OAuth for eduVPN

François Kooman

<fkooman@tuxed.net>

@fkooman

Problem

How to get and maintain an OpenVPN configuration on a device in the field?

What is OAuth?

- “Open **Authorization**” for the web;
- Authorization of *App* (client) to *API* (resource server);
- *App* X wants to use *API* Y on behalf of *user* Z (resource owner);
- **Authorization != Authentication**

What is OAuth?

- A *client* obtains an *access token* from the *authorization server* bound to a *resource owner* and *scope* to access a *resource server*.

Why OAuth Exists

- API cannot (directly) verify user credentials, e.g. no access to them (SAML);
- Prevent App from learning/leaking user credentials;
- Limit capabilities of App (scope, expiry).

Why OAuth for eduVPN?

- We do *not* control/provision the client (BYOD);
- App needs to download OpenVPN configuration;
- Channel needed to *update* this configuration *over the air*, e.g. to update the remote(s);
- Leverage Identity Federations, e.g. SURFconext, for user authentication;
- Authentication only available through browser (no ECP);
- OAuth seems the (standard) way to go, many (big) organizations on board, is that good or bad?!

OAuth

- OAuth 2.0 (RFC 6749) is a “Framework”, i.e. you are on your own;
- Lots of OPTIONAL, MAY, SHOULD;
- Needs additional RFCs to make things work:
  - The OAuth 2.0 Authorization Framework: Bearer Token Usage (RFC 6750)
  - Proof Key for Code Exchange by OAuth Public Clients (PKCE) (RFC 7636);
  - OAuth 2.0 for Native Apps (RFC 8252);
  - OAuth 2.0 Token Introspection (RFC 7662)

OAuth Flow

- “App” opens (default) system browser by opening “authorization endpoint”
  - Benefits from Single Sign On (SSO) if available in active browser session
- OAuth server authenticates user, and allows users to “Approve” the authorization
- Server sends browser back to registered URL endpoint of “App” (e.g. app registered custom scheme, …)
- App exchanges “authorization code” at “token endpoint” for access token (and refresh token)…
- Access token is used to access API

“Distributed/Federated” eduVPN

Allow users of organization A to use the VPN server of organization B as a *guest*...

“Distributed/Federated” eduVPN

Use *Bearer* token from VPN server A at VPN server B.

The Snake Pit

- (I actually like, non poisonous, non strangling snakes!)
- Public-key cryptography for Bearer tokens:
  - Short lived, e.g. 1 hour
    - Requires refresh tokens, avoid bugging user every hour
  - No (real) need to query OAuth server, avoids token “introspection”, depends at bit perhaps...;
  - Very easy to accept tokens from other OAuth servers, just configure their public key.

JSON Web Token (JWT)?

Nope.

XMLDSig all over again...

```
"alg": "none"
```

This is why we cannot have nice things!

libsodium?

Yes!

libsodium: easy to use, secure, crypto library.

Finally!

Signatures

- Public-key signatures
  - Ed25519
  - Simple JSON structure:
    - ***Issuer***: issuing OAuth server FQDN;
    - ***Subject***: (persistent opaque) user identity at issuing OAuth server;
    - ***Audience***: *
    - ***Expiry***: NOW() + 1 hour
    - ***Replay***: yes, please!

Registry

- Create a registry with Public-key information of all known (and trusted) VPN servers;
- Policy in place between VPN server operators (NRENs);
- (Offline, detached) signature over registry;
- Auto verification and import of registry by all participating VPN servers;
- Tokens can be used everywhere!

Abuse

- Stop abuse with immediate blocking of guest user;
- Work with originating VPN server to identify user, if needed (bound to policy).