Use Vault with client certificates

Jul 20, 2016 • Jo Vandeginste

Tags: vault (3)

Still learning to use Vault, I want to experiment accessing the Vault using a client certificate instead of a regular token. This proved to be not-so-easy for reasons I hadn't foreseen…

Step 1 - generate a certificate

This is well-documented on the Internet, with one caveat: many how-to's don't let you protect your key with a passphrase. I think you should. And this exactly turned out to be the unforeseen reason this whole thing was harder than expected…!

For documentation reasons, I will add my own method.

First create a config file to facilitate experimentation; please change the values in the `[ dn ]` section and eg. call the file `cert.conf`:

```
[ req ]
default_bits = 2048
prompt = no
encrypt_key = yes
default_md = sha256
distinguished_name = dn

[ dn ]
C = Country Name (2 letter code)
ST= State or Province Name (full name)
L = Locality Name (eg, city)
O = Personal
emailAddress= you@yourdomain.com
0.CN= Your Name
```

Now generate a key and certificate with passphrase using this config file (I will use `th3p@ss` as the passphrase throughout this text):

```
openssl req -config cert.conf -x509 -newkey rsa:2048 -keyout key.pem -out cert
```

The certificate will only be valid for 14 days - plenty of time to experiment :-) You can now verify the certificate:

```
$ openssl x509 -in cert.pem -noout -text
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 11244389068616569346 (0x9c0c1848572d1202)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=BE, ST=Vlaams-Brabant, L=Rotselaar, O=Personal/emailAddress=
        Validity
            Not Before: Jul 20 13:07:24 2016 GMT
            Not After : Aug  3 13:07:24 2016 GMT
        Subject: C=BE, ST=Vlaams-Brabant, L=Rotselaar, O=Personal/emailAddress
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
[...]
```

Step 2 - connect to the Vault server

Now try to contact your Vault server with it:

```
$ curl https://vault.example.com/v1/sys/health --cert cert.pem --key key.pem
curl: (35) error reading X.509 key or certificate file: Decryption has failed.
```

What? So *this* I spent a long time figuring out, until the answer became obvious: the key is encrypted (passphrase), but curl doesn't prompt for the passphrase. After some digging around, I found out that you need to put the passphrase together with the `cert.pem`:

```
$ curl https://vault.example.com/v1/sys/health --cert cert.pem:th3p@ss --key k
{"initialized":true,"sealed":false,"standby":false,"server_time_utc":146902061
```

Okay, better now! (Not really, since I see my passphrase clear text on the command line… There are ways around this but I still feel dirty now!)

Sadly, the `vault` binary does not (yet) support encrypted keys:

```
$ VAULT_ADDR=https://vault.example.com vault auth -method=cert -client-cert=ce
Error initializing client to auth: crypto/tls: failed to parse private key
```

I found no solution, so for the binary to work we need an unencrypted key. We can decrypt our current key:

```
$ openssl rsa -in key.pem -out key.insecure.pem
Enter pass phrase for key.pem:
writing RSA key
```

With this insecure key, both curl and vault will work:

```
$ curl https://vault.example.com/v1/sys/health --cert cert.pem --key key.insec
{"initialized":true,"sealed":false,"standby":false,"server_time_utc":146902074
```

```
$ VAULT_ADDR=https://vault.example.com vault auth -method=cert -client-cert=ce
Error making API request.

URL: PUT https://vault.example.com/v1/auth/cert/login
Code: 400\. Errors:

* invalid certificate or no client certificate supplied
```

Okay, it doesn't actually work, since the Vault server doesn't know my certificate yet. This is the next step.

Step 3 - uploading your client certificate

This was surprisingly easy with sufficient documentation on Vault's site. You do obviosly need a token with sufficient rights to perform these changes. I'm working on an experimental setup, so I have the root token `my-token` at hand…

First make sure the `cert` backend is enabled, then upload your new certificate (and grant it the `root` policy for testing):

```
$ VAULT_TOKEN=my-token VAULT_ADDR=https://vault.example.com vault auth-enable
Successfully enabled 'cert' at 'cert'!
$ VAULT_TOKEN=my-token VAULT_ADDR=https://vault.example.com vault write auth/c
Success! Data written to: auth/cert/certs/your.name
```

Verify the content of the `auth/cert`:

```
$ VAULT_TOKEN=my-token VAULT_ADDR=https://vault.example.com vault read auth/ce
Key             Value
---             -----
certificate     -----BEGIN CERTIFICATE-----
```

```
[...]
-----END CERTIFICATE-----
display_name    Your Name
policies        root
ttl             3600
```

Step 4 - use your certificate

Now the real test :-) We can use our client certificate now to generate tokens to actually do something on the Vault. So step 1 is generating a token:

```
$ VAULT_ADDR=https://vault.example.com vault auth -method=cert -client-cert=ce
Successfully authenticated! You are now logged in.
The token below is already saved in the session. You do not
need to "vault auth" again with the token.
token: 52xxxx8e-xxxx-e22b-xxxx-e326xxxx0b7b
token_duration: 3599
token_policies: [root]
```

So now I have a Vault token `52xxxx8e-xxxx-e22b-xxxx-e326xxxx0b7b`, which I can use for 1 hour to do anything `root` can (that was the policy I assigned to my certificate). Let's do some basic stuff! First I exported `VAULT_TOKEN` and `VAULT_ADDR` for the sake of brevity.

Look at the information about the token itself:

```
$ vault token-lookup
Key                     Value
---                     -----
accessor                98xxxx5d-xxxx-3906-xxxx-fcebxxxx8e58
creation_time           1469021773
creation_ttl            3600
display_name            cert-Your Name
explicit_max_ttl        0
id                      52xxxx8e-xxxx-e22b-xxxx-e326xxxx0b7b
meta                    map[authority_key_id: cert_name:your.name common_name:
num_uses                0
orphan                  true
path                    auth/cert/login
policies                [root]
renewable               true
role
ttl                     3468
```

Let's do some Vaulty actions with the token:

```
$ vault list /secret/                # List the currently existing keys at /s
Keys
teams/
test
$ vault read /secret/test            # Read pre-existing data that I put ther
Key                     Value
---                     -----
refresh_interval        2592000
a                       b
c                       4
$ vault write /secret/test d=5       # Overwrite the pre-existing data
Success! Data written to: secret/test
$ vault read /secret/test            # Yup, it's gone now!
Key                     Value
---                     -----
refresh_interval        2592000
d                       5
```

Now we wait for the rest of the hour to pass and verify our token has expired:

```
$ vault token-lookup
error looking up token: Error making API request.

URL: GET https://vault.example.com/v1/auth/token/lookup-self
Code: 400\. Errors:

* permission denied
```

You can get a new token any time with the `auth` command, until your certificate expires (hopefully not after it expires, but I haven't tried it)

Share this:

Jo Vandeginste

I'm a Linux SysAdmin at a university's central IT department. I try to learn some new (to me) tools and usually struggle to get them working in our corporate environment.

Found something very useful and feel grateful? Consider donating!

1PvmjQ6DAhPAUYMgSZ1Z3wnryKWjpbS54P