A Comprehensive Formal Security Analysis of OAuth 2.0

Daniel Fett
University of Trier, Germany
fett@uni-trier.de

Ralf Küsters
University of Trier, Germany
kuesters@uni-trier.de

Guido Schmitz
University of Trier, Germany
schmitzg@uni-trier.de

The OAuth 2.0 protocol is one of the most widely deployed authorization/single sign-on (SSO) protocols and also serves as the foundation for the new SSO standard OpenID Connect. Despite the popularity of OAuth, so far analysis efforts were mostly targeted at finding bugs in specific implementations and were based on formal models which abstract from many web features or did not provide a formal treatment at all.

In our work, we carry out the first extensive formal analysis of the OAuth 2.0 standard in an expressive web model. Our analysis aims at establishing strong authorization, authentication, and session integrity guarantees, for which we provide formal definitions. In our formal analysis, all four OAuth grant types (authorization code grant, implicit grant, resource owner password credentials grant, and the client credentials grant) are covered. They may even run simultaneously in the same and different relying parties and identity providers, where malicious relying parties, identity providers, and browsers are considered as well. Our modeling and analysis of the OAuth 2.0 standard assumes that security recommendations and best practices are followed, in order to avoid obvious and known attacks.

While proving security in our model, we discovered four attacks which break the security of OAuth. The vulnerabilities can be exploited in practice and are present also in the new OpenID Connect standard.

We propose fixes for the identified vulnerabilities, and then, for the first time, actually prove the security of OAuth in an expressive web model. In particular, we show that the fixed version of OAuth (with security recommendations and best practices in place) provides the authorization, authentication, and session integrity properties we specify.

1 Introduction

The OAuth 2.0 authorization framework [8] defines a web-based protocol that allows a user to grant web sites access to her resources (data or services) at other web sites (*authorization*). The former web sites are called relying parties (RP) and the latter are called identity providers (IdP).[1] In practice, OAuth 2.0 is often used for *authentication* as well. That is, a user can log in at an RP using her identity managed by an IdP (single sign-on, SSO).

Authorization and SSO solutions have found widespread adoption in the web over the last years, with OAuth 2.0 being one of the most popular frameworks. OAuth 2.0, in the following often simply called *OAuth*,[2] is used by identity providers such as Facebook, Google, Microsoft, Yahoo, GitHub, and Dropbox. This enables billions of users to log in at millions of RPs or share their data with these [13], making OAuth one of the most used single sign-on systems on the web.

OAuth is also the foundation for the new single sign-on protocol OpenID Connect, which is already in use and actively supported by PayPal ("Log In with PayPal"), Google, and Microsoft, among others. Considering the broad industry support for OpenID Connect, a widespread adoption of OpenID Connect in the next years seems likely. OpenID Connect builds upon OAuth and provides clearly defined interfaces for user authentication and additional (optional) features, such as dynamic identity provider discovery and relying party registration, signing and encryption of messages, and user logout.

OAuth defines a complex protocol. The interactions between the user and her browser, the RP, and the IdP can be performed in four different flows, or *grant types*: authorization code grant, implicit grant, resource owner password credentials grant, and the client credentials grant (we refer to these as *modes* in the following). In addition, all of these modes provide further options.

Therefore, analyzing the security of OAuth is a complex task. So far, most analysis efforts were targeted towards finding errors in specific implementations [1, 3, 9, 12, 14], rather than the comprehensive analysis of the standard itself. Probably the most detailed formal analysis carried out on OAuth so far is the one in [1]. However, none of the existing analysis efforts of OAuth account for all modes of OAuth running simultaneously, which may potentially introduce new security risks. In fact, many existing approaches analyze only the authorization code mode and the implicit mode of OAuth. Also, importantly, there are no analysis efforts that are based on a comprehensive formal web model (see below), which, however, is essential to rule out security risks that arise when running the protocol in the context of common web technologies.

2 Contributions

We perform the first extensive formal analysis of the OAuth 2.0 standard for all four modes, which can even run simultaneously within the same and different RPs and IdPs, based on a comprehensive web model which covers large parts of how browsers and servers interact in real-world setups. Our analysis also covers the case of malicious IdPs, RPs, and browsers/users.

[1]Following the OAuth 2.0 terminology, IdPs are called *authorization servers* and *resource servers*, RPs are called *clients*, and users are called *resource owners*. Here, however, we stick to the more common terms mentioned above.

[2]Note that in our work, we consider only OAuth 2.0, which is very different to its predecessor, OAuth 1.0.

Formal model of OAuth. Our formal analysis of OAuth uses an expressive Dolev-Yao style model of the web infrastructure [4] proposed by Fett, Küsters, and Schmitz (FKS). The FKS model has already been used to analyze the security of the BrowserID single sign-on system [4, 5] as well as the security and privacy of the SPRESSO single sign-on system [6]. This web model is designed independently of a specific web application and closely mimics published (de-facto) standards and specifications for the web, for instance, the HTTP/1.1 and HTML5 standards and associated (proposed) standards. It is the most comprehensive web model to date. Among others, HTTP(S) requests and responses, including several headers, such as cookie, location, strict transport security (STS), and origin headers, are modeled. The model of web browsers captures the concepts of windows, documents, and iframes, including the complex navigation rules, as well as new technologies, such as web storage and cross-document messaging (postMessages). JavaScript is modeled in an abstract way by so-called scripting processes which can be sent around and, among others, can create iframes and initiate XMLHTTPRequests. Browsers may be corrupted dynamically by the adversary.

Using the generic FKS model, we build a formal model of OAuth, closely following the OAuth 2.0 standard (RFC6749 [8]). Since this RFC does not fix all aspects of the protocol and in order to avoid known implementation attacks, we use the OAuth 2.0 security recommendations (RFC6819 [10]), additional RFCs and OAuth Working Group drafts (RFC7662 [11], [2]) and current web best practices (e.g., regarding sesssion handling) to obtain a model of OAuth 2.0 with state-of-the-art security features in place, while making as few assumptions as possible (see also below). Moreover, as mentioned above, our model includes RPs and IdPs that (simultaneously) support all four modes and can be dynamically corrupted by the adversary. Also, we model all configuration options of OAuth.

Formalization of security properties. Based on this model of OAuth, we provide three central security properties of OAuth: authorization, authentication, and session integrity, where session integrity in turn is concerned with both authorization and authentication.

Attacks on OAuth 2.0 and fixes. While trying to prove these properties, we discovered four attacks on OAuth. In the first attack, which breaks the authorization and authentication properties, IdPs inadvertently forward user credentials (i.e., username and password) to the RP or the attacker. In the second attack (IdP mix-up), a network attacker playing the role of an IdP can impersonate any victim. This severe attack, which again breaks the authorization and authentication properties, is caused by a logical flaw in the OAuth 2.0 protocol. Two further attacks allow an attacker to force a browser to be logged in under the attacker's name at an RP or force an RP to use a resource of the attacker instead of a resource of the user, breaking the session integrity property. We have verified all four attacks on actual implementations of OAuth and OpenID Connect. We present our attacks on OAuth in detail in our technical report [7] where we also show how the attacks can be exploited in OpenID Connect.

We also show how all four attacks can be fixed by changes that are easy to implement in new and existing deployments of OAuth and OpenID Connect.

Formal analysis of OAuth 2.0. Using our model of OAuth with the fixes in place, we then were able to prove that OAuth satisfies the mentioned security properties. This is the first proof which establishes central security properties of OAuth in a comprehensive and expressive web model. Additionally, it is the first proof to establish security properties for OAuth with multiple or malicious IdPs, and the first one to consider all four OAuth modes.

We emphasize that, as mentioned before, we model OAuth with security recommendations and best practices in place. Implementations not following these recommendations and best practices may be vulnerable to attacks.[3] In fact, many such attacks on specific implementations have been pointed out in the literature (e.g., [1, 3, 8–10, 14, 15]). Hence, our results also provide guidelines for secure OAuth implementations.

We moreover note that while these results provide strong security guarantees for OAuth they do not directly imply security of OpenID Connect because OpenID Connect adds specific details on top of OAuth. We leave a formal analysis of OpenID Connect to future work. The results obtained here can serve as a good foundation for such an analysis.

References

[1] C. Bansal, K. Bhargavan, A. Delignat-Lavaud, and S. Maffeis. Discovering Concrete Attacks on Website Authorization by Formal Analysis. *Journal of Computer Security*, 22(4):601–657, 2014.

[2] J. Bradley, T. Lodderstedt, and H. Zandbelt. Encoding claims in the OAuth 2 state parameter using a JWT – draft-bradley-oauth-jwt-encoded-state-05. IETF. Dec. 2015. `https://tools.ietf.org/html/draft-bradley-oauth-jwt-encoded-state-05`.

[3] E. Y. Chen, Y. Pei, S. Chen, Y. Tian, R. Kotcher, and P. Tague. OAuth Demystified for Mobile Application Developers. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*, pages 892–903, 2014.

[4] D. Fett, R. Küsters, and G. Schmitz. An Expressive Model for the Web Infrastructure: Definition and Application to the BrowserID SSO System. In *35th IEEE Symposium on Security and Privacy (S&P 2014)*, pages 673–688. IEEE Computer Society, 2014.

[5] D. Fett, R. Küsters, and G. Schmitz. Analyzing the BrowserID SSO System with Primary Identity Providers Using an Expressive Model of the Web. In *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*, Lecture Notes in Computer Science, pages 43–65. Springer, 2015.

[6] D. Fett, R. Küsters, and G. Schmitz. SPRESSO: A Secure, Privacy-Respecting Single Sign-On System for the Web. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 1358–1369. ACM, 2015.

[7] D. Fett, R. Küsters, and G. Schmitz. A Comprehensive Formal Security Analysis of OAuth 2.0. Technical Report arXiv:1601.01229, arXiv, 2016. Available at `http://arxiv.org/abs/1601.01229`.

[3]Some quirks in specific implementations might prevent some attacks. Our discussions, however, highlight the importance of following the recommendations and best practices.

[8] D. Hardt (ed.). RFC6749 – The OAuth 2.0 Authorization Framework. IETF. Oct. 2012. https://tools.ietf.org/html/rfc6749.

[9] W. Li and C. J. Mitchell. Security issues in OAuth 2.0 SSO implementations. In *Information Security - 17th International Conference, ISC 2014, Hong Kong, China, October 12-14, 2014. Proceedings*, pages 529–541, 2014.

[10] T. Lodderstedt (ed.), M. McGloin, and P. Hunt. RFC6819 – OAuth 2.0 Threat Model and Security Considerations. IETF. Jan. 2013. https://tools.ietf.org/html/rfc6819.

[11] J. Richer (ed.). RFC7662 – OAuth 2.0 Token Introspection. IETF. Oct. 2015. https://tools.ietf.org/html/rfc7662.

[12] M. Shehab and F. Mohsen. Towards Enhancing the Security of OAuth Implementations in Smart Phones. In *2014 IEEE International Conference on Mobile Services*. Institute of Electrical & Electronics Engineers (IEEE), jun 2014.

[13] SimilarTech. Facebook Connect Market Share and Web Usage Statistics. Last visited Nov. 7, 2015. https://www.similartech.com/technologies/facebook-connect.

[14] S.-T. Sun and K. Beznosov. The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems. In T. Yu, G. Danezis, and V. D. Gligor, editors, *ACM Conference on Computer and Communications Security, CCS'12*, pages 378–390. ACM, 2012.

[15] R. Wang, Y. Zhou, S. Chen, S. Qadeer, D. Evans, and Y. Gurevich. Explicating SDKs: Uncovering Assumptions Underlying Secure Authentication and Authorization. In *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, pages 399–314. USENIX Association, 2013.