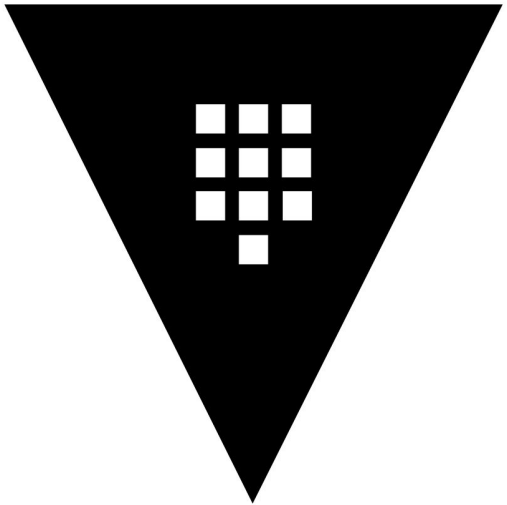
Introduction to Vault and its use at Opera

Michał Łowicki & Piotr Śliwka

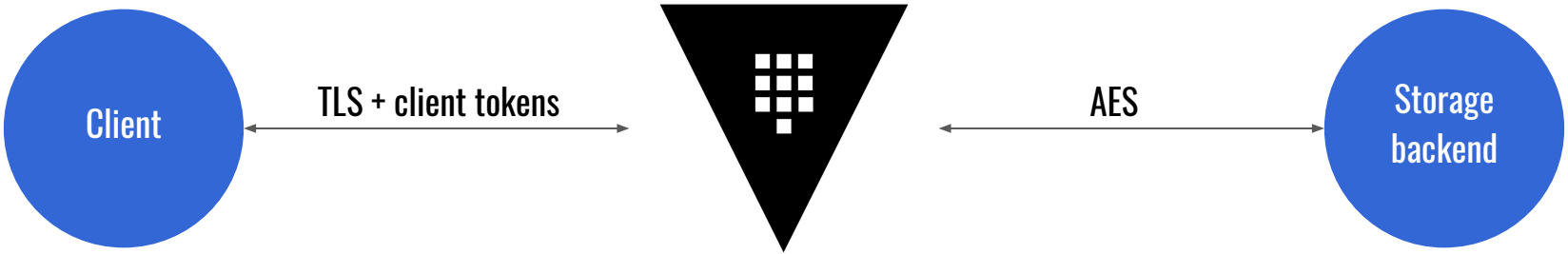


https://vaultproject.io

Secrets

- Passwords
- Tokens
- Certificates
- API keys
- Database credentials
- AWS credentials
- ...

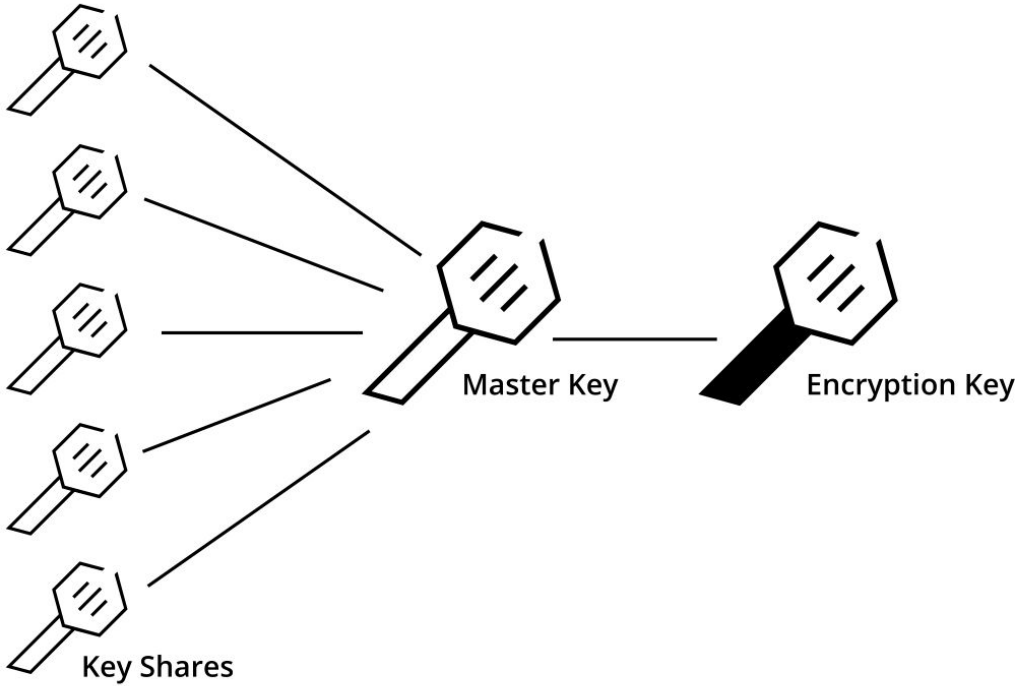
Security model



Two-man rule

Shamir's Secret Sharing

Shamir's Secret Sharing




```
> vault init -key-shares=5 -key-threshold=3
```

> vault unseal

> vault unseal

> vault unseal

```
> docker run --cap-add=IPC_LOCK
    -e 'VAULT_DEV_ROOT_TOKEN_ID=myroot'
    -e 'VAULT_DEV_LISTEN_ADDRESS=0.0.0.0:1234'
    -p 1234:1234 vault
```

```
> export VAULT_ADDR='http://0.0.0.0:1234'
> vault auth
> vault write secret/foo value=bar
> vault read -field=value secret/foo
```

Policies

```
path "secret/playground/foo" {
  capabilities = ["read", "update", "create"]
  allowed_parameters = {
    "bar" = []
  }
}
path "secret/staging/*" {
  capabilities = ["read"]
}
```

```
> vault policy-write meetup meetup.hcl
> vault write auth/radius/users/mlowicki policies=meetup
> vault auth -method=userpass -path=radius username=mlowicki
Password (will be hidden):
> vault write secret/playground/foo baz=bar
...
* permission denied
> vault write secret/playground/foo bar=baz
Success! Data written to: secret/playground/foo
```

Storage backends

- In-Memory
- Filesystem
- ZooKeeper
- S3
- Cassandra
- PostgreSQL
- ...

Secret backends

- kv (secret/ mounted by default)
- transit (cryptography-as-a-service)
- totp (multi-factor authentication)
- rabbitmq
- aws (based on IAM policies)
- consul
- ...

Transit Secrets Engine / Cryptography-as-a-service

- Encrypt / decrypt data from application
- Generate high-quality random bytes
- Cryptographic hash of given data
- HMAC (keyed-Hash Message Authentication Code)
- ...

```
> vault mount transit

> vault write -f transit/keys/mykeyring

> vault write transit/encrypt/mykeyring plaintext=$(base64
<<< "secret")

> vault write -field=plaintext transit/decrypt/mykeyring
ciphertext***** | base64 --decode
```

Auth backends

- Token
- Username & Password
- AWS
- GitHub
- LDAP
- TLS certificates
- ...

Audit backends

- File

```
> vault audit-enable file file_path=/path/to/log/file
```

- Syslog
- Socket

s/Michał/Piotrek/

Integration with infrastructure

What do we want to achieve?

Do not ever store production secrets in:

- Shared docs
- settings.py
- Dockerfiles
- Marathon app manifests
- ... and any other file within a project repo

What secret backends are we using?

- At the moment, only basic KV store

What auth backends are we using?

- RADIUS for user auth
- Plain token store for machine auth

Where do we access our secrets?

- Local developer's machine
  - shared passwords (WebUIs, admin DB accounts, etc)
  - per-machine secrets (populated during ansible-playbook runs)
- Mesos/Marathon hosted apps
  - application DB accounts
  - API keys
  - SSL private keys
  - etc

A little background on tokens

- normally issued by an auth backend
- have limited lifetime (TTL)
- have an immutable set of policies associated with each token

Personal tokens

- issued RADIUS auth backend
- based on company-wide LDAP catalogue
- 2FA with TOTP/YubiCloud
- separate policies for every team/department
    - examples: sysadmin, services-engineer, vault-admin

Personal token usage

- Manually read shared passwords:

```
$ vault read secret/infra/devpi/root

Key                 Value

---                 -----

refresh_interval    768h0m0s

password            xxxxxxxx
```

Personal token usage

- Automatically lookup secrets in Ansible templates:

```
postgres:
  image: postgres:{{ postgres_version }}
  restart: always
  environment:
    POSTGRES_USER: {{ lookup('vault', 'secret/infra/sentry/postgres').username }}
    POSTGRES_PASSWORD: "{{ lookup('vault', 'secret/infra/sentry/postgres').password }}"
```

Machine tokens

- Right now, issued only for marathon masters
- Every machine has its own token stored locally
- Issued by entitled team members
- Renewable (TTL can be extended forever)
- Orphan (outlives issuer token)
- Invalidated if not renewed within 72 hours

How to create machine token?

How to create machine token?

Using token roles...

```
$ vault read auth/token/roles/marathon-master

{

    "role_name": "marathon-master",

    "allowed_policies": "default,marathon-master",

    "orphan": true,

    "period": "72h",

    "renewable": true

}
```

How to create machine token?

… and appropriate policy …

```
$ vault read sys/policy/services-infra-engineer

[...]

path "auth/token/create/marathon-master" {

  capabilities = ["update"]

}
```

How to create machine token?

... one needs only to:

```
$ vault write -force -field=token auth/token/create/marathon-master

27ed12c6-05d2-11e8-81b5-0fe8116d1bb8
```

Integrating with Marathon-hosted apps

Marathon Vault plugin

- Avast project, hosted on GitHub
- Integrates with Marathon secrets API
- Separate secret subtrees per app
  - readable by one (and only one) Marathon app

Secret environment variables

```
{
  "env": {
    "ENV_NAME": {
      "secret": "secret_ref"
    }
  },
  "secrets": {
    "secret_ref": {
      "source": "abc/xyz@password"
    }
  }
}
```

Token Keeper

- In-house solution
- Daemon running on every machine, renewing its Vault token

Questions?