Experimental AM Protocol

This is my initial take on the problem on how to do distributed authorization. It is not compatible with the work done in the UMA Workgroup at the moment.

Motivation

The motivation for creating this proposal was an attempt to find a simple solution for mainly solving the Distributed Authorization Scenario described in the UMA Workgroup. It is also an attempt to reuse more of OAuth than the initial protocol sketch of ProtectServe did back then. Moreover it is an attempt to create several small specifications than one big one. For instance all the mechanics of how to do terms and policies is out of scope for the core protocol.

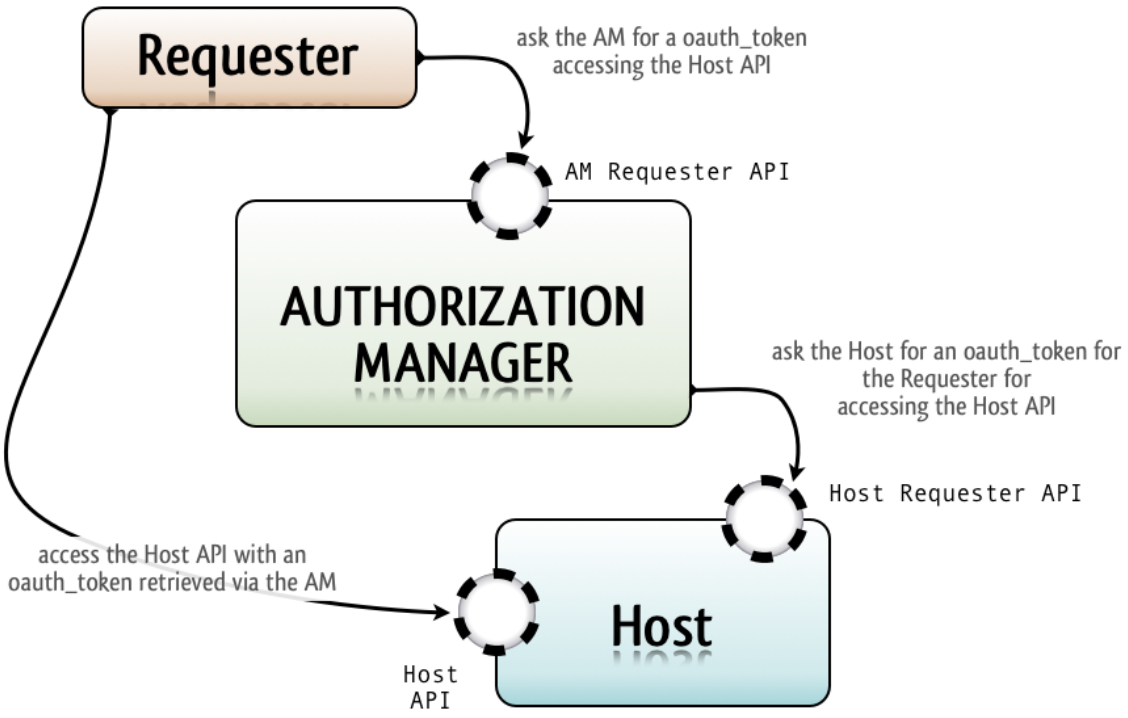
The main requirement behind this is to be able to do mass authorization without the need for each client to obtain a token for each individual server. Instead the Authorization Manager is introduced to only create a relationship with each Host once and after that any Requester only needs to create a registration with the AM instead of each Host. Moreover the new Requester could additionally be added as new Host in this step, saving some redirects.

The main idea

The OAuth specification states that the described Redirection Based Authorization is only one way for a server to provide a client with a token. Therefor this specification is describing an alternative way for a client to obtain a server without the need for it to have a direct relationship with the server. This is done with the additional component of an Authorization Manager in the middle. So the actual call of the client to a resource on the server is done as described in the OAuth specification and only the way of obtaining that token is different.

This is done by a chained token transfer. For this the Requester calls the Token Requester API on the Authorization Manager (AM) and the AM in turn calls the Token Requester API of the Host. The Host then returns a token (maybe with prior checks) to the AM which in turn is given to the Requester which then can perform the actual call with it.

Here is an overview of the process:



Roles

The following roles are defined:

- **Requester** is performing authorized requests to a Host (in OAuth terms it's a client)
- **Host** is the component which hosts a resource which a Requester wants access to (in OAuth terms it's a server)
- **Authorization Manager** (AM) is the component in the middle which is capable of negotiating between Requesters and Hosts and which can give out tokens. It might also be the central point for managing terms and policies in the future.

Protocol Overview

The protocol contains two initial steps:

1. The registration of the Requester with the AM to be able to call the **AM Requester API**
2. The registration of the AM with the Host to be able to call the **Host Requester API**

These steps only need to be done once for each pair (Requester, AM) and (AM, Host). They are long running associations. Both steps are done via normal 3-legged-OAuth. In the end the Requester owns an OAuth Token for the AM Requester API and the AM owns an OAuth Token to call the Host Requester API.

The normal protocol flow contains two additional steps:

1. The Requester obtaining an OAuth token from the Host via the AM
2. The actual call of the Requester to the protected resource on the Host

Initial Registrations

Requester - AM Registration

Each Requester which wants to use the AM's services need to register with the AM. This is usually triggered by the user on the Requester who wants the Requester to be able to access the user's resources. For instance this can be triggered by a user signing up with the Requester and trying to provide the Requester with access to all it's life-streaming services such as Twitter or Facebook (which would be the Hosts then).

The flow is a normal 3-legged OAuth Redirection-Based Authorization workflow as described in section 2 of the IETF OAuth specification. If everything is successful the Requester posses an OAuth Token for calling the AM Requester API on the AM.

AM - Host Registration

This step is usually done before the Requester registration. Here the user signs up to a new service and wants the authorization decisions being handled by an Authorization Manager. Those services could be Facebook or Twitter and again they are Hosts in this scenario.

In order to do that the AM initiates another 3-legged OAuth Redirection-Based Authorization workflow for each of them. The result is that the AM then has an OAuth token for each of these Hosts for calling the Host Requester API. Note that this is not the same as the protected resource a Requester might want to call.

The main flow

Now the Requester might want to post something on the Hosts of that user. It needs an OAuth Token of the Host to do so. Usually this is obtained by a 3-legged-OAuth flow from Requester to Host. In this case though no such thing is needed. Instead the following is happening:

1. The Requester uses the OAuth token of the AM obtained above to call the AM Requester API.
2. The AM in turn will call the Host Requester API of the Host in question
3. The Host returns a token for accessing the protected resource to the AM
4. The AM passes this token to the Requester
5. The Requester now can access the protected resource on the Host.

Open Questions

- How do the APIs look like?
- What data needs to be presented to those APIs?
- AM Requester API:
    - which resource on the server to access
    - later: terms?
- Host Requester API
    - which Requester is knocking on the door?
    - for which resource?
    - later: terms? policies? (latter might be defined on the AM)