Title: HashiCorp Vault dynamic secrets and Spring Boot

Post Body:

I am confused about the use case where HashiCorp Vault is used to provide database secrets dynamically for Spring Boot. Lets say you have two microservices: one containing the application logic and one running a database engine. The first obviously needs to authenticate towards the database and this is where dynamic secrets come into play. Vault can provide such credentials to the first microservice so you don't have to use e.g. ENV variables in a docker-compose file managing both microservices.

The App could be a Spring Boot microservice relying on Spring Cloud Vault to handle communication with HashiCorp Vault for credentials management. The microservice asks Vault for temporary db credentials (in this case they last for one hour) when it is started. During this one hour interval, the app can connect to the database and do whatever needs to be done. After one hour, the credentials expire and no communications is allowed.

The Spring Boot Cloud Vault documentation mentions

> Spring Cloud Vault does not support getting new credentials and configuring your DataSource with them when the maximum lease time has been reached. That is, if max_ttl of the Database role in Vault is set to 24h that means that 24 hours after your application has started it can no longer authenticate with the database.

In other words, after one hour, the connection is lost and there seems to be no other way to get new db credentials other then by restarting the microservice. So if have the following questions:

1. What is the added value of using Vault in this particular example if you are (seemingly) forced to restart your entire application each time the TTL expires?
2. Does the same apply when you use static secrets instead?
3. Can this issue be solved without changing microservice code? (K8S, Istio, etc.?)

My guess is the intended use of Vault with Spring Boot is different compared to my understanding.

Accepted Answer: None

Highest Rated Answer:

This article describes 4 possible solutions to mitigate the issue described in the question. Being valid approaches to solve the problem, a more generic (referring to the 'heavy rotation of dynamic secrets'-approach) and less aggressive (referring to the 'restarting the service when connectivity is lost'-approaches) should be in place.