Sikre Documentation

Release 0.1alpha

Oscar Carballal Prego

July 04, 2016

Contents

CHAPTER 1

What is Sikre?

Sikre (or the service equivalent, sikr.io) is a secure password storage API to protect all your passwords, SSH keys, SSL certificates, or other sensible information that you might have.

The principle of Sikre is "no one knows nothing" *(insert John Snow joke here)* so there is no hidden administration, no interfaces to administer the site or helpers to help you fix something that might go wrong (don't worry, even if the API fails, the data is secure). Unfortunately, that means also that we don't implement any methods for recovering data, so if you forget you master password or accidentally delete something, **no one can recover it**.

CHAPTER 2

Is there any support then?

Yes there is, I actively develop this application, and I'm all ears regarding new features or bugs that you might have find, specially if they're related to the security of the API.

Contents:

2.1 Installation

2.2 API endpoints

Sikre is a REST type API, here is the list of all it's endpoints and required methods.

2.2.1 Authentication

2.2.2 Categories

Categories is the top level data that you can have, inside the categories you can find the items and the services. An example of a category could be "Facebook", in case you have more than one Facebook password or service or for example "Google" and inside you store multiple items with your GMail accounts and other services like Analytics, etc.

GET /v1/categories Return a list of all the category objects assigned or created to/for that user. The information returned is only the `name` and the `id` of the category.

Example JSON:

```
[
  {
    "id": 4,
    "name": "Category 1"
  },
  {
    "id": 5,
    "name": "Category 2"
  },
  {
    "id": 6,
    "name": "Category 3"
  }
]
```

POST /v1/categories Saves the information for a new category. The only data needed is the category `name`.

Example JSON:

```
{
  "name": "Category 4"
}
```

The rest of the information is worked out through the `JWT token` sent in the header.

Specific categories

GET /v1/categories/<id> Return the information of a specific category, the result is the same as for the whole categories list, but only for one object.

Example JSON:

```
[
  {
    "id": 4,
    "name": "Category 1"
  },
]
```

PUT /v1/categories/<id> This method is used to edit an existing category. The UI retrieves the category and after the user modifies the content, we send back a `PUT` request with the new information. The only data needed is the category `name`.

Example JSON:

```
{
  "name": "Category 4-1"
}
```

The rest of the information is worked out through the JWT token sent in the header.

DELETE /v1/categories/<id> This method deletes a specified category **and all it's siblings**, that means, if the category contains items or services inside the items, everything will be deleted.

2.2.3 Items

2.2.4 Services

2.2.5 Generic

Generic endpoints provide information about the API and also serve the purpose of testing the API.

2.3 Reference

This is a reference of all the methods withing the API.

2.3.1 Middleware

class sikre.middleware.handle_404.**WrongURL**

process_response(*req*, *resp*, *resource=''*)
Intercept main 404 response by Falcon

If the API hits a non existing endpoint, it will trigger a customized 404 response that will redirect people to the documentation.

Raises *HTTP 404* – A falcon.HTTP_404 error

Returns A customized JSON response

Return type JSON

class sikre.middleware.headers.**BaseHeaders**

process_request(*req*, *res*)
Process the request before entering in the API

Before we process anything in the API, we reset the Origin header to match the address from the request.

Parameters **Access-Control-Allow-Origin** – Change the origin to the URL that made the request.

Raises *HTTP Error* – An HTTP error in case the Origin header doesn't match the predefined regular expression.

Returns A modified set of headers.

Return type HTTP headers

process_response(*req*, *res*, *resource*)
Process the response before returning it to the client.

In the reutrning reponse we change some values to be able to overcome the CORS protection and mask the origin server. The CORS interaction is protected by a check agains a regular expression to make sure the origin is a website-like URL.

> **Warning:** If you are really concerned about security, you can deactivate the CORS allowance by turning CORS_ACTIVE to *False* in your settings file. That will force the application to answer to the SITE_DOMAIN domain.

Parameters

- **Server** (*string*) – Changes the server name sent to the browser in the response to avoid exposure of name and version of the same.
- **Access-Control-Allow-Origin** (*string*) – Change the origin name to match the one that made the request. That way we can allow CORS anywhere.

Raises

- *HTTP Error* – An HTTP error in case the Origin header doesn't match
- the predefined regular expression.

Returns A modified set of headers

Return type HTTP headers

class sikre.middleware.https.**RequireHTTPS**
Force the connection to be HTTPS.

Middleware that intercepts all the requests and checks that is over an HTTPS protocol before continuing. The only exception to this is the DEBUG mode, in which we allow connections from non-HTTPS sources.

Raises *HTTP Bad Request* – If the connection is not HTTPS the API will complain

Returns Error mentioning the HTTPS connection is required

Return type JSON

class sikre.middleware.json.**RequireJSON**

class sikre.middleware.json.**JSONTranslator**

2.4 Support

To be written

2.5 FAQ

B

J

P

R

W