Lab 6 – Vault

Introduction

Ansible vault allows us to keep sensitive and secret data like passwords, api keys, etc in encrypted files rather than storing them in a plaintext playbook or somewhere else where someone could see it.

We will be going over Vault in this lab and the different use cases for it and how to use it.

Please refer to the **Ansible-Pod-Info.docx** file for information on connecting to your Ansible host.

Don't forget to change the XX in your inventory file based on your Pod information.

```
cd ~/ansible_labs/lab6-vault
```

1. Encrypting Existing Files

Let's say you have a file already with some sensitive data in it that you want to encrypt with vault. We can use the **ansible-vault encrypt** command to accomplish this. In our example we are going to be building up to a bigger use case in section 4. So we will be creating a vault file in a group_vars folder in order to have a seamless playbook experience.

2.1 First we need to create what will become our vault file eventually. Replace the n9k_pw value with your switch password.

```
echo 'n9k_user: admin'  >> ssh_pass.txt

echo 'n9k_pw: password'  >> ssh_pass.txt
```

If we cat this file we can see the contents in plain text which isn't good.

```
cat ssh_pass.txt
```

Now that we have a file that contains the ssh information for our switch lets encrypt it. You can see from the output it will ask us for a password twice, set this to ansible or something you will remember and then it should encrypt it.

```
ansible-vault encrypt ssh_pass.txt

New Vault password:

Confirm New Vault password:

Encryption successful
```

Now if we cat this file we will see something very different:

```


$ANSIBLE_VAULT;1.1;AES256

32303339646363643261383132396536333334306137653465326432633433
36461373336626130 36

32363037623764346332333435633533383862623233333340a3661333236
383135316430306234 35

62383234336565323133633739343833636536653431396334633430633886
535333336633163356 4

31343233303863383330a30636662656164616339346632356136353663313
638633866623930623 7

34313366623439613131313631376334373136313763363533386336363131
316564
```

3. Viewing Encrypted Files

Sometimes you will need to view encrypted files to see their values and such, there is an ansible command for that called **ansible-vault view**.

3.1 Let’s take a look at the previous password file we created using the ansible-vault view command.

```
ansible-vault view ssh_pass.txt

Vault password:

n9k_user: admin

n9k_pw: password
```

So we can see here it asks us for our vault password then gives us the value. We could use this for ansible or for other things as well to secure these files. You can use this for ssh keys, passwords, all sorts of things.

4. Create/Editing Encrypted Files

4.1 So in the previous section we took an existing file we had and made it into an encrypted file, we can also just save our self some time and if we are creating a new file we can just use the **ansible-vault create** command to create it and **ansible-vault edit** to modify it.

By also creating this in the group_vars/nxos folder we are able to tell ansible that this vault is for the group nxos and supply additional information in the nxos.yaml file in this same nxos folder for our connection information.

4.2 Encrypt SSH Password

Use Case: We need to encrypt our ssh password for our devices in ansible so we don't have to specify password every time.

4.2.1 Let's create an encrypted data file for our nexus 9k switch. This command below opens up a vim editor for us to modify our password file.

```
ansible-vault create group_vars/nxos/vault
```

Note: Enter ansible for the vault pw, this could normally be a secure pw but for our use case let's just make it easy to remember.

4.2.2 Now we should have an editor where we can specify our different ssh passwords based on devices we have. Add the following lines to the file and use the username and pw supplied from your lab handout.

Here we are supplying our username and pw for our nexus 9k switch so we don't have to supply the pw anymore and it's secured.

```
n9k_user: username_here

n9k_pw: password_here
```

Now we can hit esc and then :wq! to save our file.

Let's test our vault file by specifying the --ask-vault-pass, since we set this up as a group var we don't need to give it anything else.

```
ansible-playbook -i inventory nxos_facts_play.yaml  --ask-va
ult-pass
```

Suppose we needed to edit this file because we made a mistake, we can use the ansible-vault edit command to modify our file.

```
ansible-vault edit group_vars/nxos/vault
```

This command will give us a default editor, in our case should be VIM where we can modify the file then using VIM’s command !wq after hitting ESC to save it.

4.2.3 Here we are specifying --ask-vault-pass and will supply it our password, you could also have this stored in a file that isn’t version controlled and only allowed for say the ansible user or such to use and load it that way as well.

Here is the sample output for our group vars, can you see from this how we are accomplishing this?

```
ansible_connection: network_cli

ansible_network_os: nxos

ansible_user: "{{ n9k_user }}"

ansible_ssh_pass: "{{ n9k_pw }}"

transport: nxapi
```

By specifying in the group vars the ansible_user and ansible_ssh_pass to our variables we created in our vault file, we are able to only have to remember the main vault pw in order to run our playbook.

5. Manually Decrypting Encrypted Files

5.1 If we want to remove encryption from a file permanently we can use the **ansible-vault decrypt** command.

```
ansible-vault decrypt ssh_pass.txt
```

This will ask us for our Vault password and now the file is decrypted. So if we cat it, we should see the values we put into it earlier in the lab.

```
cat ssh_pass.txt
```

Now this file is decrypted and we would not want to store this anywhere in version control, etc.

6. Changing the Password of Encrypted Files

6.1 We can also change a password on an encrypted file by using the ansible-vault rekey command. Let’s encrypt the ssh_pass.txt again and then rekey the vault password.

First we have to encrypt it with ansible-vault since we just decrypted it earlier in the previous step. Then we can rekey it.

```
ansible-vault encrypt ssh_pass.txt
```

This will ask us for a new vault password which we will specify, you can give it ansible or something you will remember.

Now we can run our rekey command to change our ansible password for this vault file.

```
ansible-vault rekey ssh_pass.txt
```

When you enter this command it will ask for the original vault password, then ask for the new one twice and as long as everything was typed correctly now your new password is setup.

Output:

```
Rekey successful
```

7. Using Ansible Vault with a Password File

7.1 If you don’t want to have to keep typing the ansible vault password everytime you run a playbook then we have an option to add our vault password to a file then reference it during the playbook execution.

Since we have been using ansible as our password we will be echo’ing that to a .vault_pass file that we will then use to tell our playbook where to get our vault password from.

```
echo 'ansible' > .vault_pass
```

So now we can re-run our nxos playbook example we ran before and we won’t have to specify the password this time

```
ansible-playbook -i inventory nxos_facts_play.yaml --vault-p
assword-file=.vault_pass
```

8. Reading the Password File Automatically

8.1 We can also tell ansible about our password file via an environment variable so we don’t have to provide any ansible-vault flag at all. We can use the ANSIBLE_VAULT_PASSWORD_FILE environment variable and set it to our vault_pass file we created.

```
export ANSIBLE_VAULT_PASSWORD_FILE=./.vault_pass
```

8.2 Now we can execute the command without having to supply the --vault-password or --vault-password-file flag.

```
ansible-playbook -i inventory nxos_facts_play.yaml
```

9. Knowledge Check 1

Use Case: Setup ssh username and pw in vault for our IOS router and then configure a playbook to modify the MOTD, and assert to verify the change took place.

1. Your working directory is **section9_knowledgecheck**
2. Using the username and password provided on your lab sheet setup a vault store for the login information.
3. Use this vault store in a playbook configuration to configure a MOTD on the Cisco IOS device.
4. Our host group is called ios for this play, use the provided inventory file and modify it based on your device list.

5. The MOTD can say whatever you choose.
6. Utilize group_vars, ansible.cfg, with the vault so nothing else is required on the CLI but **ansible-playbook ios ios_motd.yaml**

Helpful Links

https://docs.ansible.com/ansible/latest/network/user_guide/platform_ios.html

https://docs.ansible.com/ansible/latest/modules/ios_banner_module.html

10. Securing Sensitive Data

10.1 Suppose we want to secure our MOTD message because it contains some email address or something we don't want visible in version control. We can convert our existing playbook from our previous Knowledge Check and move the MOTD into a vault file with our ssh user and pass and then configure our playbook to use the variable using Jinja2 templating which we will discuss in a future module.

Jinja2 basically takes our variable from the vault file and inserts it into the play.

```
ansible-vault edit group_vars/ios/vault
```

Now why are we getting this error when we try to edit this file?

```
ERROR! The vault password file .vault_pass was not found
```

10.2 The reason is because in our last section we told ansible to always look for a .vault_pass file in our working dir when we run a playbook, so let’s copy that from the directory above:

```
cp ../.vault_pass .
```

Now let’s try to re-run our edit command and add our MOTD in as a variable

```
ansible-vault edit /group_vars/ios/vault
```

10.3 Now let’s add our MOTD message text as a value of key motd_message

```
router_user: admin

router_pw: password

motd_message: This is my secure MOTD message, it is coming f
rom a vault file.
```

Now we can copy our existing ios_motd.yaml file to a new file and modify it use our new secure MOTD message.

```
cp ios_motd.yaml secure_change_motd.yaml
```

10.4 Now let’s modify this playbook to use our new MOTD message from our vault.

```
vim secure_change_motd.yaml
```

We just need to change the section where the MOTD line is

```
From This:
```

```
text: |

  This MOTD was written by ansible

  it is a multiline banner



To This:

text: "{{ motd_message }}"
```

We are using Jinja2 templating with the “{{ motd_message }}” which it will pull from our vault file.

Now we can run our secure_change_motd.yaml playbook and modify our MOTD. Run your other playbook to show the MOTD as well and verify it worked.

```
ansible-playbook -i inventory secure_change_motd.yaml
```

11. Knowledge Check 2

Use Case: We found a playbook setup with sensitive data in it and we want to be able to version control this file. Setup the playbook and group vars to be secure so you can push this to Git.

1. Your working directory is **section11_knowledgecheck**
2. Utilizing your previous playbook and setup files from **section9_knowledgecheck** setup the MOTD text to be in a secure vault store. You can copy these over by doing **cp -r ../section9_knowledgecheck**, to get started.
3. The playbook and working directory is provided, look through the files and make sure all sensitive information is stored in vault then referenced.
4. Verify the playbook works and is secure, once secured then push this to your git repo.

Version Control Commit

Now add all your new files to your repository and push it up, reference the GitHub lab if you get stuck or ask for help. Ensure the vault file is not added to version control for now.