OAuth Security

Torsten Lodderstedt
Mark McGloin
Phil Hunt

http://datatracker.ietf.org/doc/draft-lodderstedt-oauth-security/

OAuth WG session at IETF-80, April 1st 2011,

Objectives of the document

- Comprehensive Threat Model and Security Considerations
- Serve as foundation of OAuth core spec's security considerations section
- Facilitate a broad discussion on OAuth security in the WG (pre-requisite for consensus)
- Help others to assess OAuth Security (e.g. OMA or WAC)
- Serve as foundation for protocol/ implementation-specific security analysis
- Guide implementors wrt security

Overview on the I-D

- Gives comprehensive threats model for OAuth (threats & associated impact)
  - Scope: core, bearer, mac (partially)
  - Assumption: Static binding between client and service provider
- Describes respective countermeasures
- Considers design options and different client types
- Cross-references between associated threats and countermeasures

Example Threat Description

TOC

4.4.1.1. Threat: Malicious client obtains authorization

A malicious client could counterfeit a valid client and obtain an access authorization that way. The malicious client could even utilize screen scraping techniques in order to simulate the user consent in the authorization flow.

Countermeasures:

- The authorization server should authentication the client, if possible (see **Section 5.2.3.4**). Note: the authentication takes place after the end-user has authorized the access.
- The authorization server should validate the client's redirect_uri against the pre-registered redirect_uri, if one exists (see **Section 5.2.3.5**). Note: The validation of the redirect_uri is the only technical mean to recognize a malicious client id in advance of the authorization process. Further note this does not work for native applications because in contrast to web applications this uri is not bound to a single communication endpoint. The valid client's redirect_uri (typically with custom scheme) can be used by a malicious on any device.
- After authenticating the end-user, the authorization server should ask him/her for consent. In this context, the user shall be explained the purpose, scope, and duration of the authorization. Moreover, the authorization server must view to the end-user the meta data it associates with the particular client. It is up to the user to validate this data and approve the authorization request. (see **Section 5.2.4.3**).
- The authorization server must not perform automatic re-authorizations for clients it is unable to reliably authenticate or validate (see **Section 5.2.4.1**).
- If the authorization server automatically authenticates the end-user, it may nevertheless require some user input in order to prevent screen scraping. Examples are CAPTCHAs or user-specific secret like PIN codes.
- The authorization server may also limit the scope of tokens it issues to clients it cannot reliably authenticate (see **Section 5.1.5.1**).

Example Countermeasure Description

TOC

5.2.3.5. Validation of pre-registered redirect_uri

An authorization server may require clients to register their redirect_uri or a pattern (TBD: make definition more precise) thereof. The way this registration is performed is out of scope of this document. Every actual redirect_uri sent with the respective client_id to the end-user authorization endpoint must comply with that pattern. Otherwise the authorization server must assume the inbound GET request has been sent by an attacker and refuse it.

Note: the authorization server MUST NOT redirect the user agent back to the redirect_uri of the authorization request.

- Session fixation: allows to detect session fixation attempts already after first redirect to end-user authorization endpoint
- For clients of LOA 2/5/7, this measure also helps to detect malicious apps early in the end-user authorization process. This reduces the need for a interactive validation by the user.

The underlying assumption of this measure is that an attacker must use another redirect_uri in order to get access to the authorization code. Deployments might consider the possibility of an attacker using spoofing attacks to a victims device to circumvent this security measure. This is a countermeasure against the following threats:

- session fixation
- malicious apps (for deployment-specific clients with secret)

Note: Pre-registering clients might not scale in some deployments (manual process) or require dynamic client registration (not specified yet). With the lack of dynamic client registration, it only works for clients bound to certain deployments at development/configuration time. As soon as dynamic resource server discovery gets involved, that's no longer feasable.

Plan/Ideas

- Core spec's security considerations section
  - Draft (-01) exists: http://datatracker.ietf.org/doc/draft-lodderstedt-oauth-securityconsiderations/
  - Focuses on the guidlines implementors of the protocol must consider (WHAT and not WHY)
  - References security document
- Security Document
  - Bring it forward as WG item and RFC, complementary document to the core spec's security considerations section
  - Incorporate WG feedback (into -02)
  - Could be an umbrella document which discusses security wrt the OAuth protocol family (core, bearer, mac, discovery?, ...)