Secure Storage of Sensitive Information on EVE

Problem Statement

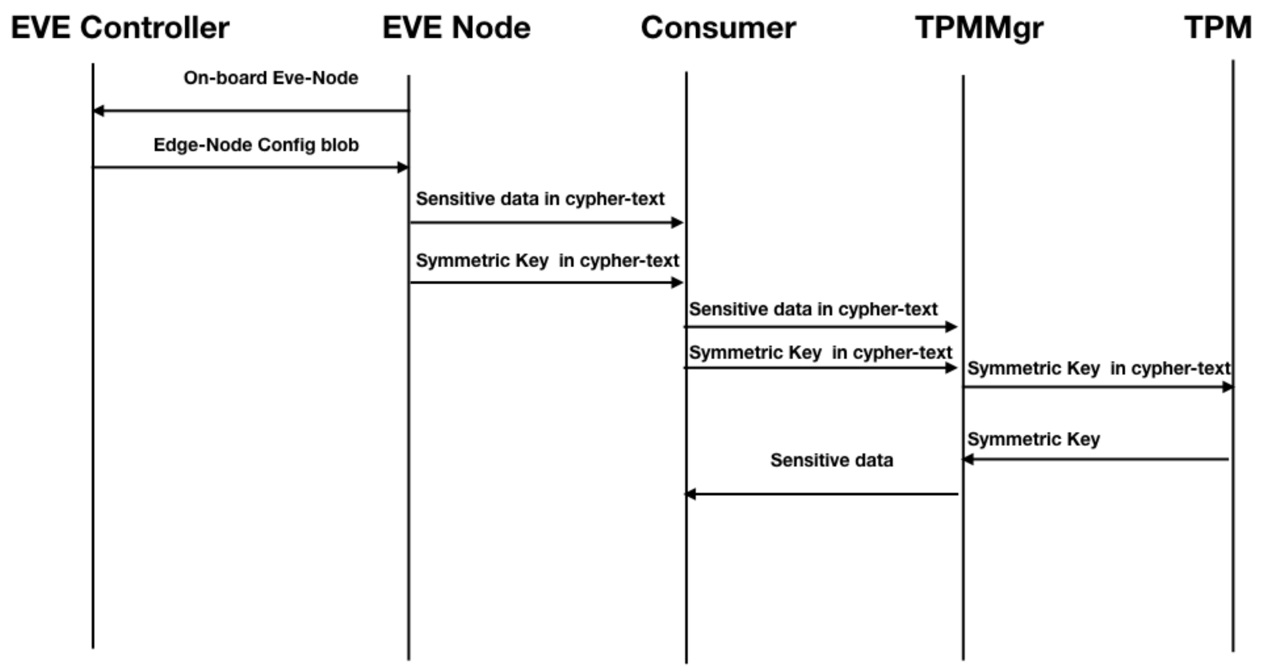
1. Currently sensitive configuration blocks for an EVE-node, not all-inclusive, are data store credentials, WLAN credentials, cloud-init (containing Azure/AWS credentials) etc.
2. EVE Controller posts these sensitive configuration, in clear-text format, as part of the configuration blob to EVE Node. The sensitive configuration blocks are exchanged between agents inside the EVE Node, as is through pubsub channel.

Motivation

As part of data at rest/transit encryption initiative, for EVE, the sensitive configuration block should be in encrypted format during the transit from EVE Controller and EVE Node, and while at rest(pubsub channel storage) inside EVE Node, until the end-user consumer uses them.

Proposed Solution

1. "Elliptic-curve Diffie–Hellman" key exchange scheme will be used to share the symmetric key used for protecting the credentials.
   An ECC Key pair is generated in the cloud. And a shared secret is generated using the private part of the ECC Key pair, and the public device cert. This shared secret is used as seed to compute an AES-256-CFB cipher. Along with this a random value is generated to be used as a ICV. The seed and ICV is used to encrypt the sensitive data in the config blob. The public part of the ECC Key pair, and ICV is published as part of the config blob. On the device, the TPM module will use the public portion of the ECC key pair along with the device private key, along with the ICV, to compute the AES cipher. And the derived cipher is used to decrypt the ciphertext config to clear text.
2. Scaling challenges: One ECC Key pair per device Vs. a global Key pair for the controller.
   How controller generates and manages the ephemeral key is up to the controller implementation .e.g. if there is a scaling concern, one such implementation may choose to share a single ephemeral pair among many edge nodes. Or it can create one pair for device and cache it for re-use in every config refresh. The only requirement from the controller is that the key pair for a given edge-node remains static and unchanged, unless there is an administrative trigger to change it.
3. Key Rotation : The Key rotation policy will be driven by controller. The edge-node will always get the public cert of the most recent ECC Key pair.
4. Transition Plan: As of now, we will be pushing both cleat-text and cipher text configuration to the device. Once all the devices migrate to a software version that supports this feature, controller will stop sending plaintext, and send sensitive data only as ciphertext.
5. Changes required in the go-tpm package.Currently ECDH APIs are not available in go-tpm that we vendor from github.com/google/go-tpm. We will fork the existing the go-tpm master, add this support and vendor it, until it is upstreamed to master.
6. Inside EVE Node, the sensitive configuration stays in encrypted format on disk (i.e., pubsub), and will only be decrypted on demand by the end consumer.
7. The end consumer module (e.g. downloader), will unwrap/unseal the symmetric key with the help of TPM-Mgr (using the new go-tpm APIs), and construct the symmetric key first, and then use the symmetric key to decrypt the sensitive configuration block, using software crypto tools (e.g. openssl or go crypto etc.)


EVE Controller
EVE Node
Consumer
TPMMgr
TPM
On-board Eve-Node
Edge-Node Config blob
Sensitive data in cypher-text
Symmetric Key in cypher-text
Sensitive data in cypher-text
Symmetric Key in cypher-text
Symmetric Key in cypher-text
Symmetric Key
Sensitive data