Lockers: An Innovative and Secure Solution for Managing Secrets

Viet Tran (IISAS)

Security vs usability

Security improvements are often done at the cost of usability

- Long passwords with upper/lowercase letters, numeric and special characters
- 2FA
- Captcha
- ...

But in secret management service, we improve both security and usability without conflicts

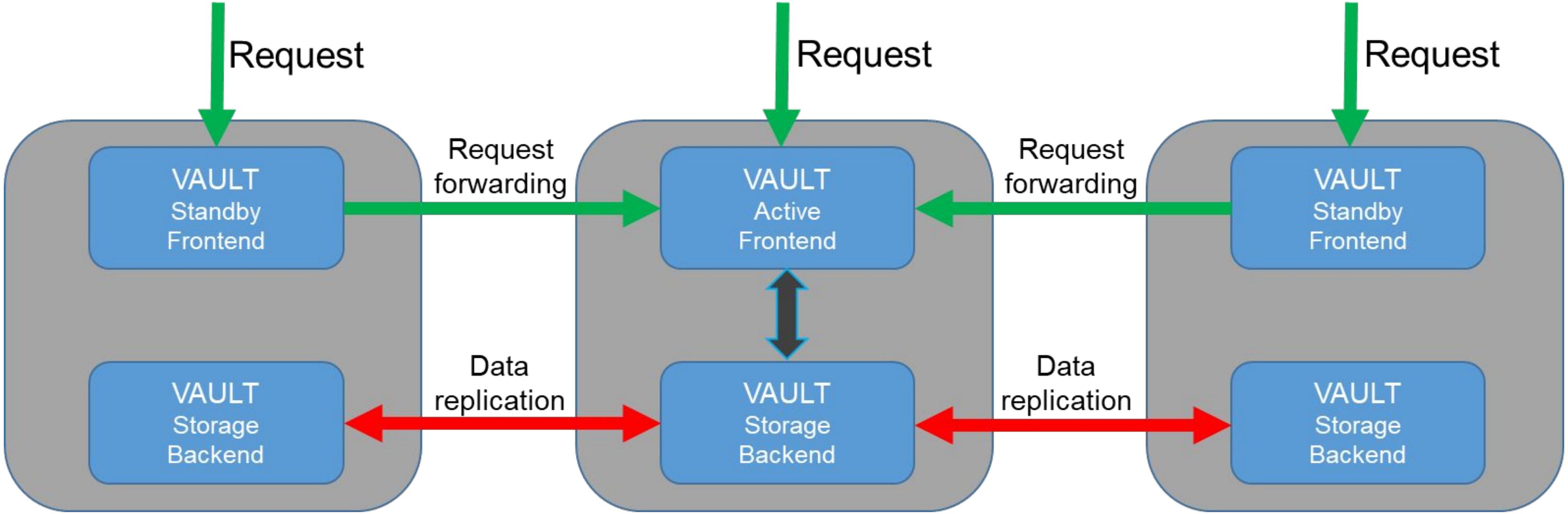
First thought of secret management service

- Just deploy HashiCorp Vault and have it
  - It works
  - It works well
  - But we can improve it a lot
  - We can improve both security and usability

Quick overviews of existing features

High-availability setup

Three nodes, geographically distributed at IISAS, INFN and IFCA



Universal endpoint via Dynamic DNS

- Three endpoints, each can serve user requests:
  - https://vault-iisas.services.fedcloud.eu:8200 (IISAS)
  - https://vault-infn.services.fedcloud.eu:8200 (INFN)
  - https://vault-ifca.services.fedcloud.eu:8200 (IFCA)
- Main, universal endpoint https://vault.services.fedcloud.eu:8200 is assigned to IFCA or INFN endpoint via Dynamic DNS

- NEW: new universal endpoint https://secrets.egi.eu/

Easy-to-use client

- Authentication via access tokens (integrated with oidc-agent and mytoken)
- Working out of the box, no setup
- Simple, easy-to-use commands

```
$ fedcloud secret put my_app_secrets mysql_password=123456 admin_password=abcdef

$ fedcloud secret list
my_app_secrets

$ fedcloud secret get my_app_secrets
key             value
--------------  -------
admin_password  abcdef
mysql_password  123456
```

Client-side encryption

- Users may need to store very sensitive secrets that absolutely nobody else can read them, by any means
    - Access tokens may be compromised
    - 2FA authentications are not suitable for automation
    - Solution: users encrypt the secrets before uploading
    - Very easy to use, fully automatic and transparent

```
$ fedcloud secret put certificate cert=@hostcert.pem key=@hostkey.pem --encrypt-key
my-secret-passphrase

$ fedcloud secret get certificate cert --decrypt-key my-secret-passphrase
```

- Security tips: use different passphrases for different secrets

NEW: Introduction of lockers

Motivations

- Authentication via access tokens from VMs is not optimal
    - Access tokens have too broad rights (to all secrets, to other services)
    - Cannot be used on shared VMs or untrusted Cloud environments

- Lockers, temporary, isolated storages for valuable items, are the solution
    - Create a locker, store valuable items there and deliver the key to recipients
    - No personal credential needed for retrieving valuable items
    - No access to other valuable items except the ones stored in lockers
    - Ideal for delivering secrets in untrusted environment like Cloud

Features of secret lockers

- Temporary, short-living:
  - Lifetimes: default 24h (client setting), max 32 days (system setting)
  - Numbers of uses: default 10 (client setting), max unlimited (system setting)
- Isolated, secure:
  - Lockers are completely isolated from each others, and from the main secret storages
  - The only way to access secrets in the lockers are locker tokens. Even creator or root do not have access by other means
- Non-personal:
  - Locker tokens cannot access other secrets outsides of the lockers
  - No personal data stored in the locker token
- Transferable:
  - As lockers are isolated and non-personal, creators can share/transfer the lockers and its content to others if needed

Creating a locker

```
$ fedcloud secret locker create
hvs.CAESIGXXX                             <= Print only the token, easy scripting

$ fedcloud secret locker create --ttl 24h --num-uses 10 --verbose
key                value
----------------   -----------------------------------------------
client_token       hvs.CAESIGXXX          <= This is the token
accessor           o3GXXXXXXXXXXXX
policies           ['default']
token_policies     ['default']
lease_duration     86400
renewable          False
orphan             False
num_uses           10
```

Checking info of the locker

```
$ fedcloud secret locker check hvs.CAESIXXX
key               value
----------------  -----------------------------------------------------
accessor          qb52XXXXXX
creation_time     1685008416
creation_ttl      86400
display_name      token-token
expire_time       2023-05-26T09:53:37.315243089Z
id                hvs.CAESIGXXX
issue_time        2023-05-25T09:53:37.315281071Z
num_uses          8
orphan            False
path              auth/token/create
policies          ['default']
renewable         False
ttl               86114
type              service
```

Accessing lockers

- Just set locker token instead of OIDC access token and use `fedcloud secret` commands `put/list/get` normally. No additional configuration needed:

```
$ fedcloud secret put mysecret password=123456 --locker-token hvs.CAESIXXX
```

- The locker token may be set as OS environment variable like access token

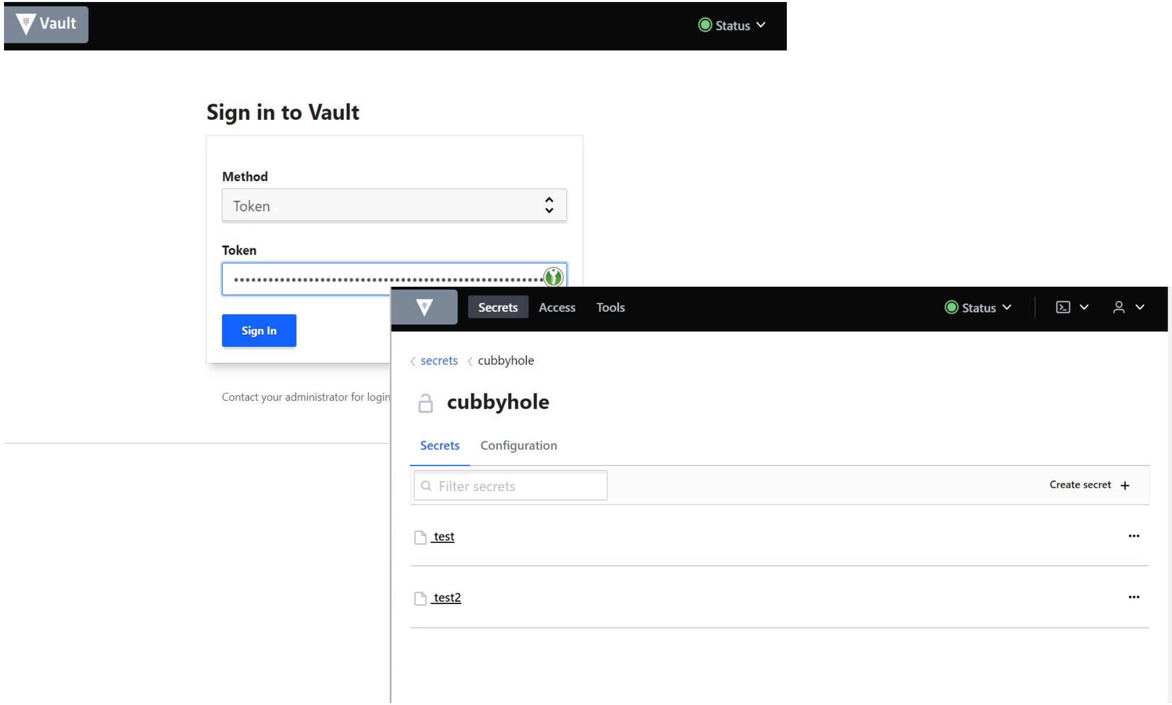
```
$ export FEDCLOUD_LOCKER_TOKEN=hvs.CAESIXXX
$ fedcloud secret get mysecret
key         value
--------   -------
password    123456
```

- Note: OIDC accounts and access tokens are not needed for accessing lockers.

Accessing lockers via GUI

Users can login to Vault's GUI via locker token and manage secrets via GUI

Warning: the GUI exhausts number of uses very fast. Remember to set larger number of uses if experimenting with GUI



Sign in to Vault

Method: Token

Token: ••

Contact your administrator for login

secrets < cubbyhole

cubbyhole

test

test2

Destroying lockers

Lockers and all contents in them are automatically destroyed when their lifetimes or numbers of uses are expired (desired feature for security)

They can be also destroyed manually if needed by revoking the locker token:

```
$ fedcloud secret locker revoke hvs.CAESIXXX
```

Special: single-use lockers

Why single-use lockers

- Sometimes, users need to deliver secrets via untrusted communications
  - It is a big trouble if some secrets are stolen (passwords, tokens, ...)
  - It is still a much bigger trouble if secret owners don’t know about that
  - Attacker may quietly abuse the secrets for long time and make much larger damages

=> Single-use lockers are the way to go

- Autodestruction after successful delivery
- Immediate detection of misbehavior if it happens

How to use single-use lockers

- Create a locker with num-uses=2 (not 1)

```
$ fedcloud secret locker create --ttl 1h --num-uses 2
hvs.CAESIXXX
```

- Store some secrets there. That will reduce number of uses to 1 (single-use)

```
$ fedcloud secret put mysecret password=123456 --locker-token hvs.CAESIXXX
```

- Send the locker token to recipient via possibly untrusted communications:
  - If the recipient can read the secrets, it is safely delivered, nobody else has read them before (and nobody can read them later)
  - If the recipient cannot read the secrets, it is a proof that secrets have been stolen. Time to alarm admins, change passwords, launch investigations and do other relevant actions

Summary about lockers

- Lockers simplify delivering secrets to untrusted VMs
  - No need of access tokens, no personal data
  - Enabling single-use secrets for detecting misbehavior
  - Enabling deliver secrets to VMs owned by others

- Very simple usage
  - Simplicity means robustness
  - Compatible with existing commands (e.g. using client-side encrypted secrets in lockers)

The locker feature is fully functional but the its API is still not very stable (public beta testing). Testers are welcomed

NEW: VO-sharing secrets

Sharing secrets in VOs

- Just login to GUI (https://secrets.egi.eu) via OIDC and EGI Checkin
- Go to folder “secrets/vos/<vo-name>/”
- Create/read/update VO-shared secrets via GUI
- See demo

Summary

- The secret management service is continuously improved:
  - Introduction of Lockers
  - VO-sharing secrets via GUI

- Client-side encryption and lockers are significant security additions
  - It is not a Vault deployment, it is a new security service for EGI FedCloud

- More features are planned
  - VO-based tokens (can access only secrets in a specific VO, nothing else)
  - CLI for managing VO tokens and accessing VO secrets