Future Proofing the OAuth 2.0 Authorization Code Grant Protocol by the application of BCM Principles

Nat Sakimura[1]

Nomura Research Institute, Chiyoda, Tokyo, JAPAN

Abstract. OAuth 2.0 Authorization Framework, while achieved an extremely large adoption, has been exposed to various attacks and a number of additional specifications to patch the problem has been created. It is expected that other attacks would come in the future requiring yet another patch specification. To avoid such future problems, a more systematic approach is needed.
This paper attempts to do it by applying BCM principles on OAuth (RFC6749). It demonstrates that additional parameters in all four messages are needed as well as the integrity protection of both authorization request and response.

Keywords: OAuth, Federation Protocol, Security, authorization request, authorization server, redirect uri, rfc6749 authorization code grant, protocol variant, code grant protocol

1 Introduction

IETF's Authorization federation framework, RFC6749[4] and RFC6750[5], which is commonly referred to as OAuth 2.0 has recently gained enormous adoption. The adoption started from the social network context and propagated to more enterprise context and it is expanding to financial institutional context as it can be seen in such documents like Open Banking Standard [6]. These higher risk use cases warrant more rigorous security analysis of the protocol. Recent research publications such as Mladenov et al. [1] and Fett et al. [2] are such examples and suggests fixes to the protocols based on the vulnerability they have found.

While these fixes do fix the found vulnerability, it does not mean they prevent unknown attacks. To further harden the protocol, a framework based approach would be useful. Basin et al[3] is one such example.

2 BCM Principles

Through the analysis of ISO/IEC 9798, David Basin, Cas Cremers, and Simon Meier [3] sets out a set of principles that cryptographic protocols should adhere. In this paper, they are referred to as BCM Principles. They are as follows:

1. P1. Positional tagging. Cryptographic message components should contain information that uniquely identifies their origin. In particular, the information should identify the protocol, the protocol variant, the message number, and the particular position within the message, from which the component was sent.
2. P2. Inclusion of identities and their roles. Each cryptographic message component should include information about the identities of all the agents involved in the protocol run and their roles, unless there is a compelling reason to do otherwise.

In this paper, RFC 6749 is evaluated against the BCM principles and protocol fix to fulfil them are proposed.

3 Application of BCM Principles on RFC6749

In RFC 6749, there are 4 message types.

1. Authorization Request
2. Authorization Response
3. Token Request
4. Token Response

In the following subsections, each of the above is evaluated against the BCM principles and required fixes are explained.

3.1 Authorization Request

Authorization request in the RFC6749 Code Grant protocol sends the following in the message:

- response_type (REQUIRED): This is set to 'code'.
- client_id (REQUIRED)
- redirect_uri (OPTIONAL)
- scope (OPTIONAL)
- state (RECOMMENDED)

Applying P1 means that the Authorization Request MUST be marked as "Authorization Request of RFC6749 Authorization Code Grant" and the sender MUST be indicated in it. Note that in the BCM framework, the protocol with optional fields are treated as distinct protocols. Thus, from BCM point of view, RFC6749 Authorization Code Grant with redirect_uri and without etc. are regarded as different.

Thus, merely stating response_type=code is not fulfilling the P1. We have to look at the list of parameters being sent to identify the protocol variant. If the authorization request is integrity protected, just looking at the list of parameter received would tell which protocol variant has been used. However, in RFC6749,

Authorization Request/Response is not integrity protected. An adversary can add/remove/tamper the parameters. Thus, this property is not fulfilled. To fulfil the property, **the message has to be integrity protected**. Section 4 talks about methods to achieve it.

Also, P1 requires the message to uniquely identify the origin. Note that client_id is not globally unique but only in the scope of the authorization server/issuer. Thus, the issuer information MUST be included in the message. When looking at the HTTP header information as well, the URI to which the message is being sent is somewhat indicating the authorization server. However, this is not granted. Same authorization endpoint URI might be shared by multiple Authorization Server, e.g., in the case of the multi-tenant server. Thus, **it must include the issuer identifier in the message to uniquely identify the origin**. This can be indirectly done by setting a unique redirect_uri per authorization server and including it in the authorization message, i.e., the redirect_uri is not optional any more.

It should also be noted that while the list of parameters and values of the parameter would make it possible to infer the protocol variant, **it would probably be better to indicate the protocol version and variant explicitly** in the message.

Applying P2 means that the Authorization Request MUST contain the information about all the agent involved and their current roles. As to the agents involved, the issuer (a unique redirect_uri) and client_id will be included in the authorization message per P1. However, roles are not indicated by them. Roles of authorization server in RFC6749 is indicated by the endpoints, i.e., the registration endpoint, discovery endpoint, authorization endpoint, and token endpoint. The roles of the client is less clearly indicated. They are authorization requester, token requester, and resource requester. They can only be inferred in RFC6749 and RFC6750 by the combination of the client_id and the protocol messages. Another agent is the user agent, which is indicated by the state parameter. Only the other agent is the protected resource. It is identified by the URL of the resource. Group identifiers can be used potentially, but in that case, the group membership must be clearly defined.

To comply with P2, these identifiers needs to be included in the protocol messages, which many of them are currently not. If they are included, attacks like code phishing and token phishing at attackers token and resource endpoint respectively can be avoided as it will be detected by the authorization server.

NOTE that the state value MUST be unique and MUST NOT be duplicated across user agents. One might want to consider using user agent fingerprint as a part of the state parameter. One could also consider Token Bind ID per [7]. State parameter at this point is not optional (recommended) but REQUIRED.

3.2 Authorization Response

RFC6749 Authorization Code Grant Authorization response includes the following parameters:

– code (REQUIRED)
– state (REQUIRED)
– other extension parameters (OPTIONAL)

Applying P1 means that the message MUST include the identifier of the authorization server directly or indirectly. In the basic assumption of RFC6749, where there is a separate logical client per authorization server, redirect_uri was supposed to fulfil this role. However, it is known that many implementation does not adhere to this rule, in which case, the above response does not fulfil it.

For the protocol variant indication, the same as in Authorization Request applies, i.e., **the message has to be integrity protected**.

Applying P2 means that the message has to contain the identifiers of all the agents and their roles as identified in 3.1.

3.3 Token Request

Token Request in RFC6749 sends the following parameters:

– grant_type (REQUIRED)
– code (REQUIRED)
– redirect_uri (REQUIRED)
– client_credential (REQUIRED in the header, in the case of the confidential client) OR
– client_id (REQUIRED in the case of the public client).

Note that redirect_uri in the previous section was used as the client assigned identifier for the authorization server.

Adhering to P2 means that all the identifiers identified in 3.1 also needs to be sent.

Note that this message is integrity protected by TLS.

3.4 Token Response

The token response includes following parameters:

– access_token (REQUIRED)
– token_type (REQUIRED)
– expires_in (RECOMMENDED)
– refresh_token (OPTIONAL)
– other parameters (OPTIONAL)

Adhering to P1 means that **redirect_uri MUST be added**.

Adhering to P2 means that the **all the identifiers identified in 3.1 MUST be added**.

4 Integrity protection

In section 3.1 and 3.2, it was noted that P1 is not fulfilled even after adding some parameters as the integrity of these messages are not protected and thus the parameters can be changed and the originator can be spoofed. Thus, some way of integrity protection has to be considered, i.e., it has to have "signature" by the originator of the message.

One way of achieving it is to sign the request and response. In OpenID Connect, there is a facility called request object and ID Token which effectively integrity protects the authorization request and response. Note that ID Token does not protect 'state'. To fully protect the response, a state hash, s_hash, needs to be introduced as in [8]. If additional parameters were to be sent, hash of those parameters in the similar fashion has to be included in the ID Token as well.

In OAuth, OAuth JAR is proposed for authorization request protection.

5 Conclusions

In this paper, RFC6749 Authorization Code Grant Protocols were analysed against the BCM principles. It was revealed that all the message lacked one or more parameters. In addition, the authorization request and response needed a mechanism to integrity protect to fulfil the BCM principles.

It should also be noted that the above amended protocol's security property is not proved. It is desirable to prove it as in Section 4 of [3] and the author looks forward to seeing one.

References

1. Mladenov, V., Mainka, C., Schwenk, J.: On the security of modern Single Sign-On Protocols: Second-Order Vulnerabilities in OpenID Connect. arXiv 1508.04324v2, (2016) ¡http://arxiv.org/abs/1508.04324v2/¿
2. Fett, D., Kuesters, R., Schmitz, G.: A Comprehensive Formal Security Analysis of OAuth 2.0", arXiv 1601.01229v2, (2016) ¡http://arxiv.org/abs/1601.01229v2/¿.
3. Basin, D., Cremers, C., Meier, S.: Provably Repairing the ISO/IEC 9798 Standard for Entity Authentication. Journal of Computer Security - Security and Trust Principles archive Volume 21 Issue 6, 817-846 (2013) ¡https://www.cs.ox.ac.uk/people/cas.cremers/downloads/papers/BCM2012-iso9798.pdf¿
4. Heardt, D.: RFC6749: The OAuth 2.0 Authorization Framework. IETF (2012)
5. Jones, M., Heardt, D.: RFC6750: The OAuth 2.0 Authorization Framework: Bearer Token Usage. IETF (2012)
6. Open Banking Working Group: THe Open Bank Standard Open Data Institute (2016).
7. Belfanz, D.: Token Binding over HTTP. IETF 2017
8. Sakimura, N., et al.: Financial API - Read Write Security Profile OIDF 2017
9. Sakimura, N., Bradley, J. OAuth 2.0 Framework: JWS Authorization Request IETF 2017