Vault

DECOUPLING SECRETS & APPLICATIONS

HashiCorp

Armon Dadgar

@armon

Derek Downey

@derek_downey

ABOUT PYTHIAN

11,400

Pythian currently manages more than 11,400 systems.

400+

Pythian currently employs more than 400 people in 200 cities in 35 countries

1997

Pythian was founded in 1997

Global Leader In IT Transformation And Operational Excellence

Unparalleled Expertise

- Top 5% in databases, applications, infrastructure, Big Data, Cloud, Data Science, and DevOps

Unmatched Certifications

- 9 Oracle ACEs, 4 Oracle ACE Directors, 1 Oracle ACE Associate
- 6 Microsoft MVPs, 1 Microsoft Certified Master
- 5 Google Platform Qualified Developers
- 1 Cloudera Champion of Big Data
- 1 Mongo DB Certified DBA Associate Level
- 1 DataStax Certified Partner, 1 MVP
- 11 AWS Certified Solutions Architects, 1 AWS Certified Developer, 1 AWS Certified SysOps Administrator

Broad Technical Experience

- Oracle, Microsoft, MySQL, Oracle EBS, Hadoop, Cassandra, MongoDB, virtualization, configuration management, monitoring, trending, and more.

SOME OF OUR CLIENTS

BRIXMOR™ Property Group

Grasshopper

AtTask

Bookbyte

ANZ

Jpay

BULLHORN

IGN Entertainment

TOWER INTERNATIONAL

tracks

freshdirect

Innovapost™

inpho

CleaverBrooks

World Wide Technology, Inc.

GIANT TIGER

ARC'TERYX

inc Research

LONE WOLF REAL ESTATE TECHNOLOGIES

une University of New England

9/11 MEMORIAL

CROWDTWIST GIVE MORE. GET MORE.

glu

FOX SPORTS

TIO™

SECRET MANAGEMENT

WHAT IS "SECRET"?

SECRET VS. SENSITIVE

SECRET	SENSITIVE
DB CREDENTIALS	PHONE NUMBERS
SSL CA/CERTIFICATES	MOTHER'S MAIDEN NAME
CLOUD ACCESS KEYS	EMAIL ADDRESSES
ENCRYPTION KEYS	DATACENTER LOCATIONS
WIFI PASSWORDS	CUSTOMER PII
SOURCE CODE	EMAIL/CHAT

SECRET

DB CREDENTIALS

SSL CA/CERTIFICATES

CLOUD ACCESS KEYS

ENCRYPTION KEYS

WIFI PASSWORDS

SOURCE CODE

✓

SENSITIVE

PHONE NUMBERS

MOTHER'S MAIDEN NAME

EMAIL ADDRESSES

DATACENTER LOCATIONS

CUSTOMER PII

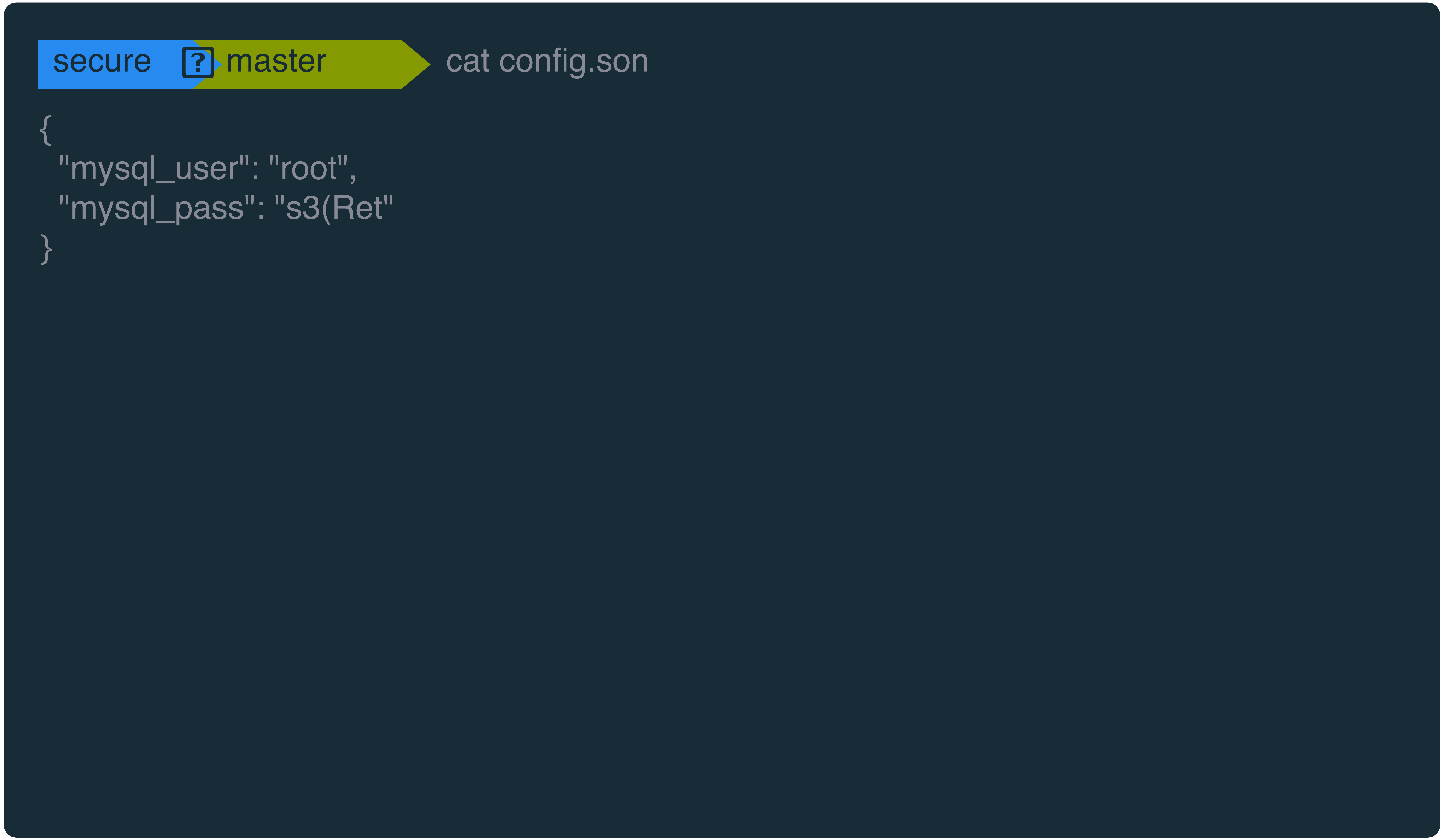
EMAIL/CHAT

✗

SECRET MANAGEMENT 1.0

HOW DO I DISTRIBUTE SECRETS?

- How do applications get secrets?
- How do humans acquire secrets?
- How are secrets updated?
- How is a secret revoked?



```
secure ? master   cat config.son
{
  "mysql_user": "root",
  "mysql_pass": "s3(Ret"
}
```

WHY NOT CONFIG MANAGEMENT?

- Centrally stored
- Eventually consistent
- No access control
- No auditing
- No revocation

WHY NOT (ONLINE) DATABASES?

- RDBMS, Consul, ZooKeeper, etc
- Not designed for secrets
- Limited access controls
- Typically plaintext storage
- No auditing or revocation abilities

HOW TO HANDLE SECRET SPRAWL?

- Secret material is distributed
- Who has access?
- When were secrets used?
- What is the attack surface?
- What do we do in the event of a compromise?

STATE OF THE WORLD 1.0

- Secret sprawl
- Decentralized keys
- Limited visibility
- Poorly defined “break glass” procedures

SECRET MANAGEMENT 2.0

Vault

MODERN SECRETS MANAGEMENT

VAULT GOALS

- Single source for secrets
- Programmatic application access (Automated)
- Operator access (Manual)
- Practical security
- Modern data center friendly

VAULT FEATURES

- Secure secret storage (in-memory, Consul, file, postgres, and more)
- Auditing
- Rich ACLs
- Multiple client authentication methods
- Leasing, renewal, and revocation
- Dynamic secrets

SECURE SECRET STORAGE

- Data is encrypted in transit and at rest
- 256bit AES in GCM mode
- TLS 1.2 for clients
- No HSM required

AUDITING

- Pluggable Audit Backends
- Request and Response Logging
- Prioritizes Safety over Availability
- Secrets Hashed in Audits
  - Searchable, but not reversible

RICH ACLS

- Role Based Policies
- Restrict access to “need to know”
- Default Deny, must be explicitly allowed

FLEXIBLE AUTH

- Pluggable Backends
- Tokens, **GitHub**, AppID, User/Pass, TLS Certs
- Machine-Oriented vs Operator-Oriented

LEASING, RENEWAL, AND REVOCATION

- Every Secret has a Lease*
- Secrets are revoked at the end of the lease unless renewed
- Secrets may be revoked early by operators
  - "Break Glass" procedure
- Dynamic Secrets make leases enforceable
  - Not possible for arbitrary secrets

DYNAMIC SECRETS

- Never provide “root” credentials to clients
- Provide limited access credentials based on role
- Generated **on demand** when requested
- Leases are enforceable via revocation
- Audit trail can identify point of compromise

DYNAMIC SECRETS

- Pluggable Backends
- AWS, Cassandra, Consul, MySQL, PostgreSQL, MSSQL, …
- Grow support over time

INTEGRATING MYSQL

MySQL user management

- Clunky to manage many users
- Difficult to manage passwords
- Password expiration only recently
- Password validation only recently

MySQL user management

- Hardcoded in applications
- Plaintext secrets
- Difficult to rotate

How does Vault help?

- Creates users with high entropy secrets
- Secrets have aggressive expiration
- Secrets can easily be revoked

Create user pattern

```
$ vault write mysql/roles/readonly \
    sql="CREATE USER '{{name}}'@'%' IDENTIFIED BY '{{password}}';GRANT SELECT ON
*.* TO '{{name}}'@'%';"
```

Read from Vault

```
$ vault read mysql/creds/readonly
Key             Value
lease_id        mysql/creds/readonly/b9b1fbb4-5ef8-1977-1fd2-ed21912e6288
lease_duration  600
lease_renewable true
password        04f9d427-5ea4-8ce4-8e92-30c5cdcb5f7e
username        root-6dd78551-dd
```

Read from Vault

```
$ vault read mysql/creds/readonly
Key             Value
lease_id        mysql/creds/readonly/b9b1fbb4-5ef8-1977-1fd2-ed21912e6288
lease_duration  600
lease_renewable true
password        04f9d427-5ea4-8ce4-8e92-30c5cdcb5f7e
username        github-6dd78551-dd
```

```
$ mysql -ugithub-6dd78551-dd -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.7.11 MySQL Community Server (GPL)

mysql> SELECT USER();
+-----------------------------+
| User()                      |
+-----------------------------+
| github-6dd78551-dd@localhost  |
+-----------------------------+
1 row in set (0.00 sec)

mysql> SHOW GRANTS;
+-----------------------------------------------+
| Grants for github-6dd78551-dd@%                |
+-----------------------------------------------+
| GRANT SELECT ON *.* TO 'github-6dd78551-dd'@'%' |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

Doesn’t MySQL do this natively?

- Proxy users (5.5+)
- Secrets have expiration (in 5.6+)
- Strong password policies can be implemented (5.6+)

But…

- Must manually add/remove users
- Must manually update passwords
- MySQL-specific authentication plugins

Why Vault?

- Centralized secret management with rest of organization
- Users easier to manage
- Vault generates high-entropy secrets by default
- Limit attack surface if secrets compromised
- “Breakglass” policies to revoke secrets

Remove single secret

```
$ vault revoke mysql/creds/readonly/6f1a7e70-cdd7-6954-eb57-b46da0c88ad5docker
Key revoked with ID ‘mysql/creds/readonly//6f1a7e70-cdd7-6954-eb57-b46da0c88ad5'.

$ mysql -uroot -p -e "SELECT user, host FROM mysql.user"
Enter password:
+----------------------+------+
| user                 | host |
+----------------------+------+
| root                 | %    |
+----------------------+------+
```

Remove all MySQL secrets

```
$ mysql -uroot -p -e "SELECT user, host FROM mysql.user"
+------------------+------+
| user             | host |
+------------------+------+
| root             | %    |
| root-6a1e1fbb-37 | %    |
| root-7dc68b1f-dd | %    |
| root-fcb6e200-87 | %    |
+------------------+------+

$ vault revoke --prefix mysql
Key revoked with ID 'mysql'.

$ mysql -uroot -p -e "SELECT user, host FROM mysql.user“
+------+------+
| user | host |
+------+------+
| root | %    |
+------+------+
```

Vault for Direct Access

- Great for third-party access: consultants, auditors, etc
- Only create a single user to grant appropriate access
- Or create user per vendor or role
- Aggressive secrets expiration limits risk of password exposure
- Authentication plugins

Vault Authentication

- Github
- LDAP

```
$ vault policies github
path "mysql/*" {
  policy = "write"
}

$ vault auth -method=github token=$GITHUB_TOKEN
Successfully authenticated!
token: 920b84f1-4ca9-33aa-4946-f046ef0b3f53
token_duration: 2591999
token_policies: [default, github]

$ vault read mysql/creds/readonly
Key             Value
lease_id        mysql/creds/readonly/6b4c559e-5008-f813-92af-19eaa41cbac4
lease_duration     600
lease_renewable true
password           8bb914dc-9619-3c87-ba4e-18f1ec602e98
username           github-dte-7a311
```

Vault for MySQL Applications

- Dynamic config via consul-template
- Secrets not stored in plaintext
- consul-template automatically renews

Vault Auditing

- Log access
- Supports writing syslog and file
- Hashes access so secrets are not stored in plaintext

Audit log

```
{"time":"2016-04-07T19:20:46Z","type":"request","auth":{"display_name":"root","policies":
["root"],"metadata":null},"request":{"operation":"read","client_token":"hmac-
sha256:ab960b87941cb0ad31477bec09b31671457c1967b15a89bf8574bae528c11ffa","pa
th":"mysql/creds/readonly","data":null,"remote_address":"127.0.0.1"},"error":""}
```

Considerations

- Current implementation requires multiple mysql mount per unique environment.
- Auditing only access requests to Vault, not whether used on DB or what was done.

OPERATING VAULT

HIGH AVAILABILITY

- Consul used for leader election
- Active/Standby
- Automatic failover

UNSEALING THE VAULT

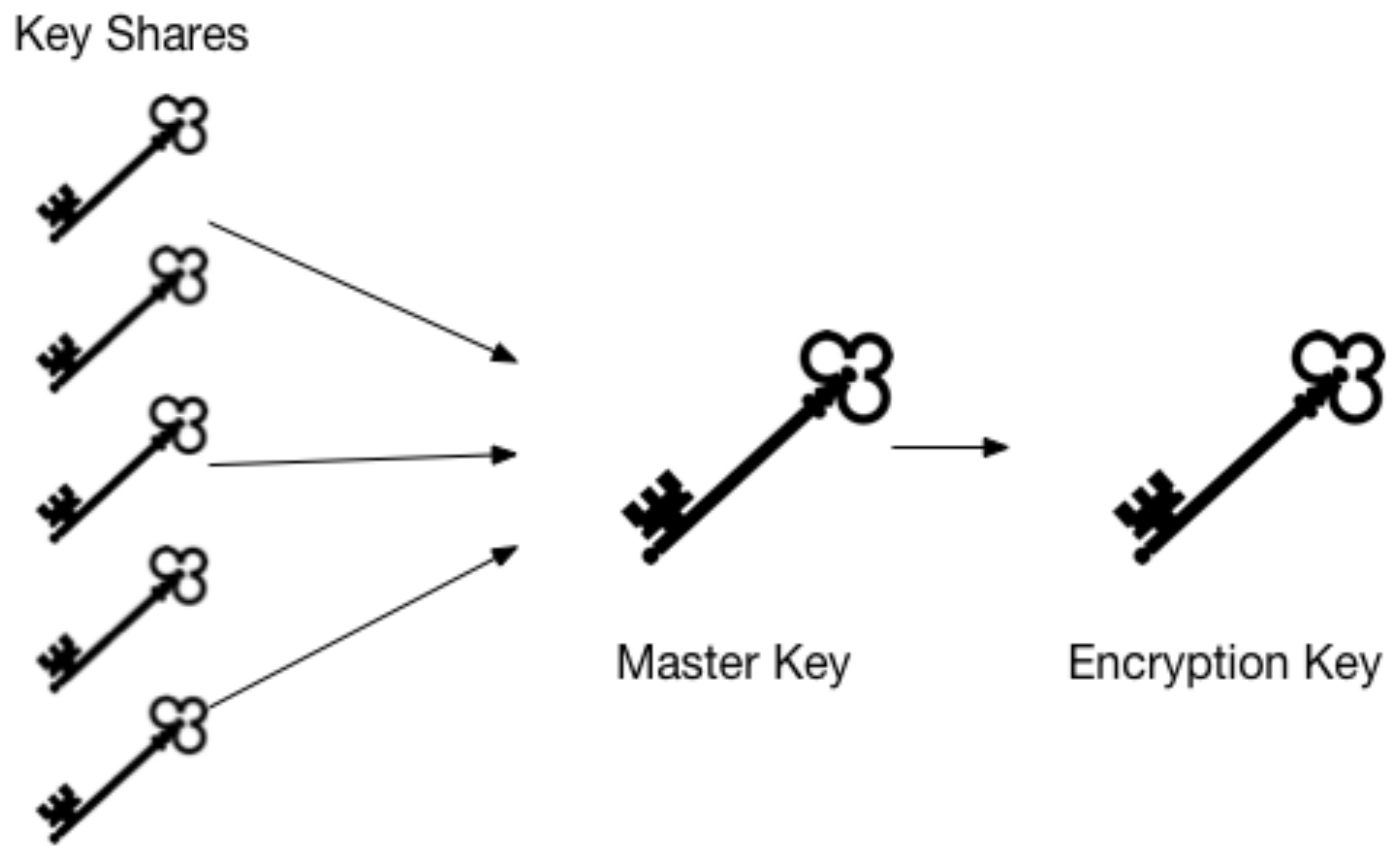
- Data in Vault encrypted
- Vault requires encryption key
- Must be provided *online*

WATCHING THE WATCHMEN

- Master Key is the “key to the kingdom”
- All data could be decrypted
- Protect against insider attack
- Two-Man Rule

SHAMIR SECRET SHARING

- Protect Encrypt Key with Master Key
- Split Master Key into **N** shares
- **T** shares to recompute Master
- Quorum of key holders required to unseal
  - Default N:5, T:3



SUMMARY

- Solves the “Secret Sprawl Problem”
- Protects against external threats (Cryptosystem)
- Protects against internal threats (ACLs and Secret Sharing)

DEMONSTRATION

Demo Github Authentication

- Setup vault, unseal (Docker containers), setup file audit
- Create github auth config
- Authenticate
- Create mysql user
- Request secret
- Use secret
- Revoke secret
- Verify Audit log

BUILDING ON VAULT

SECURITY FOUNDATION

- Base of Trust
- Core Infrastructure
- Flexible Architecture
- Foundation for Security Infrastructure

PERSONALLY IDENTIFIABLE INFORMATION

- PII information is everywhere
  - SSN, CC#, OAuth Tokens, etc.
  - Email? Physical address?
- Security of storage?
- Scalability of storage?
- Audibility of access?

PII WITH VAULT

- “transit” backend in Vault
- Encrypt/Decrypt data in transit
- Avoid secret management in client applications
- Builds on Vault foundation

TRANSIT BACKEND

- Web server has no encryption keys
- Requires two-factor compromise (Vault + Database)
- Decouples storage from encryption and access control

EXTENSIBLE

- PKI backend for Certificate Authority + Signing
    - Mutual TLS for Applications
- SSH backend for SSH key management
    - “vault ssh” CLI command, dynamic keys or one-time-passwords

VAULT IN PRACTICE

USING VAULT

- API Driven
- JSON/HTTPS
- Rich CLI for humans and scripts
- Rich client libraries

APPLICATION INTEGRATION

- Vault-aware
  - Native client libraries (go, ruby, rails, python, node, and more)
  - Secrets only in-memory
  - Safest but high-touch

CONSUL TEMPLATE INTEGRATION

- Secrets templatized into application configuration
- Vault is transparent
- Lease management is automatic
- Non-secret configuration still via Consul

```
secure ? master  cat secrets.yml.ctmpl

{{  with $secret := vault "mysql/creds/production" }}
---
production:
  adapter: mysql
  database: mysql.service.consul
  username: {{$secret.Data.username}}
  password: {{$secret.Data.password}}
  pool: {{key "production/mysql/pool"}}
{{ end }}
```

THANK YOU!

QUESTIONS?

hashicorp/vault

https://vaultproject.io https://pythian.com

security@hashicorp.com