OAuth Working Group N. Sakimura, Ed.
Internet-Draft Nomura Research Institute
Intended status: Standards Track J. Bradley
Expires: January 30, 2014 Ping Identity
N. Agarwal
Google
July 29, 2013

OAuth Transient Client Secret Extension for Public Clients
draft-sakimura-oauth-tcse-00

Abstract

The OAuth 2.0 public client utilizing code flow is susceptible to the code interception attack. This specification describe a mechanism that acts as a control against this threat.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 http://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 30, 2014.

Copyright Notice

Copyright (c) 2013 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 (http://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
2. Terminology 3
  2.1. transient client secret 3
  2.2. transient client secret hash 3
3. Protocol 3
  3.1. Client checks the server support 3
  3.2. Client creates a transient client secret 3
  3.3. Client sends the salted left hash with the authorization request 3
  3.4. Server returns the code 4
  3.5. Client sends the code and the secret to the token endpoint 4
  3.6. Server verifies tcs before returning the tokens 4
4. IANA Considerations 4
  4.1. OAuth Parameters Registry 4
5. Security Considerations 5
6. Acknowledgements 5
7. References 6
  7.1. Normative References 6
  7.2. Informative References 6
Authors' Addresses 6

1. Introduction

Public clients in OAuth 2.0 [RFC6749] is suseptible to the "code" interception attack. The "code" interception attack is an attack that a malicious client intercepts the "code" returned from the authorization endpoint and uses it to obtain the access token. This is possible on a public client as there is no client secret associated for it to be sent to the token endpoint. This is especially true on some smartphone platform in which the "code" is returned to a redirect URI with a custom scheme as there can be multiple apps that can register the same scheme.

To mitigate this attack, this extension utilizes dynamically created client secret called transient client secret whose left hash is sent as a new authorization request parameter. The "code" obtained is then sent to the token endpoint with the transient client secret and

the server compairs it with the previously received left hash of it so that it can perfom the proof of posession by the client.

2. Terminology

In addition to the terms defined in OAuth 2.0 [RFC6749], this specification defines the following terms.

2.1. transient client secret

a cryptographically random string with big enough entropy that is used to correlate the authorization request to the token request

2.2. transient client secret hash

base64url encoding of the left most 128bit of SHA256 hash of transient client secret

3. Protocol

3.1. Client checks the server support

Before starting the authorization process, the client MUST make sure that the server supports this specification. It may be obtained out-of-band or some other mechanisms such as the discovery document in OpenID Connect Discovery [OpenID.Discovery]. The exact mechanism on how the client obtains this information is out of scope of this specification.

The client that wishes to use this specification MUST stop proceeding if the server does not support this extension.

3.2. Client creates a transient client secret

The client then creates a transient client secret, "tcs", in the following manner.

tcs = high entropy cryptographic random string

NOTE: transient client secret MUST have high enough entropy to make it inpractical to guess the value.

3.3. Client sends the salted left hash with the authorization request

Then, the client calculates a transient client secret hash, "tcsh", the salted left hash of the "tcs" as follows where L is a function that calcualtes the base64url encoded left-most 128 bits of the binary input, and H is a SHA256 function.

tcsh = L (H (tcs))

The client sends the transient client secret hash with the following parameter with the OAuth 2.0 [RFC6749] Authorization Request:

tcsh REQUIRED. transient client secret hash

3.4. Server returns the code

When the server issues "code", it MUST associate the "tcsh" value with the "code" so that it can be used later.

Typically, the "tcsh" value is stored in encrypted form in the "code", but it could as well be just stored in the server in association with the code. The server MUST NOT include the "tcsh" value in the form that any entity but itself can extract it.

3.5. Client sends the code and the secret to the token endpoint

Unpon receipt of the "code", the client sends the request to the token endpoint. In addition to the parameters defined in OAuth 2.0 [RFC6749], it sends the following parameter:

tcs REQUIRED. transient client secret

3.6. Server verifies tcs before returning the tokens

Upon receipt of the request at the token endpoint, the server verifies it by calculating the transient client secret hash from "tcs" value and "tcsh" and comparing it with the previously associated "tcsh". If they are equal, then the successful response SHOULD be returned. If the values are not equal, an error response indicating "invalid_grant" as described in section 5.2 of OAuth 2.0 [RFC6749] SHOULD be returned.

4. IANA Considerations

This specification makes a registration request as follows:

4.1. OAuth Parameters Registry

This specification registers the following parameters in the IANA OAuth Parameters registry defined in OAuth 2.0 [RFC6749].

- Parameter name: tcs

- Parameter usage location: Access Token Request

- Change controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net

- Specification document(s): this document

- Related information: None

- Parameter name: tcsh

- Parameter usage location: Authorization Request

- Change controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net

- Specification document(s): this document

- Related information: None

5. Security Considerations

The security model relies on the fact that the transient client secret is not being disclosed in the front channel. It is vitally important to adhear to this principle. As such, the transient client secret has to be created in such a manner that it is cryptographically random and has high entropy that it is not practical for the attacker to guess, and if it is to be returned inside "code", it has to be encrypted in such a manner that only the server can decrypt and extract it.

6. Acknowledgements

The initial draft of this specification was created by the OpenID AB/Connect Working Group of the OpenID Foundation, by most notably of the following people:

- Naveen Agarwal, Google

- Dirk Belfanz, Google

- John Bradley, Ping Identity

- Brian Campbell, Ping Identity

- Ryo Ito, mixi

- Michael B. Jones, Microsoft

- Torsten Lodderstadt, Deutsche Telekom

- Breno de Madeiros, Google
- Anthony Nadalin, Microsoft
- Nat Sakimura, Nomura Research Institute

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, March 2012.

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

7.2. Informative References

[OpenID.Discovery]
Sakimura, N., Bradley, J., Jones, M., and E. Jay, "OpenID Connect Discovery 1.0", May 2013.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

Authors' Addresses

Nat Sakimura (editor)
Nomura Research Institute

Email: sakimura@gmail.com

John Bradley
Ping Identity

Email: jbradley@pingidentity.com

Naveen Agarwal
Google

Email: naa@google.com