Secrets for All the Things

Secrets Injection for Every Application in your Cloud Agnostic Environment

- Mike Ruth, Staff Security Engineer @Cruise Automation
- Brian Nuszkowski, Staff Security Engineer @Cruise Automation
We're building the world's most advanced **self-driving vehicles** to safely connect people with the places, things, and experiences they care about.

https://cruise-automation.github.io/webviz/worldview/

https://getcruise.com/
Secrets management is hard

Developers keep leaving secret keys to corporate data out in the open for anyone to take

By Keith Collins • May 4, 2016

Private keys used to sign firmware published by the company.

How you could be leaking your secrets onto GitHub

Hard coding credentials and pushing the code to GitHub is a common mistake that can lead to exposing sensitive info like Slack tokens or Amazon keys.

Researchers at North Carolina State University (NCSU) scanned almost 13% of GitHub’s public repositories over nearly six months. In a paper revealing the findings, they said:

We find that not only is secret leakage pervasive – affecting over 100,000 repositories – but that thousands of new, unique secrets are leaked every day.
Problems with storing secrets inline

In source

- Easily readable and exfiltrated
- Commit history does us no favors
- See: Trufflehog

In artifacts

- Easily readable and exfiltrated
- Not “DevOps friendly”. Secrets are now the same in Dev as they are in Prod
Thinking about how we solve the secrets management problem

- Where we store secrets depends on what applications and services need them, and where they’re running.

- Clouds, data centers, vehicles that drive themselves
Assessing a secrets technology

- KMS + S3
- Secrets Manager

- KMS + GCS

- K8s Secrets

- HSM
- TPM

@mf_ruth  @nuszkowski
Establishing Identity

- AWS IAM Roles
- GCP Service Accounts
- Kubernetes Service Accounts
Authenticating Services - AWS

AWS Metadata Service

AWS Signature Signing

1. HTTP (GET) - Create canonical request
2. Create string to sign
3. Calculate signature
4. Add signature to request

Amazon EC2

EC2 Instances

EC2 Metadata Service

Instance Identity Document

PKCS#7 Signature
Authenticating Services - GCP

GCP Metadata Service
- GCP Metadata Service
- Request Signed JWT
- Compute Engine
- Pass Signed JWT

Service Account Signed JWTs
- Request Self Signed JWT
- Pass Signed JWT
- Cloud Function
Authenticating Services - K8s

Kubernetes AuthN and AuthZ

- Kubernetes Pod
  - App Container
  - K8s SA
  - JWT (Bearer Token)

- Kubernetes API Server
  - rbac.authorization.k8s.io
  - JWT (Bearer Token)
These implementations are all platform & cloud specific

Do they have to be?
Enter Hashicorp Vault

Central secrets management
## Vault Namespacing

<table>
<thead>
<tr>
<th>Group</th>
<th>Permissions</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team Admin</td>
<td>admin</td>
<td>secret/&lt;prefix&gt;/&lt;namespace&gt;/*</td>
</tr>
<tr>
<td>Team Contractor</td>
<td>list</td>
<td>secret/&lt;prefix&gt;/&lt;namespace&gt;/*</td>
</tr>
<tr>
<td>App Service Account</td>
<td>list, get</td>
<td>secret/&lt;prefix&gt;/&lt;namespace&gt;/&lt;env&gt;/&lt;app&gt;/*</td>
</tr>
</tbody>
</table>
Vault Auth Backends

```json
{
    "bound_ami_id": ["ami-foo30987"],
    "bound_ec2_instance_id": ["i-12345678901230567"],
    "role_tag": ",",
    "policies": [
        "default",
        "dev",
        "prod"
    ],
    "max_ttl": 1800000,
    "disallow_reauthentication": false,
    "allow_instance_migration": false
}

{
    "type": "gce",
    "project_id": "project-123456",
    "policies": ["prod"],
    "bound_zones": ["us-east1-b", "eu-west2-a"],
    "ttl": "30m",
    "max_ttl": "24h",
    "bound_service_accounts": ["dev-1@project-123456.iam.gserviceaccount.com"
}

{
    "bound_service_account_names": "vault-auth",
    "bound_service_account_namespaces": "default",
    "policies": ["dev",
    "prod"
    ],
    "max_ttl": 1800000
}
```
But secrets retrieval is still platform & cloud specific!

Does it have to be?
Enter Daytona

Secrets fast and secure

DAYTONA
Daytona

- Automate Vault authentication, fetching of secrets, and Vault token renewal
- Deployed as a sidecar, initContainer, or entrypoint
- Support for AWS, GCP, Kubernetes Auth Backends

---

```yaml
apiVersion: v1
type: Pod
metadata:
  name: awesome-app
spec:
volumes:
  - name: vault-secrets
    emptyDir:
      medium: Memory
initContainers:
  - name: daytona
    image: gcr.io/cruise-automation/daytona@sha256:abcd123
    securityContext:
      runAsUser: 9999
      allowPrivilegeEscalation: false
    volumeMounts:
      - name: vault-secrets
        mountPath: /home/vault
    env:
      - name: K8S_AUTH
        value: "true"
      - name: K8S_AUTH_MOUNT
        value: "kubernetes-gcp-dev-cluster"
      - name: SECRET_ENV
        value: "true"
      - name: TOKEN_PATH
        value: /home/vault/.vault-token
      - name: VAULT_AUTH_ROLE
        value: awesome-app-vault-role-name
      - name: SECRET_PATH
        value: /home/vault/secrets
      - name: VAULT_SECRETS_APP
        value: secret/path/to/app
      - name: VAULT_SECRETS_GLOBAL
        value: secret/path/to/global/metrics
```
Secrets Injection

Inject Secrets to...

- Environment variables
- A single json file
- Their own individual files

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: awesome-app
spec:
  volumes:
  - name: vault-secrets
    emptyDir:
    medium: Memory
  initContainers:
    serviceAccountName: awesome-app
  - name: daytona
    image: gcr.io/cruise-automation/daytona@sha256:abcd123
    securityContext:
      runAsUser: 9999
      allowPrivilegeEscalation: false
    volumeMounts:
    - name: vault-secrets
      mountPath: /home/vault
  env:
  - name: K8S_AUTH
    value: "true"
  - name: K8S_AUTH_MOUNT
    value: "kubernetes-gcp-dev-cluster"
  - name: SECRET_ENV
    value: "true"
  - name: TOKEN_PATH
    value: /home/vault/.vault-token
  - name: VAULT_AUTH_ROLE
    value: awesome-app-vault-role-name
  - name: SECRET_PATH
    value: /home/vault/secrets
  - name: VAULT_SECRETS_APP
    value: secret/path/to/app
  - name: VAULT_SECRETS_GLOBAL
    value: secret/path/to/global/metrics
```
Vault Integration - AWS

Vault Login
AWS IAM Role used for Vault authentication

Secrets Injection
Daytona side-loads secrets

Short-lived Credentials
Vault generates temporary AWS & GCP credentials on-demand
Vault Integration - GCP

Vault Login
GCP service accounts used for Vault authentication.

Secrets Injection
Daytona side-loads secrets

Short-lived Credentials
Vault generates temporary AWS & GCP credentials on-demand
Vault Integration - Kubernetes

Vault Login
Kubernetes service accounts used for Vault authentication.

Secrets Injection
Daytona side-loads secrets

Short-lived Credentials
Vault generates temporary AWS & GCP credentials on-demand
And you can use it today

https://github.com/cruise-automation/daytona
Thank You

Questions?