

Restrict your configuration with a JWT Token

The objective is to limit the creation of a conference room to any user authenticate with a JWT token. Guests will have to wait for one of these users to come and unlock the room.

You can find the spec here: <https://github.com/jitsi/lib-jitsi-meet/blob/master/doc/tokens.md>

Packages

You need those package to user the tokens authentication:

```
yay -S --needed lua52-base64 lua52-basexx lua52-cjson lua52-jwtjitsi lua52-luaossl
```

Jicofo

/etc/jicofo/jicofo.conf

```
jicofo {  
  authentication {  
    enabled = true  
    type = JWT  
    login-url = YOUR_DOMAIN  
    enable-auto-login = true  
  }  
}
```

The restart.

```
systemctl restart jicofo
```

Jitsi meet

/etc/webapps/jitsi-meet/config.js

```
var config = {
  host: {
    // anonymous users need to use a dedicated muc without authentication
    anonymousdomain: 'guest.YOUR_DOMAIN',
  },
}
```

Prosody

/etc/prosody/conf.d/jitsi.cfg.lua

```
-- change authentication of your domain
VirtualHost "YOUR_DOMAIN"
  authentication = "token"
  app_id = "APP_ID"
  app_secret = "APP_SECRET"
  allow_empty_token = false
  modules_enabled = {
    -- keep existing modules and add
    "presence_identity";
  }
  c2s_require_encryption = false
-- add guest virtual host to allow anonymous user to join your room
VirtualHost "guest.YOUR_DOMAIN"
  authentication = "jitsi-anonymous"
  c2s_require_encryption = false
  modules_enabled = {
    -- copy the content of the modules_enabled
    -- of the VirtualHost "YOUR_DOMAIN"
    -- remove only the module "muc_lobby_rooms" of the list
    -- add presence_identity
    -- example:
    "bosh";
    "pubsub";
    "ping"; -- Enable mod_ping
    "speakerstats";
    "external_services";
```

```
"conference_duration";  
"websocket";  
"presence_identity";  
}
```

Component "conference.YOUR_DOMAIN" "muc"

```
modules_enabled = {  
  -- add this to the modules_enabled  
  "token_verification";  
}
```

Then restart

```
systemctl restart prosody
```

Generate a JWT token

Here a quick example in nodejs that will generate a valid link with a JWT token:

```
const jwt = require('jsonwebtoken')  
const crypto = require('crypto');  
const words = require('random-words')  
const yourDomain = "YOUR_DOMAIN"  
const appId = "APP_ID"  
const appSecret = "APP_SECRET"  
const userName = "YOUR_USERNAME"  
const userEmail = "YOUR_EMAIL"  
function getBody(domain, appId, name, email, room) {  
  const md5Email = crypto.createHash('md5').update(email).digest("hex");  
  const id = crypto.createHash('sha1').update(`${name}:${email}`).digest("hex")  
  return {  
    context: {  
      user: {  
        avatar: `https://gravatar.com/avatar/${md5Email}`,  
        name,  
        email,  
        id,  
      },  
      group: 'users'  
    },  
    "aud": "jitsi",
```

```
    "iss": appld,
    "sub": domain,
    room,
  }
}
const room = process.argv[2] || words({exactly: 3, join: '-'})
const data = getBody(
  yourDomain,
  appld,
  userName,
  userEmail,
  room,
)
const options = {
  algorithm: 'HS256',
  expiresIn: '2h',
}
const jwtToken = jwt.sign(data, appSecret, options)
console.log(`https://${yourDomain}/${room}?jwt=${jwtToken}`)
```

Revision #1

Created 8 June 2023 06:43:01 by Celogeek

Updated 8 June 2023 06:47:30 by Celogeek