I wrote this today for my company, and I'd appreciate feedback. I may have left out some steps or put too much detail in others.

Thanks,
Clay

Getting Started with HashiCorps' Vault

Vault is a tool developed by HashiCorp to securely store secrets. In this guide I will show step by step how I use Vault to store password information used in scripts. I'll use BASH for the scripting language in the examples, as it's fairly universal to Linux. This method works fine in other scripting languages such as Perl, Ruby, Javascript, etc. as well. I worked many days to understand the product and work it into correcting a problem that I had, so this process may not work for you exactly as shown but it should provide some insight.

Before you begin going through this guide, you must have Vault installed on a server and running. See the "getting started" guide if you need to accomplish this setup.

One word about authentication. Vault has a number of methods to authenticate users, from basic username/password
to more esoteric methods like APP-ID. As my purpose was to prevent passwords from being added to scripts, many of these methods were less than optimal. I chose to use x509 certificates, as in my opinion they are a bit more secure than the other methods for my purpose. Since the CERT backend will not accept CIDR restrictions, I've set up the server that runs Vault for my site to only accept connections from the internal (10.0.0.0) network.

Setting up vault to use TLS certificates

First, you must create an x509 certificate for Vault. You can obtain this from a Certificate Authority, but as our Vault is keeping internal information only, I chose to use a self-signed certificate. To create this, I did the following:

I first created a certificate configuration file. Here's the file I used (you'll need to update this for your purpose):

```
[req]
distinguished_name = req_distinguished_name
x509_extensions = v3_req
prompt = no

[req_distinguished_name]
C = US
ST = state
L =  city
O = company
CN = *

[v3_req]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
basicConstraints = CA:TRUE
subjectAltName = @alt_names

[alt_names]
DNS.1 = *
DNS.2 = *.*
DNS.3 = *.*.*
DNS.4 = *.*.*.*
DNS.5 = *.*.*.*.*
DNS.6 = *.*.*.*.*.*
```

```
DNS.7 = *.*.*.*.*.*
IP.1 = 10.16.19.21
```

You must put the IP address of your Vault server in the line "IP.1".

I then ran the command:

```
openssl req -x509 -batch -nodes -newkey rsa:2048 -keyout selfsigned.key -out selfsigned.crt -config selfsigned.cfr -days 9999
```

This example creates a self-signed certificate that is valid for 9,999 days. For a production server I'd set this to 365 days as I'd want to replace the certificate every year, but it's up to you.

Once the certificate has been created, you must add it to your certificate bundle if you don't want the "invalid CA" message (yes, there are other ways to accomplish this goal but I think this is the easiest).

```
cat selfsigned.crt >> /etc/pki/tls/certs/ca-bundle.crt
```

Now the Vault configuration needs to be updated to use the certificates. Update your vault.conf file's listener section:

```
listener "tcp" {
  address = "10.16.19.21:8000"
  tls_cert_file = "/root/selfsigned.crt"
  tls_key_file = "/root/selfsigned.key"
}
```

Now you'll need to restart Vault to use the new config. **BEFORE YOU KILL THE VAULT PROCESS** be sure you have the key
to unseal the vault - it seals whenever it's restarted.

Find the vault process and kill it:

```
[root@vaultserver ~]# ps -ef|grep vault
root      3235  3130  0 17:12 pts/0    00:00:00 grep vault
root     30719     1  0 Jun05 ?        00:01:15 /opt/vault/vault server -config /opt/vault/vault.conf

[root@vaultserver ~]# kill 30719
```

Restart Vault:

```
/opt/vault/vault server -config /opt/vault/vault.conf >> /var/log/vault.log 2>&1
```

You should see something like the following in the /var/log/vault.log:

```
==> Vault server configuration:

         Log Level: info
             Mlock: supported: true, enabled: true
           Backend: consul (HA available)
 Advertise Address: http://10.16.19.21:8000
       Listener 1: tcp (addr: "10.16.19.21:8000", tls: "enabled")

==> Vault server started! Log data will stream in below:
```

Note the Listener section -- you should see tls: "enabled". Your address/backend/etc. may be different.
If you check the vault status, you will find that it's sealed. You must now unseal it to continue:

```
vault unseal -address https://10.16.19.21:8000 <unseal key>
```

OK, now we have a vault that's TLS enabled so we can now use CERT authentication. Let's enable CERT:

vault auth-enable cert

You can confirm with:

vault auth --methods

Note: After restart, you'll sometimes get an error on this command. Just log into vault as the root again to resolve it.
vault auth -address https://10.16.19.21:8000 -method=userpass username=<user> password=<userpass>

You should see something like this:

```
[root@vaultserver vault]# vault auth -methods
Path       Type      Description
app-id/    app-id
cert/      cert
token/     token     token based credentials
userpass/  userpass
```

Inserting secrets into the Vault

Now we need some secrets in our vault to read. Each secret should be assigned to a policy (or more) to easily allow users to access it. To create a policy, you first must create a policy file. To do this, you'll need to know the name you want to assign to the secret(s), as the policy will be set to read these values. We'll use a couple of secrets named "foo" and "names".

Create a file called "<name>.hcl" -- note that the name can be anything, and doesn't have to correlate with the name of the policy.

Here's an example of a policy:

```
[root@vaultserver vault]# cat users.hcl
path "secret/policy/scripts/foo" {policy="read"}
path "secret/policy/scripts/names" {policy="read"}
```

Note that the secrets are going to be stored under the policy "scripts".
The path is secret/policy/<policy name>/<secret name> and the "policy" sets the access this policy has to the data.

Now we add this policy to the Vault:

vault policy-write scripts users.hcl

Now that the policy is there, we can write the secrets into the Vault under that location:

```
vault write "secret/policy/scripts/foo" value=bar policy=scripts
vault write "secret/policy/scripts/names" value="joe,bob,briggs" policy=scripts
```

OK, the secrets are in the vault, and the policy "scripts" has read access to them. Now we need to set up a user to read them.

Setting up a CERT authenticated user

For our user, we need a certificate so that user can authenticate to the vault. Again we'll use a self-signed certificate. Here's how I created my user cert:

```
openssl genrsa -des3 -out scripts.key 2048
openssl req -new -key scripts.key -out scripts.csr
openssl x509 -req -in scripts.csr -signkey scripts.key -out scripts.crt
```

Now that we have the certificate, we add this to the Vault so it can authenticate our user:

vault write auth/cert/certs/scripts display_name=scripts policies=scripts certificate=@scripts.crt

Note that we created the user and assigned them to the "scripts" policy. It doesn't matter that the user name and the policy name are the same, although you may want to avoid this in a real world situation.

Using the Vault

We now have a user and secrets, so lets actually get them out of the Vault. You'll need the certificate for the script user as well as the key file. This should be accessible to that user only (mode 600) and located somewhere they can access, like /<home directory>/.ssh

To cut down on traffic to the Vault server, I start out the script by getting a Token that I can use through this script execution.

TOKEN=$(curl -s -X PUT --cert "/home/user/.ssh/scripts.crt" --key "/home/user/.ssh/scripts.key" https://10.16.19.21:8000/v1/auth/cert/login|awk -F, '{print $5}'|awk -F: '{print $3}'|sed 's/"//g')

Now I can access the secret(s) using the token:

curl -s -H "X-Vault-Token: $TOKEN" "https://10.16.19.21:8000/v1/secret/policy/scripts/foo" | awk -F\" '{print $18}'

To use this in a script, I'd use the "$()" construction to replace the place where the password was previously:

$(curl -s -H "X-Vault-Token: $TOKEN" "https://10.16.19.21:8000/v1/secret/policy/scripts/foo"|awk -F\" '{print $18}')


Clay Bowen

6/8/15

Have to modify the user's section to the following (otherwise it'll ask for a password each time):

Setting up a CERT authenticated user

For our user, we need a certificate so that user can authenticate to the vault. Again we'll use a self-signed certificate. Here's how I created my user cert:

openssl genrsa -out scripts.key 2048

When it asks for a password, press "Enter"

Thanks,
Clay
- show quoted text -

Jack Hsu

6/15/15

Hi Clay,

I follow your instruction to the point it's giving me following error. Is it because the curl version on my Yosemite? The vault.log shows "TLS handshake error from 127.0.0.1:62947: EOF". Help is much appreciated.

$ curl -v -X PUT --cert ./scripts.crt --key ./scripts.key https://127.0.0.1:8000/v1/auth/cert/login

* Hostname was NOT found in DNS cache

*   Trying 127.0.0.1...

* Connected to 127.0.0.1 (127.0.0.1) port 8000 (#0)

* WARNING: SSL: CURLOPT_SSLKEY is ignored by Secure Transport. The private key must be in the Keychain.

* WARNING: SSL: Certificate type not set, assuming PKCS#12 format.

* SSL: Can't load the certificate "./scripts.crt" and its private key: OSStatus -25299

* Closing connection 0

curl: (58) SSL: Can't load the certificate "./scripts.crt" and its private key: OSStatus -25299

$ curl -V

curl 7.37.1 (x86_64-apple-darwin14.0) libcurl/7.37.1 SecureTransport zlib/1.2.5

Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtsp smtp smtps telnet tftp

Features: AsynchDNS GSS-Negotiate IPv6 Largefile NTLM NTLM_WB SSL libz

- show quoted text -


Clay Bowen
6/15/15

I'm using Curl 7.40, but that shouldn't make a difference.
curl 7.40.0 (x86_64-redhat-linux-gnu) libcurl/7.40.0 NSS/3.16.2.3 Basic ECC zlib/1.2.8 libidn/1.18 libssh2/1.4.2
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtsp scp sftp smb smbs smtp smtps telnet tftp
Features: AsynchDNS IDN IPv6 Largefile GSS-API Kerberos SPNEGO NTLM NTLM_WB SSL libz UnixSockets

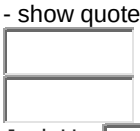
What version of OpenSSL are you running?

[root@ip-10-16-43-125 log]# rpm -qa|grep ssl
openssl-1.0.1k-1.84.amzn1.x86_64
openssl-devel-1.0.1k-1.84.amzn1.x86_64

Thanks,
Clay

And are the certificates in the same directory where you are running the curl command?

- show quoted text -



Jack Hsu

6/15/15

I'm getting further after "brew install curl --with-openssl", but the curl SSL cert verification seems to fail. The openssl version is 0.9.8zd. If I bypass the cert verification, then I got "Empty reply from server". Any idea where the problem might be in my environment? Thanks

$ curl -v -X PUT --cert ./scripts.crt --key ./scripts.key https://127.0.0.1:8000/v1/auth/cert/login

* Trying 127.0.0.1...

* Connected to 127.0.0.1 (127.0.0.1) port 8000 (#0)

* ALPN, offering http/1.1

Enter PEM pass phrase:

* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH

* successfully set certificate verify locations:

* CAfile: /usr/local/etc/openssl/cert.pem

  CApath: none

* TLSv1.2, TLS Unknown, Unknown (22):

* TLSv1.2, TLS handshake, Client hello (1):

* SSLv2, Unknown (22):

* TLSv1.2, TLS handshake, Server hello (2):

* SSLv2, Unknown (22):

* TLSv1.2, TLS handshake, CERT (11):

* SSLv2, Unknown (21):

* TLSv1.2, TLS alert, Server hello (2):

* SSL certificate problem: self signed certificate

* Closing connection 0

curl: (60) SSL certificate problem: self signed certificate

More details here: http://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle"
 of Certificate Authority (CA) public keys (CA certs). If the default
 bundle file isn't adequate, you can specify an alternate file
 using the --cacert option.
If this HTTPS server uses a certificate signed by a CA represented in
 the bundle, the certificate verification probably failed due to a
 problem with the certificate (it might be expired, or the name might
 not match the domain name in the URL).
If you'd like to turn off curl's verification of the certificate, use
 the -k (or --insecure) option.

$ curl -V

curl 7.42.1 (x86_64-apple-darwin14.3.0) libcurl/7.42.1 OpenSSL/1.0.2a zlib/1.2.5

Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtsp smb smbs smtp smtps telnet tftp

Features: IPv6 Largefile NTLM NTLM_WB SSL libz TLS-SRP UnixSockets

$ openssl version

OpenSSL 0.9.8zd 8 Jan 2015

$ curl --insecure -v -X PUT --cert ./scripts.crt --key ./scripts.key https://127.0.0.1:8000/v1/auth/cert/login

*   Trying 127.0.0.1...

* Connected to 127.0.0.1 (127.0.0.1) port 8000 (#0)

* ALPN, offering http/1.1

Enter PEM pass phrase:

* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH

* successfully set certificate verify locations:

*   CAfile: /usr/local/etc/openssl/cert.pem

  CApath: none

* TLSv1.2, TLS Unknown, Unknown (22):

* TLSv1.2, TLS handshake, Client hello (1):

* SSLv2, Unknown (22):

* TLSv1.2, TLS handshake, Server hello (2):

* SSLv2, Unknown (22):

* TLSv1.2, TLS handshake, CERT (11):

* SSLv2, Unknown (22):

* TLSv1.2, TLS handshake, Server key exchange (12):

* SSLv2, Unknown (22):

* TLSv1.2, TLS handshake, Server finished (14):

* SSLv2, Unknown (22):

* TLSv1.2, TLS handshake, Client key exchange (16):

* SSLv2, Unknown (20):

* TLSv1.2, TLS change cipher, Client hello (1):

* SSLv2, Unknown (22):

* TLSv1.2, TLS handshake, Finished (20):

* SSLv2, Unknown (20):

* TLSv1.2, TLS change cipher, Client hello (1):

* SSLv2, Unknown (22):

* TLSv1.2, TLS handshake, Finished (20):

* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-SHA

* ALPN, server accepted to use http/1.1

* Server certificate:

* subject: C=US; ST=state; L=city; O=company; CN=*

* start date: 2015-06-15 17:37:16 GMT

* expire date: 2042-10-30 17:37:16 GMT

* issuer: C=US; ST=state; L=city; O=company; CN=*

* SSL certificate verify result: self signed certificate (18), continuing anyway.

* SSLv2, Unknown (23):

> PUT /v1/auth/cert/login HTTP/1.1

> Host: 127.0.0.1:8000

> User-Agent: curl/7.42.1

> Accept: */*

>

* SSLv2, Unknown (21):

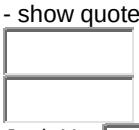
* TLSv1.2, TLS alert, Client hello (1):

* Empty reply from server

* Connection #0 to host 127.0.0.1 left intact

curl: (52) Empty reply from server

- show quoted text -



Jack Hsu 

6/15/15 

Here are the log messages. Thanks

When doing cert verification:

2015/06/15 15:16:29 http: TLS handshake error from 127.0.0.1:65073: remote error: unknown certificate authority

When bypassing cert verification:

2015/06/15 15:16:56 http: panic serving 127.0.0.1:65087: runtime error: slice bounds out of range

goroutine 531 [running]:

net/http.func·011()

/Users/mitchellh/code/3rdparty/go/src/net/http/server.go:1130 +0xbb

github.com/hashicorp/vault/builtin/credential/cert.validateConnState(0xc208128930, 0xc2080a6980, 0x0, 0x0, 0x0, 0x0, 0x0)

/Users/mitchellh/code/go/src/github.com/hashicorp/vault/builtin/credential/cert/path_login.go:158 +0x55e

github.com/hashicorp/vault/builtin/credential/cert.(*backend).pathLogin(0xc2080e0b50, 0xc208104770, 0xc20802b0d0, 0x5cf3c0, 0x0, 0x0)

/Users/mitchellh/code/go/src/github.com/hashicorp/vault/builtin/credential/cert/path_login.go:42 +0xce

github.com/hashicorp/vault/builtin/credential/cert.*backend.(github.com/hashicorp/vault/builtin/credential/cert.pathLogin)·fm(0xc208104770, 0xc20802b0d0, 0x68e610, 0x0, 0x0)

/Users/mitchellh/code/go/src/github.com/hashicorp/vault/builtin/credential/cert/path_login.go:25 +0x4d

github.com/hashicorp/vault/logical/framework.(*Backend).HandleRequest(0xc208077ef0, 0xc208104770, 0x28, 0x0, 0x0)

/Users/mitchellh/code/go/src/github.com/hashicorp/vault/logical/framework/backend.go:126 +0x813

github.com/hashicorp/vault/vault.(*Router).Route(0xc20801e4c0, 0xc208104770, 0x0, 0x0, 0x0)

/Users/mitchellh/code/go/src/github.com/hashicorp/vault/vault/router.go:209 +0x94a

github.com/hashicorp/vault/vault.(*Core).handleLoginRequest(0xc208074400, 0xc208104770, 0x0, 0x0, 0x0)

/Users/mitchellh/code/go/src/github.com/hashicorp/vault/vault/core.go:446 +0x2ed

github.com/hashicorp/vault/vault.(*Core).HandleRequest(0xc208074400, 0xc208104770, 0x0, 0x0, 0x0)

/Users/mitchellh/code/go/src/github.com/hashicorp/vault/vault/core.go:323 +0x173

github.com/hashicorp/vault/http.request(0xc208074400, 0xad4a28, 0xc2081820a0, 0xc2080ad040, 0xc208104770, 0x675150, 0x4)

/Users/mitchellh/code/go/src/github.com/hashicorp/vault/http/handler.go:74 +0x32

github.com/hashicorp/vault/http.func·002(0xad4a28, 0xc2081820a0, 0xc2080ad040)

/Users/mitchellh/code/go/src/github.com/hashicorp/vault/http/logical.go:67 +0x5a9

net/http.HandlerFunc.ServeHTTP(0xc2080e12d0, 0xad4a28, 0xc2081820a0, 0xc2080ad040)

/Users/mitchellh/code/3rdparty/go/src/net/http/server.go:1265 +0x41

net/http.(*ServeMux).ServeHTTP(0xc20803cea0, 0xad4a28, 0xc2081820a0, 0xc2080ad040)

/Users/mitchellh/code/3rdparty/go/src/net/http/server.go:1541 +0x17d

github.com/hashicorp/vault/http.func·001(0xad4a28, 0xc2081820a0, 0xc2080ad040)

/Users/mitchellh/code/go/src/github.com/hashicorp/vault/http/help.go:18 +0x128

net/http.HandlerFunc.ServeHTTP(0xc208111b20, 0xad4a28, 0xc2081820a0, 0xc2080ad040)

/Users/mitchellh/code/3rdparty/go/src/net/http/server.go:1265 +0x41

net/http.serverHandler.ServeHTTP(0xc208107440, 0xad4a28, 0xc2081820a0, 0xc2080ad040)

/Users/mitchellh/code/3rdparty/go/src/net/http/server.go:1703 +0x19a

net/http.(*conn).serve(0xc2080665a0)

/Users/mitchellh/code/3rdparty/go/src/net/http/server.go:1204 +0xb57

created by net/http.(*Server).Serve

/Users/mitchellh/code/3rdparty/go/src/net/http/server.go:1751 +0x35e

- show quoted text -


Clay Bowen

6/16/15

Hey Jack - I ran into a similar issue (https://github.com/hashicorp/vault/issues/322) when I had the audit service running. Disable the audit service (both log and syslog) and see if that resolves your issue. I know it's not optimal, but I'm hoping that the issue gets resolved soon.

Thanks,
Clay

- show quoted text -

Jack Hsu

6/16/15

Could you tell me how to disable the audit service?

Thanks
- show quoted text -
- show quoted text -
...


Clay Bowen

6/16/15

vault audit-disable file

or

vault audit-disable syslog

You have to be logged in as an administrator first, of course. And have the VAULT_ADDR environment variable set to the proper URL, or you'll have to supply the -address parameter.

Thanks,
Clay
- show quoted text -
- show quoted text -
...

Jack Hsu

6/16/15

I assume following error meaning the audit service was not enabled. But the same problem when

"curl -v -X PUT --cert ./scripts.crt --key ./scripts.key https://127.0.0.1:8000/v1/auth/cert/login".

$ vault audit-disable syslog

Error disabling audit backend: Error making API request.

URL: DELETE http://127.0.0.1:8200/v1/sys/audit/syslog

Code: 400. Errors:

* no matching backend

$ vault audit-disable file

Error disabling audit backend: Error making API request.

URL: DELETE http://127.0.0.1:8200/v1/sys/audit/file

Code: 400. Errors:

* no matching backend

- show quoted text -
- show quoted text -
...


Clay Bowen

6/16/15

Sounds like you've run into another issue. You should post it on github so Armon and the team can look at it.

Here's what I have configured:
[root@vaultserver log]# vault auth -methods
Path Type Description
app-id/ app-id
cert/ cert
token/ token token based credentials
userpass/ userpass

I'm using Consul as my storage (as the guide shows). You may want to try enabling audit (see the documentation on the audit backend) and then disabling it. I'm also using a slightly newer version of OpenSSL (1.0.1e-30). I do have the devel packages for OpenSSL on my test vault system.

Thanks,
Clay
- show quoted text -
- show quoted text -
...

Jack Hsu

6/18/15

Clay,

Instead of manually run the "vault auth-enable cert" command each time after vault server is restarted, is there a way we can add it to config file to automatically enable this? If there is such way, I'd like to know the syntax in config file.

Thanks
- show quoted text -

Armon Dadgar

6/18/15

Other recipients: jhs...@gmail.com

Hey Jack,

You actually only need to enable the auth backend once, it is not necessary to do it on every restart.
It cannot be done via a configuration file, as it is a security sensitive configuration.

Best Regards,
Armon Dadgar

From: Jack Hsu <jh...@gmail.com>
Reply: Jack Hsu <jh...@gmail.com>>
Date: June 18, 2015 at 12:19:33 PM
To: vault...@googlegroups.com <vaul...@googlegroups.com>>
Subject: Re: Getting started guide for using CERT authentication in scripts

- show quoted text -
--

Jack Hsu

6/18/15

Other recipients: jhs...@gmail.com

OK, I guess it's because I'm running in 'dev' mode (default to inmem) which doesn't persist anything from previous state of run.

Thanks
- show quoted text -


Clay Bowen

6/19/15

I don't run vault in dev mode, so I haven't tested CERT with that configuration. Did you still get the PANIC after enabling/disabling audit?

Thanks,
Clay
- show quoted text -

Andy Swanson

12/13/16

I know this is an old thread but I would just like to say I appreciate the very concise way that you presented this. I happen to be new to consul/vault and am roiling this out in a fairly large global operation. The one piece that was annoying me was how to access the secrets in vaut without using the root key and I knew there were better ways. So I did spend time in the docs but decided to hunt for better clarity. Anyway thanks for the write up as it was ver helpful and I had a test put together in minutes!

Tim
- show quoted text -

This message has been deleted.


Vishal Nayak

2/1/17

Re: [vault] Re: Getting started guide for using CERT authentication in scripts

Hi Darryl,

I am sorry, what is the issue? I can see that the authentication is
successful and a client token is returned by Vault.

Regards,
Vishal

On Wed, Feb 1, 2017 at 1:22 PM, Darryl <darryl....@gmail.com> wrote:
> Hello,
>
>   I'm new to Vault and after attempting your tutorial I'm running into the
> following issue:
>

> After running:
> TOKEN=$(curl -insecure -s -X PUT --cert "/tmp/scripts.crt" --key
> "/tmp/scripts.key" https://vault.marathon.mesos:8200/v1/auth/cert/login|awk
> -F, '{print $5}'|awk -F: '{print $3}'|sed 's/"//g')
>
> (I do have a password set on my .key file) , there's nothing in my TOKEN
> variable which is mirrored by what happens after I run the curl bare. Now if
> I add -verbose I do see what looks like a successful connect
> /tmp # curl --verbose --insecure -s -X PUT --cert "/tmp/scripts.crt" --key
> "/tmp/scripts.key"
> https://vault.marathon.mesos:0/v1/secret/policy/scripts/apps"
> 8200/v1/auth/cert/login
> *   Trying x.x.x.111...
> * TCP_NODELAY set
> * Connected to vault.marathon.mesos (x.x.x.111) port 8200 (#0)
> * ALPN, offering http/1.1
> Enter PEM pass phrase:
> * Cipher selection:
> ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
> * successfully set certificate verify locations:
> *   CAfile: /etc/ssl/certs/ca-certificates.crt
>   CApath: none
> * TLSv1.2 (OUT), TLS header, Certificate Status (22):
> * TLSv1.2 (OUT), TLS handshake, Client hello (1):
> * TLSv1.2 (IN), TLS handshake, Server hello (2):
> * TLSv1.2 (IN), TLS handshake, Certificate (11):
> * TLSv1.2 (IN), TLS handshake, Server key exchange (12):
> * TLSv1.2 (IN), TLS handshake, Request CERT (13):
> * TLSv1.2 (IN), TLS handshake, Server finished (14):
> * TLSv1.2 (OUT), TLS handshake, Certificate (11):
> * TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
> * TLSv1.2 (OUT), TLS handshake, CERT verify (15):
> * TLSv1.2 (OUT), TLS change cipher, Client hello (1):
> * TLSv1.2 (OUT), TLS handshake, Finished (20):
> * TLSv1.2 (IN), TLS change cipher, Client hello (1):
> * TLSv1.2 (IN), TLS handshake, Finished (20):
> * SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
> * ALPN, server accepted to use http/1.1
> * Server certificate:
> * subject: CN=*
> * start date: Jan 31 20:39:16 2017 GMT
> * expire date: Jan 29 20:39:16 2027 GMT
> * issuer: CN=*
> * SSL certificate verify result: self signed certificate (18), continuing
> anyway.
>> PUT /v1/auth/cert/login HTTP/1.1
>> Host: vault.marathon.mesos:8200
>> User-Agent: curl/7.52.1
>> Accept: */*
>>
> < HTTP/1.1 200 OK
> < Content-Type: application/json
> < Date: Wed, 01 Feb 2017 18:12:04 GMT
> < Content-Length: 452
> <
> {"request_id":"9d81253d-4a5b-2931-3118-
40a10cbf22e2","lease_id":"","renewable":false,"lease_duration":0,"data":null,"wrap_info":null,"warnings":null,"auth":{"c
lient_token":"8adaeda7-26e0-bec5-4a6a-aef200ced5eb","accessor":"0fb8f62e-33c4-020f-8aa4-
62c7e1a64b13","policies":["default","scripts"],"metadata":{"authority_key_id":"","cert_name":"scripts","common_name"
:"vault.marathon.mesos","subject_key_id":""},"lease_duration":2764800,"renewable":true}}
> * Curl_http_done: called premature == 0
> * Connection #0 to host vault.marathon.mesos left intact

>
> Did I miss a step?
>
> Thank you
>
>
> On Monday, June 8, 2015 at 3:51:27 PM UTC-4, Clay Bowen wrote:
>>
>> I wrote this today for my company, and I'd appreciate feedback. I may
>> have left out some steps or put too much detail in others.
>>
>> Thanks,
>> Clay
>>
> --
> This mailing list is governed under the HashiCorp Community Guidelines -
> https://www.hashicorp.com/community-guidelines.html. Behavior in violation
> of those guidelines may result in your removal from this mailing list.
>
> GitHub Issues: https://github.com/hashicorp/vault/issues
> IRC: #vault-tool on Freenode
> ---
> You received this message because you are subscribed to the Google Groups
> "Vault" group.
> To unsubscribe from this group and stop receiving emails from it, send an
> email to vault-tool+...@googlegroups.com.
> To view this discussion on the web visit
> https://groups.google.com/d/msgid/vault-tool/65fdeed0-6031-4e9b-a497-6fc46302c026%40googlegroups.com.
> For more options, visit https://groups.google.com/d/optout.

--
vn

Darryl

2/2/17

Re: [vault] Re: Getting started guide for using CERT authentication in scripts

Honestly, the issue was a typo "-insecure" rather than "--insecure", caught it right after I posted that. Apologies

- show quoted text -
- show quoted text -

To view this discussion on the web visit https://groups.google.com/d/msgid/vault-tool/CAFy0tpzxfVOrTVw0iKMPLtDaPyq18vLij11FN6S-gFVXQCXxYA%40mail.gmail.com.

For more options, visit https://groups.google.com/d/optout.