The Case for OAuth 3.0

Justin Richer ⁝ 7-9 minutes ⁝ 12/14/2019

It's been about a year since I proposed that we should move past OAuth 2.0, and a lot has happened since then to tackle some of the issues that I raised. I helped lead the TxAuth session at IETF 106 in Singapore last week, and we had a number of side meetings and follow ons from that to discuss the implications. So far, we're seeing two major approaches: extending OAuth 2.0, and building a new protocol from the ground up. I think there's room for both in the world, for different reasons.

Extending OAuth 2.0

OAuth 2 has been one of the most wildly successful security protocols on the internet to date, and it's been extended in all kinds of interesting and specialized ways. For example, the Rich Authorization Request (RAR) draft specification details a way to move beyond the limits of the scope parameter, while the Pushed Authorization Request (PAR) draft specification provides a way to avoid a lot of the problems with the front channel. When combined through the JOSE-based request (JAR) and response (JARM) mechanisms backported from OpenID Connect, we've got a pretty rich and powerful intent registration pattern.

These extensions don't always play nicely with each other, nor do they always play nicely with legacy applications. The PAR, CIBA, and Device Authorization Grant specs each define their own endpoint for doing essentially the same intent registration step, but in slightly different ways. And take PKCE for another example; it allows a native client to protect itself against code theft and injection attacks, but if the AS doesn't support the PKCE extension, the request will still succeed but without any of the PKCE protections and with the client being none the wiser of its vulnerability. And those Rich Authorization Requests? They need to live in a world that already has OAuth 2.0 scopes, OpenID Connect claims, Proof of Posession aud (audience) , and the new resource indicator field deployed. The combinatorics of these alone are frightening, and any AS that wants to support all of these for different APIs certainly has its work cut out.

On top of that, we've got a whole series of Best Current Practice documents for OAuth 2 to deal with native apps, single-page apps, JWTs in general, JWTs as access tokens, and OAuth 2 as a whole. Developers today need to navigate all of these extensions and guides to be able to deploy OAuth 2 in a safe and reasonable fashion. It's no mean feat, but I can recommend a good book on the topic.

Toward OAuth 3.0

With as much as we've been able to do in the legacy space, sometimes it's best to take a clean room approach. I've been working on the XYZ project for most of 2019 as a way to showcase what a new OAuth-style delegation protocol could look like if we could shed the

chains of backwards compatibility. In building out this project, I've discovered that we can make the protocol both simpler and more powerful if we take a small step back from the assumptions that OAuth 2.0 made a decade ago. The XYZ website has more details about the project, but I wanted to highlight a few things here.

For one, user interaction in OAuth 2.0 is assumed to be all about the user in a browser. While that's a powerful pattern, it's limiting in today's world of native applications and multi-device authorizations. In XYZ, the client can declare to the server how it can interact, whether through a browser or an agent or through a challenge-response mechanism or something else entirely. Furthermore, while OAuth 2 requires the client to know ahead of time whether it wants to get the user involved, XYZ allows the AS to decide at runtime if the user's presence is required for a given request, or if it can just return an access token right away. The user can get involved only when needed and only in ways that the client knows how to deal with.

Another major aspect is that the transaction is the most important aspect of the XYZ process. A client is a piece of software that can start a transaction, and an access token is the result of a transaction. Everything else in the process modifies or adapts the transaction over time to accomplish the delegation and authorization process. This reification of what was an implied underlying element allows us to more cleanly map how everything fits together over the life of the protocol.

XYZ takes much inspiration from UMA 2, a multi-party delegation protocol built on top of OAuth 2. UMA 2 and OAuth 2 coexist somewhat reasonably, but ultimately UMA 2 can't collapse back into vanilla OAuth 2 for the single-user use case because OAuth 2 starts with the user and assumes there's only one the whole time.

XYZ also adds other features like key proofing mechanisms, multi-user delegation, multi-device processing, support for ephemeral and static clients, and resource-first flows. All of this comes together in a protocol that lets us solve OAuth 2.0's use cases in a more clean and consistent manner, all while enabling new use cases to grow up right along side of them. This is why I have submitted XYZ as an internet draft to the IETF, and why I am proposing we start work on OAuth 3.0 using this as the basis. This work is happening on the TxAuth list in the IETF, which isn't yet a working group but could become one.

Defining OAuth 2.1

Somewhere past tacking on new extensions but shy of defining a new system is a growing desire to define an OAuth 2.1. This work would take all the Best Current Practice documents and roll them up into a core set of requirements. For instance, the implicit grant type could be deprecated in favor of using the authorization code grant type, and PKCE could be mandated for every authorization code transaction. This type of draft would be mostly compatible with OAuth 2.0, to the point where it would essentially be OAuth 2.0 but cleaned up and with a specific set of extensions turned on and best practices made mandatory.

Since publication of this article, Aaron Parecki has written a great blog post about what OAuth 2.1 could be.

All I Want Is Everything

Interestingly, I predicted much of this movement way back in the summer of 2016 in a talk I gave entitled “The Next Three Years”. I believed then as I believe now: technology is always growing and changing, and it’s up to all of us to make sure it grows the best way possible.

To me, it makes the most sense to build out all of these paths simultaneously. OAuth 2.0 isn’t going away any time soon, and people will still need to build it out and use it while we work on the next generation. I still believe the extensions and profiles of OAuth 2.0 that we’re building today make sense to do, evidenced by the fact I’m involved with many of these efforts myself. And even after something new is available, people will keep using the old system for many years to come. After all, OAuth 1.0 is still out there, and SAML is still deployed even in light of OpenID Connect. An OAuth 2.1 revision would let us clean up some things, while OAuth 3.0 would let us address many of the long-standing issues that we’ve only managed to back-patch so far.

And to be clear, I don’t actually care if the new work is called OAuth 3.0 or TxAuth or some other name, but I do think that it’s a fitting change set for a major revision of a powerful and successful protocol. We need something new, while we patch what we have, and now’s the time to build it. Come join the conversation in TxAuth, play with XYZ, and let’s start building the future.