HashiCorp Vault

Vault Dynamic Secrets

Agenda

1. Dynamic Secrets
2. Dynamic Cloud Credentials
3. Dynamic Database Secrets
4. Other Secrets Engines
5. Q&A

01

Dynamic Secrets

What is a Dynamic Secret?

- Credentials (username/password, certificate) that are created when they are accessed
- Secrets do not exist until they are read
- Time-bound via TTL
  - Can be renewed*
  - Cleans itself up at its TTL
- Built in revocation mechanism

Why Dynamic Secrets?

Static Secrets

- Manage Credentials
  (e.g. create username and password for application A)
- Manually created, typically have a long life due to management overhead
- Manual lifecycle management

Dynamic Secrets

- Manage Intentions
  (e.g. Spring application needs database access)
- Dynamically created when needed at read time (do not exist until read)
- Automatic lifecycle: create, revoke, & rotate

Why Dynamic Secrets?

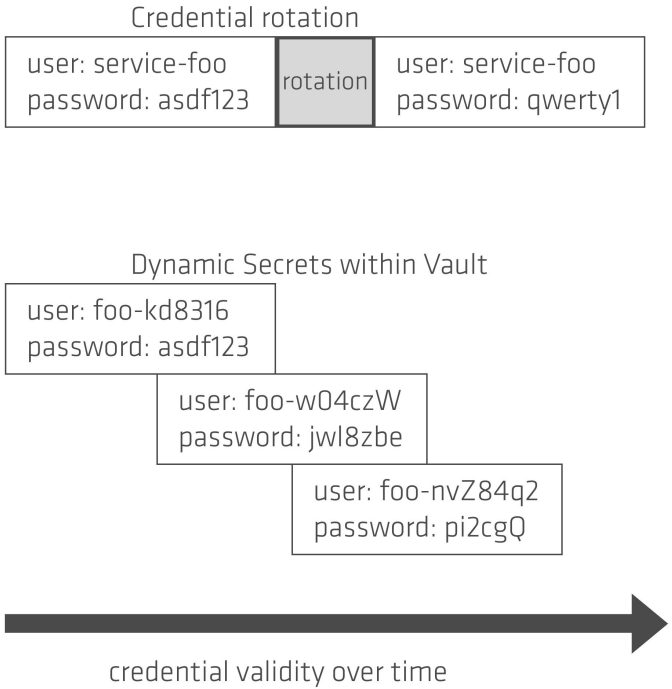
Static Secrets

- Often shared across applications and instances, hard to determine where secret is being used
- Exist at rest, can be leaked by operator, application, or logs
- Revocation requires operator intervention or action

Dynamic Secrets

- Vault knows which secrets each client has, simple to revoke and limit blast radius
- Do not exist until read, created on demand when needed
- Finite lifespan, automatically revoked / deleted / rotated via TTL.
- Unique credentials per client make forensics easy in the event of compromise or leak

Why Dynamic Secrets?



- No deadlock period during credential rotation
- Application logic for handling rotation scheduling not needed

Dynamic Credentials Example

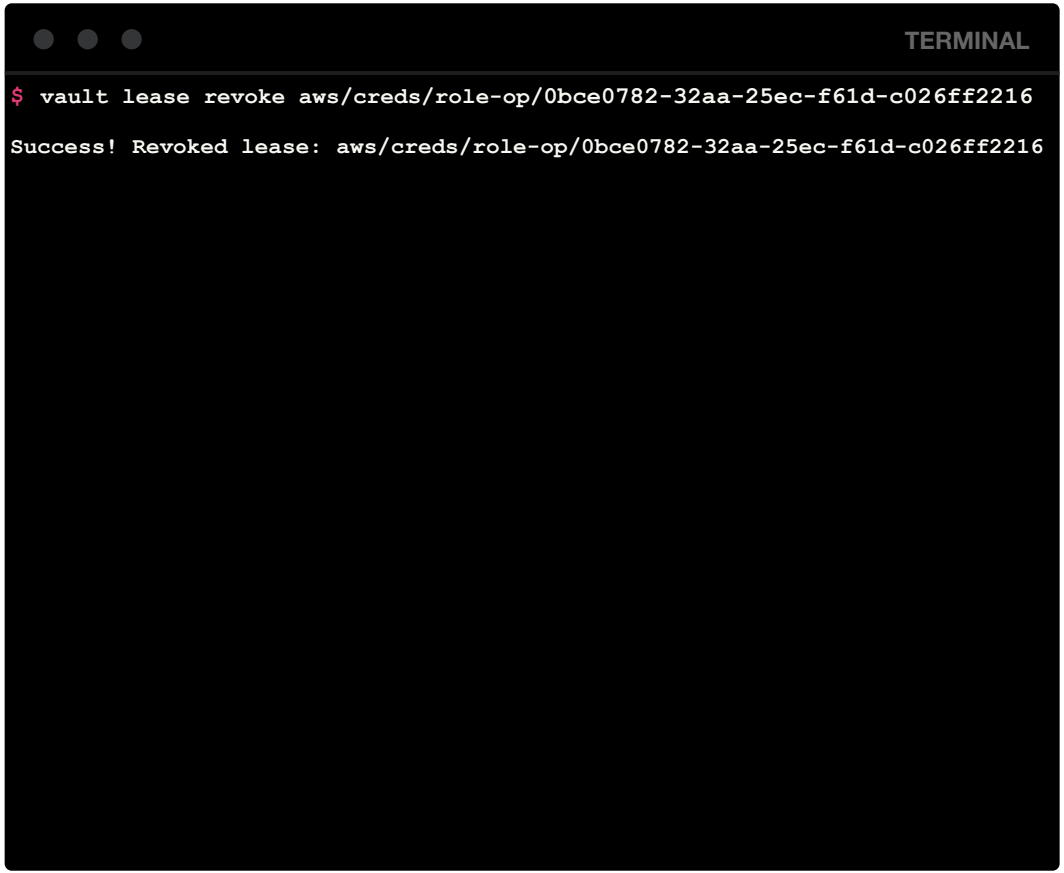
Generate AWS secret

CODE EDITOR

```
$ vault read aws/creds/role-op

Key                Value
---                -----
lease_id           aws/creds/role-op/0bce0782-32aa-25ec-f61d-c026ff2216
lease_duration     288h
lease_renewable    true
access_key         AKIAJELUDIANQGRXCTZQ
secret_key         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
security_token     <nil>
```

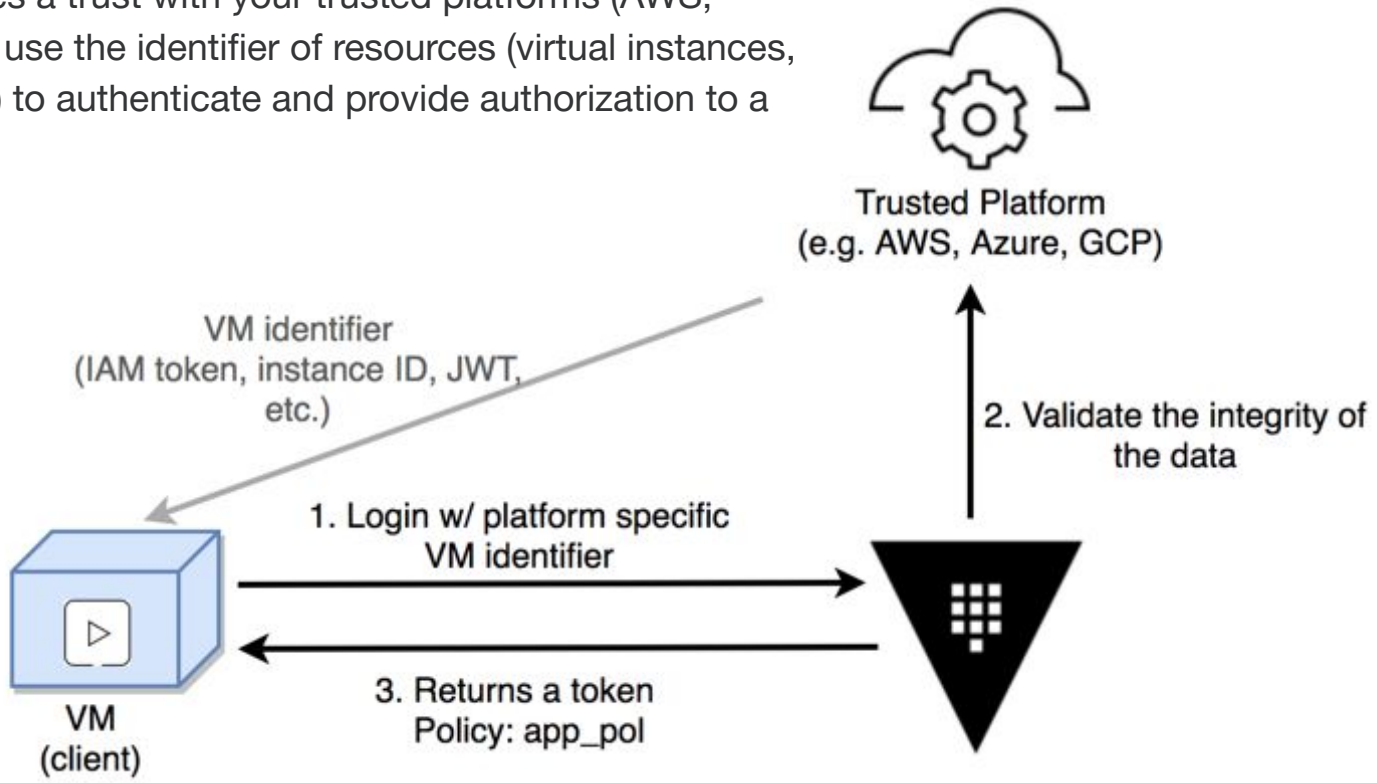
Dynamic Credentials Example

Revoke AWS secret



Trust and Platform Integration

Vault establishes a trust with your trusted platforms (AWS, Azure, GCP) to use the identifier of resources (virtual instances, containers, etc) to authenticate and provide authorization to a Vault token



TLDR: Dynamic Secrets

- Reduce time spent managing secrets
- Help teams achieve compliance objectives
- Improve security posture
  - Create a moving target for attackers
  - Minimize the risk of exposing credentials
  - Make forensics easier
  - Credential rotation & revocation becomes SOP

Dynamic Secret Types

Cloud credentials

Database credentials

Other secrets

Dynamic Secret Engines

Cloud Credentials

- AWS
- Azure
- AliCloud
- GCP

Database Secrets

- DB2
- Cassandra
- Couchbase
- Elasticsearch
- HanaDB
- InfluxDB
- MongoDB
- MongoDB Atlas
- MSSQL
- MySQL/MariaDB
- Oracle
- PostgresSQL
- Redshift
- Snowflake

Other secrets

- Active Directory
- Consul
- Terraform
- Nomad
- OpenLDAP
- PKI (Certificate)
- RabbitMQ
- Venafi

02

Dynamic Cloud Credentials

Dynamic Cloud Credentials

- Generate short-lived cloud credentials
- Scoped to specific policies in each cloud's policy language
- Secure privileged access flows
  - Operators need highly privileged cloud access for key administrative tasks, how can this be done securely?
  - Operators can generate short lived privileged credentials with an approval flow using **Vault Control Groups**
- Generate short-lived credentials for Terraform runs
  - Temporary cloud credentials with instance creation powers limited to the life of a single Terraform run

Azure Secrets Engine

[Documentation]

- Dynamically generates service principals along with role and group assignments
- Vault roles can be mapped to Azure roles
- Service principals are associated with a lease, when lease expires the service principal is deleted
- Calling an existing service principle will generate a dynamic password which is deleted when lease expires

GCP Secrets Engine

Documentation

- Dynamically generates service account keys and OAuth tokens based on IAM policies
- Service account keys are associated with a lease, when lease expires the account key is revoked
- New Service Accounts do not need to be created for batch jobs or short-term access
- Supports rolesets, static accounts, access tokens, and service account keys

AWS Secrets Engine

[Documentation]

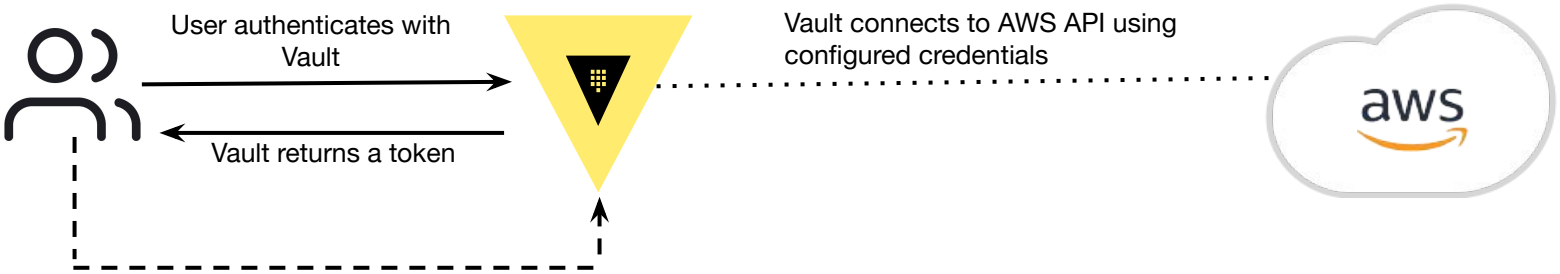
- Dynamically generates credentials based on IAM policies, can be mapped to internal auth methods like LDAP/OIDC
- No clicking in the UI is required, credentials are revoked when Vault lease expires
- Three supported credential types:
  - ***iam_user:*** Dynamically generates ephemeral IAM user, attaches IAM policies and generates an access key and secret key
  - ***assumed_role:*** Typically used for cross-account access, Vault calls sts:AssumeRole and returns the access key, secret key, and session token
  - ***federation_token:*** Vault calls sts:GetFederationToken passing AWS policy and returns access key, secret key and session token

Configure AWS Dynamic Credentials

CODE EDITOR

```
# Enable AWS Secrets Engine
$ vault secrets enable aws

# Configure credentials for Vault to communicate to AWS for generation
# of IAM credentials

$ vault write aws/config/root \
    access_key=AKIAJWVN5Z4FOFT7NLNA \
    secret_key=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 \
    region=us-east-1

# Configure a Vault role that maps to a set of AWS permissions and
# an AWS credential type for credential generation

$ vault write aws/roles/my-role \
    credential_type=iam_user \
    policy_document=-<<EOF
{
  "Version: 2022-03-25",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:*",
      "Resource": "*"
    }
   ]
 }
EOF
```

Configure AWS Dynamic Credentials



Using this token user generates a new AWS credential pair by reading from the /creds endpoint with the name of the role:

```
$ vault read aws/creds/my-role
```

Vault returns credentials, each time the command is run new credentials will generate

```
Key                Value
---                -----
lease_id           aws/creds/my-role/f3e92392-7d9c-09c8-c921-575d62fe80d8
lease_duraton      768h
lease_renewable    true
access_key          AKIAIOWQXTLW36DV7IEA
secret_key          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
security_token      <nil>
```

03

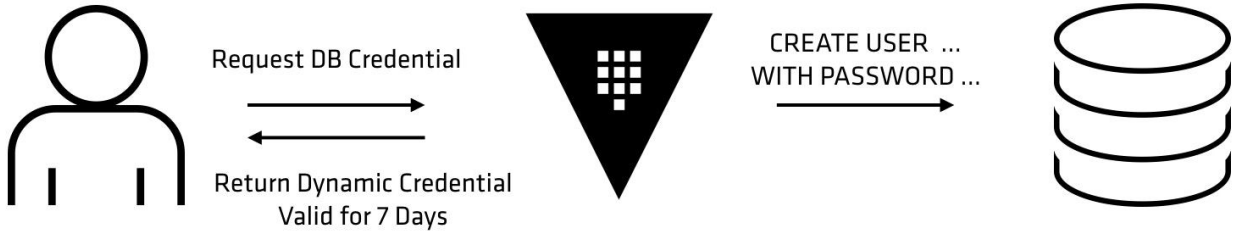
Dynamic Database Credentials

Database Credential Types

- Dynamic user/application credentials
- Root credential rotation
- Static Roles

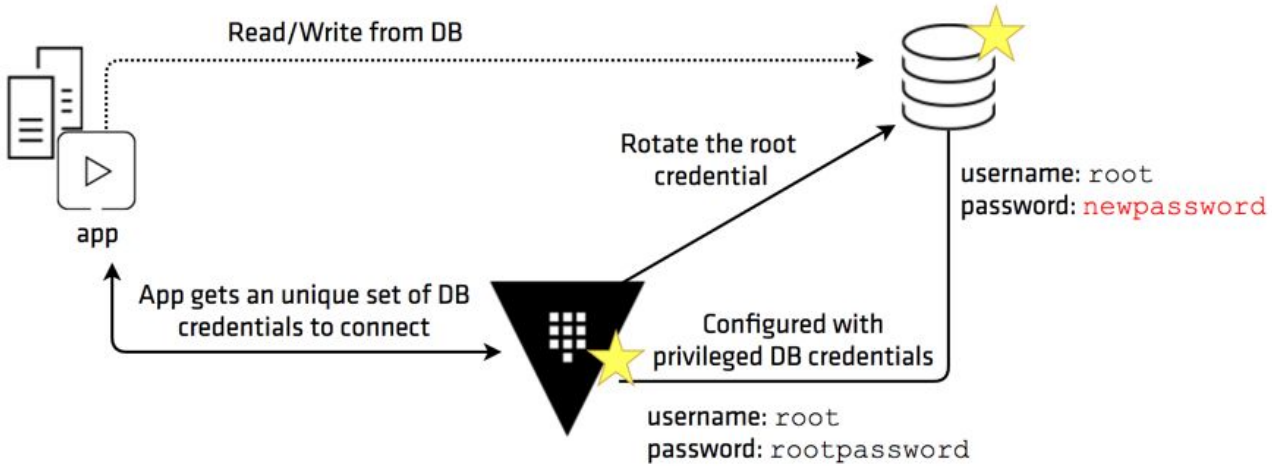
Dynamic Database Credentials

- On demand short-lived credentials for application and user requests
- Can be scoped to specific grant statements
- Revoked at TTL expiration
- Applications or users that need occasional access provision it as needed and credentials do not exist when not in use



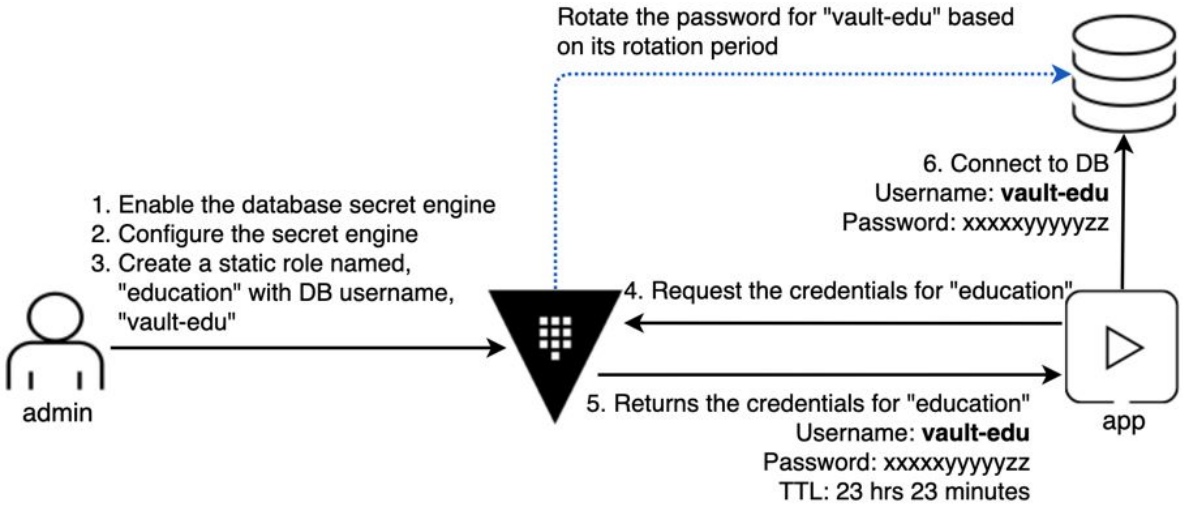
Root Credential Rotation

- Periodically rotate root database password
- Maintain GRC / Security policy compliance
- Rotate root credentials after initial database configuration - **only Vault will have the privileged credentials**



Static Database Roles

- Automatic rotation of database user account passwords
- Ideal for longer-lived connections i.e. service accounts
- Align with security best practices and compliance policy



04

Other Dynamic Credentials

Other Secret Engines

- Consul
- LDAP
- Nomad
- PKI (Certificates)
- RabbitMQ
- Terraform Cloud
- TOTP
- Venafi (Certificates)

LDAP Secrets Engine

- Manages LDAP credentials and performs dynamic credential creation
  - Integrates with LDAP v3 including OpenLDAP, Active Directory and IBM Resource Access Control Facility (RACF)
  - Does not require instances to be manually registered in advance to gain access
- Service Account Check-Out
  - Allows a library of service accounts to be checked out by an person or machine
  - Passwords rotate each time a service account is checked out
  - Accounts automatically check back in and rotate at TTL expiration

Terraform Cloud Secrets Engine

- Enables the generation, management, and revocation of credentials for Terraform Cloud (TFC) and Terraform Enterprise (TFE)
- Generates Terraform API tokens dynamically for Organizations, Teams, and Users

PKI Secrets Engine

- Generates dynamic X.509 certificates
- Provides a facility for services to get certificates without having to generate a private key and CSR, submitting to a CA, and waiting for signing
- Each instance of an application can have a unique certificate improving application security posture and blast radius
- Works with keys stored in an external KMS via the managed keys system (Vault 1.10+)

Resources

- Vault Secrets Engines
- Blog: Why We Need Dynamic Secrets
- Getting Started with Dynamic Secrets
- Database Credential Rotation Tutorial Collection
- LDAP Secrets Engine Tutorial
- Azure Secrets Engine Tutorial
- Terraform Cloud Secrets Engine
- Inject Secrets into Terraform Using the Vault Provider
- PKI Secrets Engine with Managed Keys

Q & A

Thank You

customer.success@hashicorp.com
www.hashicorp.com