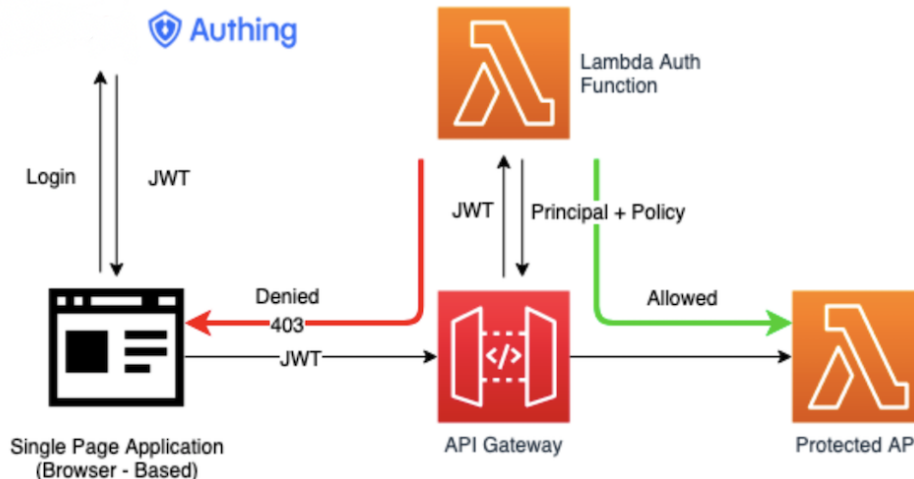


# API Gateway&Authing Authentication POC

## POC Requirements

The [API Gateway custom authorizers](#) with Authing will be used as an authentication method. The architecture will be :



## Details

This is an example of how to protect API endpoints with [Authing](#), JSON Web Tokens (jwt) and a [custom authorizer lambda function](#).

Custom Authorizers allow you to run an AWS Lambda Function before your targeted AWS Lambda Function. This is useful for Microservice Architectures or when you simply want to do some Authorization before running your business logic.


## Prerequisite

1. NodeJS 14.x or above
2. An AWS China Region account(AWS Global account does not work)
3. An [Authing](#) account

## Set up


1. Install [NodeJS Authing SDK dependencies](#), run `npm install`.
2. Prepare an [Authing application](#), use the default application or [create a new one](#).

Please select the type of application created



Self-built Application

Create mobile, web, and IoT applications and use Authing for authentication



Integrated Application

Single sign-on to purchased applications and SaaS services (such as Alibaba Cloud, AWS, etc.).

Application Name \*

CelloSquare-Test

Subdomain \*

https://

cellosquare-test

.authing.cn

Cancel

create

Note: after the Authing application created, please make sure the verification is all **none**.

Authorization configuration

Save

Authorization Flow ?

☒ authorization\_code

☐ implicit

☒ refresh\_token

☒ password

☐ client\_credentials

Return Type ?

☐ code id\_token token

☐ code id\_token

☐ code token

☒ code

☐ id\_token token

☐ id\_token

☐ none

Id\_token signature algorithm ?

☒ HS256

☐ RS256

Don't enforce Https for implicit mode callback ?

☐

Enable id\_token encryption ?

☐

Client Verification Method for Fetching Token

☐ client\_secret\_post

☐ client\_secret\_basic

☒ none

Client Verification Method for Validating Token

☐ client\_secret\_post

☐ client\_secret\_basic

☒ none

Client Verification Method for Revoking Token



☐ client\_secret\_post


☐ client\_secret\_basic

☒ none

3. Gather all the Authing application information used by JavaScript SDK and we will use this in Step 4, including
- **App ID:** Authing unique application ID for every one, and all the SDK will need this information to init.
  - **App Secret:** Authing issue for every applications to be identified by the platform, and some Authing SDK may need this to init. This should be always used in backend in production for security reason. But we need to broken the rule in this demo, because we need keep this demo simple enogh. And if we want to use this App Secret only in backend then we must use HTTPS as the call back URL.
  - **Subdomain:** The endpoint for Authing API and all the SDK need this to init.

● Applications / CelloSquare-Test



**CelloSquare-Test** 

App ID: 618b91ed336e00cd99440d79 

Experience Login Tutorial

Configuration Login Control Branding Access Authorization Advanced Configuration

Endpoint information

App Secret: e757\*\*\*\*\*  Refresh 

Issuer: https://cellosquare-test.authing.cn/oidc


Service Discovery Address: https://cellosquare-test.authing.cn/oidc/.well-known/openid-configuration

JWKS Public Key Endpoint: https://cellosquare-test.authing.cn/oidc/.well-known/jwks.json

Token Endpoint: https://cellosquare-test.authing.cn/oidc/token

User Information Endpoint: https://cellosquare-test.authing.cn/oidc/me

Logout Endpoint: https://cellosquare-test.authing.cn/oidc/session/end

Auth Config  Save

\* Subdomain: https:// cellosquare-test .authing.cn

Protocol Type: ☒ OIDC ☐ OAuth 2.0 ☐ SAML2

4. Please modify the `serverless.yml` according to your keys.
- Line 27, modify the prefix in the `serverless.yml`, make sure it is unique in site and bucketname.
  - Line 28, modify the `authId` to your App ID.
  - Line 29 modify the `authUrl` to Subdomain.
  - Line30 modify the `authKey` to App Secret.
5. Please deploy Serverless framework

```
sls deploy
```

5. Run your frontend locally or deploy your frontend to host of your choosing. In both cases, make sure to configure the `Login Callback URL` in your Authing console. An example of how to run your frontend locally:

```
cd front;  
python -m http.server
```

And also need start Serverless offline:

```
sls offline
```

It will work like the following:

```
ericacn@08f8bc6ad9d4 aws-apigateway-authing-main % sls offline  
Serverless: Running "serverless" installed locally (in service node_modules)  
offline: Starting Offline: dev/cn-northwest-1.  
offline: Offline [http for lambda] listening on http://localhost:3002  
offline: Function names exposed for local invocation by aws-sdk:  
  * auth: aws-apigateway-authing-dev-auth  
  * publicEndpoint: aws-apigateway-authing-dev-publicEndpoint  
  * privateEndpoint: aws-apigateway-authing-dev-privateEndpoint  
  * s3ConfigModify: aws-apigateway-authing-dev-s3ConfigModify  
offline: Configuring Authorization: api/private auth  
  
  POST | http://localhost:3000/dev/api/public  
  POST | http://localhost:3000/2015-03-31/functions/publicEndpoint/invocations  
  POST | http://localhost:3000/dev/api/private  
  POST | http://localhost:3000/2015-03-31/functions/privateEndpoint/invocations  
  
offline: [HTTP] server ready: http://localhost:3000 🚀  
offline:  
offline: Enter "rp" to replay the last request
```

## Custom Authorizers functions

[Custom authorizers functions](#) are executed before a Lambda function is executed and return an Error or a Policy document.

The Custom authorizer function is passing an `event` object to API Gateway as below:

```
{  
  "type": "TOKEN",  
  "authorizationToken": "<Incoming bearer token>",  
  "methodArn": "arn:aws:execute-api:<Region id>:<Account id>:<API id>/<Stage>/<Method>/<Path>"  
}
```

You will have to change this policy to accommodate your needs. The default reply provided, will only authorize one endpoint!

## Workflow

### Frontend

The frontend is a bare bones vanilla JavaScript implementation.

### Issues you may meet

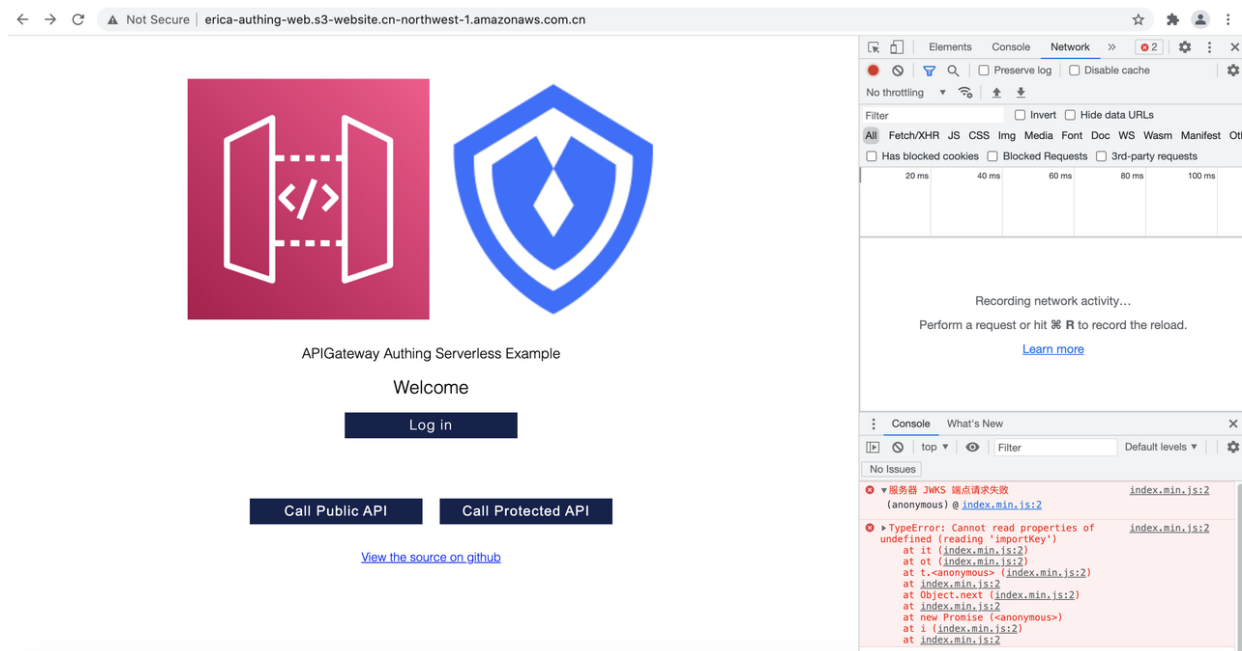
1. Make sure you have installed two plugins, [serverless-offline](#) and [serverless-s3-sync](#), otherwise you will get the following error when running "sls deploy"

Serverless Error -----

Serverless plugin "serverless-offline" not found. Make sure it's installed and listed

2. The error like the following:

"TypeError: Cannot read properties of undefined (reading 'importKey')"



The reason is that there are two signature algorithms, HTTP with HS256 are used in test case. In the production, please choose HTTPS with RS256.

Authorization configuration Save

Authorization Flow :

<input checked="" type="checkbox"/> authorization_code	<input type="checkbox"/> implicit	<input checked="" type="checkbox"/> refresh_token	<input checked="" type="checkbox"/> password
<input type="checkbox"/> client_credentials			

Return Type :

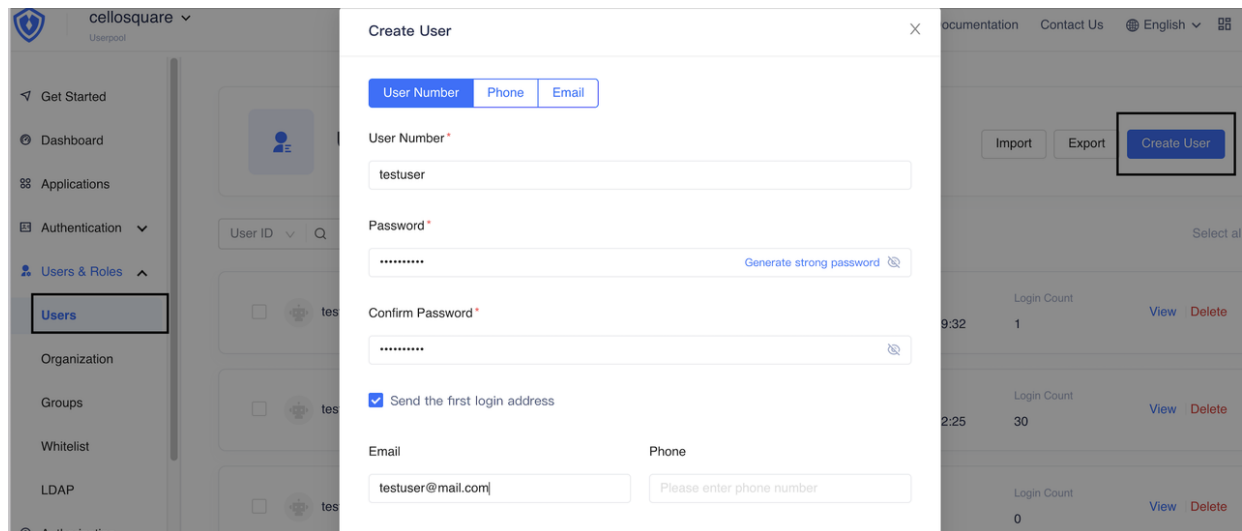
<input type="checkbox"/> code id_token token	<input type="checkbox"/> code id_token	<input type="checkbox"/> code token	<input checked="" type="checkbox"/> code
<input type="checkbox"/> id_token token	<input type="checkbox"/> id_token	<input type="checkbox"/> none	

Id\_token signature algorithm :

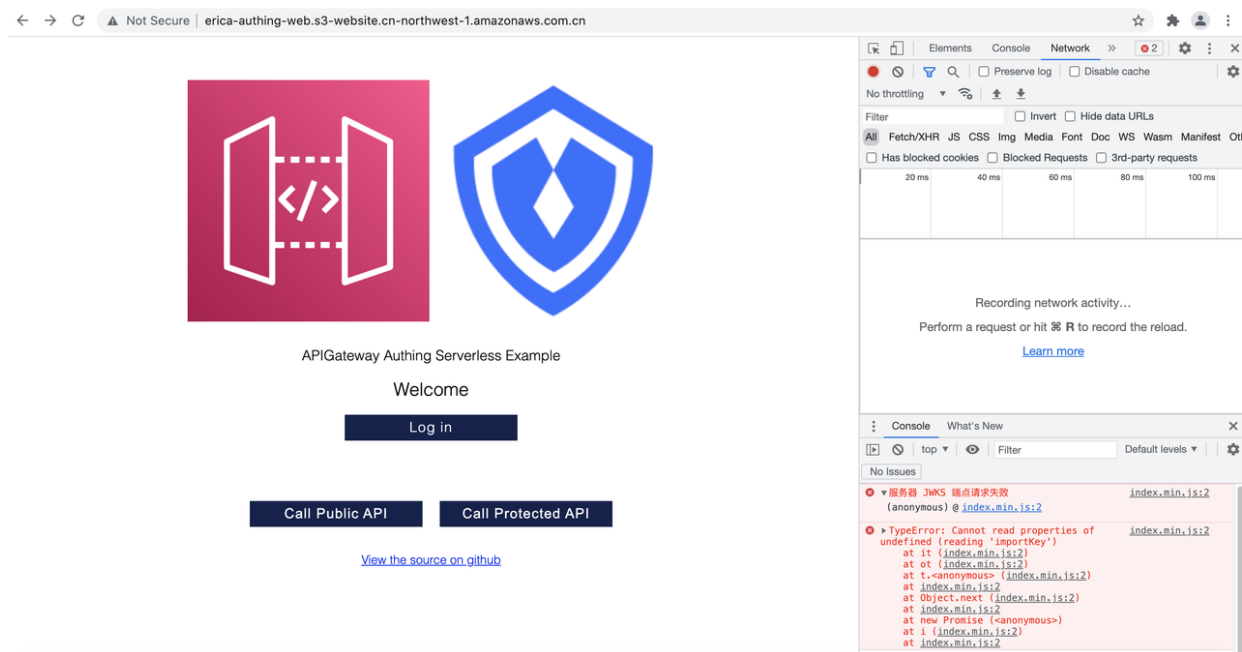
<input checked="" type="radio"/> HS256	<input type="radio"/> RS256
--	-----------------------------

## How it works

1. Prepare a user in the Authing.



2. Open your link in the Browser and Click “Log in” .



3. Enter your user's name and password that is created in the first Step.

The screenshot shows a web browser window with the URL `cellosquare-test.authing.cn/login?app_id=618b91ed336e00cd99440d79&uid=E4ff8Og3cg_ICzDRjvwVq&nonce=9484501786986597&state=5619689274743838&scope=...`. The page is titled "CelloSquare-Test" and has a login form with fields for "Phone" and "Password". The "Password" field is filled with "testuser". Below the form is a "Login" button. There are links for "Forgot password?" and "No account yet, sign up now". The page also has a language selector set to "English" and a "Help" link.

The network console on the right shows a list of requests. The first request is a GET request to `manifest.json` with a status of 200. The console also shows a warning message: "Manifest: property 'start\_url' ignored, should be same origin as document."


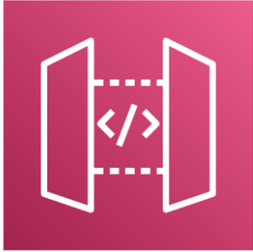
4. If you log in successfully, your user's email will show on the page.

The screenshot shows a web browser window with the URL `erica-authing-web.s3-website.cn-northwest-1.amazonaws.com.cn/?code=MA7Akj6RCboAPsoSik_nXDBdWbXao7nadyjtyeRUxCS&state=5619689274743838`. The page is titled "APIGateway Authing Serverless Example" and displays the email "Welcome testuser@mail.com". There is a "Logout" button and two buttons: "Call Public API" and "Call Protected API". A link "View the source on github" is also present.

The network console on the right shows a list of requests. The first request is a GET request to `app.js` with a status of 200. The console also shows a warning message: "TypeError: Cannot read properties of undefined (reading 'importKey')".

6. Try to call the different API, public API and protected API.

→ ↻ ⚠ Not Secure | erica-authing-web.s3-website.cn-northwest-1.amazonaws.com.cn/?code=MA7Akj6RCboAPsoSlk\_nXDBdWbXAo7nadyjtyeRUxCS&state=5619689274743...



APIGateway Authing Serverless Example

Welcome testuser@mail.com

Logout

Hi 从 Public API

Call Public API Call Protected API

[View the source on github](#)

Elements Network

No throttling

Filter

Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest

Name	Status	T...	Ini...	S...	T...	Waterfall
authing-co...	200	p...	26...	(...	4...	
jwtks.json	204	p...	Pr...	0 B	3...	
token	204	p...	Pr...	0 B	2...	
token	200	xhr	in...	1...	4...	
jwtks.json	200	xhr	in...	6...	1...	
jwtks.json	200	xhr	in...	6...	2...	
public	200	f...	ag...	2...	6...	

15 requests | 6.2 kB transferred | 1.6 MB resources | Finish: 19.70 s


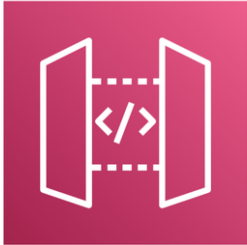
Console

1 Issue

TypeError: Cannot read properties of undefined (reading 'importKey')

Message: {message: 'Hi 从 Public API'}

← → ↻ ⚠ Not Secure | erica-authing-web.s3-website.cn-northwest-1.amazonaws.com.cn/?code=MA7Akj6RCboAPsoSlk\_nXDBdWbXAo7nadyjtyeRUxCS&state=5619689274743...



APIGateway Authing Serverless Example

Welcome testuser@mail.com

Logout

Hi 从 Private API. Only logged in users can see this

Call Public API Call Protected API

[View the source on github](#)

Elements Network

No throttling

Filter

Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest

Name	Status	T...	Ini...	S...	T...	Waterfall
token	204	p...	Pr...	0 B	2...	
token	200	xhr	in...	1...	4...	
jwtks.json	200	xhr	in...	6...	1...	
jwtks.json	200	xhr	in...	6...	2...	
public	200	f...	ag...	2...	6...	
private	200	f...	ag...	3...	1...	
private	200	p...	Pr...	0 B	3...	

17 requests | 6.5 kB transferred | 1.6 MB resources | Finish: 47.06 s

Console

1 Issue

Message: {message: 'Hi 从 Private API. Only logged in users can see this'}

## Reference

This project is refer to the [Serverless](#) examples in the [Github](#).

## License Summary

This sample code is made available under the MIT-0 license. See the LICENSE file.