Using (Hashicorp) Vault to Issue Intermediate Certs through Roles

Jump to bottom

Jason Riddle edited this page on Feb 11, 2016 · 1 revision

Prerequisites.

Start the vault server.

```
vault server -dev
```

Export `VAULT_ADDR` .

```
export VAULT_ADDR='http://127.0.0.1:8200'
export VAULT_TOKEN=$(cat ~/.vault-token)
```

Mount the `pki-root` backend.

```
http PUT "${VAULT_ADDR}/v1/sys/mounts/pki-root" \
  X-Vault-Token:${VAULT_TOKEN} \
  type=pki \
  description='root pki'
```

Mount the `pki-intermediate` backend.

```
http PUT "${VAULT_ADDR}/v1/sys/mounts/pki-intermediate" \
  X-Vault-Token:${VAULT_TOKEN} \
  type=pki \
  description='intermediate pki'
```

Set the `max-lease-ttl` for the `pki-intermediate` backend.

```
http PUT "${VAULT_ADDR}/v1/sys/mounts/pki-intermediate/tune" \
  X-Vault-Token:${VAULT_TOKEN} \
  max-lease-ttl=8760h
```

Set the Root CA.

(Optional) Generate a root CA (I generated one with cfssl).

```
$ curl -O https://storage.googleapis.com/configs.kuar.io/ca-csr.json

$ cat ca-csr.json
{
  "CN": "Kubernetes CA",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "Kubernetes",
      "OU": "CA",
      "ST": "Oregon"
    }
  ]
}

$ cfssl gencert -initca ca-csr.json | cfssljson -bare ca

$ ls
ca-csr.json    ca-key.pem     ca.csr         ca.pem
```

Upload a root CA to Vault.

```
http POST "${VAULT_ADDR}/v1/pki-root/config/ca" \
  X-Vault-Token:${VAULT_TOKEN} \
  pem_bundle="$(cat ca-key.pem ca.pem)"
```

(OR) Generate a new self-signed CA certificate and private key using Vault..

```
http POST "${VAULT_ADDR}/v1/pki-root/root/generate/exported" \
  X-Vault-Token:${VAULT_TOKEN} \
  common_name='Kubernetes Root CA' \
  ttl=8760h \
| tee >(jq -r .data.certificate > ca.pem) \
  >(jq -r .data.issuing_ca > issuing_ca.pem) \
  >(jq -r .data.private_key > ca-key.pem)
```

..and Verify.

```
$ ls
# ca-key.pem     ca.pem         issuing_ca.pem
```

```
$ openssl rsa -noout -text -in ca-key.pem
# Private-Key: (2048 bit)
# <Snip>

$ openssl x509 -noout -text -in issuing_ca.pem
# Certificate:
# <Snip>

$ openssl x509 -noout -text -in ca.pem
# Certificate:
# <Snip>

$ openssl x509 -noout -issuer -subject -in ca.pem
# issuer= /CN=Kubernetes CA
# subject= /CN=Kubernetes CA
```

Create a Signed Cert for the `pki-intermediate` mount.

Generate a new private key and a CSR for signing..

```
http POST "${VAULT_ADDR}/v1/pki-intermediate/intermediate/generate/exported" \
  X-Vault-Token:${VAULT_TOKEN} \
  common_name=intermediate.com \
  ip_sans=127.0.0.1 \
| tee >(jq -r .data.csr > vault.csr) \
  >(jq -r .data.private_key > vault.pem)
```

..and Verify.

```
openssl req -noout -subject -in vault.csr
# subject=/CN=intermediate.com

openssl req -noout -text -in vault.csr
# < Full Text >
```

Generate a signed cert using `vault.csr` .

```
http POST "${VAULT_ADDR}/v1/pki-root/root/sign-intermediate" \
  X-Vault-Token:${VAULT_TOKEN} \
  common_name=intermediate.com \
  csr=@vault.csr \
  ttl=72h \
| tee >(jq -r .data.certificate > vault.cert) \
  >(jq -r .data.issuing_ca > vault_issuing_ca.pem)
```

Set the `pki-intermediate` signed cert to `vault.cert`.

```
http POST "${VAULT_ADDR}/v1/pki-intermediate/intermediate/set-signed" \
  X-Vault-Token:${VAULT_TOKEN} \
  certificate=@vault.cert
```

Configure and create a role.

Configure a role using the `pki-intermediate` mount.

```
http POST "${VAULT_ADDR}/v1/pki-intermediate/roles/example-dot-com" \
  X-Vault-Token:${VAULT_TOKEN} \
  allowed_domains=example.com \
  allow_subdomains=true \
  max_ttl=8h
```

Generate credentials for the `example-dot-com` role.

```
http POST "${VAULT_ADDR}/v1/pki-intermediate/issue/example-dot-com" \
  X-Vault-Token:${VAULT_TOKEN} \
  common_name=blah.example.com \
| tee >(jq -r .data.certificate > blah.cert) \
  >(jq -r .data.issuing_ca > blah_issuing_ca.pem) \
  >(jq -r .data.private_key > blah-key.pem)
```

..and Verify.

```
$ ls
# blah-key.pem        blah.cert           blah_issuing_ca.pem

$ openssl rsa -noout -text -in blah-key.pem
# Private-Key: (2048 bit)
# <Snip>

$ openssl x509 -noout -text -in blah_issuing_ca.pem
# Certificate:
# <Snip>

$ openssl x509 -noout -text -in blah.cert
# Certificate:
# <Snip>

$ openssl x509 -noout -issuer -subject -in blah.cert
# issuer= /CN=intermediate.com
# subject= /CN=blah.example.com
```

Clone this wiki locally

```
https://github.com/jason-riddle/generating-certs.wiki.git
```