Vault Configuration

How does Vault encrypt data?

Static secrets

Dynamic secrets

Secret management

Copyright © 2021 by Elephant Scale, All Rights Reserved

How does Vault encrypt data?

How does Vault encrypt data?

Static secrets

Dynamic secrets

Secret management

Copyright © 2021 by Elephant Scale, All Rights Reserved

How does Vault encrypt data?

- Vault encrypts data by leveraging a few key sources.
  - The libraries that Vault uses, or the cryptography modules
  - Golang's crypto and
  - x/crypto libraries that are part of the golang language.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Vault cryptography libraries

- Kerckhoffs's principle
  - A cryptosystem should be secure even if everything about the system, except the key, is public knowledge
  - No security-by-obscurity
- Shannon's maxim
  - The enemy knows the system
- HashiCorp principle
  - the source code for how you encrypt and decrypt data should be open source

Copyright © 2021 by Elephant Scale, All Rights Reserved

Cryptography at HashiCorp

- Leverage those Golang crypto and x/crypto libraries
  - for heavy lifting associated with encrypting and decrypting data
- Those libraries contain the methods and functions
  - implementations of various algorithms like AES256
  - for internally encrypting data and decrypting data
  - allowing you to leverage cryptography without

Copyright © 2021 by Elephant Scale, All Rights Reserved

Vault key management

- How does Vault manage keys that are associated with these cryptographic functions?
- Vault handles this all through its own internal keyring
  - open source
  - not require a user to integrate into something like an HSM (Hardware security module) unless they want to.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Entropy and Cybersecurity

- Entropy is the foundation upon which all cryptographic functions operate.
  - It is a measure of the randomness or diversity of a data-generating function.
  - Data with full entropy is completely random and no meaningful patterns can be found.
  - Low entropy data provides the ability or possibility to predict forthcoming generated values

Copyright © 2021 by Elephant Scale, All Rights Reserved

Vault cryptography integrations

- Two key things that we need to focus on
  - entropy
  - links to other cryptographic standards or follows cryptographic standards

Copyright © 2021 by Elephant Scale, All Rights Reserved

Entropy in Vault

- The entropy of Vault's encryption varies depending upon what system Vault is being run on
  - Golang's crypto and x/crypto libraries use a randomized function that calls different entropy pools
    - Entropy pool for Windows
    - Entropy pool for Linux

Copyright © 2021 by Elephant Scale, All Rights Reserved

Is your entropy sufficient?

- What is sufficient entropy for Vault?
  - In many ways, those random number generators are sufficiently random
  - For some Vault Enterprise customers
    - Seal Wrap

Copyright © 2021 by Elephant Scale, All Rights Reserved

Seal Wrap

- Seal Wrap allows to leverage external cryptographic modules
  - Similar to HSM (Hardware Security Module)
  - protect and wrap the cryptographic infrastructure of Vault
  - operate within very rigorous cryptographic environments in a way that doesn't violate
    - their story around entropy,
    - their story around key rotation,
    - key management, etc.

Copyright © 2021 by Elephant Scale, All Rights Reserved

US standard of FIPS 140-2

- FIPS 140-2 level one, two, and three environments
- with Seal Wrap, Vault can
  - Be wrapped with another layer of cryptography
  - from a separate, very secure source such as an HSM

Copyright © 2021 by Elephant Scale, All Rights Reserved

Static secrets

How does Vault encrypt data?

Static secrets

Dynamic secrets

Secret management

Copyright © 2021 by Elephant Scale, All Rights Reserved

Key/Value Secrets Engine

- Vault can be used to store any secret in a secure manner.
- The secrets may be
  - SSL certificates and keys for your organization's domain
  - credentials to connect to a corporate database server, etc.
- Storing such sensitive information in plaintext is not desirable.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Static secret scenario

- Personas
  - **devops** with privileged permissions to write secrets
  - **apps** reads the secrets from Vault
- Challenge
  - Developers use a single admin account to access a third-party app (e.g. Splunk)
  - and anyone who knows the user ID and password can log in as an admin
  - SSH keys to connect to remote machines are shared and stored as a plaintext
  - An app integrates with LDAP, and its configuration information is in a plaintext

Copyright © 2021 by Elephant Scale, All Rights Reserved

Solution

- Use Vault
  - centralized secret storage
  - Secure any sensitive information
- Vault encrypts these secrets using 256-bit AES in GCM mode
  - with a randomly generated nonce prior to writing them to its persistent storage.
  - The storage backend never sees the unencrypted value
  - even if an attacker gained access to the raw storage, they wouldn't be able to read your secrets.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Lab: Static Secret

- Continue with lab21
- **NOTE:** For the purpose of this lab, we will use the root token to work with Vault.
- Best practice
  - root tokens are only used for just enough initial setup or in emergencies.
  - As a best practice, use tokens with appropriate set of policies based on your role in the organization.
- https://github.com/elephantscale/vault-consul-labs-answers/tree/main/lab21

Copyright © 2021 by Elephant Scale, All Rights Reserved

Dynamic secrets

How does Vault encrypt data?

Static secrets

Dynamic secrets

Secret management

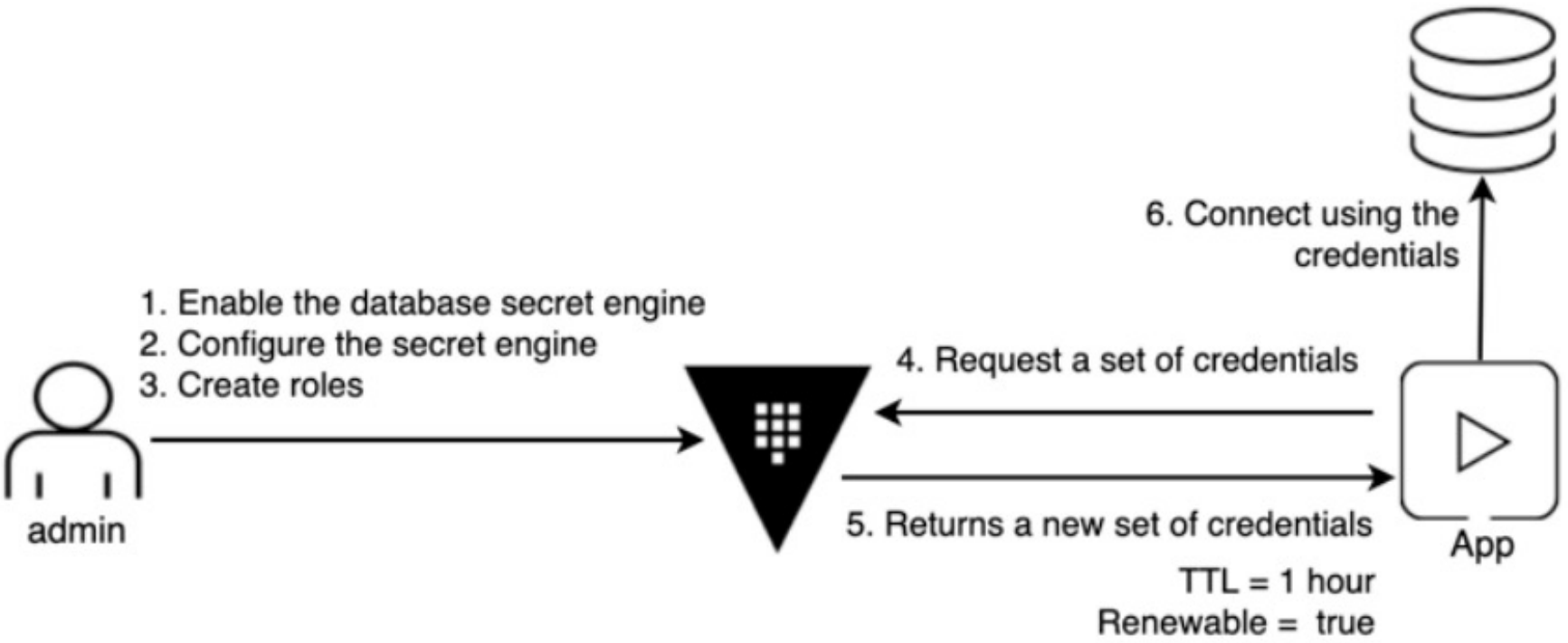
Copyright © 2021 by Elephant Scale, All Rights Reserved

Dynamic Secrets: Database Secrets Engine

- Challenge
  - Data protection is a top priority, and database credential rotation is a critical part of any data protection initiative. Each role has a different set of permissions granted to access the database. When a system is attacked by hackers, continuous credential rotation becomes necessary and needs to be automated.
- Solution
  - Applications ask Vault for database credentials rather than setting them as environment variables. The administrator specifies the TTL of the database credentials to enforce its validity so that they are automatically revoked when they are no longer used.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Dynamic secrets



Copyright © 2021 by Elephant Scale, All Rights Reserved

Start Postgres

- The lab requires a Postgres database. Docker provides a Postgres server image that satisfies this requirement.
  - NOTE: This lab works for an existing Postgres database given appropriate credentials and connection information.
  - Pull a Postgres server image with `docker`.

```
1 docker pull postgres:latest
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Create a user

- Create a Postgres database with a root user named root with the password rootpassword.

```
docker run \
       --name postgres \
       --env POSTGRES_USER=root \
       --env POSTGRES_PASSWORD=rootpassword \
       --detach  \
       --publish 5432:5432 \
       postgres
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Talk to DB

- The credentials generated by the Vault role in the "create a role" step requires a role named ro that has been granted the ability to read all tables.
- Connect to the Postgres database via the CLI within the postgres container.

```
1 docker exec -it postgres psql
```

- Create a role named ro.

```
1 CREATE ROLE ro NOINHERIT;
```

- Grant the ability to read all tables to the role named ro

```
1 Grant the ability to read all tables to the role named ro.
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Start Vault

- In another terminal, start a Vault dev server with root as the root token.

```
1 vault server -dev -dev-root-token-id root
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Export environment variables

```
1 export VAULT_ADDR=http://127.0.0.1:8200
```

- Export an environment variable for the vault CLI to authenticate with the Vault server.

```
1 export VAULT_TOKEN=root
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Scenario

- In this lab, you are going to configure the PostgreSQL secrets engine, and create a read-only database role. The Vault-generated PostgreSQL credentials will only have read permission.
  - Enable the database secrets engine
  - Configure PostgreSQL secrets engine
  - Create a role
  - Request PostgreSQL credentials
  - Manage leases
  - Define a password policy
  - Define a username template

Copyright © 2021 by Elephant Scale, All Rights Reserved

Enable the database secrets engine

- Enable the database secrets engine at the database/ path.

```
1 vault secrets enable database
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Configure PostgreSQL secrets engine

- (Persona: admin)
- The database secrets engine supports many databases through a plugin interface. To use a Postgres database with the secrets engine requires further configuration with the postgresql-database-plugin plugin and connection information.
- Configure the database secrets engine with the connection credentials for the Postgres database.

```
vault write database/config/postgresql \
     plugin_name=postgresql-database-plugin \
     connection_url="postgresql://{{username}}:{{password}}@localhost:5432/postgres?sslmode=disable" \
     allowed_roles=readonly \
     username="root" \
     password="rootpassword"
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Create a role

- (Persona: admin)
- In configure Postgresql secrets engine step, you configured the PostgreSQL secrets engine with the allowed role named readonly. A role is a logical name within Vault that maps to database credentials. These credentials are expressed as SQL statements and assigned to the Vault role.
- Define the SQL used to create credentials.

```
tee readonly.sql <<EOF
CREATE ROLE "{{name}}" WITH LOGIN PASSWORD '{{password}}' VALID UNTIL '{{expiration}}' INHERIT;
GRANT ro TO "{{name}}";
EOF
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Create the role named readonly

```
vault write database/roles/readonly \
      db_name=postgresql \
      creation_statements=@readonly.sql \
      default_ttl=1h \
      max_ttl=24h
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Request PostgreSQL credentials

- (Persona: apps)
- The applications that require the database credentials read them from the secret engine's readonly role.
- Read credentials from the readonly database role.

```
1 vault read database/creds/readonly
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Result

- You will get something like this

```
(base) mark@mark-workstation:~$ vault read database/creds/readonly
Key                Value
---                -----
lease_id           database/creds/readonly/eUcaTQetKjQyvmEdmfs37L8D
lease_duration     1h
lease_renewable    true
password           -EzpkpKSWqXw94pHDER4
username           v-token-readonly-1R8polxUd27lhfJslPU8-1620273816
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Validation

- Connect to the Postgres database via the CLI within the postgres container.
- Your system prompt is replaced with a new prompt root=#. Commands issued at this prompt are executed against the Postgres database running within the container.
- List all the database users.

```
1 SELECT usename, valuntil FROM pg_user;
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Users

- You will see this kind of output

```
root=# SELECT usename, valuntil FROM pg_user;
                    usename                    |        valuntil
-----------------------------------------------+------------------------
 root                                          |
 v-token-readonly-1R8polxUd27lhfJslPU8-1620273816 | 2021-05-06 05:03:41+00
(2 rows)
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Manage leases

- (Persona: admin)
- The credentials are managed by the lease ID and remain valid for the lease duration (TTL) or until revoked. Once revoked the credentials are no longer valid.
- List the existing leases.

```
1 vault list sys/leases/lookup/database/creds/readonly
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Leases

- All valid leases for database credentials are displayed.
- Create a variable that stores the first lease ID.

```
1 LEASE_ID=$(vault list -format=json sys/leases/lookup/database/creds/readonly | jq -r ".[0]")
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Lease renew

- Renew the lease for the database credential by passing its lease ID.

```
1 vault lease renew database/creds/readonly/$LEASE_ID
```

```
Key                Value
---                -----
lease_id           database/creds/readonly/eUcaTQetKjQyvmEdmfs37L8D
lease_duration     1h
lease_renewable    true
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Revoke

- Revoke the lease without waiting for its expiration.

```
1 vault lease revoke database/creds/readonly/$LEASE_ID
```

- Observe success status

Copyright © 2021 by Elephant Scale, All Rights Reserved

List leases

- List the existing leases.

```
1 vault list sys/leases/lookup/database/creds/readonly
```

- The lease is no longer valid and is not displayed.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Read new creds

- Read new credentials from the readonly database role.

```
1 vault read database/creds/readonly
```

```
Key                Value
---                -----
lease_id           database/creds/readonly/iOU9So8m8iBspondStVcXppm
lease_duration     1h
lease_renewable    true
password           9CE-bA449zIjlfAPvkqh
username           v-token-readonly-DPR7a33Khp4pSZTbuBRo-1620276553
```

- [illegible]

Copyright © 2021 by Elephant Scale, All Rights Reserved

Revoke

- Revoke all the leases with the prefix database/creds/readonly.

```
1 vault lease revoke -prefix database/creds/readonly
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Define a password policy

- The database secret engines generate passwords that adhere to a default pattern that may be overridden with a new password policy. A policy defines the rules and requirements that the password must adhere to and can provide that password directly through a new endpoint or within secrets engines.
- The passwords you want to generate adhere to these requirements.
  - length of 20 characters
  - at least 1 uppercase character
  - at least 1 lowercase character
  - at least 1 number
  - at least 1 symbol

Copyright © 2021 by Elephant Scale, All Rights Reserved

Password policy

- Define this password policy in a file named example_policy.hcl.

```
tee example_policy.hcl <<EOF
length=20

rule "charset" {
  charset = "abcdefghijklmnopqrstuvwxyz"
  min-chars = 1
}

rule "charset" {
  charset = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  min-chars = 1
}

rule "charset" {
  charset = "0123456789"
  min-chars = 1
}

rule "charset" {
  charset = "!@#$%^&*"
  min-chars = 1
}
EOF
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Create policy

- Create a Vault password policy named example with the password policy rules defined in example_policy.hcl.

```
vault write sys/policies/password/example policy=@example_policy.hcl
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Generate

- Generate a password from the example password policy.

```
vault read sys/policies/password/example/generate
```

```
Key          Value
---          -----
password     zUKMWPOr821D%DZHklt%
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Apply the password policy

- Configure the database secrets engine with the example password policy.

```
vault write database/config/postgresql \
     password_policy="example"
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Read creds

- Read credentials from the readonly database role.

```
1 vault read database/creds/readonly
```

- You will get similar output

```
(base) mark@mark-workstation:~$ vault read database/creds/readonly
Key                Value
---                -----
lease_id           database/creds/readonly/mqiGqFkjSADfba6zNsijV5mC
lease_duration     1h
lease_renewable    true
password           *bDTIi#Sl&EE3xX4@evE
username           v-token-readonly-OKTpYkC7wj9lqa3mgBwS-1620277767
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Define a username template

- The database secret engines generate usernames that adhere to a default pattern. A customized username template may be provided to meet the needs of your organization.
  - Ensure that custom username templates include enough randomness to prevent the same username being generated multiple times.
- Read credentials from the readonly database role.

```
1 vault read database/creds/readonly
```

```
Key                Value
---                -----
lease_id           database/creds/readonly/5ix3yTUwLJyNvb0CwAuHipDH
lease_duration     1h
lease_renewable    true
password           2PGeTe4EqDVueP@6Cwnp
username           v-token-readonly-SqapepE0EGKAQyQcXxDN-1620277969
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Configure

- Configure the database secrets engine with the username template.

```
vault write database/config/postgresql \
    username_template="myorg-{{.RoleName}}-{{unix_time}}-{{random 8}}"
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Read creds

- Read credentials from the readonly database role.

```
1 vault read database/creds/readonly
```

```
Key                 Value
---                 -----
lease_id            database/creds/readonly/4sYzt5kxfL5VV1Hgqq30ombo
lease_duration      1h
lease_renewable     true
password            k1NEHfC76bsBJ@ByLVQI
username            myorg-readonly-1620278291-FIFEqmw4
```

Copyright © 2021 by Elephant Scale, All Rights Reserved

Secret management

How does Vault encrypt data?

Static secrets

Dynamic secrets

Secret management

Copyright © 2021 by Elephant Scale, All Rights Reserved

Secrets engines

- Static Secrets: Key/Value Secrets Engine
- Versioned Key/Value Secrets Engine
- Cubbyhole Response Wrapping
- Dynamic Secrets: Database Secrets Engine
- Couchbase Secrets Engine
- Database Secrets Engine with MongoDB

Copyright © 2021 by Elephant Scale, All Rights Reserved

Secrets engines cont'd

- Database Root Credential Rotation
- Database Static Roles and Credential Rotation
- Active Directory Service Account Check-out
- OpenLDAP Secrets Engine
- Azure Secrets Engine
- A dozen more secret engines

Copyright © 2021 by Elephant Scale, All Rights Reserved

Versioned Key/Value Secrets Engine

- The Static Secrets lab introduced the basics of working with key-value secrets engine.
- Vault 0.10 introduced K/V Secrets Engine v2 with Secret Versioning.
- Let us look at the key-value secrets engine v2 features.

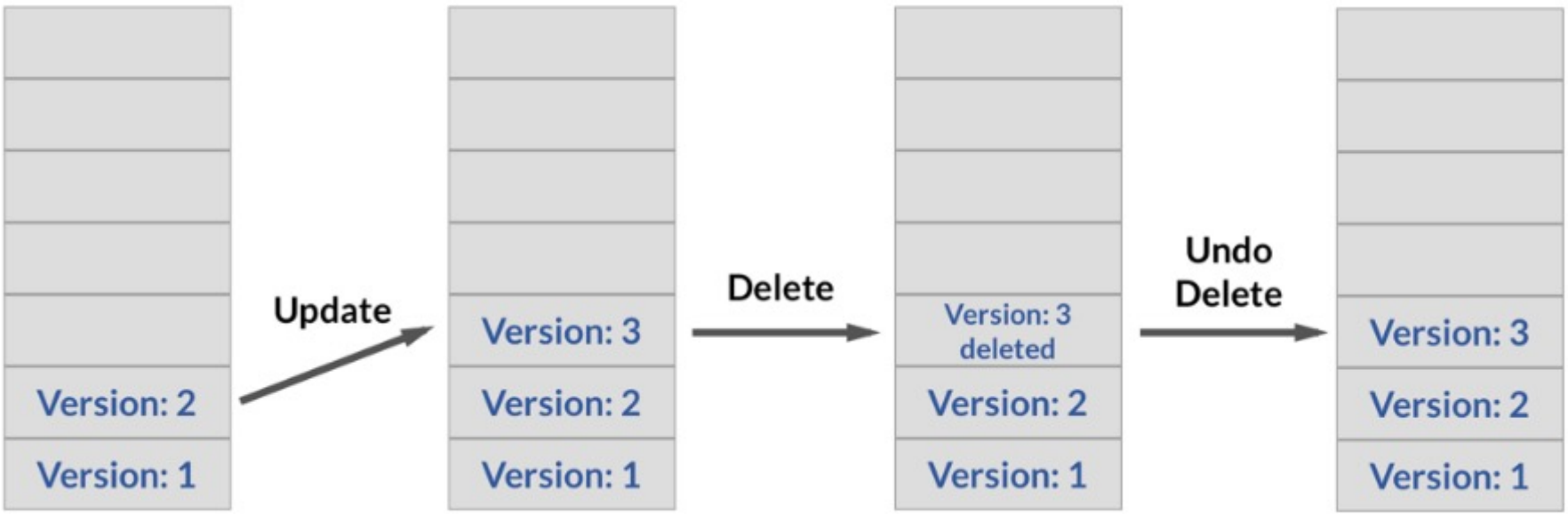
Copyright © 2021 by Elephant Scale, All Rights Reserved

Versioned Key/Value scenario

- The KV secrets engine v1 does not provide a way to version or roll back secrets
- This made it difficult to recover from
  - unintentional data loss or
  - overwrite when more than one user is writing at the same path.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Versioned Key/Value solution



Copyright © 2021 by Elephant Scale, All Rights Reserved

Versioned Key/Value solution

- Run the version 2 of KV secrets engine
  - it can retain a configurable number of secret versions.
  - This enables older versions' data to be retrievable in case of unwanted deletion or updates of the data. In addition,
  - its Check-and-Set operations can be used to protect the data from being overwritten unintentionally.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Cubbyhole Secret Engine

- The term
  - cubbyhole comes from an Americanism where you get a "locker" or "safe place" to store your belongings or valuables.
  - It is not possible to reach into another token's cubbyhole even as the root user
  - By contrast, the secrets in the key/value secrets engine are accessible to any token

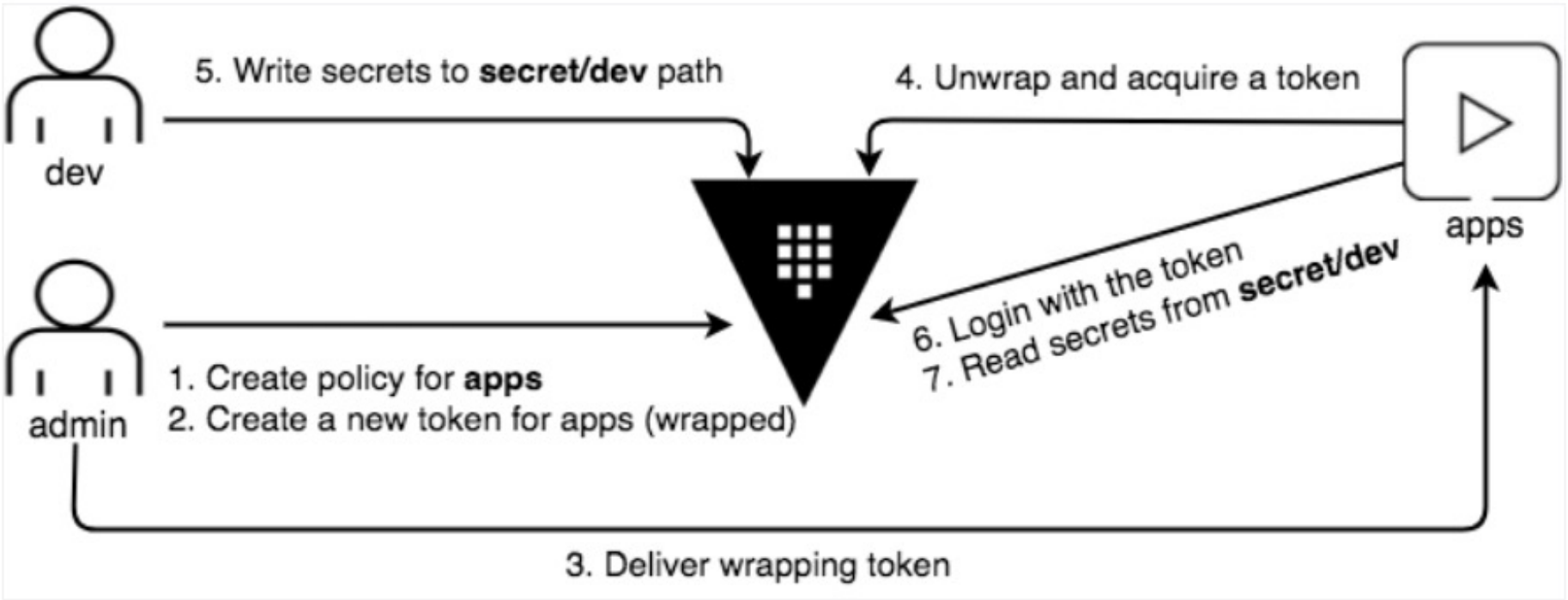
Copyright © 2021 by Elephant Scale, All Rights Reserved

Cubbyhole Scenario

- Personas
  - The end-to-end scenario described in this tutorial involves two personas:
    - admin with privileged permissions to create tokens
    - apps trusted entity retrieving secrets from Vault
- Challenge
  - In order to tightly manage the secrets, you set the scope of who can do what using the Vault policy and attach that to tokens, roles, entities, etc.
  - How can you securely distribute the initial token to the trusted entity?

Copyright © 2021 by Elephant Scale, All Rights Reserved

Cubbyhole solution



Copyright © 2021 by Elephant Scale, All Rights Reserved

Cubbyhole solution

- the initial token is stored in the cubbyhole secrets engine.
- The wrapped secret can be unwrapped using the single-use wrapping token.
- Even the user or the system created the initial token won't see the original value.
- The wrapping token is short-lived and can be revoked just like any other tokens so that the risk of unauthorized access can be minimized.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Couchbase Secrets Engine

- Vault provides powerful dynamic credential lifecycle management for a wide range of database solutions.
- Let's look at the use of the database secrets engine to dynamically generate credentials for Couchbase Server database users.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Couchbase challenge

- Credential management is critical for secrets hygiene, but managing the lifecycle of credentials across numerous heterogeneous platforms such as database solutions can be cumbersome and time-consuming.
- An application requires credentials to access a specific database platform, but those credentials should never be hard coded into the application or allowed to persist past their useful lifetime.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Couchbase solution

- Vault provides a databases secrets engine with support for credential lifecycle management across a range of database solutions.
- Administrators can define credential attributes, such as attached policies and time to live values, such that the credential provides least privileged access for only the allowed time-frame and is revoked when no longer needed.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Database Secrets Engine with MongoDB

- Data protection is a top priority
  - and database credential rotation is a critical part of any data protection initiative.
- Vault's database secrets engine generates database credentials dynamically
- Each app instance can get unique credentials

Copyright © 2021 by Elephant Scale, All Rights Reserved

Database Root Credential Rotation

- Vault's database secrets engine gives every service instance gets a unique set of database credentials
- This reduces the manual tasks performed by the database administrator and makes the database access more efficient and secure.

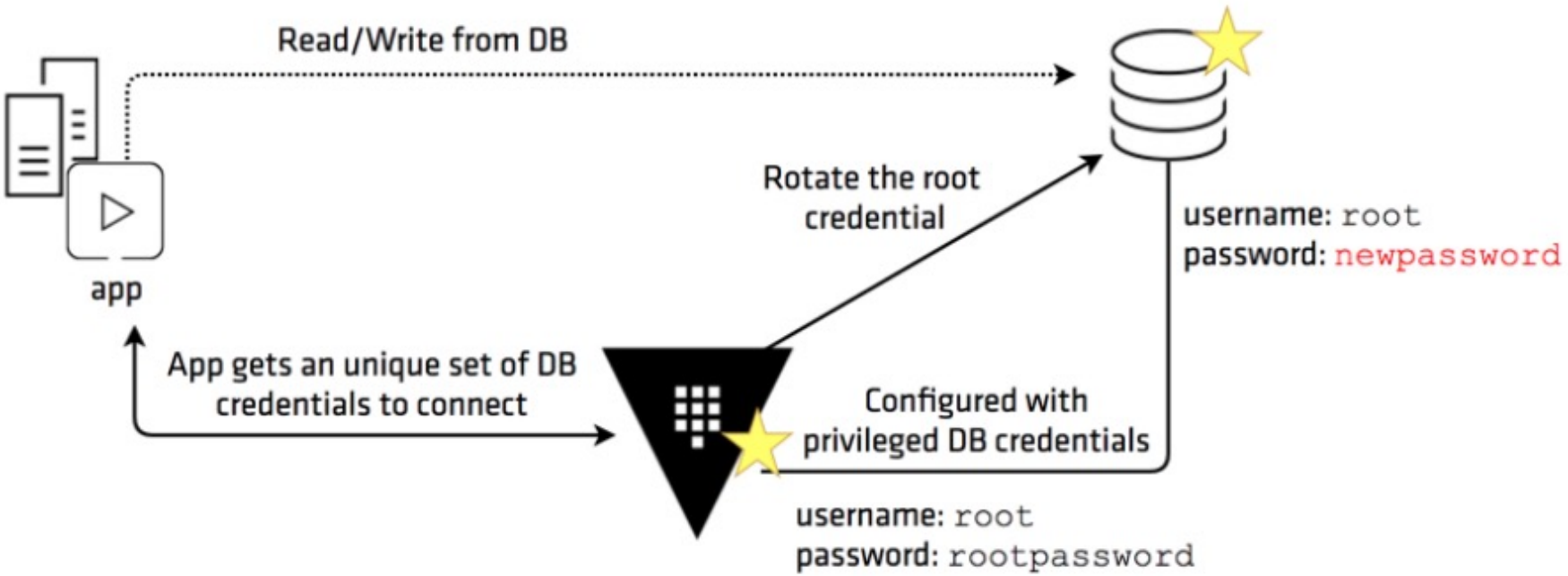
Copyright © 2021 by Elephant Scale, All Rights Reserved

Database Root Credential Rotation - challenge

- Vault is managing the database credentials on behalf of the database administrator
  - it must also be given a set of highly privileged credentials
- Credentials are often long-lived and never change once configured on Vault

Copyright © 2021 by Elephant Scale, All Rights Reserved

Database Root Credential Rotation - solution



Copyright © 2021 by Elephant Scale, All Rights Reserved

Database Root Credential Rotation - solution

- Use the Vault's `/database/rotate-root/:name` API endpoint to rotate the root credentials stored for the database connection.
- **Best Practice** : Use this feature to rotate the root credentials immediately after the initial configuration of each database.

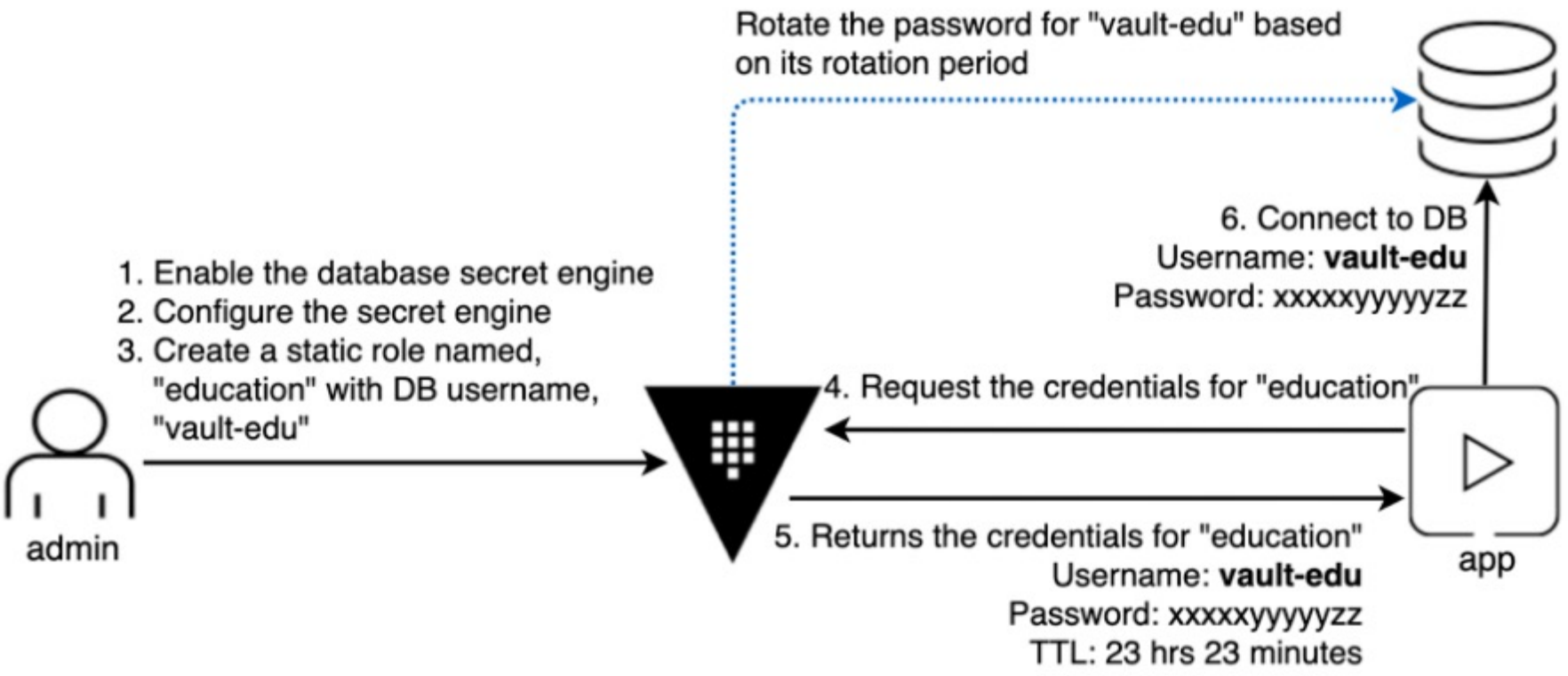
Copyright © 2021 by Elephant Scale, All Rights Reserved

Database Static Roles and Credential Rotation

- Vault creates a unique set of username and password with specified time-to-live (TTL) every time a client requests.
- This allows each application to have its own database credentials.
- Adopting the database secrets engine requires some code change in those applications.

Copyright © 2021 by Elephant Scale, All Rights Reserved

Database Static Roles and Credential Rotation - solution



Copyright © 2021 by Elephant Scale, All Rights Reserved

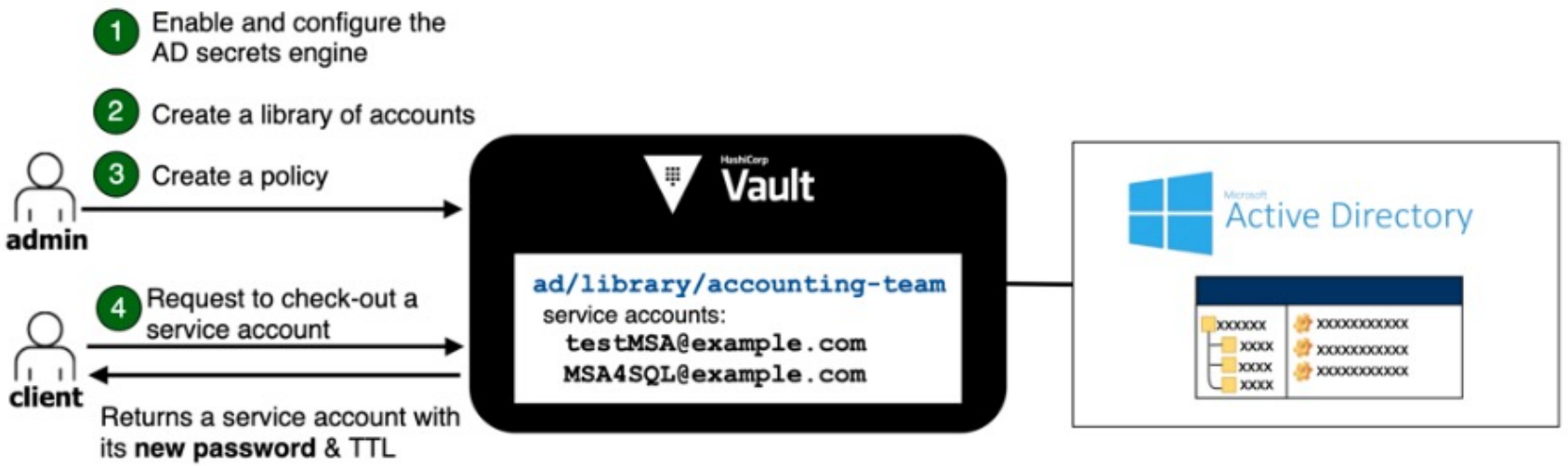
Active Directory Service Account

- Vault 0.10.2 introduced the Active Directory (AD) secrets engine which was designed to rotate the shared AD passwords dynamically.
- This allowed companies to reduce the risk of damage from a password leak.
- Challenge
  - Service accounts are limited based on Client Access License (CAL)
  - This makes credential rotation cumbersome

Copyright © 2021 by Elephant Scale, All Rights Reserved

Active Directory Service Account - solution

- The requester first checks out the account
- When done, they check the service account back
- Whenever a service account is checked back in, Vault rotates its password



Copyright © 2021 by Elephant Scale, All Rights Reserved