DDR Demo

Auto Renewed Certificates – DDR

<https://github.com/hashicorp/demo-vault-auto-renewed-certs/tree/feat/vault-pki>

# Proposed Scope

## Vault Namespaces

* ✅ **[⭐ Enterprise]** Vault Namespaces: /pki-root, /pki-issuing
  1. ✅ SE Access to Root: Request signed certificate, subject to CG approval + List & Read all
  2. ✅ SE Access to Issuing: Read/Write All

## Root CA

* ⚠️ AWS KMS ECC Key
  1. ✅ Resource policy limits signing to Vault user; management operations to IAM
* ⚠️ AWS IAM User for Vault Managed Key
* ⚠️ **[⭐ Enterprise]** Vault Managed Key
* ✅ PKI Mount
* ✅ PKI Key and Self-Signed Certificate
* ~~PKI Role for Issuing CA~~ *(N/A)*
* ✅ PKI Operators Group
* ✅ ACL Policy with **[⭐ Enterprise]** M-of-N Control Group
* ✅ Username & Password – Alice, Bob, and Charlie
* ✅ CRL, AIA, and OCSP Configs
* *~~[Stretch]~~* ***~~[⭐ Enterprise]~~*** *~~Enterprise MFA for Approving Signing Reqs.~~*

## Issuing CA

* ✅ PKI Mount
* ✅ Issuing CA CSR
  1. Module variable enables SE to choose whether to have the Issuing CA’s CSR as a module output, or whether the module should handle signing
  2. If demonstrating Control Groups, signing via module should be disabled
* ✅ PKI Role for App Certs
  1. 7-day certificates
  2. Permitted names to match …?
  3. **[⭐ Enterprise]** Cert Metadata
* ✅ ACL Policy for App Certs
* ✅ AWS Auth Backend
* ✅ AWS Auth Role for EC2 Instance
* ✅ App Entity & Entity Alias
  1. Initially with no permissions so SE can demonstrate Vault Agent acting in the background and handling it all seamlessly?
* ✅ Group with ACL policy attached
* ✅ CRL, AIA, and OCSP Configs
* ✅ *[Stretch] Tidy Configs*

## EC2 Auto Scaling

* ✅ VMs retrieve their “backend” certificates from Vault
* ✅ Load balancer fronts all VMs and uses an AWS ACM certificate for public trust
* ✅ “Hello World” web server app
  1. Python app deployed with user-data
* ✅ Vault Agent
  1. AWS Auto-Auth
  2. Fetch Certificate
     + Instance ID in certificate metadata
  3. Auto-Reload Web App
  4. Deployed through user-data
* ❓ Sentinel policy to block access by EC2 instance
* ✅ *[Stretch] Web app prints certificate information*

## Kubernetes / VSO

* TBD

## Route 53

* ✅ Point to EC2 instance
* ✅ Follow “<customer>-<app>.guilherme-pamplonasantos.sbx. …” pattern.

# Web App ID [TK]

1. Two parts
   1. Front end
   2. N back ends
2. Front end:
   1. The public side of things
   2. Give it a Let's Encrypt cert so it looks valid (we have patterns for doing that on the Instruqt side with Caddy which can be adapted here)
   3. It simply shows the status of all the back ends, reading the json blob from 3b and showing it good or bad
3. Back End
   1. Listens on the port with TLS
   2. GET / returns a json blob:  
      {  
       "cn": "<Common name of cert this backend has>",  
       "serial": "<Serial number of the cert this backend has>",  
       "issued-on": "YYYY-mm-ddTHH:MM:SS",  
       "expires-on": "YYYY-mm-ddTHH:MM:SS",  
      }
4. Something polling a KV entry in Vault looking for an integer N
   1. Launches N copies of the back end, listening on <ip>:<base port +N>
   2. Configures the front end with the list of N back-ends
   3. Kicks the front end to reload its config
5. Advantages
   1. Front end uses a cert already trusted by SE browser, no funny stuff there
   2. Can show scaling quickly (or not, default to 1)
   3. Shows obviously cert validity
   4. You can enable/disable the backend identity from being able to get backend certs, show renewal failing (you no longer have permission to fetch this cert), then turn back on, look how fast things recover

# DDR Platform Limitations

## AWS IAM User

Service Control Policy blocking IAM User creation by DDR role. Sent problem statement to Thomas for forwarding to security and see if we can solve.

## SE Read-Only Access

The DDR platform is configured such that HCP Terraform shares the same Vault identity as the Solutions Engineer. So, adding a read-only policy to the PKI Root namespace doesn’t work – Vault becomes blocked from configuring all necessary resources.