply because it is receiving authorization responses from more than one authorization server at the same redirection endpoint and the client is insufficiently able to determine that the response received is associated with the correct authorization server.

This specification enables the authorization server to dynamically provide the client using it with additional information about the current protocol interaction that can be validated by the client and that enables the client to dynamically provide the authorization server with additional information about the current protocol interaction that can be validated by the authorization server. This enables them to abort interactions in which endpoints from multiple authorization servers would otherwise be used.

The mitigation data provided by the authorization server to the client is an issuer URL, which is used to identify the authorization server, and a client ID, which is used to verify that the response is from the correct authorization server and is intended for this client. The issuer URL is defined in Section 3 of [OAuth.Discovery]. If supported by the authorization server, the issuer URL can also be used to obtain a consistent set of metadata describing the authorization server configuration, as also described in [OAuth.Discovery].

This mitigation data is returned to the client in the authorization response. The syntax for returning the mitigation data from the

authorization server is dependent upon the OAuth response type being used. The syntax used with the existing response types registered in the IANA "OAuth Authorization Endpoint Response Types" registry [IANA.OAuth.Parameters] as of the time of this writing is defined by this specification. Two of these response types are defined by RFC 6749 [RFC6749]; the rest are defined by [OAuth.Responses].

The mitigation data provided by the client to the authorization server is the existing "state" value defined by RFC 6749 [RFC6749], but adding also sending it from the client to the token endpoint. This is used by the authorization server to verify that the authorization code and state both belong to the same protocol interaction.

1.1. Requirements Notation and Conventions

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 RFC 2119 [RFC2119].

1.2. Terminology

This specification uses the terms "Access Token", "Authorization Code", "Authorization Endpoint", "Authorization Grant", "Authorization Server", "Client", "Client Authentication", "Client Identifier", "Client Secret", "Grant Type", "Protected Resource", "Redirection URI", "Refresh Token", "Resource Owner", "Resource Server", "Response Type", and "Token Endpoint" defined by OAuth 2.0 [RFC6749], the terms "Claim Name", "Claim Value", and "JSON Web Token (JWT)" defined by JSON Web Token (JWT) [JWT].

2. The OAuth Issuer

The OAuth issuer serves as a concrete identifier for the authorization server. As defined in [OAuth.Discovery], the OAuth issuer is the URL of the authorization server's configuration information location, which uses the "https" scheme and has no query or fragment components. Also as specified there, when discovery is supported, the authorization server's metadata is retrieved as a JSON document [RFC7159] from a path derived from this URL. This metadata document contains a consistent set of metadata describing the authorization server configuration.

Implementations supporting this specification MAY also support discovery or they MAY simply use the issuer URL as a concrete identifier for the authorization server. This specification does not

rely upon the authorization server publishing or the client retrieving a discovery metadata document.

3. Mitigation Data Returned in Authorization Response

Mitigating the attacks relies on the authorization server returning additional data about the interaction and the client checking that data. The mitigation data returned is the client ID and the issuer URL. The syntax for returning the mitigation data from the authorization server is dependent upon the OAuth response type being used.

3.1. Mitigation Data Returned in Authorization Response Parameters

Some OAuth response types do not already return the issuer URL and client ID in the authorization response. When this is the case, the mitigation data is returned as additional OAuth response parameters.

These new response parameters are defined for this purpose:

client_id
: Client that this response is intended for. It MUST contain the OAuth 2.0 client ID of the client as its value.

iss
: Issuer URL for the authorization server issuing the response. The "iss" value is a case-sensitive URL using the "https" scheme that contains scheme, host, and optionally, port number and path components and no query or fragment components.

As of the time of this writing, these are the existing response types that are registered in the IANA "OAuth Authorization Endpoint Response Types" registry [IANA.OAuth.Parameters] that do not already return the issuer URL and client ID in the authorization response: "code", "code token", "none", and "token". Therefore, the client ID and issuer are returned using the new authorization response parameters when using these response types. To avoid duplication, as discussed in Section 7.2, it is NOT RECOMMENDED to also return them in this manner when the response type already returns these values in the authorization response.

3.1.1. Example Authorization Response using Response Parameters

The following example authorization response is to a request that used the "code" response type. It uses the "iss" and "client_id" response parameters to return the mitigation information to the client.

The example successful authorization response follows (with line breaks within lines for display purposes only):

```
HTTP/1.1 302 Found
Location: https://client.example.org/cb?
  code=Qcb0Orv1zh30vL1MPRsbm-diHiMwcLyZvn1arpZv-Jxf_11jnpEX3Tgfvk
  &state=nrsz6AnHzPSVVBYRVTXV6ZTXQeg_eih7hdpewHNXmZ8
  &iss=https://server.example.com
  &client_id=5d9e8a36-569d-4c40-8d6b-6e279ac1c5f1
```

3.2. Mitigation Data Returned in JWT

As of the time of this writing, these are the existing response types that are registered in the IANA "OAuth Authorization Endpoint Response Types" registry [IANA.OAuth.Parameters] that already return the issuer URL and client ID in the authorization response: "code id_token", "code id_token token", "id_token", and "id_token token". All of these return these values as the "iss" (issuer) claim value and as an "aud" (audience) claim value in a signed ID Token, which is a JSON Web Token [JWT], as specified in "OpenID Connect Core 1.0" [OpenID.Core]. When using these response types, the client MUST use the client ID and issuer values returned in the ID Token for validating the mitigation data.

3.2.1. Example Authorization Response using JWT

The following example authorization response is to a request that used the "id_token token" response type. It uses the "iss" and "aud" claims in the ID Token to return the mitigation information to the client.

The example successful authorization response follows (with line breaks within lines for display purposes only):

```
HTTP/1.1 302 Found
Location: https://client.example.org/cb#
  access_token=jHkWEdUXMU1BwAsC4vtUsZwnNvTIxEl0z9K3vx5KF0Y
  &token_type=Bearer
  &id_token=eyJraWQiOiIxZTlnZGs3IiwiYWxnIjoiUlMyNTYifQ.
  ewogImlzcyI6ICJodHRwczovL3NlcnZlci5leGFtcGxlLmNvbSIsCiAic3ViIjog
  IjI0ODI4OTc2MTAwMSIsCiAiYXVkIjogInM2QmhkUmtxdDMiLAogIm5vbmNlIjog
  Im4tMFM2X1d6QTJNaiIsCiAiZXhwIjogMTMxMTI4MTk3MCwKICJpYXQiOiAxMzEx
  MjgwOTcwLAogImF0X2hhc2giOiAiNzdRbVVQdGpQZnpXdEYyQW5wSzlSUSIKfQ.
  kdqTmftlaXg5WBYBr1wkxhkqCGZPc0k8vTiV5g2jj67jQ7XkrDamYx2bOkZLdZrp
  MPIzkdYB1nZI_G8vQGQuamRhJcEIt21kblGPZ-yhEhdkAiZIZLu38rChalDS2Mh0
  glE_rke5XXRhmqqoEFFdziFdnO3p61-7y51co84OEAZvARSINQaOWIzvioRfs4zw
  IFOaT33Vpxfqr8HDyh31zo9eBW2dSQuCa071z0ENWChWoPliK1JCo_Bk9eDg2uwo
  2ZwhsvHzj6TMQ0lYOTzufSlSmXIKfjlOsb3nftQeR697_hA-nMZyAdL8_NRfaC37
  XnAbW8WB9wCfECp7cuNuOg
  &state=af0ifjsldkj
```

Decoding the ID Token in the response will yield the following claims, which includes the mitigation information in the "iss" and "aud" claims:


```
{
 "iss": "https://server.example.com",
 "sub": "248289761001",
 "aud": "s6BhdRkqt3",
 "nonce": "n-0S6_WzA2Mj",
 "exp": 1311281970,
 "iat": 1311280970,
 "at_hash": "77QmUPtjPfzWtF2AnpK9RQ"
}
```


4. Validating the Authorization Response

Upon receiving the mitigation data in an authorization response, the client MUST validate that the response was intended for it and that the authorization server configuration information that it obtained at client registration time is consistent with the authorization server configuration information contained in the metadata referenced by the issuer URL.

The client MUST validate the authorization server configuration as follows:

1. Compare the issuer URL for the authorization server that the client received when it registered at the authorization server that it made the request to with the issuer value returned in the "iss" response parameter or the "iss" claim in the ID Token, depending upon the response type being used. If they do not exactly match, the client MUST NOT proceed with the authorization.

2. Verify that the response is intended for this client by confirming that the client's client identifier for the authorization server the request was made to matches the value of the "client_id" response parameter or that the client's client identifier is an audience value of the ID Token, depending upon the response type being used. If not, the client MUST NOT proceed with the authorization.

5. Mitigation Data Sent to the Token Endpoint

Mitigating the attacks also relies on the client sending additional data about the interaction to the token endpoint, for response types that use it, and the authorization server checking that data. The mitigation data sent is the same state value that is sent in the authorization request and returned in the authorization response. This specification defines the new "state" token request parameter for passing this additional information.

As of the time of this writing, these are the existing response types that are registered in the IANA "OAuth Authorization Endpoint Response Types" registry [IANA.OAuth.Parameters] that use the token endpoint: "code", "code id_token", "code id_token token", and "code token". The state value is to be sent in the "state" token request parameter when using these response types, and any new response types registered that use the token endpoint.

5.1. Example Token Request

The following example token request is part of a protocol interaction that used the "code" response type. It uses the "state" request parameter to send mitigation information to the authorization server.

The example of token request follows (with line breaks within lines for display purposes only):

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

grant_type=authorization_code
  &code=SplxlOBeZQQYbYS6WxSbIA
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
  &state=ZSGXNBavNc-B3kU3DeJnZoWWOzYxsbvj7jp-S0x_z8U
```

6. Validating the Token Request

When the authorization server receives a token request at the token endpoint that contains a value in the "state" parameter, it MUST validate that the state value received exactly matches the state value previously received in the corresponding authorization request. If the recorded state value and the state value received do not exactly match, the authorization server MUST NOT proceed with the authorization.

7. Security Considerations

7.1. IdP Mix-Up and Malicious Endpoint Attacks

The attacks mitigated by this extension are described in detail in "A Comprehensive Formal Security Analysis of OAuth 2.0" [arXiv.1601.01229v2] and "On the security of modern Single Sign-On Protocols: Second-Order Vulnerabilities in OpenID Connect" [arXiv.1508.04324v2]. To mitigate these attacks, clients configured to use more than one authorization server should use authorization servers that return issuer and client ID information and should validate that a consistent set of authorization server endpoints are being used when using response types that utilize multiple endpoints.

When registering, clients SHOULD NOT allow multiple authorization servers to return the same issuer value, and MUST NOT allow multiple authorization servers to return the same issuer and client ID value pair.

7.2. Duplicate Information Attacks

If a protocol is defined to return the same information in multiple locations, this can create an additional attack surface. Knowing

that the information is supposed to be the same, recipients will often be lazy and use the information from only one of the locations, not validating that all the supposedly duplicate instances are the same. This can enable attackers to create illegal protocol messages that have different values in the multiple locations and those illegal messages will not be detected or rejected by these lazy recipients.

For this reason, if an OAuth profile is being used that returns the mitigation information defined by this specification in one location, it SHOULD NOT also be returned in another. In particular, if a JWT containing the client ID and issuer values is being returned in the authorization response, they SHOULD NOT also be returned as individual authorization response parameters.

7.3. Cut-and-Paste Attacks

OAuth authorization responses are sent as redirects to redirection URIs, with the response parameters typically passed as URI query parameters or fragment values. A "cut-and-paste" attack is performed by the attacker creating what appears to be a legitimate authorization response, but that substitutes some of the response parameter values with values of the attacker's choosing. Sometimes this is done by copying or "cutting" some values out of a legitimate response and replacing or "pasting" some of these values into a different response, the original version of which may have also been legitimate, creating a combination of response values that are not legitimate and that may cause behaviors sought by the attacker. The Code Substitution threat described in Section 4.4.1.13 of [RFC6819] is one example of the use of a cut-and-paste attack.

A concern with returning the mitigation information as new individual authorization response parameters whose values are not cryptographically bound together is that cut-and-paste attacks against their values will not be detected. A security analysis has not been done of the effects of the new attacks that the use of cut-and-paste against these new values will enable.

To prevent replay of the state in another browser instance by an attacker, the state value MUST be tied to the browser instance in a way that cannot be forged by an attacker. Section 4 of [I-D.bradley-oauth-jwt-encoded-state] provides several examples of how a client can accomplish this.

In the replay attack, the attacker can set cookies in the browser. Using an unsigned cookie to bind state to the browser is not sufficient.

8. IANA Considerations

8.1. OAuth Parameters Registration

This specification registers the following parameters in the IANA "OAuth Parameters" registry [IANA.OAuth.Parameters] established by RFC 6749 [RFC6749].

8.1.1. Registry Contents

- o Parameter name: "client_id"
- o Parameter usage location: Authorization Response
- o Change controller: IESG
- o Specification document(s): Section 3.1 of [[this specification]]
- o Related information: None

- o Parameter name: "iss"
- o Parameter usage location: Authorization Response
- o Change controller: IESG
- o Specification document(s): Section 3.1 of [[this specification]]
- o Related information: None

- o Parameter name: "state"
- o Parameter usage location: Token Request
- o Change controller: IESG
- o Specification document(s): Section 5 of [[this specification]]
- o Related information: None

9. References

9.1. Normative References

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

[JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <http://tools.ietf.org/html/rfc7519>.

[OAuth.Discovery]
Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Discovery", draft-jones-oauth-discovery-00 (work in progress), November 2015, <http://tools.ietf.org/html/draft-jones-oauth-discovery-00>.

[OAuth.Responses]

de Medeiros, B., Ed., Scurtescu, M., Tarjan, P., and M. Jones, "OAuth 2.0 Multiple Response Type Encoding Practices", February 2014, <http://openid.net/specs/oauth-v2-multiple-response-types-1_0.html>.

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

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

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

[RFC6819] Lodderstedt, T., Ed., McGloin, M., and P. Hunt, "OAuth 2.0 Threat Model and Security Considerations", RFC 6819, DOI 10.17487/RFC6819, January 2013, <http://www.rfc-editor.org/info/rfc6819>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <http://www.rfc-editor.org/info/rfc7662>.

9.2. Informative References

[I-D.bradley-oauth-jwt-encoded-state]
Bradley, J., Lodderstedt, T., and H. Zandbelt, "Encoding claims in the OAuth 2 state parameter using a JWT", draft-bradley-oauth-jwt-encoded-state-05 (work in progress), December 2015.

[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, <http://www.rfc-editor.org/info/rfc7591>.

[arXiv.1508.04324v2]
Mladenov, V., Mainka, C., and J. Schwenk, "On the security of modern Single Sign-On Protocols: Second-Order Vulnerabilities in OpenID Connect", arXiv 1508.04324v2, January 2016, <http://arxiv.org/abs/1508.04324v2/>.

[arXiv.1601.01229v2]
Fett, D., Kuesters, R., and G. Schmitz, "A Comprehensive Formal Security Analysis of OAuth 2.0", arXiv 1601.01229v2, January 2016, <http://arxiv.org/abs/1601.01229v2/>.

Appendix A. Implementation Notes

The authorization server can compare the two state values either by recording the complete state value between the authorization request and the token request, possibly in the same data structure in which the authorization code issued was recorded, or by recording only a cryptographic hash of the state value, possibly resulting in substantial size savings.

Appendix B. Acknowledgements

Alfred Albrecht, John Bradley, Brian Campbell, Joerg Connotte, William Denniss, Sebastian Ebling, Florian Feldmann, Daniel Fett, Roland Hedberg, Phil Hunt, Ralf Kuesters, Torsten Lodderstedt, Christian Mainka, Vladislav Mladenov, Anthony Nadalin, Justin Richer, Nat Sakimura, Antonio Sanso, Guido Schmitz, Joerg Schwenk, Hannes Tschofenig, and Hans Zandbelt all contributed to the discussions that led to the creation of this specification.

This specification is partially based on the OpenID Connect Core 1.0 specification, which was produced by the OpenID Connect working group of the OpenID Foundation.

Appendix C. Open Issues

- o We need to do a security analysis of the cut-and-paste attacks that may be enabled when mitigation information is returned to the client using individual authorization response parameters.

Appendix D. Document History

[[to be removed by the RFC Editor before publication as an RFC]]

-01

- o Simplified by no longer specifying the signed JWT method for returning the mitigation information.
- o Simplified by no longer depending upon publication of a discovery metadata document.
- o Added the "state" token request parameter.
- o Added examples.
- o Added John Bradley as an editor.

-00

- o Created the initial version.

Authors' Addresses

Michael B. Jones
Microsoft

Email: mbj@microsoft.com
URI: http://self-issued.info/

John Bradley
Ping Identity

Email: ve7jtb@ve7jtb.com
URI: http://www.thread-safe.com/