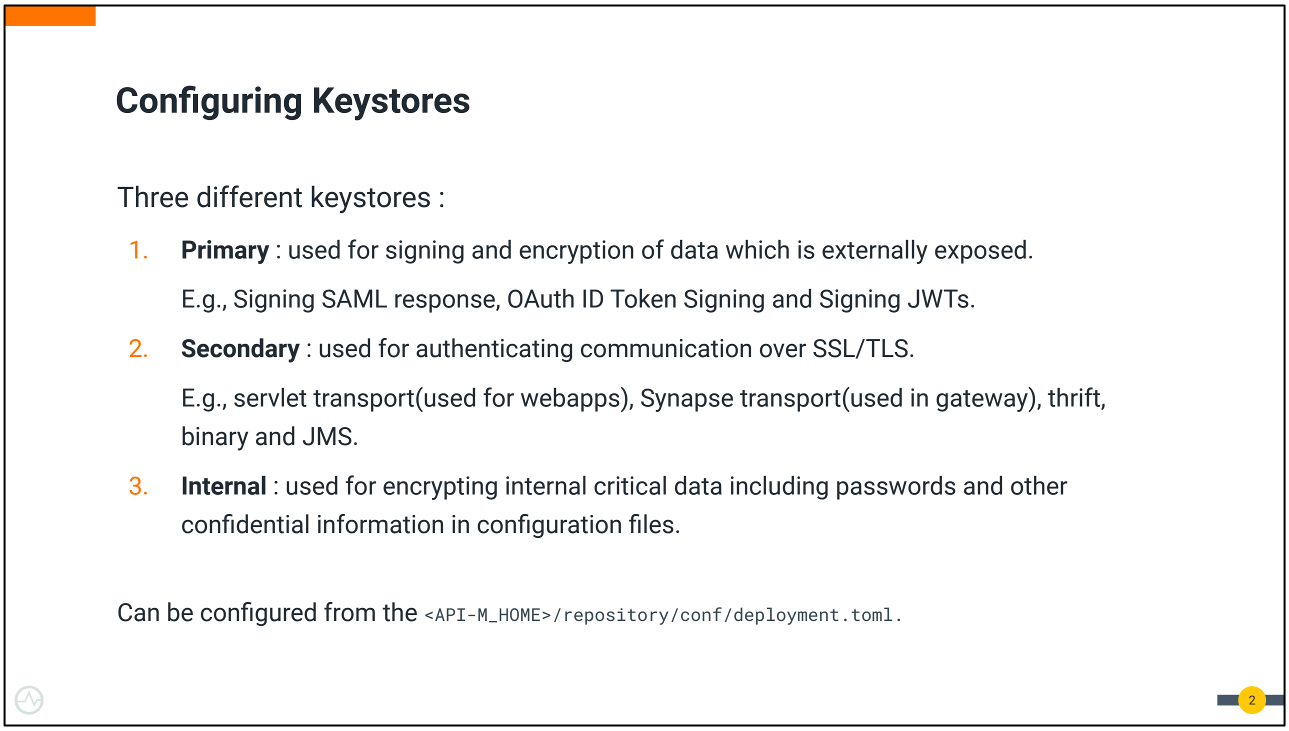
WSO2 API Manager 3.2.0 Developer Fundamentals

Product Security

WSO2 Training



Configuring Keystores

Three different keystores :

1. **Primary** : used for signing and encryption of data which is externally exposed.

   E.g., Signing SAML response, OAuth ID Token Signing and Signing JWTs.

2. **Secondary** : used for authenticating communication over SSL/TLS.

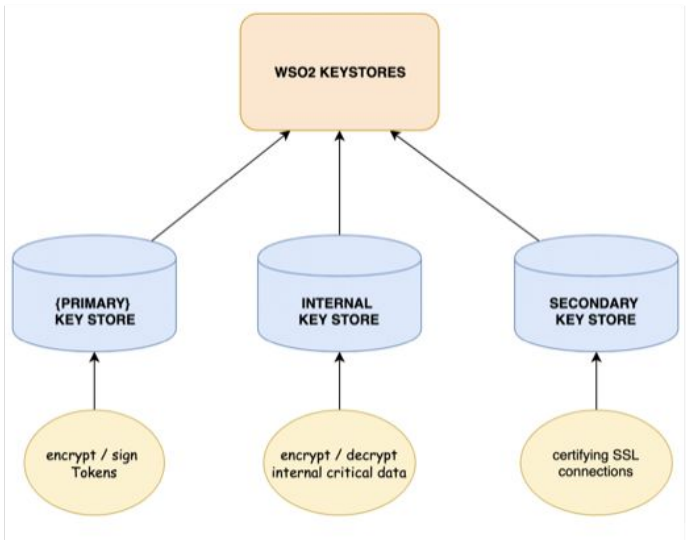
   E.g., servlet transport(used for webapps), Synapse transport(used in gateway), thrift, binary and JMS.

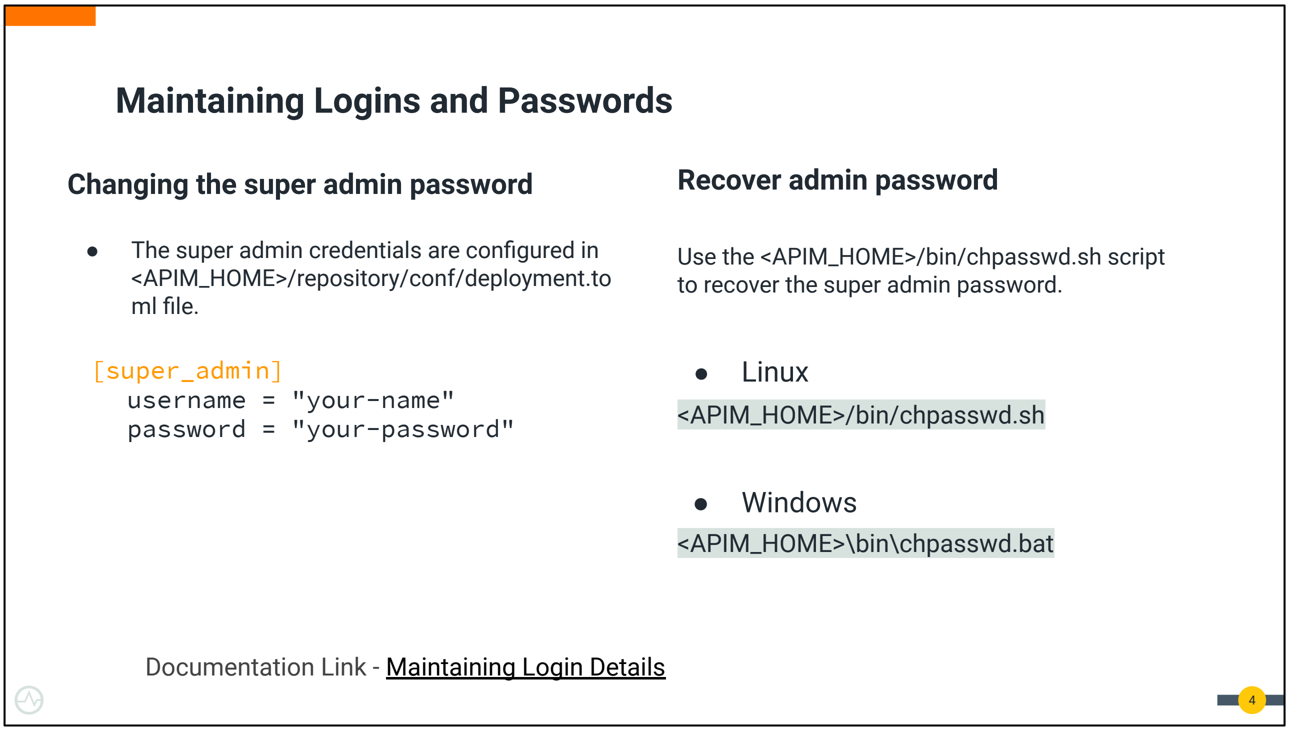
3. **Internal** : used for encrypting internal critical data including passwords and other confidential information in configuration files.

Can be configured from the `<API-M_HOME>/repository/conf/deployment.toml.`

WSO2 products use asymmetric cryptography by default for the purposes of authentication and data encryption.
In asymmetric cryptography, keystores (with key pairs and certificates) are created and stored for the product.
Keystore is a repository where private keys and certificates can be stored.
Possible to have multiple keystores so that the keys used for different use cases are kept unique.

Configuring Keystores





Maintaining Logins and Passwords

Changing the super admin password

- The super admin credentials are configured in <APIM_HOME>/repository/conf/deployment.toml file.

```
[super_admin]
  username = "your-name"
  password = "your-password"
```

Recover admin password

Use the <APIM_HOME>/bin/chpasswd.sh script to recover the super admin password.

- Linux

`<APIM_HOME>/bin/chpasswd.sh`

- Windows

`<APIM_HOME>\bin\chpasswd.bat`

Documentation Link - Maintaining Login Details

Important
In the first start up, the super admin is created in the configured user store. After that, during the login, password validation is done from the user store.

To change the password afterwards, it should be done through the API Manager management console.

- Log in to management console as the super admin (https://localhost:9443/carbon),
- Go to **Users and Roles** > **List** > **Users**
- Click on the **Change Password** button in the Actions section of admin user.
- Enter the new password and click Save.

Securing Passwords

Secure-Vault

- Store encrypted passwords and map them with aliases, which are used in configuration files instead of the actual passwords.
- These encrypted passwords will be decrypted and resolved during runtime only.

For example, if the admin user password is admin, you can define an alias (such as admin_password) and map that particular alias to the actual password (admin). At runtime, the product will look up this alias in the secure vault, decrypt it, and use the mapped password.

Encrypting passwords in configuration files

Working with encrypted passwords

- Encrypting passwords in product configuration files
- Encrypting secured endpoint passwords
- Encrypting passwords for mediation flow
- Using encrypted passwords in mediation flow

Link - Working with Encrypted Passwords

Encrypting Passwords with Cipher Tool

The following methods enable you to encrypt passwords by changing the configuration files :

1. Encrypting passwords using the automated process.
2. Encrypting passwords using the manual process.
3. Changing already encrypted passwords.
4. Resolving already encrypted passwords.

Link - Encrypting Passwords with Cipher Tool

These files are stored in the <API-M_HOME>/conf/security directory

Encrypting passwords using the automated process

- This automated process can be used for passwords that are used in configuration files.
- Add a [secrets] section in the deployment.toml file with the passwords and aliases that needs to be encrypted.
- Replace the actual password usages with the alias.
- Run the ciphertool to encrypt the passwords.
  - On Linux: `./ciphertool.sh -Dconfigure`
  - On Windows: `./ciphertool.bat -Dconfigure`

Encrypting passwords using the automated process

Secret Definition

```
Format
alias = “[password]”

[secrets]
admin_password = "[admin]"
keystore_password = "[wso2carbon]"
key_password = "[wso2carbon]"
truststrore_password = "[wso2carbon]"
```

Replace Actual Password with Alias

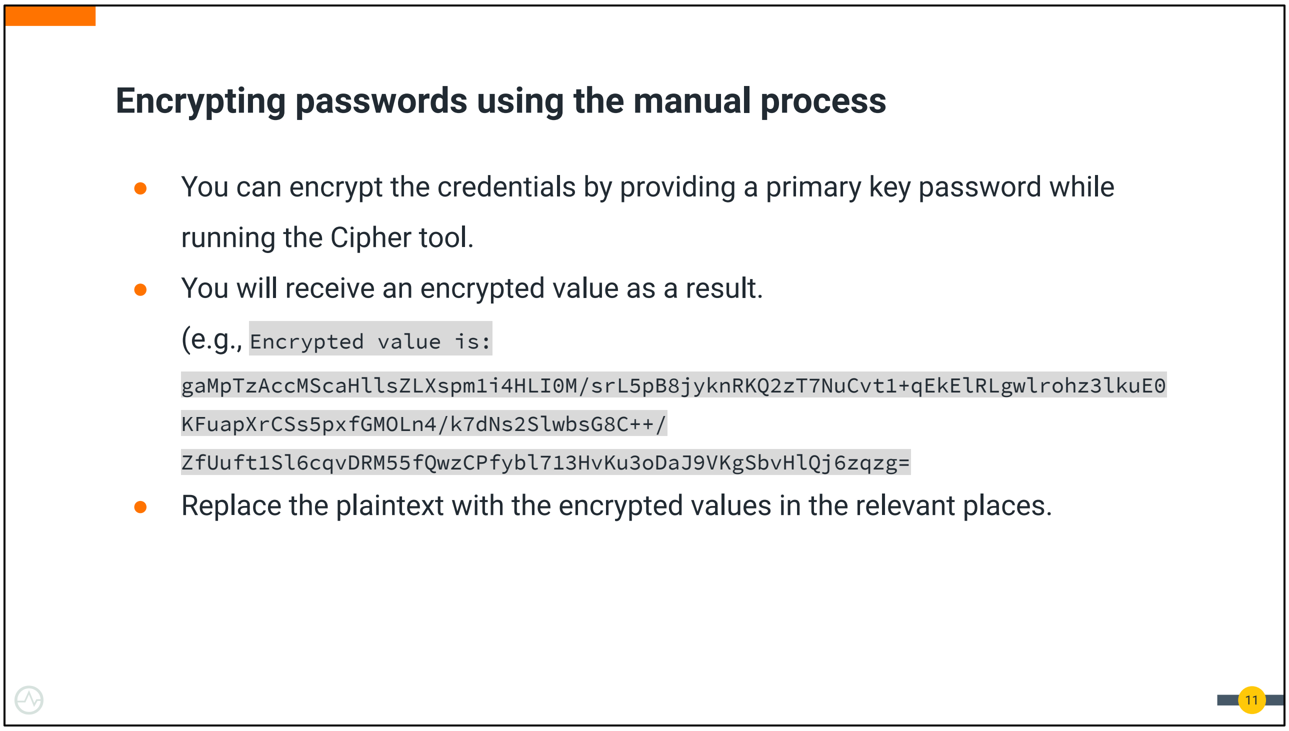
```
[super_admin]
username="admin"
password="$secret{admin_password}"

[keystore.tls]
password = "$secret{keystore_password}"
key_password = "$secret{key_password}"

[truststore]
password = "$secret{truststrore_password}"
```

Encrypting passwords using the manual process

- This manual process can be used for encrypting any password in a configuration file.
- The **log4j2.properties file** and **jndi.properties** file are two such files which require the manual password encryption process.



Encrypting passwords using the manual process

- You can encrypt the credentials by providing a primary key password while running the Cipher tool.
- You will receive an encrypted value as a result.
  (e.g., `Encrypted value is:`
  ```
  gaMpTzAccMScaHllsZLXspm1i4HLI0M/srL5pB8jyknRKQ2zT7NuCvt1+qEkElRLgwlrohz3lkuE0
  KFuapXrCSs5pxfGMOLn4/k7dNs2SlwbsG8C++/
  ZfUuft1Sl6cqvDRM55fQwzCPfybl713HvKu3oDaJ9VKgSbvHlQj6zqzg=
  ```
- Replace the plaintext with the encrypted values in the relevant places.

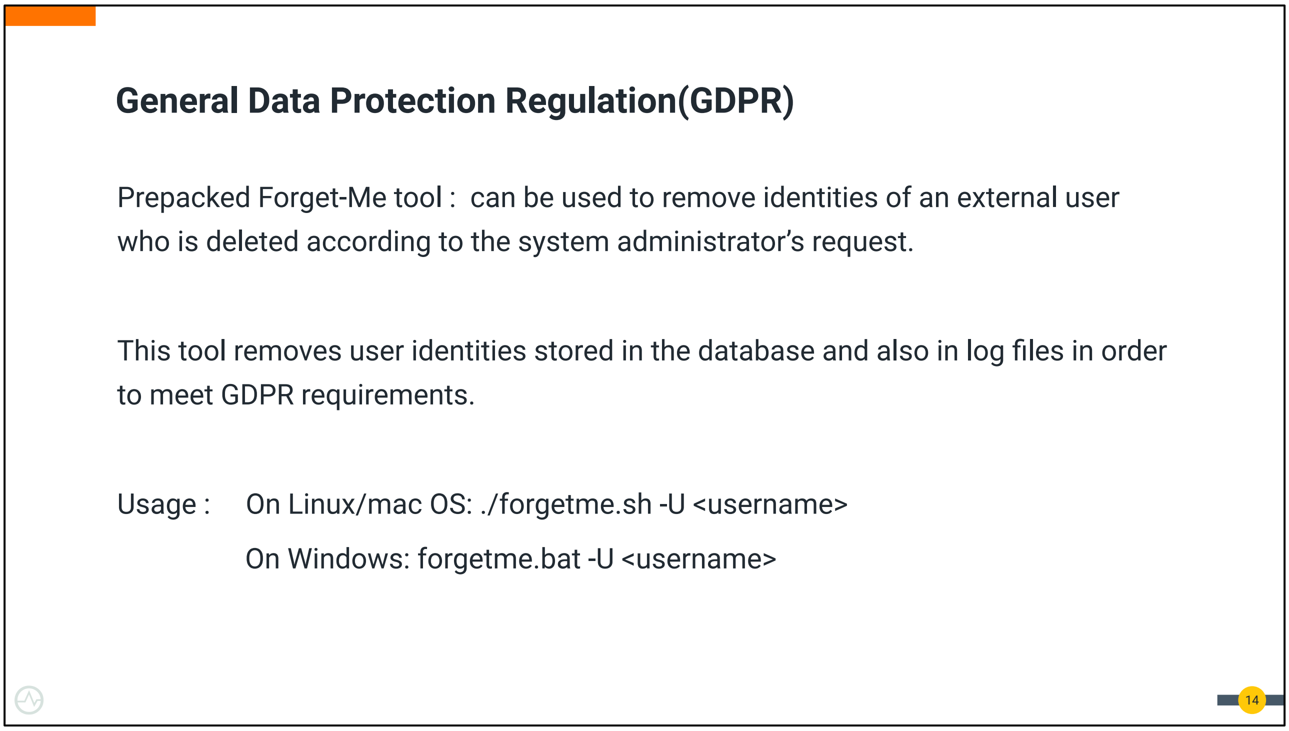
1) ./ciphertool.sh -Dconfigure
2) ./ciphertool.sh
3) Enter Plain Text Value :admin

Encrypting passwords for mediation flow

1. Encrypt passwords manually as shown before.
2. Select Browse under Resources to access the registry browser and go to the `/_system/config/repository/components/secure-vault` location.
3. Add the aliases and the encrypted value as a property.

Using encrypted passwords in mediation flow

Instead of hard coding the admin user's password as `<Password>admin</Password>`, you can encrypt and store the password using the AdminUser. Password alias as follows: `<Password>{wso2:vault-lookup('AdminUser.Password')}</Password>`.



General Data Protection Regulation(GDPR)

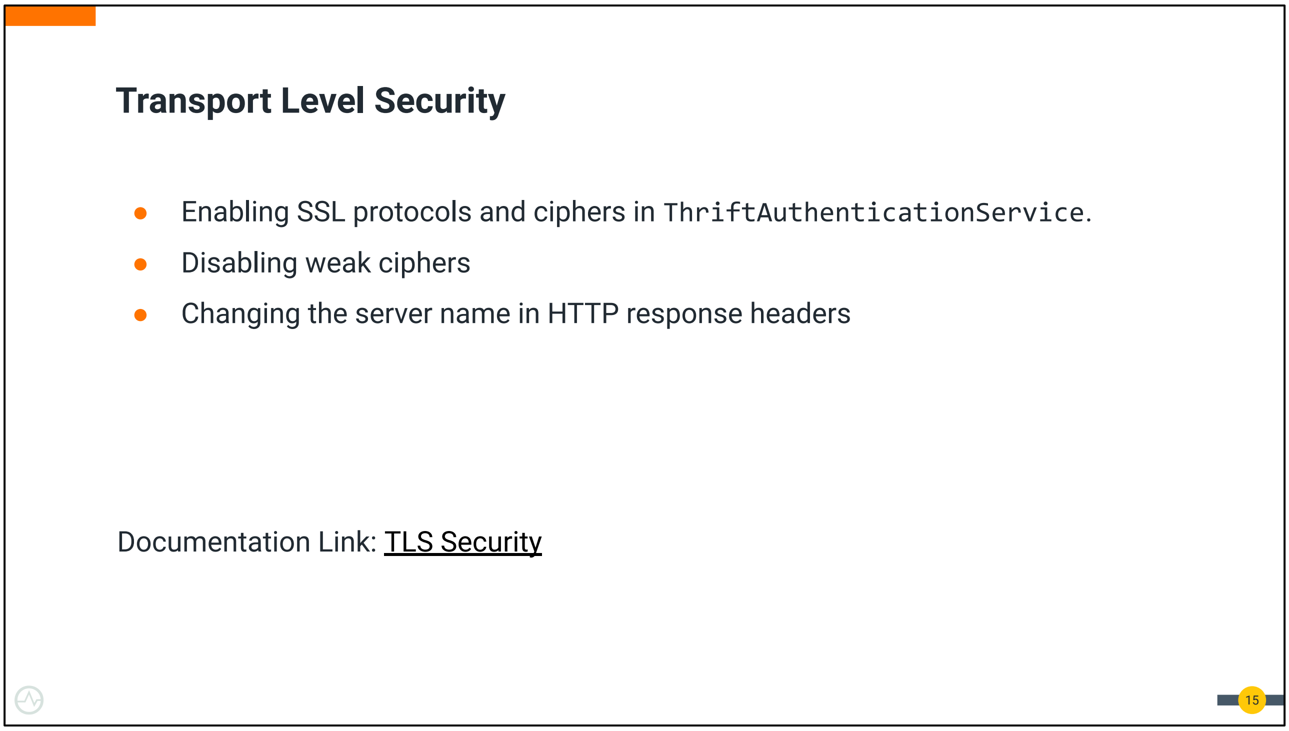
Prepacked Forget-Me tool : can be used to remove identities of an external user who is deleted according to the system administrator’s request.

This tool removes user identities stored in the database and also in log files in order to meet GDPR requirements.

Usage : On Linux/mac OS: ./forgetme.sh -U <username>

On Windows: forgetme.bat -U <username>

```
<API-M_HOME>/repository/components/tools/forget-me/conf
```



Transport Level Security

- Enabling SSL protocols and ciphers in `ThriftAuthenticationService`.
- Disabling weak ciphers
- Changing the server name in HTTP response headers

Documentation Link: TLS Security

A cipher is an algorithm for performing encryption or decryption. When you set the sslprotocol of your server to TLS, the TLS and the default ciphers get enabled without considering the strength of the ciphers.

This is a security risk as weak ciphers, also known as EXPORT ciphers, can make your system vulnerable to attacks such as the Logjam attack on Diffie-Hellman key exchange. The Logjam attack is also called the Man-in-the-Middle attack. It downgrades your connection's encryption to a less-secured level (e.g., 512 bit) that can be decrypted with sufficient processing power.

It is recommended to change this by configuring the server name in the deployment.toml file.

By default, all WSO2 products pass "WSO2 Carbon Server" as the server value in HTTP headers when sending HTTP responses. This means that information about the WSO2 product stack will be exposed through HTTP responses. It is recommended to change this by configuring the server name in the deployment.toml file.

```
[transport.https.properties]
server="WSO2 Carbon Server"

[transport.http.properties]
server="WSO2 Carbon Server"
```

Mutual SSL

Secure your backend by enabling mutual SSL between the API Gateway and your backend.

- Export the certificates.
- Enable dynamic SSL profiles.

Documentation Link : Mutual SSL