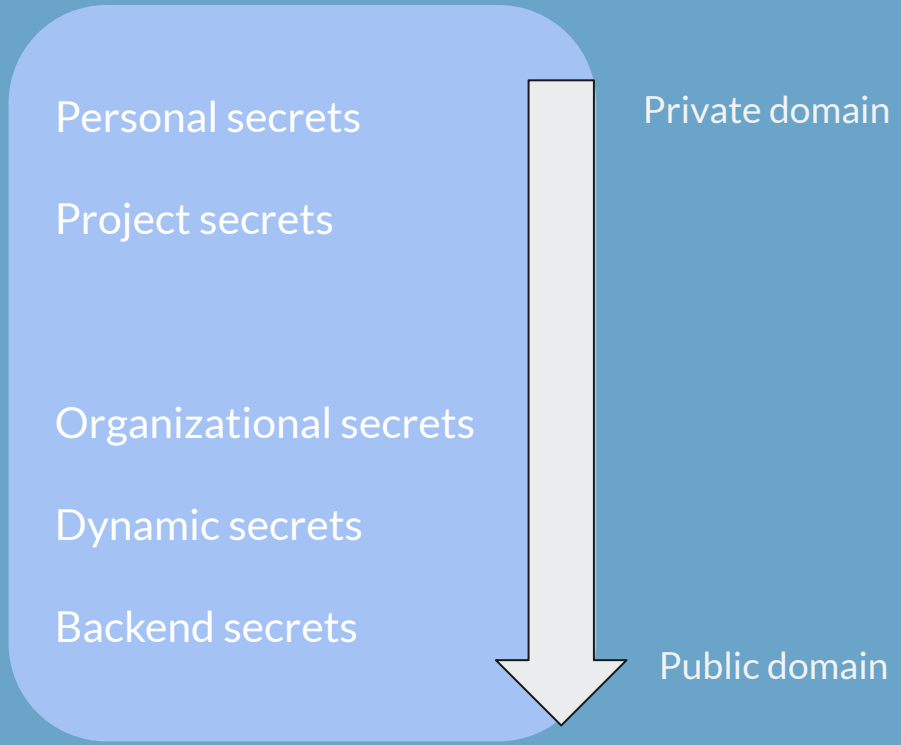
Keeping secrets secret

Centralised storage and management of passwords / API keys/ credentials in a scalable manner

Daniel Ivanov
sertys@gmail.com

What is a secret?



Personal secrets

- Login credentials
- developer keys
- SSH credentials
- X.509 certificates
- TOTP hashes

Project secrets

Cloud keys

Environment variables

Github accounts

Database credentials

Salts

CI-related credentials

Git-secret / git-crypt / git actions

Docker secrets

Blackbox

Organizational secrets

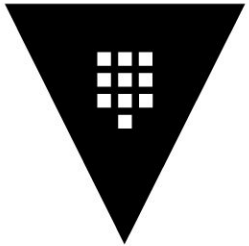
are public secrets

- Common points of entry
- Shared accounts
- Company financial requisites
- IP material
- Database sensitive data

Public secrets demand

- ★ security in transit and at rest
- ★ encryption
- ★ flexible authentication
- ★ access control
- ★ version control
- ★ solid auditing
- ★ easy access
- ★ high availability

Introducing



HashiCorp
Vault

A tool for centralised or clustered management of secrets.

Build custom or use out-of-the box?

The eternal dilemma

Security in transit and at rest

The barrier - an encryption/decryption layer between components

When a Vault server is started, it starts in a *sealed* state. In this state, Vault is configured to know where and how to access the physical storage, but doesn't know how to decrypt any of it.

Unsealing is the process of constructing the master key necessary to read the decryption key to decrypt the data, allowing access to the Vault. API requests are secured via TLS for transit.The Shamir key sharing provides key splitting for the master key initiation. Lets us position guardians for the data and distribute responsibility in a predictable and configurable way.

Encryption

Core barrier encryption(as in storage encryption)

AES-256 (w/ GCM-96)

An internal barrier encrypts and decrypts the data before it hits the storage or the API router

Transport level encryption

TLS(by default)

Since the API exposes HTTP, it is securable in a plethora of manners depending on infrastructure.

Flexible authentication

Vault does not discriminate machines nor humans

Flexible authentication which is also pluggable and supports :

AppRoles
AliCloud
AWS
Azure
Google cloud
JWT
Kubernetes
Github
LDAP
Okta
Radius
TLS certificates
Tokens
Username/Password

...and your imagination + coding skills

Access control

Vault is in essence a RESTful API and it’s policies are path-based. Everything is DENY by default. This gives for flexible and granular control

```
# Permit reading only "secret/foo". An attached token cannot read "secret/food"
# or "secret/foo/bar".
path "secret/foo" {
  capabilities = ["read"]
}

# Permit reading everything under "secret/bar". An attached token could read
# "secret/bar/zip", "secret/bar/zip/zap", but not "secret/bars/zip".
path "secret/bar/*" {
  capabilities = ["read"]
}

# Permit reading everything prefixed with "zip-". An attached token could read
# "secret/zip-zap" or "secret/zip-zap/zong", but not "secret/zip/zap
path "secret/zip-*" {
  capabilities = ["read"]
      }
```

Version control

Vault can be just a KV-store if so you desire

Vault supports various secrets engines.
Think of them as functional implementations.
The more basic are the KV engines - Version 1, Version 2.

Version 1

- Speed
- No-locking
- No versioning

Version 2

- Locking
- Metadata overhead
- Versioning

Solid auditing

Storing detailed logs for data access and manipulation is as easy as :

```
$ vault audit enable file file_path=/var/log/vault_audit.log
```

And you can send audit logs to stdout(for container logging) or a socket(tcp,udp,unix) for off-site storage.

Audit deviced also support HTTP header passthrough for integrated logging.

Auditability is key for when fecal matter hits the fan as it provides investigation paths and damage control.

Easy access

Vault provides a web UI, CLI and a RESTful API

Vault supports multiple management and access vectors that use the same core

CLI for sysadmins
Web UI for users
API for developers

High availability

Vault has pluggable storage

With Vault you can use distributed or centralised storage. The core is stateless which avails to replication of service. You can store your secrets in:

- Azure
- Cassandra
- CockroachDB
- Consul
- CouchDB
- DynamoDB
- Filesystem
- In-Memory
- MySQL
- PostgreSQL
- S3
- Swift
- Zookeeper
- + more

Still not convinced?

You can still use Vault for Encryption as a service in pseudo-TSM manner

Next Steps

Dynamic secrets
Security concerns
Deployment patterns
Plugins