Multi-cloud & the Chamber of Secrets

Michael Kehoe
Sr Staff Security Engineer
Agenda

01 $ whoami
02 Background of Confluent's infra
03 Problem Introduction
04 Defining a secret strategy
05 Implementing a secret policy
06 Implementing controls
07 Conclusion
08 Q & A
$ whoami
$ whoami

- Sr Staff Security Engineer - Confluent
  - Cloud Architecture & Reliability

- Previously:
  - Sr Staff SRE @ LinkedIn
  - PhoneSat intern @ NASA

- Background in:
  - Networks
  - Microservices
  - Traffic Engineering
  - KV Databases
  - Incident Management

- Twitter: @michaelkkehoe
- LinkedIn: linkedin.com/in/michaelkkkehoe
- Website: michael-kehoe.io
Confluent Architecture
Confluent Architecture - Control Plane

Confluent VPC/VNet

Confluent VPC

Confluent VPC

Customer On-Premises Network
- Monitoring & Orchestration Apps
- Confluent Cloud Admins

Customer Cloud VCP/VNet
- Kafka Clients/Connect/Streams/Apps
- ksqlDB CLI Clients

Confluent Schema Registry

Confluent Audit Logs

Confluent Public APIs

Fully Managed Connectors

Confluent Cloud Cluster

ksqlDB App(s)

Confluent Cloud Console

Confluent VPC

Confluent VPC

AWS
Confluent Architecture - Customer infra

x thousands across 3 CSPs
Problem Statement
Problem Statement

Global Multi-Cloud
AWS, Azure, GCP over multiple regions

3rd party secrets
Some of the secret mechanisms we control, in some cases, we have secrets for 3rd party services

Control-plane vs data-plane
We need to be able to serve secrets in our control-plane & customer data-plane infra

Ownership
Finding who owns a secret is hard
Defining a secret strategy - Finding the blind spots
## Finding the blind spots

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>01</strong></td>
<td><strong>AWS</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>02</strong></td>
<td><strong>GCP</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>03</strong></td>
<td><strong>Azure</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>04</strong></td>
<td><strong>Hashicorp Vault</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>05</strong></td>
<td><strong>3rd party systems</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Defining a secret strategy - Building an inventory
Building an inventory - Take 1

- Manually retrieve inventory
  - Manually pull an inventory from each system and collate it into a spreadsheet

- Parse permissions/policy
  - Manually retrieve policy/permission and evaluate security/value of secrets

- Find owners
  - Find who owns the secret

- File tickets for remediation
  - Ask owners to update information, add controls
Building an inventory - Take 2

- **Automated inventory retrieval**
  Use DivvyCloud/Steampipe to create inventory

- **Parse permissions/policy**
  Automated parsing of policy and permissions against predetermined risk levels

- **Find owners**
  All secrets now have metadata for ownership

- **File tickets for remediation**
  Based on metadata, automated tickets can be filed
Implementing a secret policy - Defining a strategy
Creating a Strategy

Define what is high-value
Create a definition of what we consider keys-to-the-kingdom and inventory against the sensitivity of the credential

Define approved systems/ controls
Define what credential types should be used and what controls they require

Utilize the best of Vault
1. Utilize dynamic engines as much as possible
2. Terraform IaC against Vault

Implement policy for secret lifetimes
For security and compliance reasons, implement a static credential rotation policy

Force ownership
Ensure that every secret has defined ownership.
- Use of IaC in Vault
- Use of IaC (tags) with CSPs

Improved inventory & monitoring
Daily inventory of all known secrets & usage monitoring of select HV secrets.
Implementing a secret policy - Allowing exceptions
Allowing exceptions

Because only a sith deals in absolutes...

There will always be exceptions
There will always be an edge-case that needs to be accounted for

Create alternative controls
If a secret can not have a preventative control... create a monitoring control

Inventory your exceptions
Ensure that you document the exception
Implementing secret controls - Preventative Controls
Building preventative controls

For the best case

- **Utilize Vault engines**
  Constantly rotate secrets using dynamic engines (we built our own engines)

- **Utilize CSP native dynamic identities**
  Make the CSP responsible for managing the credential

- **Utilize IP restrictions**
  In the unideal case, place IP restrictions on the use of the credential
Implementing secret controls - Building Monitoring controls
Building monitoring controls

For when you may not be able to implement a preventive control

- Utilize existing logging pipelines
  Utilize the existing logging we do of our systems

- Create “known usage locations”
  Create a list of IPs that we expect to be using the credential

- Alert infosec oncall
  Ensure that you document the exception
Conclusion
Conclusion

Know where your secrets are
Do a deep inventory of any place you may have a secret

Know how to secure them
Create standards for how secrets should be protected

Make secrets easy to manage
Make it easy to manage the creation/update of the secret

Build monitoring controls
Monitor for last-use/where they are being used from