OAUTH WG
Internet-Draft
Intended status: Informational
Expires: April 25, 2013

G. Fletcher
AOL
T. Lodderstedt
Deutsche Telekom AG
Z. Zeltsan
Alcatel-Lucent

October 22, 2012

OAuth Use Cases
draft-ietf-oauth-use-cases-03

Abstract

This document lists the OAuth use cases. The provided list is based on the Internet Drafts of the OAUTH working group and discussions on the group's mailing list.

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 April 25, 2013.

Copyright Notice

Table of Contents

1. Introduction

This document describes the use cases that have been discussed on the oauth WG mailing list and introduced by the Internet Drafts submitted to the group. The selected use cases illustrate the use of the OAuth flows by the clients of the various profiles and types. The document also includes those cases that are not directly supported by the OAuth 2.0 [I-D.ietf-oauth-v2], but were considered during its development. The document provides a list of the requirements derived from the use cases. The use cases supported by OAuth 2.0 are indicated.

The document's objective is to help with understanding of the OAuth 2.0 protocol design.

The following section provides the abbreviated descriptions of the use cases.

2. OAuth use cases

This section describes the use cases that have been discussed by the oauth WG.

2.1. Web server

Description:

Alice accesses an application running on a web server at www.printphotos.example and instructs it to print her photographs that are stored on a server www.storephotos.example. The application at www.printphotos.example receives Alice's authorization for accessing her photographs without learning her authentication credentials with www.storephotos.example.

Pre-conditions:

- Alice has registered with www.storephotos.example to enable authentication

- The application at www.printphotos.example has established authentication credentials with the application at www.storephotos.example

Post-conditions:

A successful procedure results in the application www.printphotos.example receiving an authorization code from

www.storephotos.example. The code is bound to the application at www.printphotos.example and to the callback URL supplied by the application. The application at www.printphotos.example uses the authorization code for obtaining an access token from www.storephotos.example. The application at www.storephotos.example issues an access token after authenticating the application at www.printphotos.example and validating the authorization code that it has submitted. The application at www.printphotos.example uses the access token for getting access to Alice's photographs at www.storephotos.example.

Note: When an access token expires, the service at www.printphotos.example needs to repeat the OAuth procedure for getting Alice's authorization to access her photographs at www.storephotos.example. Alternatively, if Alice wants to grant the application a long lasting access to her resources at www.storephotos.example, the authorization server associated with www.storephotos.example may issue the long-living tokens. Those tokens can be exchanged for short-living access tokens required to access www.storephotos.example.

Requirements:

- The server www.printphotos.example, which hosts an OAuth client, must be capable of issuing the HTTP redirect requests to Alice's user agent - a browser

- Application at www.storephotos.example must be able to authenticate Alice. The authentication method is not in the OAuth

scope

- Application at www.storephotos.example must obtain Alice's authorization of the access to her photos by www.printphotos.example

- Application at www.storephotos.example may identify to Alice the scope of access that www.printphotos.example has requested while asking for Alice's authorization

- Application at www.storephotos.example must be able to authenticate the application at www.printphotos.example and validate the authorization code before issuing an access token. The OAuth 2.0 protocol [I-D.ietf-oauth-v2] specifies one authentication method that MAY be used for such authentication - Client Password Authentication.

- Application at www.printphotos.example must provide a callback URL to the application at www.storephotos.example (note: the URL can

be pre-registered with www.storephotos.example)

- Application at www.storephotos.example is required to maintain a record that associates the authorization code with the application at www.printphotos.example and the callback URL provided by the application

- Access tokens are bearer's tokens (they are not associated with a specific application, such as www.printphotos.example) and should have a short lifespan

- Application at www.storephotos.example must invalidate the authorization code after its first use

- Alice's manual involvement in the OAuth authorization procedure (e.g., entering an URL or a password) should not be required. (Alice's authentication to www.storephotos.example is not in the OAuth scope. Her registration with www.storephotos.example is required as a pre-condition)

Note: OAuth 2.0 supports this use case

2.2. User-agent

Description:

Alice has on her computer a gaming application. She keeps her scores in a database of a social site at www.fun.example. In order to upload Alice's scores, the application gets access to the database with her authorization.

Pre-conditions:

- Alice uses a gaming application implemented in a scripting language (e.g., JavaScript) that runs in her browser and uses OAuth for accessing a social site at www.fun.example
- There is no a web site supporting this application and capable of handling the OAuth flow, so the gaming application needs to update the database itself
- The application is registered with the social site at www.fun.example and has an identifier
- Alice has registered with www.fun.example for identification and authentication

- An auxiliary web server at www.help.example is reachable by Alice's browser and capable of providing a script that extracts an access token from an URL's fragment

Post-conditions:

A successful procedure results in Alice's browser receiving an access token. The access token is received from www.fun.example as a fragment of a redirection URL of an auxiliary web server www.help.example. Alice's browser follows the redirection, but retains the fragment. From the auxiliary web server at www.help.example Alice's browser downloads a script that extracts access token from the fragment and makes it available to the gaming application. The application uses the access token to gain access to Alice's data at www.fun.example.

Requirements:

- Registration of the application running in the Alice's browser with the application running on www.fun.example is required for identification

- Alice's authentication with www.fun.example is required

- Application running at www.fun.example must be able to describe to Alice the request made by the gaming application running on her computer and obtain Alice's authorization for or denial of the requested access

- After obtaining Alice's authorization the application running at www.fun.example must respond with an access token and redirect Alice's browser to a web server (e.g., www.help.example) that is capable of retrieving an access token from an URL

Note: OAuth 2.0 supports this use case

2.3. Native Application

Description:

Alice wants to upload (or download) her photographs to (or from) storephotos.example using her smartphone. She downloads and installs a photo app on her smartphone. In order to enable the app to access her photographs, Alice needs to authorize the app to access the web site on her behalf. The authorization shall be valid for a prolonged duration (e.g. several months), so that Alice does not need to authenticate and authorize access on every execution of the app. It shall be possible to withdraw the app's authorization both on the

smartphone as well as on the site storephotos.example.

Pre-conditions:

- Alice has installed a (native) photo app application on her smartphone

- The installed application is registered with the social site at

storephotos.example and has an identifier

- Alice holds an account with storephotos.example

- Authentication and authorization shall be performed in an interactive, browser-based process. The smartphone's browser is used for authenticating Alice and for enabling her to authorize the request by the Mobile App

Post-conditions:

A successful procedure results in Alice's app receiving the access and refresh tokens. The app obtains the tokens by utilizing the Authorization Code flow. The application uses the access token to gain access to Alice's data at storephotos.example. The refresh tokens are persistently stored on the device for use in subsequent app executions. If a refresh token exists on app startup, the app directly uses the refresh token to obtain a new access token.

Requirements:

- Alice's authentication with storephotos.example is required

- Registration of the application running on Alice's smartphone is required for identification and registration and may be carried out on a per installation base

- The application at storephotos.example provides a capability to view and delete the apps' authorizations. This implies that the different installations of the same app on the different devices can be distinguished (e.g., by a device name or a telephone number)

- The app must provide Alice an option to logout. The logout must result in revocation of the refresh tokens on the authorization server

Note: OAuth 2.0 supports this use case

2.4. In-App-Payment (based on Native Application)

Description:

Alice has installed on her computer a gaming application (e.g., running as native code or as a widget). At some point she wants to play the next level of the game and needs to purchase an access to the advanced version of the game from her service provider at www.sp.example. With Alice's authorization the application accesses her account at www.sp.example and enables her to make the payment.

Pre-conditions:

- Alice has registered and has an account with her service provider at www.sp.example
- The application is registered with the service provider at www.sp.example. This enables the server provider to provide Alice with all necessary information about the gaming application (including the information about the purchasing price)
- Alice has a Web user-agent (e.g., a browser or a widget runtime) installed on her computer

Post-conditions:

A successful procedure results in the gaming application invoking the user browser and directing it to the authorization server of the service provider. The HTTP message includes information about the gaming application's request to access Alice's account. The authorization server presents to Alice the authentication and authorization interfaces. The authorization interface shows Alice the information about the application's request including the requested charge to her account. After Alice successfully authenticates and authorizes the request, the authorization server enables Alice to save the transaction details including the authorization code issued for the gaming application. Then the authorization server redirects Alice's browser to a custom scheme URI (registered with the operating system). This redirection request contains a one-time authorization code and invokes a special application that is able to extract the authorization code and present it to the gaming application. The gaming application presents the authorization code to the authorization server and exchanges it for a one-time access token. The gaming application then uses the access token to get access to Alice's account and post the charges at www.sp.example.

Requirements:

Note: The focus is on the requirements that are specific to this use case. The requirements that are common to the native applications are listed in the preceding use case.

- An authorization server associated with the server at www.sp.example must be able to provide Alice with information about the access request that the gaming application has made (including the amount that is to be charged to her account with the service provider and the purpose for the charge) over a secure transport

- An authorization server associated with the server at www.sp.example must be able to obtain Alice's authorization decision on the request over a secure transport

- An authorization server associated with the server at www.sp.example must be able to generate on demand a one-time authorization code and a one-time access token according to the scope authorized by Alice

- An authorization server associated with the server at www.sp.example must be able to call back to the gaming application with the authorization result over a secure transport

- An authorization server associated with the server at www.sp.example must enable the gaming application to exchange an authorization code for an access token over a secure transport

- An authorization server associated with the server at www.sp.example must verify the authorization code and invalidate it after its first use

- An authorization server associated with the server at www.sp.example must enable Alice to save the details of the requested transaction, including the authorization code

- An authorization server associated with the server at www.sp.example must keep a record linking the requested transaction with the authorization code and the respective access token

- An authorization server associated with the server at www.sp.example must enable the resource server www.sp.example to obtain the transaction information that is linked to the issued access token

- Resource server at www.sp.example must verify access token and invalidate it after its first use

- A resource server at www.sp.example must enable the gaming application to post charges to Alice's account according to the access token presented over a secure transport

- The gaming application must provide a custom scheme URI to the authorization server associated with www.sp.example (note: it can be preregistered with the authorization server)

- Alice's manual involvement in the OAuth authorization procedure (e.g., entering an URL or a password) should not be required. (Alice's authentication to www.sp.example is not in the OAuth scope)

Note: OAuth 2.0 does not directly support this use case

2.5. Device with an input method

Description:

Alice has a device, such as a gaming console, that does not support an easy data-entry method. She also has access to a computer with a browser. The application running on the Alice's device gets authorized access to a protected resource (e.g., photographs) stored on a server at www.storephotos.example

Pre-conditions:

- Alice uses a gaming console, which does not have an easy data-entry method, for accessing her photographs at www.storephotos.example

- Alice is able to connect to www.storephotos.example using a computer that runs a browser

- Auhtorization server associated with www.storephotos.example is able to generate an authorization code that is suitable for reading and writing by a human (e.g., an alphanumeric string that is not too long)

- The gaming device supports input of the characters that can be found in an authorization code

- Alice has registered with the authorization server associated with www.storephotos.example for identification and authentication

Post-conditions:

- Alice, interacting with an authorization server associated with www.storephotos.example, authorizes access to her photographs by her gaming console. She uses her browser-equipped computer for OAuth authorization

- The authorization server associated with www.storephotos.example responds to Alice's authorization by displaying an authorization code in her browser's window

- Alice enters the displayed code into an input field on the gaming console

- The gaming console exchanges with the authorization server the authorization code for an access token

- Alice's gaming console uses the access token to access the photographs on www.storephotos.example

Requirements:

- Alice's authentication with the authorization server is required

- Alice is required to perform authorization of her gaming console by interacting with the authorization server associated with www.storephotos.example. To that end she has to direct her browser to the authorization server

- After authorizing the access and getting an authorization code displayed in her browser, Alice has to enter the displayed code into an input field on the gaming console

- The gaming console should be able to exchange the authorization

code for an access token through interaction with the authorization server associated with www.storephotos.example

- The URL of the authorization server and the authorization code must be suitable for manual entry
- The authorization code must be composed of the characters that are appropriate for input to the gaming console
- Because the authorization code is relatively short and its character set is limited, the code's lifetime should be configured appropriately

Note: OAuth 2.0 supports this use case

2.6. Client password (shared secret) credentials

Description:

The company GoodPay prepares the employee payrolls for the company GoodWork. In order to do that the application at www.GoodPay.example gets authenticated access to the employees' attendance data stored at www.GoodWork.example.

Pre-conditions:

- The application at www.GoodPay.example has established through a registration an identifier and a shared secret with the application running at www.GoodWork.example
- The scope of the access by the application at www.GoodPay.example to the data stored at www.GoodWork.example has been defined

Post-conditions:

A successful procedure results in the application at www.GoodPay.example receiving an access token after authenticating to the application running at www.GoodWork.example.

Requirements:

- Authentication of the application at www.GoodPay.example to the application at www.GoodWork.example is required

- The authentication method must be based on the identifier and shared secret, which the application running at www.GoodPay.example submits to the application at www.GoodWork.example in the initial HTTP request

- Because in this use case GoodPay gets access to GoodWork's sensitive data, GoodWork shall have a pre-established trust with GoodPay on the security policy and the authorization method's implementation

Note: OAuth 2.0 supports this use case

2.7. Assertion

Description:

Company GoodPay prepares the employee payrolls for the company GoodWork. In order to do that the application at www.GoodPay.example gets authenticated access to the employees' attendance data stored at

www.GoodWork.example.
This use case describes an alternative solution to the one described by the use case Client password credentials.

Pre-conditions:

- The application at www.GoodPay.example has obtained an authentication assertion from a party that is trusted by the application at www.GoodWork.example

- The scope of the access by the application at www.GoodPay.example to the data stored at www.GoodWork.example has been defined

- The application at www.GoodPay.example has established trust relationship with the asserting party and is capable of validating its assertions

Post-conditions:

A successful procedure results in the application at www.GoodPay.example receiving an access token after authenticating to the application running at www.GoodWork.example by presenting an assertion (e.g., SAML assertion).

Requirements:

- Authentication of the application at www.GoodPay.example to the application at www.GoodWork.example is required
- The application running at www.GoodWork.example must be capable of validating assertion presented by the application running at www.GoodPay.example
- Because in this use case GoodPay gets access to GoodWork's sensitive data, GoodWork shall establish trust with GoodPay on the security policy and the authorization method's implementation

Note: OAuth 2.0 supports this use case

2.8. Access token exchange

Description:

Alice uses an application running on www.printphotos.example for printing her photographs that are stored on a server at www.storephotos.example. The application running on www.storephotos.example, while serving the request of the application at www.printphotos.example, discovers that some of the requested

photographs have been moved to www.storephotos1.example. The application at www.storephotos.example retrieves the missing photographs from www.storephotos1.example and provides access to all requested photographs to the application at www.printphotos.example. The application at www.printphotos.example carries out Alice's request.

Pre-conditions:

- The application running on www.printphotos.example is capable of interacting with Alice's browser

- Alice has registered with and can be authenticated by authorization server
- The applications at www.storephotos.example has registered with authorization server
- The applications at www.storephotos1.example has registered with authorization server
- The application at www.printphotos.example has registered with authorization server

Post-conditions:

A successful procedure results in the application at www.printphotos.example receiving an access token that allows access to Alice's photographs. This access token is used for the following purposes:

- By the application running at www.printphotos.example to get access to the photographs at www.storephotos.example
- By the application running at www.storephotos.example to obtain from authorization server another access token that allows it to retrieve the additional photographs stored at www.storephotos1.example

As the result, there are two access token issued for two different applications. The tokens may have different properties (e.g., scope, permissions, and expiration dates).

Requirements:

- The applications at www.printphotos.example and www.storephotos.example require different access tokens

- The application at www.printphotos.example is required to provide its callback URL to the application at www.storephotos.example
- Authentication of the application at www.printphotos.example to

the authorization server is required

- Alice's authentication by the authorization server is required

- The authorization server must be able to describe to Alice the request of the application at www.printphotos.example and obtain her authorization (or rejection)

- If Alice has authorized the request, the authorization server must be able to issue an access token that enables the application at www.printphotos.example to get access to Alice's photographs at www.storephotos.example

- The authorization server must be able, based on the access token presented by the application at www.printphotos.example, to generate another access token that allows the application at www.storephotos.example to get access to the photographs at www.storephotos1.example. In this context the authorization server must validate the authorization of the application at www.storephotos.example to obtain the token.

- The application at www.storephotos.example must be able to validate an access token presented by the application running at www.printphotos.example

- The application at www.storephotos1.example must be able to validate the access token presented by the application running at www.storephotos.example

Note: This use case is indirectly supported by Assertion frmamework for OAuth 2.0 [I-D.ietf-oauth-assertions] and its extensions SAML 2.0 Bearer Assertion Profiles for OAuth 2.0 [I-D.ietf-oauth-saml2-bearer] and JSON Web Token (JWT) Bearer Token Profiles for OAuth 2.0 [I-D.ietf-oauth-jwt-bearer]

2.9. Multiple access tokens

Description:

Alice uses a communicator application running on a web server at www.communicator.example to access her email service at www.email.example and her voice over IP service at www.voip.example. Email addresses and telephone numbers are obtained from Alice's address book at www.contacts.example. Those web sites all rely on

the same authorization server, so the application at www.communicator.example can receive a single authorization from Alice for getting access to these three services on her behalf at once.
The authorization server needs to issue different access tokens for the involved services due to security and privacy policy. One typical reason is the use of the symmetric secrets for signing self-contained access tokens. In this use case, using a particular token for more than a single service introduces a security risk.

Note: This use case is especially useful for native applications since a web browser needs to be launched only once.

Pre-conditions:

- The same authorization server serves Alice and all involved servers
- Alice has registered with the authorization server for authentication and for authorization of the requests of the communicator application running at www.communicator.example
- The email application at www.email.example has registered with the authorization server for authentication
- The VoIP application at www.voip.example has registered with the authorization server for authentication
- The address book at www.contacts.example has registered with the authorization server for authentication

Post-conditions:

A successful procedure results in the application at www.communicator.example receiving three different access tokens: one for accessing the email service at www.email.example, one for accessing the contacts at www.contacts.example, and one for accessing the VoIP service at www.voip.example.

Requirements:

- The application running at www.communicator.example must be authenticated by the authorization server
- Alice must be authenticated by the authorization server
- The application running at www.communicator.example must be able

to get a single Alice's authorization for access to the multiple

services (e.g., email and VoIP)

- The application running at www.communicator.example must be able to recognize that all three applications rely on the same authorization server
- A callback URL of the application running at www.communicator.example must be known to the authorization server
- The authorization server must be able to issue the separate service-specific tokens (with different, scope, permissions, and expiration dates) for access to the requested services (such as email and VoIP)

Note: OAuth 2.0 does not support this use case

2.10. Gateway for browser-based VoIP applets

Description:

Alice accesses a social site on a web server at www.social.example. Her browser loads a VoIP applet that enables her to make a VoIP call using her SIP server at www.sipservice.example. The application at www.social.example gets Alice's authorization to use her account with www.sipservice.example without learning her authentication credentials with www.sipservice.example.

Pre-conditions:

- Alice has registered with www.sipservice.example for authentication
- The application at www.social.example has established authentication credentials with the application at www.sipservice.example

Post-conditions:

A successful procedure results in the application at www.social.example receiving access token from www.sipservice.example

with Alice's authorization.

Requirements:

- The server at www.social.example must be able to redirect Alice's browser to www.sipservice.example

- The application running at www.sipservice.example must be capable of authenticating Alice and obtaining her authorization of a request from www.social.example

- The server at www.sipservice.example must be able to redirect Alice's browser back to www.social.example

- The application at www.social.example must be able to translate the messages of the Alice's VoIP applet into SIP and RTP messages

- The application at www.social.example must be able to add the access token to the SIP requests that it sends to www.sipservice.example

- Application at www.sipservice.example must be able to authenticate the application at www.social.example and validate the access token

- Alice's manual involvement in the OAuth authorization procedure (e.g., entering an URL or a password) should not be required. (Alice's authentication to www.sipservice.example is not in the OAuth scope)

Note: OAuth 2.0 does not support this use case

2.11. Signed Messages

Description:

Alice manages all her personal health records in her personal health data store at a server at www.myhealth.example, which manages authorization of access to Alice's participating health systems. Alice's Primary Care Physician (PCP), which has a Web site at

www.pcp.example, recommends her to see a sleep specialist (www.sleepwell.example). Alice arrives at the sleep specialist's office and authorizes it to access her basic health data at her PCP's web site. The application at www.pcp.example verifies that Alice has authorized www.sleepwell.example to access her health data as well as enforces that www.sleepwell.example is the only application that can retrieve that data with that specific authorization.

Pre-conditions:

- Alice has a personal health data store that allows for discovery of her participating health systems (e.g. psychiatrist, sleep specialist, PCP, orthodontist, ophthalmologist, etc)

- The application at www.myhealth.example manages authorization of access to Alice's participating health systems
- The application at www.myhealth.example can issue authorization tokens understood by Alice's participating health systems
- The application at www.pcp.example stores Alice's basic health and prescription records
- The application at www.sleepwell.com stores results of Alice's sleep tests

Post-conditions:

- A successful procedure results in just the information that Alice authorized being transferred from the Primary Care Physician (www.pcp.example) to the sleep specialist (www.sleepwell.example)
- The transfer of health data only occurs if the application at www.pcp.example can verify that www.sleepwell.example is the party requesting access and that the authorization token presented by www.sleepwell.example is issued by the application at www.myhealth.example with a restricted audience of www.sleepwell.example

Requirements:

- The application at www.sleepwell.example interacting with www.myhealth.example must be able to discover the location of the PCP system (e.g., XRD discovery)

- The application at www.sleepwell.example must be capable of requesting Alice's authorization of access to the application at www.pcp.example for the purpose of retrieving basic health data (e.g. date-of-birth, weight, height, etc). The mechanism Alice uses to authorize this access is out of scope for this use case

- The application at www.myhealth.example must be capable of issuing a token bound to www.sleepwell.example for access to the application at www.pcp.example. Note that a signed token (JWT) can be used to prove who issued the token

- The application at www.sleepwell.example must be capable of issuing a request (which includes the token issued by www.myhealth.example) to the application at www.pcp.example

- The application at www.sleepwell.example must sign the request before sending it to www.pcp.example

- The application at www.pcp.example must be capable of receiving the request and verifying the signature

- The application at www.pcp.example must be capable of parsing the message and finding the authorization token

- The application at www.pcp.example must be capable of verifying the signature of the authorization token

- The application at www.pcp.example must be capable of parsing the authorization token and verifying that this token was issued to the application at www.sleepwell.com

- The application at www.pcp.example must be capable of retrieving the requested data and returning it to the application at www.sleepwell.example

Note: OAuth 2.0 does not support this use case

2.12. Signature with asymmetric secret

Description:

Alice accesses an application running on a web server at www.printphotos.example and instructs it to print her photographs that are stored on a server www.storephotos.example. The application at www.printphotos.example, which does not have a shared secret with www.storephotos.example, receives Alice's authorization for accessing her photographs without learning her authentication credentials with www.storephotos.example.

Pre-conditions:

- Alice has registered with www.storephotos.example to enable authentication
- The application at www.printphotos.example has a private and a matching public keys

Post-conditions:

A successful procedure results in the application at www.printphotos.example receiving an access token from www.storephotos.example for accessing the Alice's photographs.

Requirements:

- The application at www.printphotos.example must be capable of issuing the HTTP redirect requests to Alice's user agent - a browser
- The application at www.storephotos.example must be able to authenticate Alice
- The application running at www.storephotos.example must be able to obtain the public key of the application at www.printphotos.example
- The application running at www.printphotos.example is required to

sign using its private key the requests to the application at www.storephotos.example

- The application at www.storephotos.example must obtain Alice's authorization of the access to her photos by www.printphotos.example

- The application at www.storephotos.example is required to identify to Alice the scope of access that www.printphotos.example has requested while asking for Alice's authorization

- The application at www.storephotos.example must be able to authenticate the application at www.printphotos.example by validating a signature of its request using the public key of www.printphotos.example

- The application at www.printphotos.example must provide a callback URL to the application at www.storephotos.example (note: the URL can be pre-registered with www.storephotos.example)

- The application at www.storephotos.example must be capable of issuing the HTTP redirect requests to Alice's browser

- Alice's manual involvement in the OAuth authorization procedure (e.g., entering an URL or a password) should not be required. (Alice's authentication to www.storephotos.example is not in the OAuth scope)

Note: OAuth 2.0 does not support this use case

3. Authors of the use cases

The major contributors of the use cases are as follows:

W. Beck, Deutsche Telekom AG

G. Brail, Sonoa Systems
B. de hOra
B. Eaton, Google
S. Farrell, NewBay Software
G. Fletcher, AOL

Y. Goland, Microsoft
B. Goldman, Facebook
E. Hammer-Lahav, Yahoo!
D. Hardt
R. Krikorian, Twitter
T. Lodderstedt, Deutsche Telekom
E. Maler, PayPal
D. Recordon, Facebook
L. Shepard, Facebook
A. Tom, Yahoo!
B. Vrancken, Alcatel-Lucent
Z. Zeltsan, Alcatel-Lucent

4. Security considerations

The OAuth 2.0 specification [I-D.ietf-oauth-v2] provides the implementers with security guidelines for all OAuth 2.0 client profiles. In addition, a comprehensive OAuth security model and background for the protocol design are provided by [I-D.ietf-oauth-v2-threatmodel].

5. IANA considerations

This Internet Draft includes no request to IANA.

6. Acknowledgements

The authors thank Igor Faynberg and Hui-Lan Lu for their invaluable help with preparing this document. Special thanks are to the draft reviewers Thomas Hardjono and Melinda Shore, whose suggestions have helped to improve the draft.

7. References

7.1. Normative References

[I-D.ietf-oauth-v2]
Hardt, D., "The OAuth 2.0 Authorization Framework", draft-ietf-oauth-v2-31 (work in progress), August 2012.

7.2. Informative References

[I-D.ietf-oauth-v2-threatmodel]
Lodderstedt, T., McGloin, M., and P. Hunt, "OAuth 2.0 Threat Model and Security Considerations", draft-ietf-oauth-v2-threatmodel-07 (work in progress), August 2012.

[I-D.ietf-oauth-assertions]
Campbell, B., Mortimore, C., Jones, M., and Y. Goland, "Assertion Framework for OAuth 2.0", draft-ietf-oauth-assertions-05 (work in progress), September 2012.

[I-D.ietf-oauth-saml2-bearer]
Campbell, B. and C. Mortimore, "SAML 2.0 Bearer Assertion Profiles for OAuth 2.0", draft-ietf-oauth-saml2-bearer-14 (work in progress), September 2012.

[I-D.ietf-oauth-jwt-bearer]
Jones, M., Campbell, B., and C. Mortimore, "JSON Web Token (JWT) Bearer Token Profiles for OAuth 2.0", draft-ietf-oauth-jwt-bearer-02 (work in progress), September 2012.

Authors' Addresses

George Fletcher
AOL

Email: gffletch@aol.com

Torsten Lodderstedt
Deutsche Telekom AG

Email: torsten@lodderstedt.net

Zachary Zeltsan
Alcatel-Lucent
600 Mountain Avenue
Murray Hill, New Jersey
USA

Phone: +1 908 582 2359
Email: Zachary.Zeltsan@alcatel-lucent.com