OAuth

OAuth is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords.[1] This mechanism is used by companies such as Amazon,[2] Google, Facebook, Microsoft and Twitter to permit the users to share information about their accounts with third party applications or websites.

The OAuth logo, designed by American blogger Chris Messina

Generally, OAuth provides to clients a "secure delegated access" to server resources on behalf of a resource owner. It specifies a process for resource owners to authorize third-party access to their server resources without sharing their credentials. Designed specifically to work with Hypertext Transfer Protocol (HTTP), OAuth essentially allows access tokens to be issued to third-party clients by an authorization server, with the approval of the resource owner. The third party then uses the access token to access the protected resources hosted by the resource server.[3]

OAuth is a service that is complementary to and distinct from OpenID. OAuth is unrelated to OATH, which is a *reference architecture* for *authentication*, not a *standard* for *authorization*. However, OAuth is directly related to OpenID Connect (OIDC), since OIDC is an authentication layer built on top of OAuth 2.0. OAuth is also unrelated to XACML, which is an authorization policy standard. OAuth can be used in conjunction with XACML, where OAuth is used for ownership consent and access delegation whereas XACML is used to define the authorization policies (e.g., managers can view documents in their region).

Contents

History

OAuth began in November 2006 when Blaine Cook was developing the Twitter OpenID implementation. Meanwhile, Ma.gnolia needed a solution to allow its members with OpenIDs to authorize Dashboard Widgets to access their service. Cook, Chris Messina and Larry Halff from Magnolia met with David Recordon to discuss using OpenID with the Twitter and Magnolia APIs to delegate authentication. They concluded that there were no open standards for API access delegation.[4]

The OAuth discussion group was created in April 2007, for the small group of implementers to write the draft proposal for an open protocol. DeWitt Clinton from Google learned of the OAuth project, and expressed his interest in supporting the effort. In July 2007, the team drafted an initial specification. Eran Hammer joined and coordinated the many OAuth contributions creating a more formal specification. On 4 December 2007, the OAuth Core 1.0 final draft was released.[5]

At the 73rd Internet Engineering Task Force (IETF) meeting in Minneapolis in November 2008, an OAuth BoF was held to discuss bringing the protocol into the IETF for further standardization work. The event was well attended and there was wide support for formally chartering an OAuth working group within the IETF.

The OAuth 1.0 protocol was published as RFC 5849, an informational Request for Comments, in April 2010.

Since 31 August 2010, all third party Twitter applications have been required to use OAuth.[6]

The OAuth 2.0 framework was published as RFC 6749, and the Bearer Token Usage as RFC 6750, both standards track Requests for Comments, in October 2012.

OAuth 2.0

OAuth 2.0 is not backwards compatible with OAuth 1.0. OAuth 2.0 provides specific authorization flows for web applications, desktop applications, mobile phones, and smart devices. The specification and associated RFCs are developed by the IETF OAuth WG;[7] the main framework was published in October 2012.

Facebook's Graph API only supports OAuth 2.0.[8] Google supports OAuth 2.0 as the recommended authorization mechanism for all of its APIs.[9] Microsoft also supports OAuth 2.0 for various APIs and its Azure Active Directory service,[10] which is used to secure many Microsoft and third party APIs.

The OAuth 2.0 Framework[11] and Bearer Token Usage[12] were published in October 2012.

Security

OAuth 1.0

On 23 April 2009, a session fixation security flaw in the 1.0 protocol was announced. It affects the OAuth authorization flow (also known as "3-legged OAuth") in OAuth Core 1.0 Section 6.[13] Version 1.0a of the OAuth Core protocol was issued to address this issue.[14]

OAuth 2.0

In January 2013, the Internet Engineering Task Force published a threat model for OAuth 2.0.[15] Among the threats outlined is one called "Open Redirector"; in the spring of 2014, a variant of this was described under the name "Covert Redirect" by Wang Jing.[16][17][18][19]

OAuth 2.0 has been analyzed using formal web protocol analysis. This analysis revealed that in setups with multiple authorization servers, one of which is behaving maliciously, clients can become confused about the authorization server to use and may forward secrets to the malicious authorization server (AS Mix-Up Attack).[20] This prompted the creation of a new best current practice internet draft that sets out to define a new security standard for OAuth 2.0.[21] Assuming a fix against the AS Mix-Up Attack in place, the security of OAuth 2.0 has been proven under strong attacker models using formal analysis.[20]

One implementation of OAuth 2.0 with numerous security flaws has been exposed.[22]

In April–May 2017, about one million users of Gmail (less than 0.1% of users as of May 2017) were targeted by an OAuth-based phishing attack, receiving an email purporting to be from a colleague, employer or friend wanting to share a document on Google Docs.[23] Those who clicked on the link within the email were directed to sign in and allow a potentially malicious third-party program called "Google Apps" access their "email account, contacts and online documents".[23] Within "approximately one hour",[23] the phishing attack was stopped by Google, who advised those who had given "Google Apps" access to their email to revoke such access and change their passwords.

Uses

OAuth can be used as an authorizing mechanism to consume secured RSS/ATOM feeds. Consumption of RSS/ATOM feeds that require authentication has always been an issue. For example, an RSS feed from a secured Google Site could not have been consumed using Google Reader. Instead, three-legged OAuth would have been used to authorize that RSS client to access the feed from the Google Site. It can also be used as a means to login without creating an account on any site and all the benefits of the host of the OAuth system.

OAuth and other standards

OpenID vs. pseudo-authentication using OAuth

OAuth is an *authorization* protocol, rather than an *authentication* protocol. Using OAuth on its own as an authentication method may be referred to as pseudo-authentication. The following diagrams highlight the differences between using OpenID (specifically designed as an authentication protocol) and OAuth for authentication.

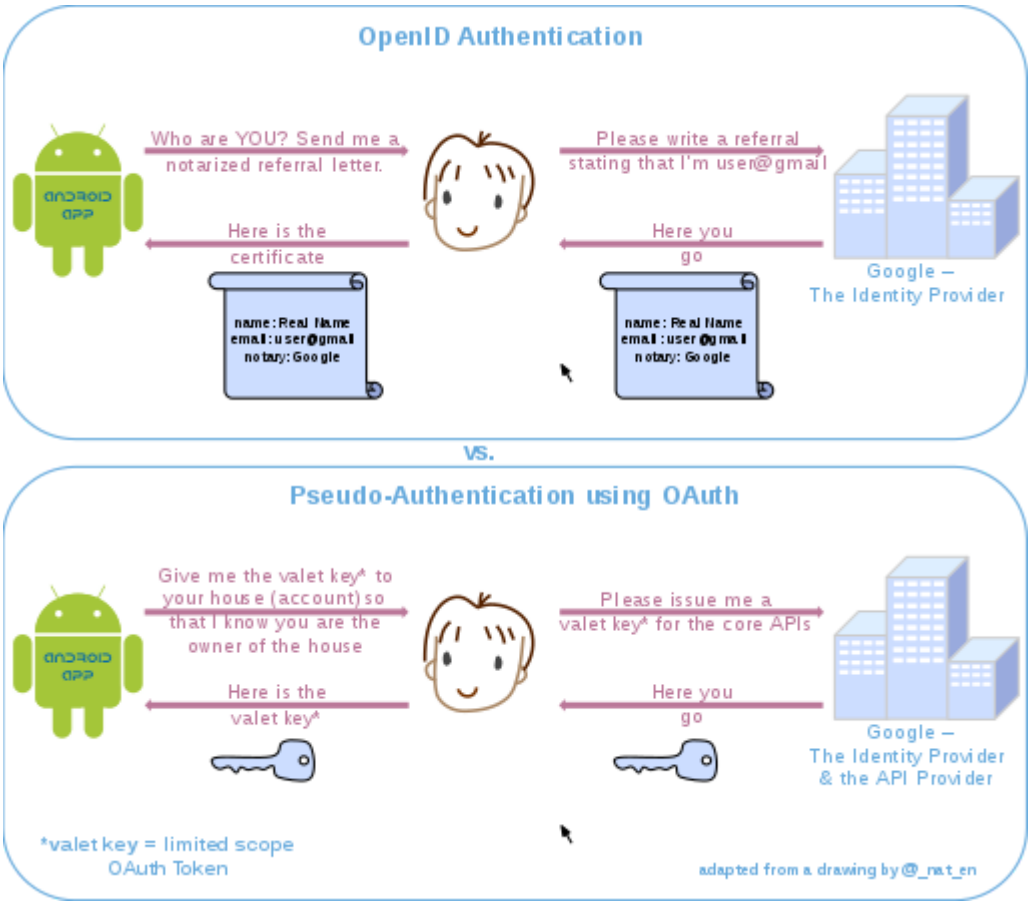
The communication flow in both processes is similar:

1. (Not pictured) The user requests a resource or site login from the application.
2. The site sees that the user is not authenticated. It formulates a request for the identity provider, encodes it, and sends it to the user as part of a redirect URL.
3. The user's browser requests the redirect URL for the identity provider, including the application's request
4. If necessary, the identity provider authenticates the user (perhaps by asking them for their username and password)
5. Once the identity provider is satisfied that the user is sufficiently authenticated, it processes the application's request, formulates a response, and sends that back to the user along with a redirect URL back to the application.
6. The user's browser requests the redirect URL that goes back to the application, including the identity provider's response
7. The application decodes the identity provider's response, and carries on accordingly.

8. (OAuth only) The response includes an access token which the application can use to gain direct access to the identity provider's services on the user's behalf.

The crucial difference is that in the OpenID *authentication* use case, the response from the identity provider is an assertion of identity; while in the OAuth *authorization* use case, the identity provider is also an API provider, and the response from the identity provider is an access token that may grant the application ongoing access to some of the identity provider's APIs, on the user's behalf. The access token acts as a kind of "valet key" that the application can include with its requests to the identity provider, which prove that it has permission from the user to access those APIs.

Because the identity provider typically (but not always) authenticates the user as part of the process of granting an OAuth access token, it's tempting to view a successful OAuth access token request as an authentication method itself. However, because OAuth was not designed with this use case in mind, making this assumption can lead to major security flaws.[24]



OAuth and XACML

XACML is a policy-based, attribute-based access control authorization framework. It provides:

- An access control architecture.
- A policy language with which to express a wide range of access control policies including policies that can use consents handled / defined via OAuth.
- A request / response scheme to send and receive authorization requests.

XACML and OAuth can be combined together to deliver a more comprehensive approach to authorization. OAuth does not provide a policy language with which to define access control policies. XACML can be used for its policy language.

Where OAuth focuses on delegated access (I, the user, grant Twitter access to my Facebook wall), and identity-centric authorization, XACML takes an attribute-based approach which can consider attributes of the user, the action, the resource, and the context (who, what, where, when, how). With XACML it is possible to define policies such as

- Managers can view documents in their department
- Managers can edit documents they own in draft mode

XACML provides more fine-grained access control than OAuth does. OAuth is limited in granularity to the coarse functionality (the scopes) exposed by the target service. As a result, it often makes sense to combine OAuth and XACML together where OAuth will provide the delegated access use case and consent management and XACML will provide the authorization policies that work on the applications, processes, and data.

Lastly, XACML can work transparently across multiple stacks (APIs, web SSO, ESBs, home-grown apps, databases...). OAuth focuses exclusively on HTTP-based apps.

Controversy

Eran Hammer resigned from his role of lead author for the OAuth 2.0 project, withdrew from the IETF working group, and removed his name from the specification in July 2012. Hammer cited a conflict between web and enterprise cultures as his reason for leaving, noting that IETF is a community that is "all about enterprise use cases" and "not capable of simple." "What is now offered is a blueprint for an authorization protocol", he noted, "that is the enterprise way", providing a "whole new frontier to sell consulting services and integration solutions."[25]

In comparing OAuth 2.0 with OAuth 1.0, Hammer points out that it has become "more complex, less interoperable, less useful, more incomplete, and most importantly, less secure." He explains how architectural changes for 2.0 unbound tokens from clients, removed all signatures and cryptography at a protocol level and added expiring tokens (because tokens couldn't be revoked) while complicating the processing of authorization. Numerous items were left unspecified or unlimited in the specification because "as has been the nature of this working group, no issue is too small to get stuck on or leave open for each implementation to decide."[25]

David Recordon later also removed his name from the specifications for unspecified reasons. Dick Hardt took over the editor role, and the framework was published in October 2012.[11]

See also

- List of OAuth providers
- Data portability
- IndieAuth
- Mozilla Persona
- OpenID
- SAML
- User-Managed Access

References

1. Whitson Gordon. "Understanding OAuth: What Happens When You Log Into a Site with

Google, Twitter, or Facebook" (http://lifehacker.com/5918086/understanding-oauth-what-happens-when-you-log-into-a-site-with-google-twitter-or-facebook). Retrieved 15 May 2016.
2. Amazon & OAuth 2.0 (https://login.amazon.com/)
3. "RFC 6749 - The OAuth 2.0 Authorization Framework" (http://tools.ietf.org/html/rfc6749). Retrieved 15 May 2016.
4. "Introduction" (https://oauth.net/about/introduction/). *oauth.net*. Retrieved 21 November 2018.
5. "OAuth Core 1.0" (http://oauth.net/core/1.0/). 4 December 2007. Retrieved 16 October 2014.
6. Chris Crum (31 August 2010). "Twitter Apps Go OAuth Today" (http://www.webpronews.com/twitter-apps-go-oauth-today-2010-08/). *WebProNews.com*. Retrieved 31 July 2017.
7. "Web Authorization Protocol (oauth)" (http://datatracker.ietf.org/wg/oauth/). IETF. Retrieved 8 May 2013.
8. "Authentication - Facebook Developers" (https://developers.facebook.com/docs/authentication/). *Facebook for Developers*.
9. "Using OAuth 2.0 to Access Google APIs | Google Identity Platform" (https://developers.google.com/identity/protocols/OAuth2). *Google Developers*. Retrieved 4 January 2020.
10. "v2.0 Protocols - OAuth 2.0 Authorization Code Flow" (https://docs.microsoft.com/en-us/azure/active-directory/active-directory-v2-protocols-oauth-code). *Microsoft Docs*.
11. "RFC6749 - The OAuth 2.0 Authorization Framework" (https://tools.ietf.org/html/rfc6749). Dick Hardt. October 2012. Retrieved 10 October 2012.
12. "RFC6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage" (https://tools.ietf.org/html/rfc6750). Michael Jones, Dick Hardt. October 2012. Retrieved 10 October 2012.
13. "OAuth Security Advisory: 2009.1" (http://oauth.net/advisories/2009-1). 23 April 2009. Retrieved 23 April 2009.
14. OAuth Core 1.0a (http://oauth.net/core/1.0a)
15. [rfc:6819 OAuth 2.0 Threat Model and Security Considerations]. Internet Engineering Task Force. Accessed January 2015.
16. "OAuth Security Advisory: 2014.1 "Covert Redirect" " (http://oauth.net/advisories/2014-1-covert-redirect/). OAuth. 4 May 2014. Retrieved 10 November 2014.
17. "Serious security flaw in OAuth, OpenID discovered" (https://www.cnet.com/news/serious-security-flaw-in-oauth-and-openid-discovered/). CNET. 2 May 2014. Retrieved 10 November 2014.
18. "Math student detects OAuth, OpenID security vulnerability" (http://phys.org/news/2014-05-math-student-oauth-openid-vulnerability.html). Phys.org. 3 May 2014. Retrieved 11 November 2014.
19. "Covert Redirect" (http://tetraph.com/covert_redirect/). Tetraph. 1 May 2014. Retrieved 10 November 2014.
20. Fett, Daniel; Küsters, Ralf; Schmitz, Guido (2016). "A Comprehensive Formal Security Analysis of OAuth 2.0". *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*. New York, New York, USA: ACM Press: 1204–1215. arXiv:1601.01229 (https://arxiv.org/abs/1601.01229). Bibcode:2016arXiv160101229F (https://ui.adsabs.harvard.edu/abs/2016arXiv160101229F). doi:10.1145/2976749.2978385 (https://doi.org/10.1145%2F2976749.2978385). ISBN 9781450341394.
21. Bradley, John; Labunets, Andrey; Lodderstedt, Torsten; Fett, Daniel. "OAuth 2.0 Security Best Current Practice" (https://tools.ietf.org/html/draft-ietf-oauth-security-topics-13.html). *tools.ietf.org*. Retrieved 29 July 2019.
22. "Hacking Facebook with OAuth 2.0 and Chrome" (http://homakov.blogspot.co.uk/2013/02/hacking-facebook-with-oauth2-and-chrome.html). 12 February 2013. Retrieved 6 March 2013.
23. "Google Docs phishing email 'cost Minnesota $90,000' " (https://web.archive.org/web/20170508194157/http://www.bbc.co.uk/news/technology-39845545). BBC News. 8 May 2017. Archived from the original (https://www.bbc.co.uk/news/technology-39845545) on 8 May 2017.

24. "End User Authentication with OAuth 2.0" (http://oauth.net/articles/authentication/). *OAuth Community Site*. Retrieved 8 March 2016.
25. "OAuth 2.0 and the Road to Hell" (https://hueniverse.com/oauth-2-0-and-the-road-to-hell-8eec45921529). Eran Hammer. 28 July 2012. Retrieved 17 January 2018.

External links

- The OAuth 1.0 Protocol (RFC 5849)
- The OAuth 2.0 Authorization Framework (RFC 6749)
- The OAuth 2.0 Authorization Framework: Bearer Token Usage (RFC 6750)
- OAuth.net (https://oauth.net)
- OAuth Working Group's Mailing List (https://www.ietf.org/mailman/listinfo/oauth)
- OAuth Beginner's Guide and Resource Center (https://hueniverse.com/oauth) by Hueniverse
- OAuth end points cheat sheet (https://www.cheatography.com/kayalshri/cheat-sheets/oauth-end-points/)
- OAuth with Spring framework integration (https://www.devglan.com/spring-security/spring-boot-security-oauth2-example)
