Security WG Meeting, 4/17/19

Attendees: Anthony, Tingyu, Trevor, Jim (Dell), Bryon, Jim, Toby (Intel), Colin (Kong), Brad (Beechwoods), Ian, Tony (Canonical), Malini (VMWare), Ike, Eno. Others may have joined after the meeting started and attendance was captured.

Agenda

Old Business

- Kong on ARM – now in DevOps court
- Docs
    - Securing service secrets – version 7 had no additional comments. Are we good? Can this go to TSC?
        - No need to send it to TSC, as it is an internal working document. Considered final for Edinburgh.
    - Security issue process – version 5 includes issue of how to handle dependency security issues. Is this good? Can this go to TSC?
        - No further inputs. Considered ready for TSC next week.
- Tingyu's design of vault initialization and DB initialization still being worked
    - URLs for path of secrets will be the name of the micro service (to include scripts) – an update/change from the document Tingyu shared last week.
    - Quick discussion on which credential generation algorithm to be used
        - Recommendation to use plugin (from Vault) to generate the username/password
        - No objections to this plan
- Brandon working on Go Client Module for accessing Vault (see https://github.com/edgexfoundry-holding/go-mod-core-security)
- Fuji roadmapping
    - From https://github.com/bnevis-i/security-secret-store/pull/1 (Thanks Bryon)

~~Phase 0 (tasks expected to be done in Edinburgh)~~

~~1. Create Vault namespace standard for per-service and shared secrets.~~

Phase 1

1. Develop test infrastructure that simulates EdgeX supported bring-up models supported by System Management Agent.
2. Create PKI at runtime that is unique for each boot (remove static PKI).
3. Block startup of core services until PKI is available.
4. Remove TLS skip-verify overrides from client services.
5. Revoke previously generated tokens on every reboot.
6. Generate per-service tokens at system startup.
7. Revoke Vault root token.
8. Implement Vault cubbyhole response-wrapping.
9. Implement Vault secrets client library (integrate with registration service client library?)

Phase 2

1. Generate unique-per-installation PGP key pair.
2. Derive PGP passphrase with an HMAC-KDF using hardware fingerprint as IKM and random salt.
3. Pass PKI and Vault token secrets via tmpfs volumes.
4. Revoke CA and intermediates after creating leaf certificates.
5. Token issuance driven by service registration.
6. Automated revocation of Vault tokens for failed services.
7. Self-token-rotation (token issuing service).

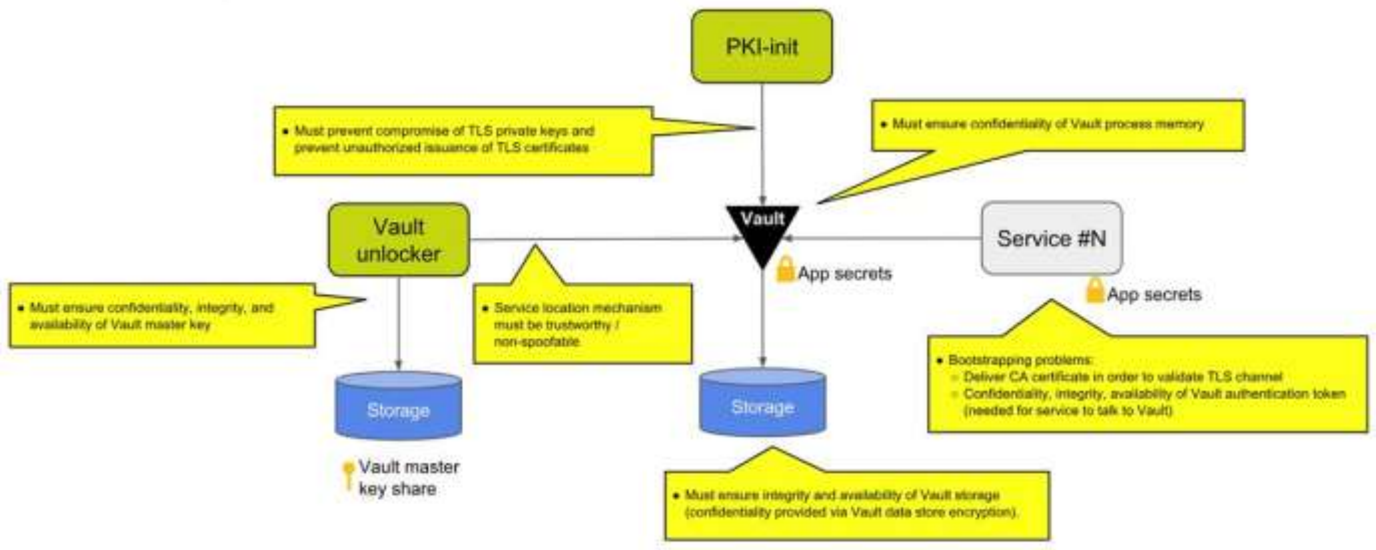
~~Phase 3~~

1. ~~TPM hardware secure storage (unauthenticated) for Vault master key.~~
2. ~~Use TPM persistent handle and NVRAM for Vault master key.~~
3. ~~Implement additional TPM authentication scenarios (simple PCR, PCR policy, and (HMAC-KDF?) password).~~
4. ~~TPM-based PKI.~~
5. ~~Once-per-boot decryption of Vault master key.~~
6. ~~Token-issuing-token encryption at rest or recovery of token-issuing-token from HW secure storage.~~

~~Phase 4~~

1. ~~PKCS11 hardware secure storage for Vault master key~~
2. ~~Implement Mandatory Access Control for EdgeX services.~~

Secret Management using Vault
General Requirements


PKI-init
• Must prevent compromise of TLS private keys and prevent unauthorized issuance of TLS certificates
• Must ensure confidentiality of Vault process memory
Vault unlocker
Vault
Service #N
App secrets
App secrets
• Must ensure confidentiality, integrity, and availability of Vault master key
• Service location mechanism must be trustworthy / non-spoofable
Storage
Storage
Vault master key share
• Bootstrapping problems:
◦ Deliver CA certificate in order to validate TLS channel
◦ Confidentiality, integrity, availability of Vault authentication token (needed for service to talk to Vault)
• Must ensure integrity and availability of Vault storage (confidentiality provided via Vault data store encryption).

Discussion on above for Fuji Road mapping:

TLS keys/certificates would also be important for service-to-service communications (via TLS)

Key goals for Fuji are:

Providing key ingredients/elements that would help eventually building hardware secure store (HWRoT) for securing Vault Master Key. This includes:

- Protection of Vault Master Key
- Generation of PKI
- Distribution of per service Vault secrets.

As an ancillary issue, how should each service be bootstrapped? Does each service have access to the Vault Master Key?

2nd goal of security for Fuji release is how to ensure the services running are those expected (and authorized). Preventing fake services from running in place of EdgeX services. Preventing other services from running that shouldn't be. Service authenticity.

- This work may just be a document.
- Could be some devops work here as well (signing containers for example)
- It must consider Docker or another package management (Snaps)
- How do we secure the Docker Registry – insuring what you are running is what is expected

Other future goals to be discussed at the Seoul Face to Face

How do we accomplish service to service authentication/secure communications?

Do we (and if so how) securely provide service updates? Is this an EdgeX job or something else?

Device identification and authentication/authorization – can we trust the device connected to EdgeX?

- How do devices identify themselves today – by GUID? In some cases, with nothing.
- Is device identification / trust the responsibility of the device service (most feel it is)? The DS has the job of trusting a new device or not. Can/should we make this part of the SDK that allows for implementers to provide the means according to the protocol and device?

New Business

- none