OAuth

Thursday 2 September 2010 05:40

At the end of August twitter decided to drop Basic Authentication for access to their API. This means an application can no longer connect to their system using only a username and password combination belonging to an account. While this might make sense for the big services out there relying on a connection with twitter, for the two things I do with it it doesn't.

I can do whatever I want with my twitter account using a web browser. I can log in using my username and password and that works as expected. But if you want to have the same functionality as a browser, but use the API, that browser suddenly needs more than the username and password to get the same data from the system. This makes sense if that browser is operated by a third party and you don't want to give your password to that third party. The OAuth system passes a valid login into twitter for that application on to that application without telling it your password. This makes certainly sense if the third party is a website.

But I'm merely running some local cronjobs accessing my twitter account. To keep them functioning I needed to register them as an application with twitter so I get a consumerkey and consumersecret for the OAuth authentication. Using that I can then treat myself as a third party and generate tokens that allow my cronjobs to access my account without knowing my password. No problems there.

But since the cronjobs are just Perl scripts I might like to share, there is no point including the consumersecret in those scripts. Because if the secret is out there anyone can use that to pretend to be my registered application, which is of course something we don't want. So I put the consumer token pair in my configuration, and anyone who wants to use the same script will have to register their own application and can use their own consumer tokens in their own configuration.

This approach of having everyone use their own registered application's consumerkey and consumersecret makes it basically an inconvenient way of using a username and password.

So why is twitter doing this? My guess is that this extra layer of authentication gives them a kill switch for every registered application. With only a username and password they can't prevent people accessing the API without blocking them also from the site. Using the tokens of a registered application they can expire the tokens of an application to prevent that application from accessing the API, but users can still use their own username and password in their browser or other applications using the API. And this makes it a business model: If your application is generating a lot of traffic, twitter will notice that by the number of authentications it's handling for that registered application, and if you don't pay up your tokens could be pulled or get a limited rate. Unless you teach your users to register their own application and use their own consumerkey and consumersecret to access their own data.

by Roland van Ipenburg http://www.xs4all.nl/~ipenburg/blog/posts/work/2010/09/02/oauth/