Author: Paul Feuvraux

JWT-like as session authentication for Muonium

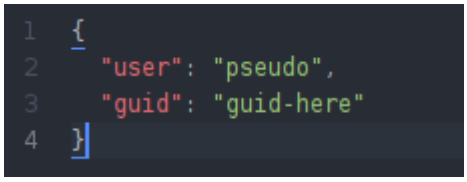
Introduction

The backend of the Muonium webapp is begin turned into an API, we'll need a tokens-based session authentication so that the users can open several sessions with the same account. We might also want to open the API for third parties.

Protocol

Connection

The user sends their credentials, if the credentials are validated, then a GUID and a key are generated randomly. The payload is created such as:



Once created, the Payload is base64encoded such as **:**

ewoidXNlciI6ICJwc2V1ZG8iLAoiZ3VpZCI6ICJndWlkLWhlcmUiCn0K

We next hmac the b64encoded Payload such as:

$sign = (key, Payload_{b64})$, where *Payload* is the base64encoded payload.

Then, we create the token such as:

$token = (Payload_{b64} :: sign)$, where "::" mark string concatenation.

$token$ is sent to the client, and stored into a Redis database as a packet such as:

$packet = (token :: key)$, where key is the randomly generated key for the hmac process.

The storing scheme of the packets is the following:

$['GUID'][' packet]$, where $GUID$ is the value and $packet$ the value.

Verifying

For each request, the server needs to verify the session. Server receives $token = (Payload_{b64} :: sign)$, the Payload is b64decoded and Server parses $GUID$ from $Payload$. Then it fetches $packet$ as value from the Redis database with $GUID$ as a key.

Once fetched, Server decodes $Payload$ from $packet$ and parses $sign$ and key .

Author: Paul Feuvraux

Then Server proceed to a verification through HMAC such as

$sign_{verifier} = HMAC - SHA384(key, Payload_{b64})$, where $Payload_{b64}$ is the Payload from the token, sent by the client.

Server checks if $sign_{verifier} = sign$, if so, then the client's session is authenticated and the request is processed.