Consent Tokens for Privileged Operations

Consent tokens are security tokens signed by Cisco and are used to enable or disable restricted actions on the router. Cisco verifies that the user requesting the consent token is the legitimate owner of the device before issuing the token.

Key characteristics of consent tokens:

- Device-specific: Each token is associated with a particular device.
- Time-limited: Tokens must be used within a defined time window.
- Single-use: Each token can be used only once.

Feature Name	Release Information	Feature Description
Customer consent token configuration	Release 26.1.1	Introduced in this release on: NCS 5700 fixed port routers; NCS 5700 line cards [Mode: Compatibility; Native] This feature allows you to enable the customer consent token workflow so that the key name is linked to the key package. This chapter introduces these changes: CLI: A new command, **consent-token customer**, has been added.

Consent tokens are used when

- disabling secure ZTP, or
- enabling or disabling lawful intercept.

Owner tokens

An owner token or a customer consent token functions similarly to a consent token. Unlike a consent token which is signed by Cisco, an owner token or a customer consent token is signed by a customer key (rooted

in the customer's OC). This allows customers to control who is authorized to perform restricted actions and to enforce stricter authentication requirements. Owner tokens or customer consent tokeens can only be used if the customer has installed an OV/OC.

Customer consent tokens are used to clear device ownership.

Provisioning Cisco-signed consent tokens

Summary

Consent tokens use a challenge-response process. The challenge string generated on the router is valid for a defined time interval. The response string for this challenge must be provided within this time interval to authenticate the user to perform the restricted action.

If you use Cisco's consent token workflow, you need to contact Cisco TAC for every request to enable or disable certain privileged operations.

Workflow

These are the stages used in provisioning a Cisco-signed consent token:

1. Generate the challenge string on the router.

   ```
   Router# request consent-token generate-challenge <feature>
   ```

   This produces a challenge string containing the device ID, a nonce, and the requested action. Here, **<feature>** can be any supported security feature, namely, **secure-ztp**,**lawful-intercept**, **factory-reset**, and so on.

2. Submit the challenge string to a Cisco TAC engineer.

   Cisco verifies that the requester is the legitimate device owner and is authorized to perform the requested action. If verified, the TAC engineer provides a signed response string.

3. Paste the response string provided by the TAC engineer when prompted, to install the signed response on the router.

   ```
   Router# request consent-token accept-response
   ```

   The router validates the signature and confirms that the device ID and nonce match its own records. If valid, the requested feature is enabled or disabled.

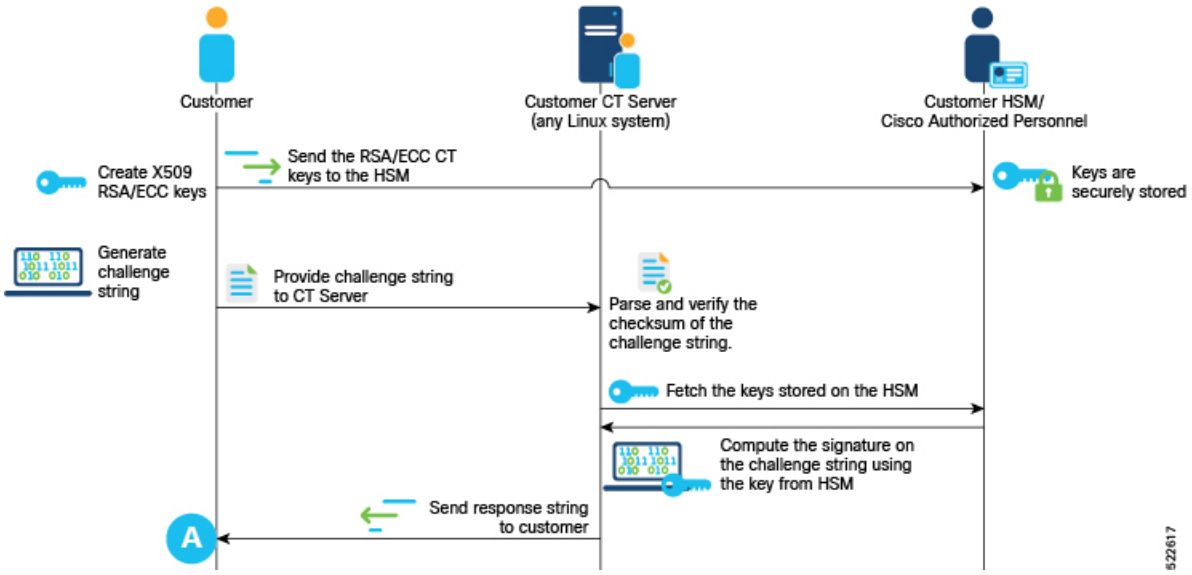
Provisioning customer consent tokens

Summary

This process allows you to set up a server on your premises to handle the customer consent token based challenge-responses.

Workflow

Figure 1: Workflow for the Consent Token Server



These are the stages involved in provisioning customer consent tokens:

1. Generate customer consent token keys using the openssl commands.
2. Onboard the customer consent tokens keys on to the router using key packages.
3. Enable the customer consent token to link the key name with the key package.

   ```
   Router# configure
   Mon Feb  2 13:22:58.355 UTC
   Router(config)#consent-token customer cert-name CT_KEY key-name key1 product-name prod1
   Router(config)#commit
   ```

   Here **cert-name** indicates the name of consent token certificate added through a key package, **key-name** indicates the name of the consent token key, and **product-name** indicates the product name for consent token.
4. Set up the consent token server on your premises using the sample script, `ct_sim.py`, available on Github at https://github.com/ios-xr/consent-token-signing-server.
5. Generate the challenge for features that support customer consent token workflow.
6. Sign the challenge string using your consent token signing server.

   The server parses the challenge, verifies the checksum, and computes the signature using the key fetched from HSM (Hardware Security Module).
7. Accept the response generated by the consent token signing server.

   The requested feature is now enabled or disabled.