sysPass Documentation

Release 3.0.0

Rubén Domínguez

Jan 08, 2019

Contents

sysPass is a password management system written in PHP that allows a centralized and collaborative passwords management.

CHAPTER 1

Features

- Encrypted passwords using AES-256 CTR
- Interface based on Material Design Lite with HTML5 and AJAX
- Multiuser with users, groups and profiles management
- Advanced profile management with 29 access levels
- MySQL/MariaDB, OpenLDAP and Active Directory authentication
- Activity notifications by email and in-app
- Public links to accounts without login
- Accounts changes history and restore points
- Accounts associated files management with images preview
- Multilanguage
- Portable backup format and export to encrypted XML
- Actions and events audit with the ability to send messages to a remote Syslog in CEF format
- API for integrating with other applications
- Import from KeePass and CSV
- One step installation

CHAPTER 2

What sysPass does not do

- It does not store the master password in the server
- It does not send any data to an external service
- It does not encrypt the accounts' password individually, it uses a master password for all instead
- It does not perform password changes on the servers
- It does not encrypt the accounts' data, only the password and custom fields data, because you wouldn't be able to perform searches
- It isn't like APT: doesn't have Super Cow Powers!!

2.1 Installation

2.1.1 Debian 9 Installation

Warning: Work in progress

Prerequisites

- Web server (Apache/Nginx/Lighttpd) with SSL enabled.
- MariaDB >= 10.1
- PHP >= 7.0
- **PHP modules**
  - mysql
  - curl
  - json

  - gd
  - xml
  - mbstring
  - intl
  - readline
  - ldap (optional)
  - mcrypt (optional for importing older XML export files)
- Latest sysPass version https://github.com/nuxsmin/sysPass/releases

Installation

Debian GNU/Linux package installation.

```
apt install locales apache2 libapache2-mod-php7.0 php-pear php7.0 php7.0-cgi php7.0-
↪cli \
php7.0-common php7.0-fpm php7.0-gd php7.0-json php7.0-mysql php7.0-readline \
php7.0-curl php7.0-intl php7.0-ldap php7.0-mcrypt php7.0-xml php7.0-mbstring

service apache2 restart
```

Optional for enabling SSL.

In order to increase your sysPass instance security, please consider to use SSL. See *Security* and the following resources for Debian:

- Sites only accessible from LAN: https://doc.debian.org/configuration/Self-Signed_Certificate
- Sites accessible from Internet, you could use Let's Encrypt, see https://certbot.eff.org/

Directories and permissions

Create a directory for sysPass within the web server root.

```
mkdir /var/www/html/syspass
```

Unpack sysPass files.

```
cd /var/www/html/syspass
tar xzf syspass.tar.gz
```

Setup directories permissions. The owner should match the web server running user.

```
chown apache -R /var/www/html/syspass
chmod 750 /var/www/html/syspass/app/config /var/www/html/syspass/app/backup
```

Installing dependencies

From sysPass root directory, download and install Composer (https://getcomposer.org/download/)

```
cd /var/www/html/syspass
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
↪'93b54496392c062774670ac18b134c3b3a95e5a5e5c8f1a9f115f203b75bf9a129d5daa8ba6a13e2cc8a1da0806388a8
↪') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-
↪setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Then install sysPass dependencies

```
php composer.phar install --no-dev
```

Environment configuration

Please, point your web browser to the following URL and follow the installer steps

https://IP_OR_SERVER_ADDRESS/syspass/index.php

Note: More information about how sysPass works on *Application*

Warning: It's very advisable to take a look to security advices on *Security*

2.1.2 CentOS 7 Installation

Prerequisites

- Web server (Apache/Nginx/Lighttpd) with SSL enabled.
- MariaDB >= 10.1
- PHP >= 7.0
- **PHP modules**
  - mysqlnd
  - curl
  - json
  - gd
  - xml
  - mbstring
  - intl
  - readline
  - ldap (optional)
  - mcrypt (optional for importing older XML export files)
- Latest sysPass version https://github.com/nuxsmin/sysPass/releases

Installation

CentOS 7 package installation.

```
yum install httpd php-ldap php-mcrypt php-mbstring php-gd php-mysqlnd php-pdo php-
↪json php-xml php-ldap php-xml mariadb-server wget
```

Automated start/stop Apache web server and MariaDB server.

```
systemctl enable httpd.service
systemctl enable mariadb.service
systemctl start httpd.service
systemctl start mariadb.service
```

Setting up MariaDB.

```
/usr/bin/mysql_secure_installation
```

Enabling firewall ports.

```
firewall-cmd --permanent --zone=public --add-service=http
firewall-cmd --permanent --zone=public --add-service=https
firewall-cmd --reload
```

Optional for enabling SSL.

In order to increase your sysPass instance security, please consider to use SSL. See *Security* and the following resources for Debian:

- Sites only accessible from LAN: https://doc.debian.org/configuration/Self-Signed_Certificate
- Sites accessible from Internet, you could use Let's Encrypt, see https://certbot.eff.org/

Directories and permissions

Create a directory for sysPass within the web server root.

```
mkdir /var/www/html/syspass
```

Unpack sysPass files.

```
cd /var/www/html/syspass
tar xzf syspass.tar.gz
```

Setup directories permissions. The owner should match the web server running user.

```
chown apache -R /var/www/html/syspass
chmod 750 /var/www/html/syspass/app/config /var/www/html/syspass/app/backup
```

SELinux

sysPass needs to be allowed to write its configuration and some other files (backup, cache, temp, etc). We have 2 choices:

Note: Please, run only one of the choices

- Change SELinux's context and user:

```
setsebool -P httpd_can_connect_ldap 1
chcon -R -t httpd_sys_rw_content_t /var/www/html/syspass/app/{config,backup,cache,tmp}
```

- Disable SELinux by editing the file "/etc/sysconfig/selinux" and setting "SELINUX" variable's value to "permissive". You need to restart the system.

Installing dependencies

From sysPass root directory, download and install Composer (https://getcomposer.org/download/)

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
↪'93b54496392c062774670ac18b134c3b3a95e5a5e5c8f1a9f115f203b75bf9a129d5daa8ba6a13e2cc8a1da0806388a8
↪') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-
↪setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Then install sysPass dependencies

```
php composer.phar install --no-dev
```

Environment configuration

Please, point your web browser to the following URL and follow the installer steps

https://IP_OR_SERVER_ADDRESS/syspass/index.php

Note: Please, follow installer steps and after the successful finishing, you will be able to log into the application

To know how sysPass works, please see *Application*

Warning: It's advisable to read the security recommendations on *Security*

2.1.3 Docker Installation

Docker based installations allow to run the application in an isolated environment besides try out multiple versions without installing any package on the host system.

sysPass can be ran in Docker containers which has been compiled on top of latest Debian stable version (Stretch) and avoiding any **package compilation**.

Docker images can be got from Docker Hub and they are complied automatically from Docker source files on https://github.com/nuxsmin/docker-syspass

There are two ways for installing:

- Using Docker Compose (recommended): deploys a fully working sysPass environment including application and database services.
- Using Docker: deploys each service (application and database) separately.

Docker Compose

In order to deploy using this method, you need to issue the following steps:

1. Install Docker engine from https://docs.docker.com/install/
2. Install Docker Compose from https://docs.docker.com/compose/install/
3. Download "docker-compose.yml" sysPass' file from https://raw.githubusercontent.com/nuxsmin/docker-syspass/master/docker-compose.yml or use the following one:

```
version: '2'
services:
  app:
    container_name: syspass-app
    image: nuxsmin/docker-syspass:3.0
    restart: always
    ports:
      - "80"
      - "443"
    links:
      - db
    volumes:
      - syspass-config:/var/www/html/sysPass/app/config
      - syspass-backup:/var/www/html/sysPass/app/backup
  db:
    container_name: syspass-db
    restart: always
    image: mariadb:10.2
    environment:
      - MYSQL_ROOT_PASSWORD=syspass
    ports:
      - "3306"
    volumes:
      - syspass-db:/var/lib/mysql

volumes:
  syspass-config: {}
  syspass-backup: {}
  syspass-db: {}
```

4. Run "docker-compose" tool for setting up the environment:

```
docker-compose -p syspass -f docker-compose.yml up -d
```

This will download the latest sysPass stable image and the database (MariaDB) one.

5. Take a look to deployment's logs:

   docker-compose -p syspass -f docker-compose.yml logs -f

Note: Docker Compose will create an isolated network for all sysPass services making possible to use DNS resolution between containers. You can use "syspass-db" for setting up the database hostname in sysPass installation page.

It will create two fixed volumes for sysPass application, one for "config" directory and the other for "backup" directory. An additional fixed volume will be created for the database container's data.

Warning: sysPass container will publish 80 and 443 host's ports to the outside. You could change this behavior by tweaking the Docker Compose's file.

Docker

By this way all the services need to be deployed manually. The following steps are needed:

1. Install Docker engine from https://docs.docker.com/install/
2. Create network for sysPass services:

```
docker network create syspass-net
```

3. Create fixed volunes for sysPass services:

```
docker volume create syspass-app-config
docker volume create syspass-app-backup
docker volume create syspass-db-data
```

4. Setup sysPass database container:

```
docker run --name syspass-db \
--network syspass-net \
--restart unless-stopped \
--env MYSQL_ROOT_PASSWORD=syspass \
--volume syspass-db-data:/var/lib/mysql \
--detach mariadb:10.2
```

5. Setup sysPass application container:

```
docker run --name syspass-app \
--network syspass-net \
--publish 80:80 \
--restart unless-stopped \
--volume syspass-app-config:/var/www/html/sysPass/app/config \
--volume syspass-app-backup:/var/www/html/sysPass/app/backup \
--detach nuxsmin/docker-syspass:latest
```

6. Connection data will be displayed in application container's console:

```
docker logs -f syspass-app
```

Tip: You can install sysPass extensions (plugins) by setting "COMPOSER_EXTENSIONS" environment variable when deploying the sysPass application container. Example: "–env COMPOSER_EXTENSIONS='syspass/plugin-authenticator'"

Database Access

You can get access to the database using the following connection data:

- User: root
- Password: syspass

You may install other sysPass images from Docker Hub

Note: Please follow the installer steps in order to setup the sysPass application instance.

More information about how sysPass works on *Application*

Warning: It's very advisable to take a look to security advices on *Security*

2.1.4 Hosting Mode

The hosting mode is for those installations that are running on a external hosting, where is not possible to create neither database nor connection user for it.

Note: It won't create neither database (except tables) nor connection user

The steps to perform the installation are the following:

- Create an user/password for sysPass connection at the hosting panel.
- Create the sysPass database (not tables) and give permissions to the previous user on it.
- Start the sysPass installation and use the user/password that was previously created for sysPass (the two first fields in the installation page).
- Provide a MySQL/MariaDB user with administration rights (it could be the same as previous if it has enough permissions), in order to create sysPass database tables. This user is used only for the installation process and it often would be the user/password for the hosting management.
- If database connection and permissions are right, the installation should finish successfully.

Note: In case of errors, you could take a look to the web server error logs.

2.2 Configuration

2.2.1 LDAP Configuration

Active Directory

Tips

- Checks if connection user is member of group "Account Operators"

OpenLDAP

In order to setup an OpenLDAP server correctly, you can follow the article at https://wiki.debian.org/LDAP/OpenLDAPSetup which describes the steps to configure a fully operational server under a Debian like distribution.

In OpenLDAP, to the use the group membership feature you need to add an 'overlay' called 'membeof'. It's a module that adds an internal attribute to those users which belongs to a group.

These are the steps to configure that module:

- Create the file 'ldap_memberof_add.ldif' with this content:

```
dn: cn=module,cn=config
objectClass: olcModuleList
cn: module
olcModulePath: /usr/lib/ldap
olcModuleLoad: memberof
```

- Create the file 'ldap_memberof_config.ldif' with this content:

```
dn: olcOverlay=memberof,olcDatabase={1}hdb,cn=config
objectClass: olcMemberOf
objectClass: olcOverlayConfig
objectClass: olcConfig
objectClass: top
olcOverlay: memberof
olcMemberOfDangling: ignore
olcMemberOfRefInt: TRUE
olcMemberOfGroupOC: groupOfNames
olcMemberOfMemberAD: member
olcMemberOfMemberOfAD: memberOf
```

- Modify the LDAP configuration by running these commands:

```
ldapadd -D cn=admin,cn=config -w "password" -H ldapi:/// -f memberof_add.ldif
ldapadd -D cn=admin,cn=config -w "password" -H ldapi:/// -f memberof_config.ldif
```

Tips

- Check whether the sysPass 'admin' user is the same in OpenLDAP, you need to add this user to the LDAP group that have access permissions to sysPass.
- The username and email ofthe LDAP users are populated from 'displayname','fullname' and 'mail' attributes.
- You could use ldaps by setting a connection URI like 'ldaps:/ /my_ldap_server'.
- You could install phpLDAPadmin to create and manage the LDAP objects.

Links

- LDAP Debian Wiki: https://wiki.debian.org/LDAP/OpenLDAPSetup
- 'memberof' overlay config: http://www.cbjck.de/2012/05/enabling-the-memberof-overlay-for-openldap/

2.3 Application

sysPass is an application that uses a MySQL/MariaDB database to store the data of all its components except for the configuration, which is stored in an XML file within 'app/config' directory.

Warning: It's important that 'app/config' directory is not accessible from the web service, because it could reveal important information.

2.3.1 Encryption

Warning: If you already use a sysPass version **<= 2.0**, it's advisable to update to 2.1 version and then to 3.0, in order to use the new security improvements on the encryption mechanisms (CVE-2017-5999)

sysPass encryption is based on AES-256 in CTR mode by using PHP's OpenSSL module. It uses the Defuse/php-encryption library for the encryption modules and functions management.

The encrypted data (up to 3.0 version) are:

- Accounts' passwords (always)
- Accounts' public links (always)
- Custom fields' data (if set)
- sysPass XML format export (if set)
- PHP's session data (if set)

In order to use the application, for every user first login, either a master password or a temporary master key (see *Temporary Master Key*) will be needed. That is so because the master password is not stored in the web server but a a generated Blowfish hash is saved in order to check if the user is using the correct master password.

After logging in with the master password, it's encrypted and stored within the user's data in the database. The encryption key is generated using a derived key from user's password and login, and a secure random salt generated by openssl_random_pseudo_bytes (stored in "config.xml" file).

On next user logins the master password is got from the user's data and decrypted by using the derived key. After this, the master password is encrypted again for storing it in the user's PHP session, so every time the master password is needed it must be decrypted using a session-based generated key. This key is regenerated every 120 seconds.

The master password will be prompted again if:

- The user changes its login password. The previous password will be requested.
- It has been changed by the administrator.
- The user changes its login username.
- The configuration salt is changed.

Note: A temporary master key (see *Temporary Master Key*) could be used instead of the real master password

Temporary Master Key

A temporary master key could be generated to be used by the application users, so it won't be needed to tell the real master password.

For the temporary master key generation the real master password is encrypted using a secure key generated by openssl_random_pseudo_bytes. Then a Blowfish generated hash of it is stored in the database ""Config" table." in order to check it when the temporary master key is provided on login.

Note: The real master password is never stored unencrypted. For checking the temporary master key a Blowfish generated hash **is only used**

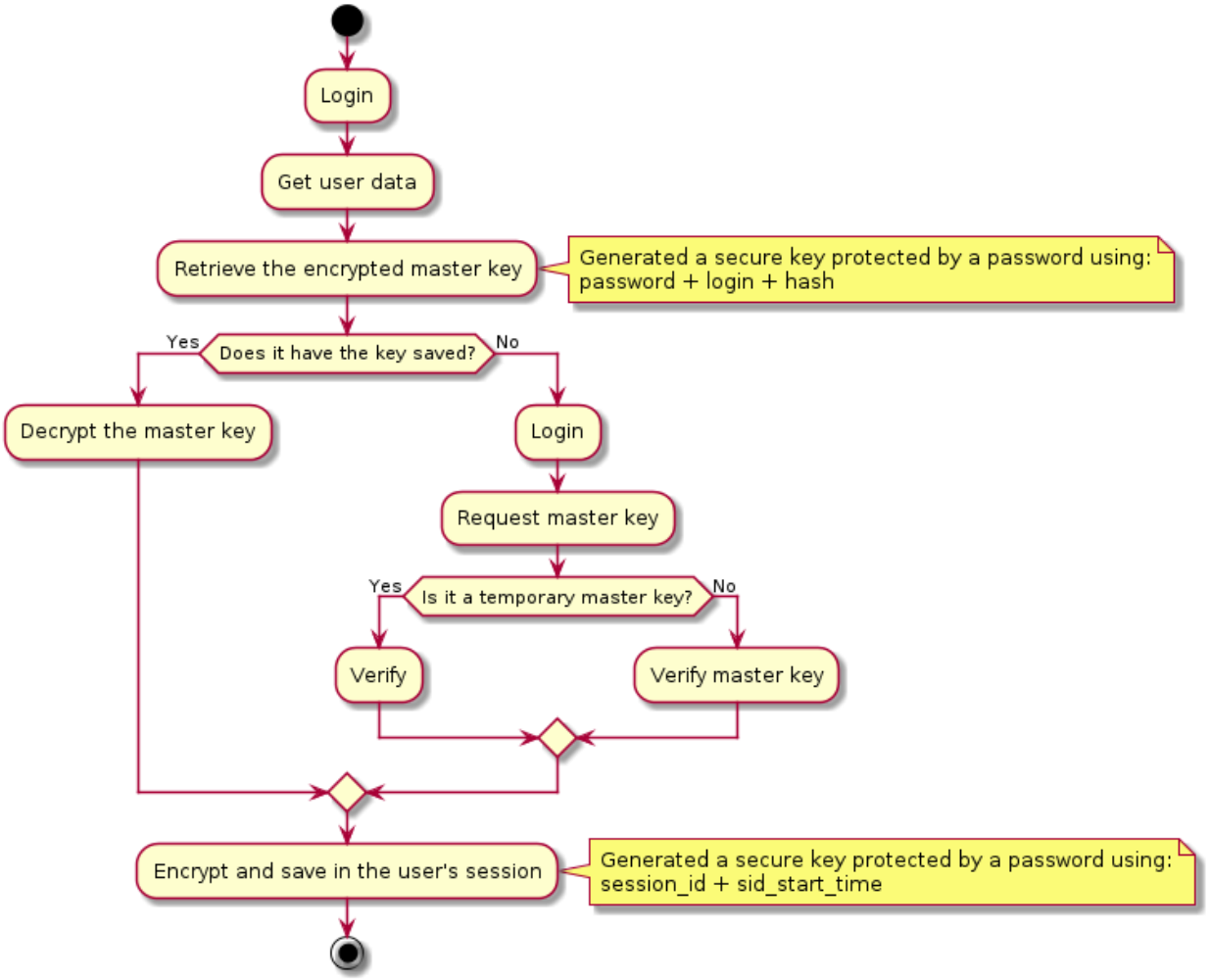
PKI-RSA

In order to improve the security of the sent data, RSA (PKI) is being used for encrypting the passwords that are being sent from the application forms. This prevents to send sensitive date through plain channels.

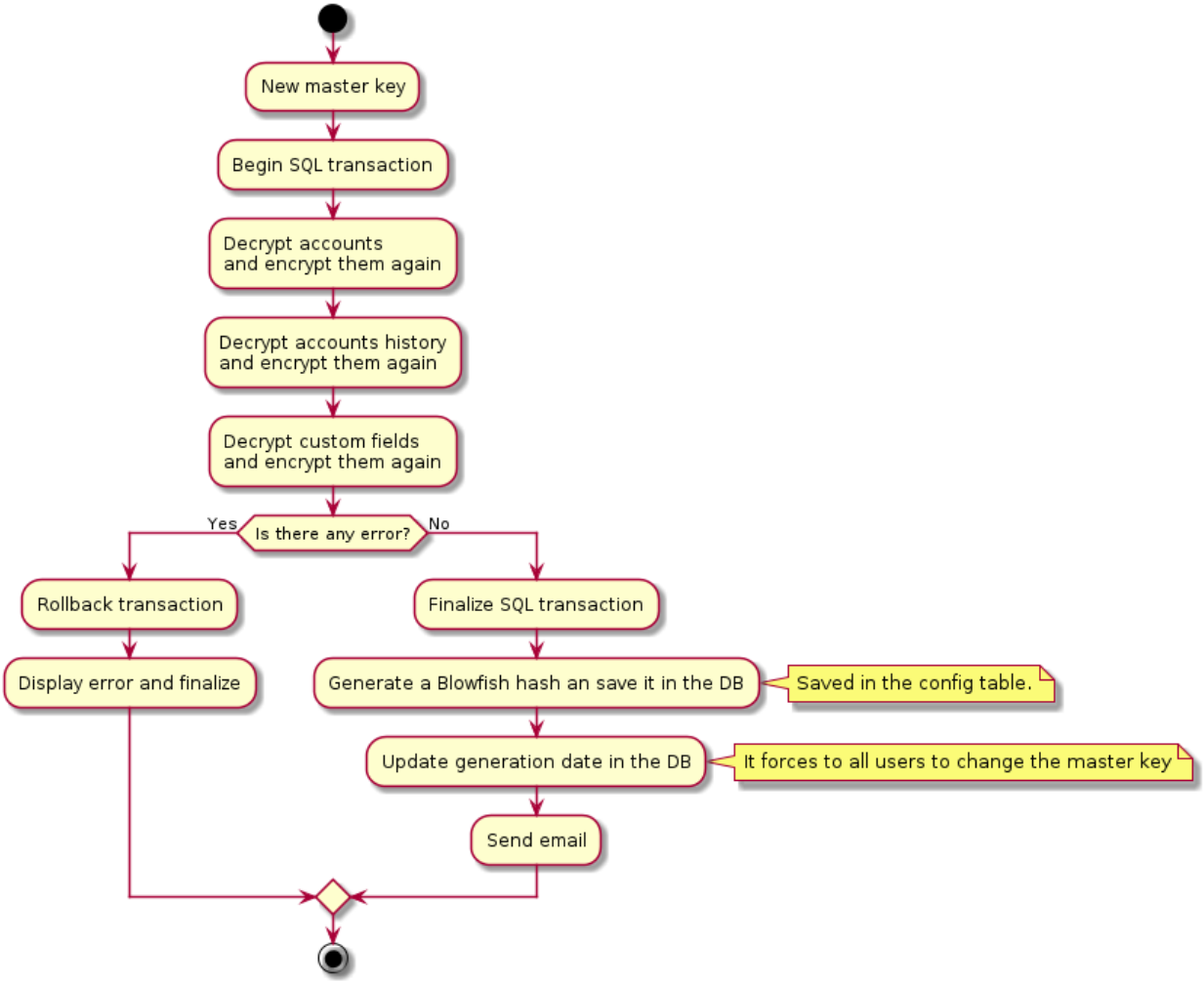
Public and private RSA keys are generated within the application "config" directory.

2.3.2 Diagrams

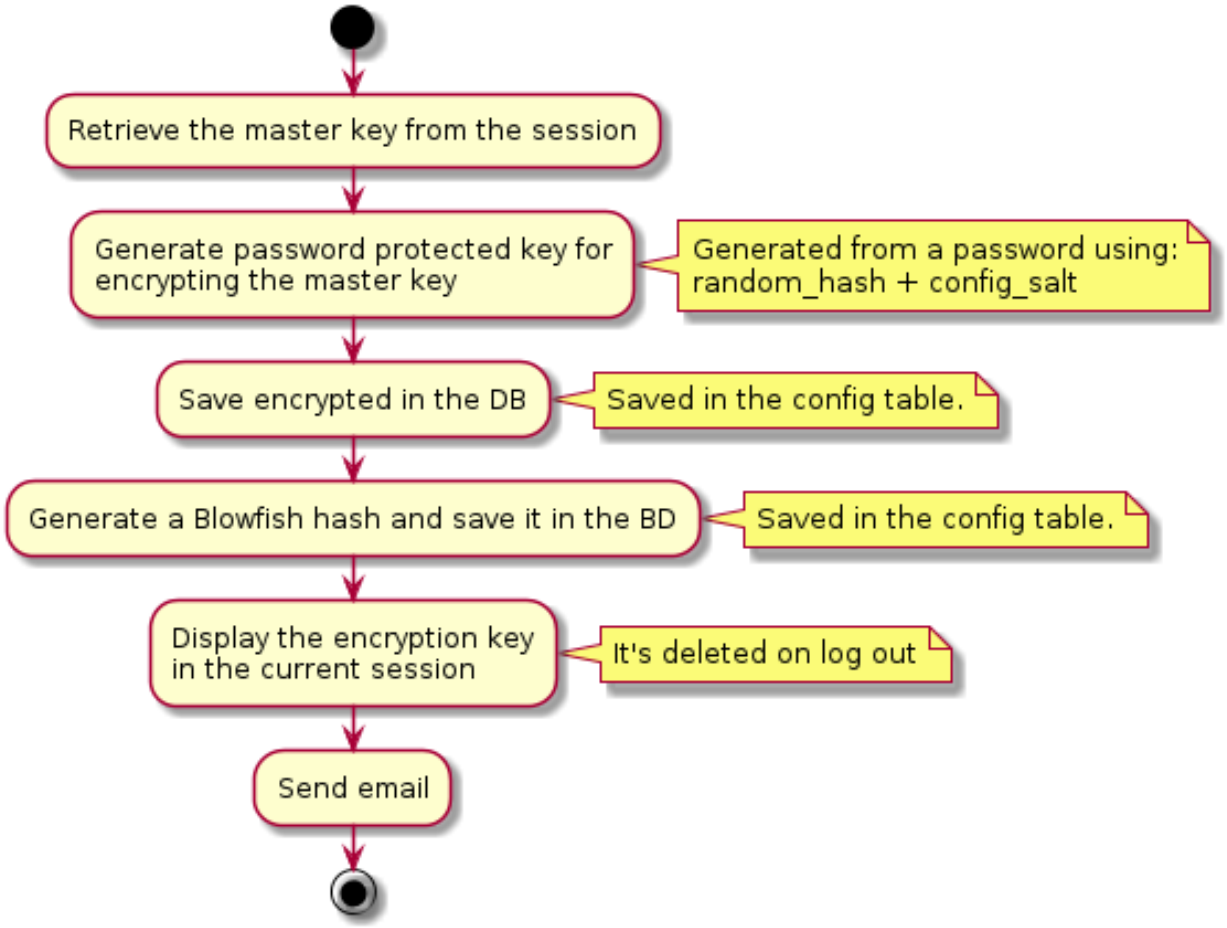
Login Process



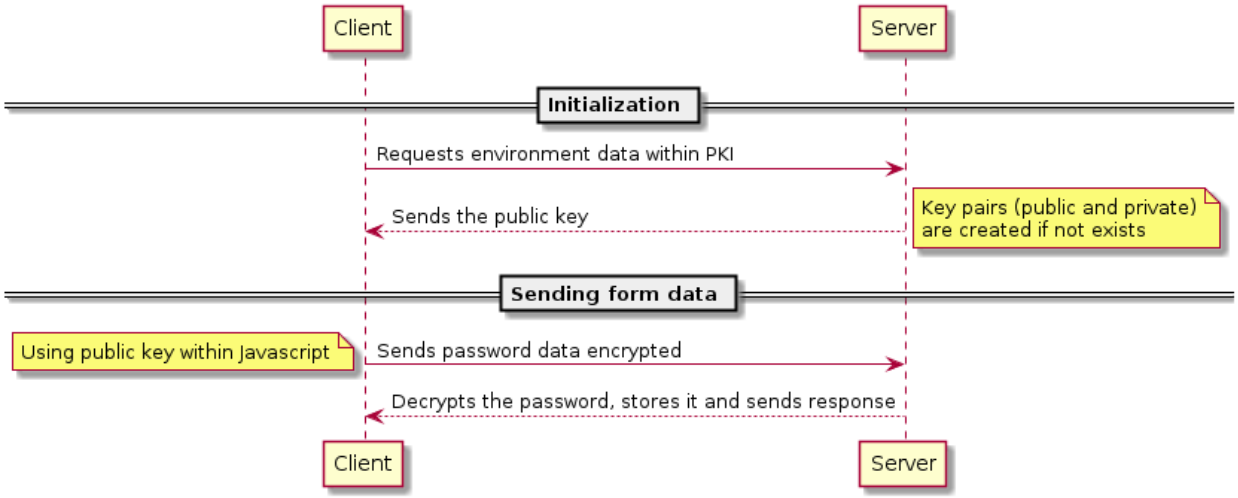
Master Password Process



Temporary Master Key Process



PKI Process



Warning: Be aware that the highest security risk is in the users themselves, because a compromised password could cause a security leak.

A sysPass compromised server could be dangerous if the database is placed alongside the web server, because the network data could be sniffed so the passwords would be revealed.

2.3.3 Security

sysPass has some security mechanisms to mitigate some kind of events and actions that could compromise the application security. Among them are:

- Security token generation for sending forms
- Removing of unwanted characters from received data
- Type casting of received data
- Hash generation for export and backup files name
- RSA (PKI) encryption is used for sending passwords within forms

Although these actions, it's needed to secure the web server components and communications by:

- Using HTTPS
- Limiting access to 'app/config' and 'app/backup' directories

In order to limit the access to the directories through Apache, '.htaccess' files could be used within the directories or by modifying the site configuration:

```
# Apache 2.4
<Directory "/var/www/html/sysPass">
  Options -Indexes -FollowSymLinks -Includes -ExecCGI
  <RequireAny>
      Require expr "%{REQUEST_URI} =~ m#.*/index\.php(\?r=)?#"
      Require expr "%{REQUEST_URI} =~ m#.*/api\.php$#"
      Require expr "%{REQUEST_URI} =~ m#^$#"
  </RequireAny>
</Directory>

<Directory "/var/www/html/sysPass/public">
  Require all granted
</Directory>
```

Danger: 'app/config' directory shouldn't be accessible through the web server, it could reveal private data.

2.3.4 Authentication

For sysPass authentication it could be possible to use several methods:

- MySQL/MariaDB database (by default)
- LDAP directory (OpenLDAP, eDirectory, Active Directory, freeIPA, etc)

Note: If LDAP option is enabled, the database authentication is used when the LDAP service is unavailable or the user doesn't exist.

For the database authentication, a generated Blowfish hash from user's password is checked, so the password is **never** stored.

If LDAP is enabled:

- The user's Blowfish generated hash is stored in order to check it, if the LDAP service is unavailable.
- Neither the user's login nor name nor email can be modified.

2.3.5 Authorization

For sysPass authorization it could be possible to use several methods:

- Auth Basic (by default)
- Two Factor 2FA (Authenticator Plugin)

The Auth Basic authorization could be enabled through the configuration module, so if the HTTP authorization header with the user's data is sent, it will be checked whether the sysPass user's login matches against the Auth Basic one.

The 2FA authorization, through the Authenticator Plugin, is done by generating an OTP token from Google Authenticator or similar applications. This authorization could be enabled from the user's preferences.

2.3.6 Permissions

sysPass permissions are set in users' profile. By default only accounts searching can be done.

There are 29 permission types:

- **Accounts**
  - Create - allows to create new accounts
  - View - allows to view the accounts' details[1]
  - View Password - allows to view the accounts' password[1]
  - Edit - allows to modify the accounts and its files[1]
  - Edit Password - allows to modify the accounts' password[1]
  - Delete - allows to delete accounts[1]
  - Files - allows to view account's files
  - Share Link - allows to create public links
  - Private - allows to create private accounts
  - Private for Group - allows to create private accounts only accessible by the account's main group
  - Permissions - allows to view and modify the accounts' permissions[1]
  - Global Search - allows to perform a searching in all the accounts except in the private ones[2]
- **Management**

[1] Only the accounts that the user and its group are granted
[2] When the account access is not granted, he/she will only be able to perform a 'Request for Account Modification'

  - Users - allows full access to the users management[3]
  - Groups - allows full access to the user groups management
  - Profiles - allows full access to the user profiles management
  - Categories - allows full access to categories management
  - Clients - allows full access to clients management
  - Custom Fields - allows full access to custom fields management
  - API Authorizations - allows full access to API authorizations management
  - Public Links - allows full access to the public links management
  - Accounts - allows full access to accounts management
  - Files- allows full access to files management
  - Tags - allows full access to the tags management
- **Configuration**
  - General - allows full access to the site, accounts, wiki, ldap and email configuration
  - Encryption - allows full access to the master password configuration
  - Backup - allows full access to perform backups[4]
  - Import - allows full access to import XML and CSV files
- **Others**
  - Event Log - allows full access to the event log

ACL

Users and Groups

- User profiles allow to set which actions could be done by the user
- An user can only display or modify accounts if:
  - Is the account's owner
  - Is member of account's primary group
  - Is member of account's secondary groups
  - Is listed as a secondary user of the account
  - His/Her main group is listed as a secondary group of the account
  - Is included through a group and the 'Secondary Groups Access' option is enabled
- Private accounts can only be accessed by the owner
- Private accounts for groups can only be accessed by the users of the main group
- Application Admin: allows full access to all the application modules and accounts, except private ones
- Accounts Admin: allows full access to all the accounts, except private ones

[3] 'Application Admin' users cannot be modified by other users
[4] Only 'Application Admin' users can download the backup or XML files

API

API's access permissions are complementary to the accounts access permissions, so users and groups ACLs will be applied when an account is either listed or accessed.

Notes

2.3.7 Accounts Searching

The accounts searching performs a query for the entered text within the fields 'name', 'login', 'url' and 'notes'.

Results filtering could be done by selecting category, client or tags.

The tag filtering is cumulative ('OR'), so it will be included all the accounts with selected tags.

There are special filters that could be entered in the text field. You could use either one or several special parameters separated by blank spaces:

Filter	Description
user:"login"	Get the accounts in which the user with login 'login' has access
owner:"login"	Get the accounts in which the user with login 'login' is the owner
group:"group_name"	Search for accounts which 'group_name' has access rights
maingroup:"group_name"	Get the accounts which have the main group with name 'group_name'
file:"file_name"	Search for accounts which contain a file with the name 'file_name'
client:"client_name"	Search for accounts by client name
category:"category_name"	Search for accounts by category name
id:"account_id"	Returns the account for the given ID
is|not:expired	Search for accounts with expired password
is|not:private	Get the private accounts for the current user
op:and|or	Operator used by special parameters

2.3.8 API

sysPass API relies on JSON-RPC v2 schema for client-server communication.

The API access URL is "https://server_name/api.php"

Example of JSON-RPC payload:

```
{
  "jsonrpc": "2.0",
  "method": "account/search",
  "params": {
    "authToken": "auth_token_for_api"
  },
  "id": 1
}
```

Methods

Accounts

account/search

Search for accounts

Parameter	Type	Required	Description
authToken	string	yes	User's API token
text	string	no	Text to search for
count	int	no	Number of results to display
categoryId	int	no	Category's Id for filtering
clientId	int	no	Client's Id for filtering
tagsId	array	no	Tags' Id for filtering
op	string	no	Operator used for filtering. It can be either 'or' or 'and'

account/view

Get account's details

Parameter	Type	Required	Description
authToken	string	yes	User's API token
tokenPass	string	yes	API token's pass
id	int	yes	Account's Id

account/viewPass

Get account's password

Parameter	Type	Required	Description
authToken	string	yes	User's API token
tokenPass	string	yes	API token's pass
id	int	yes	Account's Id
details	int	no	Whether to return account's details within response

account/editPass

Edit account's password

Parameter	Type	Required	Description
authToken	string	yes	User's API token
tokenPass	string	yes	API token's pass
id	int	yes	Account's Id
pass	string	yes	Account's password
expireDate	int	no	Expire date in UNIX timestamp format

account/create

Create account

Parameter	Type	Required	Description
authToken	string	yes	User's API token
tokenPass	string	yes	API token's pass
name	string	yes	Account's name
categoryId	int	yes	Account's category Id
clientId	int	yes	Account's client Id
pass	string	yes	Account's password
tagsId	array	no	Account's tags Id
userGroupId	int	no	Account's user group Id
parentId	int	no	Account's parent Id
login	string	no	Account's login
url	string	no	Account's access URL or IP
notes	string	no	Account's notes
private	int	no	Set account as private. It can be either 0 or 1
privateGroup	int	no	Set account as private for group. It can be either 0 or 1
expireDate	int	no	Expire date in UNIX timestamp format

account/edit

Edit account

Parameter	Type	Required	Description
authToken	string	yes	User's API token
tokenPass	string	yes	API token's pass
id	int	yes	Account's Id
name	string	no	Account's name
categoryId	int	no	Account's category Id
clientId	int	no	Account's client Id
tagsId	array	no	Account's tags Id
userGroupId	int	no	Account's user group Id
parentId	int	no	Account's parent Id
login	string	no	Account's login
url	string	no	Account's access URL or IP
notes	string	no	Account's notes
private	int	no	Set account as private. It can be either 0 or 1
privateGroup	int	no	Set account as private for group. It can be either 0 or 1
expireDate	int	no	Expire date in UNIX timestamp format

account/delete

Delete an account

Parameter	Type	Required	Description
authToken	string	yes	User's API token
id	int	yes	Account's Id

Categories

category/search

Search for categories

Parameter	Type	Required	Description
authToken	string	yes	User's API token
text	string	no	Text to search for
count	int	no	Number of results to display

category/view

Get category's details

Parameter	Type	Required	Description
authToken	string	yes	User's API token
tokenPass	string	yes	API token's pass
id	int	yes	Category's Id

category/create

Create category

Parameter	Type	Required	Description
authToken	string	yes	User's API token
name	string	yes	Category's name
description	string	no	Category's description

category/edit

Edit category

Parameter	Type	Required	Description
authToken	string	yes	User's API token
id	int	yes	Category's Id
name	string	yes	Category's name
description	string	no	Category's description

category/delete

Delete category

Parameter	Type	Required	Description
authToken	string	yes	User's API token
id	int	yes	Category's Id

Clients

client/search

Search for clients

Parameter	Type	Required	Description
authToken	string	yes	User's API token
text	string	no	Text to search for
count	int	no	Number of results to display

client/view

Get client's details

Parameter	Type	Required	Description
authToken	string	yes	User's API token
tokenPass	string	yes	API token's pass
id	int	yes	Client's Id

client/create

Create client

Parameter	Type	Required	Description
authToken	string	yes	User's API token
name	string	yes	Client's name
description	string	no	Client's description
global	int	no	Set client as global. It can be either 0 or 1

client/edit

Edit client

Parameter	Type	Required	Description
authToken	string	yes	User's API token
id	int	yes	Client's Id
name	string	yes	Client's name
description	string	no	Client's description
global	int	no	Set client as global. It can be either 0 or 1

client/delete

Delete client

Parameter	Type	Required	Description
authToken	string	yes	User's API token
id	int	yes	Client's Id

Tags

tag/search

Search for tags

Parameter	Type	Required	Description
authToken	string	yes	User's API token
text	string	no	Text to search for
count	int	no	Number of results to display

tag/view

Get tag's details

Parameter	Type	Required	Description
authToken	string	yes	User's API token
tokenPass	string	yes	API token's pass
id	int	yes	Tag's Id

tag/create

Create tag

Parameter	Type	Required	Description
authToken	string	yes	User's API token
name	string	yes	Tag's name

tag/edit

Edit tag

Parameter	Type	Required	Description
authToken	string	yes	User's API token
id	int	yes	Tag's Id
name	string	yes	Tag's name

tag/delete

Delete tag

Parameter	Type	Required	Description
authToken	string	yes	User's API token
id	int	yes	Tag's Id

User Groups

usergroup/search

Search for user groups

Parameter	Type	Required	Description
authToken	string	yes	User's API token
text	string	no	Text to search for
count	int	no	Number of results to display

usergroup/view

Get user group's details

Parameter	Type	Required	Description
authToken	string	yes	User's API token
tokenPass	string	yes	API token's pass
id	int	yes	User group's Id

usergroup/create

Create user group

Parameter	Type	Required	Description
authToken	string	yes	User's API token
name	string	yes	User group's name
description	string	no	User group's description
usersId	array	no	User group's users Id

usergroup/edit

Edit user group

Parameter	Type	Required	Description
authToken	string	yes	User's API token
id	int	yes	User group's Id
name	string	yes	User group's name
description	string	no	User group's description
usersId	array	no	User group's users Id

usergroup/delete

Delete user group

Parameter	Type	Required	Description
authToken	string	yes	User's API token
id	int	yes	User group's Id

Configuration

config/backup

Perform an application and database backup

Parameter	Type	Required	Description
authToken	string	yes	User's API token
path	string	no	Server path to store the application and database backup

config/export

Export application data in XML format

Parameter	Type	Required	Description
authToken	string	yes	User's API token
path	string	no	Server path to store the XML file
password	string	no	Password used to encrypt the exported data

2.3.9 Features

sysPass implements the following features:

- **Security**
  - Database authentication
  - LDAP directory authentication
  - Auth Basic authorization
  - Two Factor authorization (using Authenticator Plugin)
- **Permissions**
  - Module access control by profiles
  - Application administrator users
  - Accounts administrator users
  - Accounts user access control (read or write)
  - Accounts group access control (read or write)
- **Items**

  - Encrypted and unencrypted custom fields for accounts, clients, categories and users
  - Accounts public links access without user/password
  - Accounts expiry date configuration
  - Accounts' files management
  - Accounts' tags management
  - Clients management
  - Categories management
  - Public links management
  - API's authorizations management
  - Accounts management
  - Accounts' history management
  - Plugins management
  - Users management
  - User groups management
  - User profiles management
  - In-App notifications management
- **Configuration**
  - Language configuration
  - Visual theme configuration
  - Logging and audit configuration
  - Proxy configuration
  - Accounts configuration
  - Public links configuration
  - Wiki links configuration
  - LDAP configuration
  - Import users and groups from LDAP
  - Email notifications configuration
  - Master password change
  - Temporary master key generation
  - Application and database backups
  - XML format exporting using encryption or not
  - Importing from sysPass or KeePass XML formats and CSV format

2.3.10 Plugins

sysPass allows to use plugins through an architecture that implements observer pattern which is characterized by emitting a message to all subscribed observers.

Plugins must be installed in 'plugins' directory within the target module and they contain the following base structure:

```
plugins/
└── PluginName (1)
    ├── base.php
    ├── CODE_OF_CONDUCT.md
    ├── composer.json
    ├── LICENSE
    ├── README.md
    ├── src
    │   ├── lib
    │   │   ├── Controllers
    │   │   ├── Models
    │   │   ├── Plugin.php
    │   │   ├── Services
    │   │   └── Util
    │   ├── locales
    │   │   ├── en_US
    │   │   │   └── LC_MESSAGES
    │   │   │       ├── PluginName.mo (2)
    │   │   │       └── PluginName.po (2)
    │   ├── public
    │   │   ├── css
    │   │   │   ├── plugin.css
    │   │   │   ├── plugin.css.map
    │   │   │   ├── plugin.min.css
    │   │   │   └── plugin.scss
    │   │   └── js
    │   │       ├── plugin.js
    │   │       └── plugin.min.js
    │   └── themes
    │       └── material-blue
    │           └── views (3)
    │               ├── login
    │               │   └── index.inc
    │               └── userpreferences
    │                   └── preferences-security.inc
    └── version.json (4)
```

Directory and file names need to be set in the following way:

1. Directory name within the plugin name: Example: **Authenticator**
2. Filename within the plugin name in lowercase: Example: **authenticator.po**
3. View's name should match with the controller's name in MVC pattern. It could be overridden by setting the name of the view in the controller's code
4. 'version.json' file is used by JavaScript code for checking if the plugin is up-to-date.

Plugin (whithin 'Plugin.php' file) is the main class which will receive sysPass' events through the observer pattern. It must extends the abstract class 'SPPluginPluginBase' which is responsible to make the plugin's data available.

Methods

The following methods must be implemented in 'Plugin' class

init

Method that is called every time the plugin is executed

```
/**
 * Initialization
 */
public function init() {}
```

updateEvent

Method that is called when an event is emitted

```
/**
 * Update event
 *
 * @param string $event Event's name
 * @param mixed  $object
 */
public function updateEvent($event, $object) {}
```

getEvents

Method that returns an array of strings with the events that the plugin will be subscribed to

```
/**
 * Returns the events implemented by the observer
 *
 * @return array
 */
public function getEvents()
{
    return ['user.preferences', 'main.prelogin.2fa', 'login.preferences'];
}
```

getJsResources

Method that returns an array of strings with the Javascript resources required by the plugin

```
/**
 * Returns JS resources required by the plugin
 *
 * @return array
 */
public function getJsResources()
{
```

(continues on next page)

(continued from previous page)

```
    return ['plugin.min.js'];
}
```

getAuthor

Method that returns the plugin's author

```
/**
 * Returns the plugin's author
 *
 * @return string
 */
public function getAuthor()
{
    return 'Rubén D.';
}
```

getVersion

Method that returns an array of integers with the plugin's version

```
/**
 * Returns the plugin's version
 *
 * @return array
 */
public function getVersion()
{
    return [1, 0];
}
```

getCompatibleVersion

Method that returns an array of integers with the minimum sysPass compatible version

```
/**
 * Returns the minimum sysPass compatible version
 *
 * @return array
 */
public function getCompatibleVersion()
{
    return [2, 0];
}
```

getCssResources

Method that returns an array of strings with the CSS resources required by the plugin

```
/**
 * Returns the CSS resources required by the plugin
 *
 * @return array
 */
public function getCssResources()
{
    return [];
}
```

getName

Method that returns the plugin's name

```
/**
 * Returns the plugin's name
 *
 * @return string
 */
public function getName()
{
    return self::PLUGIN_NAME;
}
```

getData

Method that returns the plugin's data

```
/**
 * @return array|AuthenticatorData[]
 */
public function getData()
{
    return (array)parent::getData();
}
```

Example

```
namespace SP\Modules\Web\Plugins\Authenticator;

use Psr\Container\ContainerInterface;
use SP\Core\Context\ContextInterface;
use SP\Core\Events\Event;
use SP\Core\UI\ThemeInterface;
use SP\DataModel\PluginData;
use SP\Modules\Web\Plugins\Authenticator\Controllers\PreferencesController;
use SP\Modules\Web\Plugins\Authenticator\Models\AuthenticatorData;
use SP\Modules\Web\Plugins\Authenticator\Util\PluginContext;
use SP\Mvc\Controller\ExtensibleTabControllerInterface;
use SP\Plugin\PluginBase;
use SP\Util\Util;
use SplSubject;
```

(continues on next page)

(continued from previous page)

```
/**
 * Class Plugin
 *
 * @package SP\Modules\Web\Plugins\Authenticator
 */
class Plugin extends PluginBase
{
    const PLUGIN_NAME = 'Authenticator';
    const VERSION_URL = 'https://raw.githubusercontent.com/sysPass/plugin-
↪Authenticator/master/version.json';
    const RECOVERY_GRACE_TIME = 86400;
    /**
     * @var ContainerInterface
     */
    private $dic;

    /**
     * Receive update from subject
     *
     * @link  http://php.net/manual/en/splobserver.update.php
     *
     * @param SplSubject $subject <p>
     *                            The <b>SplSubject</b> notifying the observer of an␣
↪update.
     *                            </p>
     *
     * @return void
     * @since 5.1.0
     */
    public function update(SplSubject $subject)
    {
    }

    /**
     * Inicialización del plugin
     *
     * @param ContainerInterface $dic
     */
    public function init(ContainerInterface $dic)
    {
        if (!is_array($this->data)) {
            $this->data = [];
        }

        $this->base = dirname(__DIR__);
        $this->themeDir = $this->base . DIRECTORY_SEPARATOR . 'themes' . DIRECTORY_
↪SEPARATOR . $dic->get(ThemeInterface::class)->getThemeName();

        $this->setLocales();

        $this->dic = $dic;
    }

    /**
     * Evento de actualización
     *
```

(continues on next page)

(continued from previous page)

```
     * @param string $eventType Nombre del evento
     * @param Event  $event     Objeto del evento
     *
     * @throws \SP\Core\Exceptions\InvalidClassException
     * @throws \Exception
     */
    public function updateEvent($eventType, Event $event)
    {
        switch ($eventType) {
            case 'show.userSettings':
                /** @var ExtensibleTabControllerInterface $source */
                $source = $event->getSource(ExtensibleTabControllerInterface::class);

                (new PreferencesController($source, $this, $this->dic))
                    ->setUp();
                break;
            case 'login.finish':
                $this->checkLogin($event);
                break;
        }
    }

    /**
     * Comprobar 2FA en el login
     *
     * @param Event $event
     *
     * @throws \SP\Core\Context\ContextException
     */
    private function checkLogin(Event $event)
    {
        $session = $this->dic->get(ContextInterface::class);
        $pluginContext = $this->dic->get(PluginContext::class);

        $data = $this->getDataForId($session->getUserData()->getId());

        if ($data !== null && $data->isTwofaEnabled()) {
            $pluginContext->setTwoFApass(false);
            $session->setAuthCompleted(false);

            $eventData = $event->getEventMessage()->getExtra();

            if (isset($eventData['redirect'][0])
                && is_callable($eventData['redirect'][0])
            ) {
                $session->setTrasientKey('redirect', $eventData['redirect'][0](
↪'authenticatorLogin/index'));
            } else {
                $session->setTrasientKey('redirect', 'index.php?r=authenticatorLogin/
↪index');
            }
        } else {
            $pluginContext->setTwoFApass(true);
            $session->setAuthCompleted(true);
        }
    }
```

(continues on next page)

(continued from previous page)

```
    /**
     * Devolver los datos de un Id
     *
     * @param $id
     *
     * @return AuthenticatorData|null
     */
    public function getDataForId($id)
    {
        return isset($this->data[$id]) ? $this->data[$id] : null;
    }

    /**
     * @return array|AuthenticatorData[]
     */
    public function getData()
    {
        return (array)parent::getData();
    }

    /**
     * Devuelve los eventos que implementa el observador
     *
     * @return array
     */
    public function getEvents()
    {
        return ['show.userSettings', 'login.finish'];
    }

    /**
     * Devuelve los recursos JS y CSS necesarios para el plugin
     *
     * @return array
     */
    public function getJsResources()
    {
        return ['plugin.min.js'];
    }

    /**
     * Devuelve el autor del plugin
     *
     * @return string
     */
    public function getAuthor()
    {
        return 'Rubén D.';
    }

    /**
     * Devuelve la versión del plugin
     *
     * @return array
     */
    public function getVersion()
    {
```

(continues on next page)

(continued from previous page)

```
        return [2, 0, 1];
    }

    /**
     * Devuelve la versión compatible de sysPass
     *
     * @return array
     */
    public function getCompatibleVersion()
    {
        return [3, 0];
    }

    /**
     * Devuelve los recursos CSS necesarios para el plugin
     *
     * @return array
     */
    public function getCssResources()
    {
        return ['plugin.min.css'];
    }

    /**
     * Devuelve el nombre del plugin
     *
     * @return string
     */
    public function getName()
    {
        return self::PLUGIN_NAME;
    }

    /**
     * Establecer los datos de un Id
     *
     * @param                   $id
     * @param AuthenticatorData $AuthenticatorData
     *
     * @return Plugin
     */
    public function setDataForId($id, AuthenticatorData $AuthenticatorData)
    {
        $this->data[$id] = $AuthenticatorData;

        return $this;
    }

    /**
     * Eliminar los datos de un Id
     *
     * @param $id
     */
    public function deleteDataForId($id)
    {
        if (isset($this->data[$id])) {
            unset($this->data[$id]);
```

(continues on next page)

(continued from previous page)

```
        }
    }

    /**
     * @param mixed $pluginData
     */
    public function onLoadData(PluginData $pluginData)
    {
        $this->data = Util::unserialize(
            AuthenticatorData::class,
            $pluginData->getData()
        );
    }
}
```

Events

When an event is emitted the generating class instance is included as an argument, so it could be possible to access to the class events.

Events may include 'SPCoreEventsEventMessage' class which may contain additional data to pass into the plugin.

Currently, the generated events are the following:

Event	Class	Description
acl.deny		
check.notification		
check.tempMasterPassword		
clear.eventlog		
clear.track		
copy.account.pass		
create.account		
create.authToken		
create.category		
create.client		
create.customField		
create.itemPreset		
create.notification		
create.plugin		
create.publicLink		
create.publicLink.account		
create.tag		
create.tempMasterPassword		
create.user		
create.userGroup		
create.userProfile		
database.query		
database.rollback		
database.transaction.begin		
database.transaction.end		
database.transaction.rollback		

Continued on next page

Table 1 – continued from previous page

Event	Class	Description
delete.account		
delete.account.selection		
delete.accountFile		
delete.accountFile.selection		
delete.accountHistory		
delete.accountHistory.selection		
delete.authToken		
delete.authToken.selection		
delete.category		
delete.client		
delete.client.selection		
delete.customField		
delete.customField.selection		
delete.itemPreset		
delete.notification		
delete.notification.selection		
delete.plugin		
delete.plugin.selection		
delete.publicLink		
delete.publicLink.selection		
delete.tag		
delete.tag.selection		
delete.user		
delete.user.selection		
delete.userGroup		
delete.userGroup.selection		
delete.userProfile		
delete.userProfile.selection		
download.accountFile		
download.backupAppFile		
download.backupDbFile		
download.configBackupFile		
download.exportFile		
download.logFile		
edit.account		
edit.account.bulk		
edit.account.pass		
edit.account.restore		
edit.authToken		
edit.category		
edit.client		
edit.customField		
edit.itemPreset		
edit.notification		
edit.plugin.available		
edit.plugin.disable		
edit.plugin.enable		
edit.plugin.reset		
edit.plugin.unavailable		

Continued on next page

Table 1 – continued from previous page

Event	Class	Description
edit.publicLink.refresh		
edit.tag		
edit.user		
edit.user.pass		
edit.user.password		
edit.userGroup		
edit.userProfile		
expire.tempMasterPassword		
import.ldap.end		
import.ldap.groups		
import.ldap.start		
import.ldap.users		
ldap.bind		
ldap.check.connection		
ldap.check.group		
ldap.check.params		
ldap.connect		
ldap.connect.tls		
ldap.getAttributes		
ldap.search		
ldap.search.group		
ldap.unbind		
list.accountFile		
login.auth.browser		
login.auth.database		
login.auth.ldap		
login.checkUser.changePass		
login.checkUser.disabled		
login.finish		
login.info		
login.masterPass		
login.masterPass.temporary		
login.preferences.load		
login.session.load		
plugin.load		
plugin.load.error		
refresh.authToken		
refresh.masterPassword		
refresh.masterPassword.hash		
request.account		
request.user.passReset		
reset.min.css		
restore.accountHistory		
run.backup.end		
run.backup.process		
run.backup.start		
run.export.end		
run.export.start		
run.export.verify		

Continued on next page

Table 1 – continued from previous page

Event	Class	Description
run.import.csv		
run.import.end		
run.import.keepass		
run.import.start		
run.import.syspass		
save.config.account		
save.config.dokuwiki		
save.config.general		
save.config.ldap		
save.config.mail		
save.config.wiki		
search.category		
search.client		
search.tag		
search.userGroup		
send.mail		
send.mail.check		
session.cookie_httponly		
session.gc_maxlifetime		
session.save_handler		
session.timeout		
show.account		
show.account.bulkEdit		
show.account.copy		
show.account.create		
show.account.delete		
show.account.edit		
show.account.editpass		
show.account.history		
show.account.link		
show.account.pass		
show.account.request		
show.account.search		
show.accountFile		
show.authToken		
show.authToken.create		
show.authToken.edit		
show.category		
show.category.create		
show.category.edit		
show.client		
show.client.create		
show.client.edit		
show.config		
show.customField		
show.customField.create		
show.customField.edit		
show.itemPreset		
show.itemPreset.create		

Continued on next page

Table 1 – continued from previous page

Event	Class	Description
show.itemPreset.edit		
show.itemlist.accesses		
show.itemlist.items		
show.itemlist.security		
show.notification		
show.notification.create		
show.notification.edit		
show.plugin		
show.publicLink		
show.publicLink.create		
show.publicLink.edit		
show.tag		
show.tag.create		
show.tag.edit		
show.user		
show.user.create		
show.user.edit		
show.user.editPass		
show.userGroup		
show.userGroup.create		
show.userGroup.edit		
show.userProfile		
show.userProfile.create		
show.userProfile.edit		
show.userSettings		
track.add		
track.delay		
unlock.track		
update.masterPassword.customFields		
update.masterPassword.end		
update.masterPassword.hash		
update.masterPassword.start		
upgrade.app.end		
upgrade.app.start		
upgrade.authToken.end		
upgrade.authToken.process		
upgrade.authToken.start		
upgrade.config.end		
upgrade.config.process		
upgrade.config.start		
upgrade.customField.end		
upgrade.customField.process		
upgrade.customField.start		
upgrade.db.end		
upgrade.db.process		
upgrade.db.start		
upgrade.publicLink.end		
upgrade.publicLink.process		
upgrade.publicLink.start		

Continued on next page

Table 1 – continued from previous page

Event	Class	Description
upload.accountFile		
wiki.aclCheck		
wiki.getPage		
wiki.getPageHTML		
wiki.getPageInfo		

2.3.11 Backup Strategies

Note: Work in progress

Docker

Please perform backups regularly by using in-app tools or external ones (recommended). You need to copy the following data:

- "syspass-app-config" volume
- "syspass-app-backup" volume
- sysPass database

Example:

```
docker run --rm \
--volumes-from syspass-app \
--volume $PWD:/backup \
alpine sh -c "exec tar xzf /backup/syspass-app-backup.tar.gz /var/www/html/sysPass"

docker run --rm \
--network syspass-net \
--volume $PWD:/backup \
mariadb:10.2 sh -c 'exec mysqldump -h syspass-db -u root -p"syspass" syspass > /
↪backup/syspass-db-dump.sql'
```

These commands will create "syspass-app-backup.tar.gz" and "syspass-db-dump.sql" files within the current directory

2.4 Updating

2.4.1 General

For the sysPass updating the following steps are needed:

1. Download the application from https://github.com/nuxsmin/sysPass/releases and uncompress the files
2. Set the sysPass directory owner and permissions
3. Copy the files ("config.xml", "key.pem" y "pubkey.pem") within the "config" directory from the current version to the new one
4. Open the application from a web browser

If the application requires a database upgrade:

1. **Perform a database backup**
2. Enter the updating code which could be found in the "config/config.xml" file within the tag "upgradeKey"

Note: After the updating, it will show a message and you could take a look to the updating details in the event log

2.4.2 2.1 Version

This version includes some improvements on the sysPass security by the following features:

- It uses Defuse/php-encryption library for the data encryption with OpenSSL by using AES-256 CTR (CVE-2017-5999)
- Improvements on the session keys security
- API authorizations password
- Improvements on the public links security
- Failed log in attempts detection. A delay is set after several attempts

This upgrade requires to re-encrypt all the accounts and encrypted data, so the master password and a valid user login (for registering changes) will be needed.

Though it's a safe process, it's advisable to make a full sysPass backup.

Important Changes

Because the encryption data changes, the following items need to be regenerated:

- Public links: the links are now an snapshot of the linked account, so if the account is updated, the link needs to be renewed.
- API authorizations: As of this version, a password is needed for those authorizations that require encrypted data.
- Temporary master password: it needs to be regenerated if it's being used.

Process

For the sysPass updating the following steps are needed:

1. Download the application from https://github.com/nuxsmin/sysPass/releases and uncompress the files
2. Set the sysPass directory owner and permissions
3. Copy the files ("config.xml", "key.pem" y "pubkey.pem") within the "config" directory from the current version to the new one
4. Open the application from a web browser

If the application requires a database upgrade:

1. **Perform a database backup**
2. Enter the updating code which could be found in the "config/config.xml" file within the tag "upgradeKey"
3. Please, enter the sysPass master password.
4. Please, enter a valid user login

Note: During the upgrade, it will display the encryption tasks processes.

Note: After the updating, it will show a message and you could take a look to the updating details in the event log

2.4.3 3.0 Version

This version only can be updated from v2.1

Important Changes

- This version performs a fully database structure change, so **it's very important to make a full database backup using external tools like "mysqldump"**
- "config" directory is moved off to "/app/config"
- Composer PHP package manager is used to install and keep up-to-date sysPass dependencies

Process

The following steps need to be performed in order to update sysPass:

1. Download or clone sysPass repository from either https://github.com/nuxsmin/sysPass/releases or https://github.com/nuxsmin/sysPass.git
2. Set user and group permissions on sysPass directory
3. Copy "config.xml", "key.pem" y "pubkey.pem" from the old "config" directory to "/app/config" directory
4. From sysPass root directory, download and install Composer: https://getcomposer.org/download/
5. Install dependencies

```
$ php composer.phar install --no-dev
```

6. Set up the correct permissions on directories. Please note that "config" and "backup" directories are now within "/app"
7. Point your browser to sysPass web server URL
8. **Perform a full database backup using external tools like "mysqldump"**
9. Enter the upgrade key located in "app/config/config.xml" file within the "upgradeKey" tag

2.5 HOWTOs

2.5.1 How to test a sysPass update

Note: This procedure tells the steps to follow to try out a sysPass update without modifying the current installation

1. Make a database backup. It could be made either through the sysPass utility, MySQL workbench or mysqldump tool
2. Create a new database (eg. syspass21)
3. Create an user (eg. sp_admin21) and set the permissions over the newly created database
4. Import the backup in the newly created database. You could use the above tools
5. Create a new directory and unpack the new sysPass version package[1]
6. Copy all files within the "config" directory to the new path and check out the permissions[1]
7. Modify the "config/config.xml" file to set the correct database connection parameters ("dbname", "dbuser" and "dbpass"). Please check out that "dbHost" is correct
8. Point the browser to the application URL and follow the steps for upgrading

Notes

2.5.2 How to restore sysPass

Note: This procedure requires to have a database and application backup

1. Restore the database backup. It could be made either through the sysPass utility, MySQL workbench or mysqldump tool
2. Create the connection user (see 'config.xml' file) and set the correct permissions on the restored database
3. Restore the application backup
4. oint the browser to the application URL

2.6 Frequently Asked Questions

2.6.1 What is sysPass?

sysPass is a password manager that allows to save passwords using bidirectional encryption with a master password to a database. Passwords are associated to accounts, and these have detailed information about it like: customer, category, notes, files, etc.

The initial idea was to make servers and services passwords accesible in a multiuser environment with security applied and make a portable bundle to store on a flash drive.

2.6.2 Where can I install sysPass?

The application can be installed on any system that has Apache, PHP and MySQL installed.

2.6.3 How do I install sysPass

You can download the application from https://github.com/nuxsmin/sysPass/releases/latest and follow steps on *Installation*

[1] See *Installation* for more details

2.6.4 Which authentication methods are used?

sysPass uses MySQL/MariaDB or LDAP as authentication backends.

If LDAP is used and it is for some reason not possible to connect to the configured LDAP server, it will use MySQL as backend. In this case, user login data will be the last used on user login by LDAP.

More information on: *Authentication*

2.6.5 What is the encryption for?

The database passwords encryption allows that in case of anyone get access to the database or a data exporting is performed, it won't be readable without the master key.

This solution is very convenient when you run the application from a flash drive, because if you lose it, the information is secured.

The encryption schema used is rijndael-256 in CBC mode.

More information on: *Encryption*

2.6.6 What is portable?

It means that you can run the application without really installing it.

This application can be portable by installing Apache, PHP and MySQL on a flash drive. You can use any available LAMP bundles like WAMP, XAMPP, etc.

The backup tool allows you to make a backup of whole the environment (application and database) for example to store it on a flash drive or put it somewhere safe as a backup.

2.6.7 Is there a master password for each account/user?

The master password is global for all accounts and users.

Each time a user is added, his personal password is changed or the master password is reset, the user needs to enter the master password on the next session login.

Each time the master password is changed, the users that are logged in, will only be able to view accounts details, until the new password is entered.

More information on: *Encryption*

2.6.8 What are Wiki links?

It allows you to link the accounts with a name pattern to an external Wiki that allow to pass the account name as a parameter in the URL.

There are two types of links, the one that links to a Wiki search page (and in which the account name is passed as a parameter), and the other that links to the account page in the Wiki.

2.6.9 What are categories?

Its goal is to classify the accounts to make more precise searches.

2.6.10 What are user groups?

These groups are used to give users access to accounts that have a certain group set as primary or secondary group

2.6.11 What is customer field?

Like categories, it is possible to do searches based on the customer. This field can be treated generically as department, company, division, etc..

En futuras versiones se podrán asociar usuarios a clientes.

2.6.12 Is there an account history?

Yes, each time an account is modified or deleted, the application saves a copy of the last state.

You can switch to a history point at account details page. If the master password that was used to save account history point differs from current, the password won't be shown.

2.6.13 What are profiles?

Profiles are used to define actions that the users can do.

There are 16 access levels that can be activated and it allows to define which modules can be accessed by the users in which are defined.

2.6.14 What is maintenance mode?

This mode is used to disable the users to log in to the application while you are doing operations on database, updating, etc.

The user that enables the maintenance mode, will be the only one that can use the application until a session log out. After that it will be needed to disable it in the "config/config.xml" file within the tag "maintenance"

2.6.15 Can I change Master Password?

Yes, you need to know the current one. It's advisable to make a database backup before this process.

2.6.16 I don't remember Master Password, can I decrypt the passwords?

No, it's not possible view the passwords without the Master Password.

2.6.17 Does backup runs on Windows?

Yes, it uses the PHP PHAR library to get it working.

2.6.18 The language doesn't change

Please take a look to the locales installed on your system (server), because sysPass uses the GNU gettext system for internationalization.

The installed locales should be on the UTF-8 variant.