**Submitted By:Ruchira Pokhriyal**
**Student ID: 801085619**

**Question 1: Document what Vault is used for?**

Solution:
***A vault, as the name suggests provides some specific services to ensure that the contents within it are protected.***
→For instance, a vault can be used to provide secure access to sensitive information like passwords, certificates, API keys etc.
→Vault has a built-in support for secret revocation and can revoke not just single secrets, but all the secrets read by a specific user. It provides a secure way to store credentials that employees share to access web services. Vault provides acceptable level of security for DevOps engineers.
→Vault provides the dynamic secrets feature for scripts wherein an AWS access key can be generated for the duration of a script and then revoked. This is an excellent security measure. Another notable highlight is the audit log mechanism of Vaults which could tell us about the secrets that employees' access, and when an employee leaves, it is easier to understand which keys have been rolled and which haven't.
→Additionally, Vaults capabilities are not limited to just storing and protecting sensitive data, it provides enhanced security features like encryption and decryption of data stored elsewhere- the primary use being, allowing the applications to encrypt their data while still storing it in the primary data store. This takes the load of encryption off of a developer's shoulders.
→Furthermore, Vault manages the encryption process between the server and applications. This way, Vault is able to create a centralized "secret storage" which is more secure and easier to manage.

**Question 2: Describe how it might be used with our existing infrastructure (nginx and MySQL)**
Solution:
Vault can easily integrate and manage access to the existing Nginx and MySQL infrastructure. We can use MySQL secrets engine to dynamically generate database credentials, based on the configured roles-which means, the services requiring access to a database don't need to hardcode the credentials anymore, instead, these can be requested from the Vault itself.
Vault could be useful for authentication of user login using LDAP too (users trying to login using LDAP would be authenticated by Vault). The LDAP auth method allows authentication using an existing LDAP server and credentials (username & password) which allows Vault to be integrated into environments using LDAP, without the trouble of duplicating username & password configuration in multiple places. MYSQL and NGINIX access would be managed by Vault, by positioning it between the user and applications. Vault can also be used for securing HTTP API in our nginx.conf. We can configure Nginx as a reverse proxy and can intercept a received HTTP redirect by defining a named location.

**Question 3: What hard-coded secrets do we currently have?**
Solution:
*Current hard-coded secrets:*
Passcodes for Nginx, LDAP, Kerberos, MYSQL, SSH (hard-coded into the server)

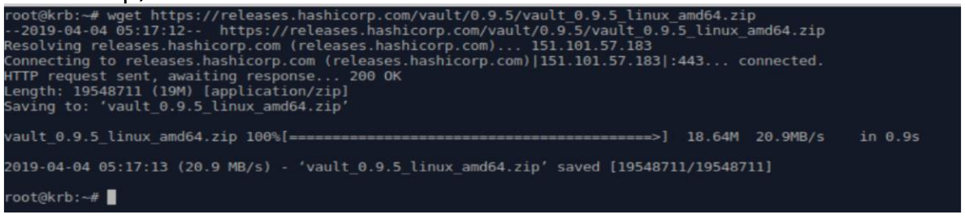
**Submitted By:Ruchira Pokhriyal**
**Student ID: 801085619**

Certificates and private SSH keys (hard-coded into our infrastructure)

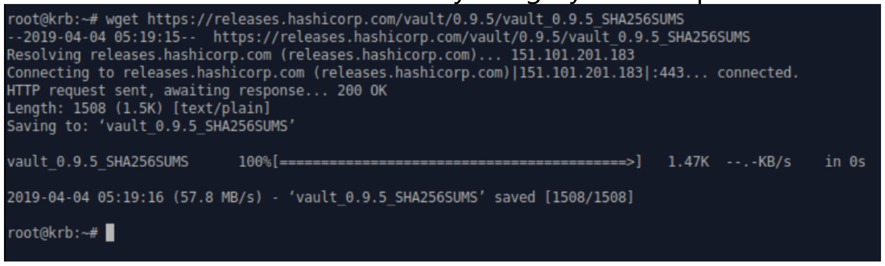
**Question 4: Does Vault have plugins to deal with these secrets?**
Solution:
Yes, it does. Vault has the authentication plugins for auth method, Active Directory, and password generator. Additionally, it uses plugin abstraction for supporting MySQL.

**Question 5: Describe the setup process for Vault and how it integrates with our existing infrastructure.**
Solution:
Setup process for Vault in brief:
→First, users must download the Vault binary from the Vault website. This would be a zip file.
→Next, users would unzip the download file. This could be done using Tar in Linux and WinZip in Windows.
→Then, users would move the executable to a folder within our PATH and point Vault to the Vault server.
→Then, we test for authentication using LDAP credentials.
→Once authenticated, access would be granted to certain policies and secrets in the Vault until the token expires. Then the secrets can be created, read, updated and deleted.

Installation Screen-Shots:
 I manually downloaded and installed Vault's executable into my client container (provided by HashiCorp).

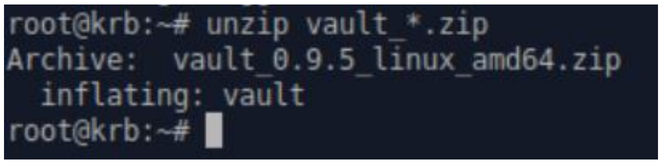

```
root@krb:~# wget https://releases.hashicorp.com/vault/0.9.5/vault_0.9.5_linux_amd64.zip
--2019-04-04 05:17:12--  https://releases.hashicorp.com/vault/0.9.5/vault_0.9.5_linux_amd64.zip
Resolving releases.hashicorp.com (releases.hashicorp.com)... 151.101.57.183
Connecting to releases.hashicorp.com (releases.hashicorp.com)|151.101.57.183|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19548711 (19M) [application/zip]
Saving to: 'vault_0.9.5_linux_amd64.zip'

vault_0.9.5_linux_amd64.zip 100%[==========================================>]  18.64M  20.9MB/s    in 0.9s

2019-04-04 05:17:13 (20.9 MB/s) - 'vault_0.9.5_linux_amd64.zip' saved [19548711/19548711]

root@krb:~# 
```

Downloaded the checksum to verify integrity of the zip file:


```
root@krb:~# wget https://releases.hashicorp.com/vault/0.9.5/vault_0.9.5_SHA256SUMS
--2019-04-04 05:19:15--  https://releases.hashicorp.com/vault/0.9.5/vault_0.9.5_SHA256SUMS
Resolving releases.hashicorp.com (releases.hashicorp.com)... 151.101.201.183
Connecting to releases.hashicorp.com (releases.hashicorp.com)|151.101.201.183|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1508 (1.5K) [text/plain]
Saving to: 'vault_0.9.5_SHA256SUMS'

vault_0.9.5_SHA256SUMS      100%[==========================================>]   1.47K  --.-KB/s    in 0s

2019-04-04 05:19:16 (57.8 MB/s) - 'vault_0.9.5_SHA256SUMS' saved [1508/1508]

root@krb:~# 
```

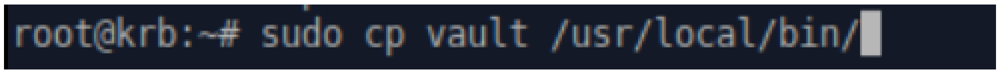
unzipped the Vault binary into working directory:

**Submitted By:Ruchira Pokhriyal**
**Student ID: 801085619**



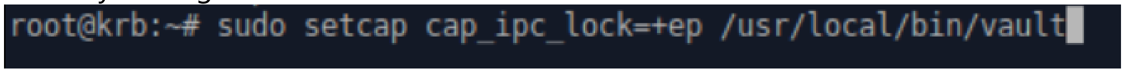
```
root@krb:~# unzip vault_*.zip
Archive:  vault_0.9.5_linux_amd64.zip
  inflating: vault
root@krb:~#
```

Moved the Vault executable into a directory in the system's path to make it accessible from the shell:



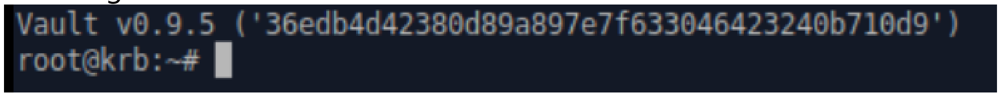
```
root@krb:~# sudo cp vault /usr/local/bin/
```

Setting Linux capability flag on the binary to add extra security by letting the binary perform memory locking:



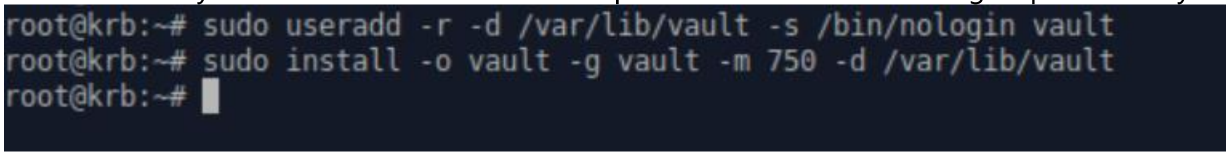
```
root@krb:~# sudo setcap cap_ipc_lock=+ep /usr/local/bin/vault
```

Checking Vault's Version:



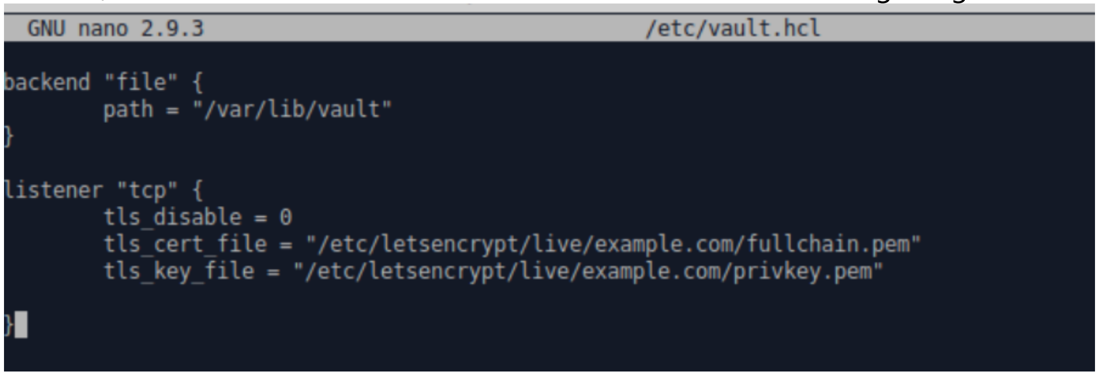
```
Vault v0.9.5 ('36edb4d42380d89a897e7f633046423240b710d9')
root@krb:~#
```

Created vault system user and set the ownership to this user and the vault group exclusively:



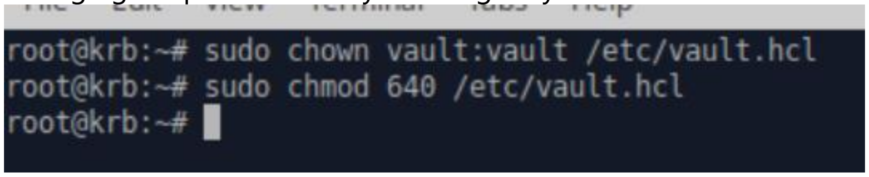
```
root@krb:~# sudo useradd -r -d /var/lib/vault -s /bin/nologin vault
root@krb:~# sudo install -o vault -g vault -m 750 -d /var/lib/vault
root@krb:~#
```

Created a new configuration file called vault.hcl which instructs Vault to store encrypted secrets on-disk, and that Vault should listen for connections via HTTPS using the generated certificates:



```
  GNU nano 2.9.3                              /etc/vault.hcl

backend "file" {
        path = "/var/lib/vault"
}

listener "tcp" {
        tls_disable = 0
        tls_cert_file = "/etc/letsencrypt/live/example.com/fullchain.pem"
        tls_key_file = "/etc/letsencrypt/live/example.com/privkey.pem"

}
```

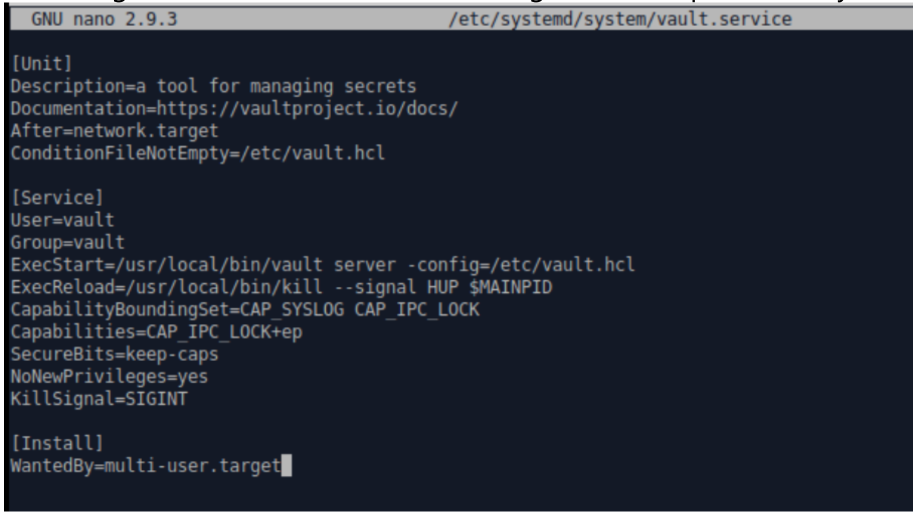
Changing file permissions by allowing only the vault user to read it:



```
root@krb:~# sudo chown vault:vault /etc/vault.hcl
root@krb:~# sudo chmod 640 /etc/vault.hcl
root@krb:~#
```

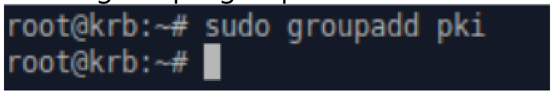
**Submitted By:Ruchira Pokhriyal**
**Student ID: 801085619**

Following allows Vault to run in the background as a persistent system service daemon:



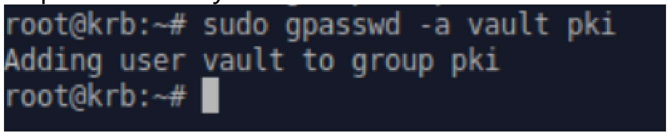
```
  GNU nano 2.9.3                         /etc/systemd/system/vault.service

[Unit]
Description=a tool for managing secrets
Documentation=https://vaultproject.io/docs/
After=network.target
ConditionFileNotEmpty=/etc/vault.hcl

[Service]
User=vault
Group=vault
ExecStart=/usr/local/bin/vault server -config=/etc/vault.hcl
ExecReload=/usr/local/bin/kill --signal HUP $MAINPID
CapabilityBoundingSet=CAP_SYSLOG CAP_IPC_LOCK
Capabilities=CAP_IPC_LOCK+ep
SecureBits=keep-caps
NoNewPrivileges=yes
KillSignal=SIGINT

[Install]
WantedBy=multi-user.target
```

Creating the pki group:



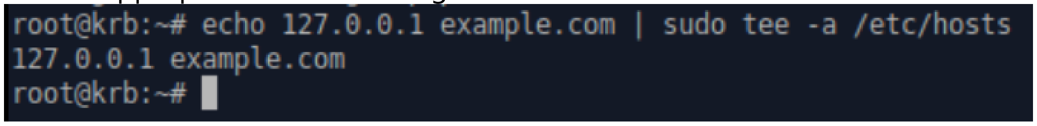
```
root@krb:~# sudo groupadd pki
root@krb:~#
```

Adding the vault user to the pki group will grant Vault access to the certificates to serve requests securely over HTTPS:



```
root@krb:~# sudo gpasswd -a vault pki
Adding user vault to group pki
root@krb:~#
```

Following would append the line *127.0.0.1 example.com* to */etc/hosts* so that the HTTP requests to example.com are routed to localhost. Here, I could replace "example.com" with my own domain. Finally, I can initialize vault with systemctl start vault and start reading and writing secrets appropriate to our existing infrastructure.



```
root@krb:~# echo 127.0.0.1 example.com | sudo tee -a /etc/hosts
127.0.0.1 example.com
root@krb:~#
```

**Question 6: Document alternatives to Vault. What are the tradeoffs for each of these solutions?**

Solution:

Below are a few alternatives to Vault:

**Hardware Security Module(s) (HSMs):**

→Expensive and not very cloud friendly.
→Once an HSM is up and running, configuring it is generally very tedious,
→The API to request secrets is difficult to use.

On the other hand, Vault can do several things that HSMs can't, for instance generating dynamic secrets.
→Vault can generate access keys according to a specific policy on the fly.
→Vault can also be used along with HSMs to provide easy configuration and better security.

**Chef, Puppet:**
→Chef, Puppet perform single-key encrypted storage for storing secrets.
→Chef has encrypted data bags.
→Puppet has encrypted Hiera.
→This creates a security issue as the sensitive encrypted data is always one secret (a password, a key, etc.) away from being decrypted.
→And since, in an elastic environment, all the servers need to have access to the secret to decrypt the data, this secret itself is generally not well protected.
→Access to the encrypted data isn't always logged, so if there is an intrusion, it isn't clear what data has been accessed and by who.

→In case of Vault, the configuration management requires fewer secrets, and in many cases doesn't ever have to persist them to disk. This gives better protection of sensitive information.
→Vault conducts a special feature when it starts called Unsealing- In this, Vault encrypts data onto physical storage and requires multiple keys to read it. So, if an intruder wants to gain access to the physical encrypted storage, it can't be read without multiple keys and these multiple keys are distributed to multiple individuals.
→For an unsealed Vault, every interaction is logged in via the audit devices and to gain access to any data, an access token is required. Again, enhanced security as the token is associated with an identity coming from a system such as LDAP, GitHub etc. The identity too is written to the audit log.

**Consule:**
→Although, Consule can successfully be used to store sensitive information and gate access using ACLs, it is not originally designed for this purpose. Hence, data isn't encrypted in transit nor at rest.
→Consule does not have pluggable authentication mechanisms.
→There isn't any per-request auditing mechanism as well.

→Vault on the other hand is specifically designed as a secret management solution.
→Hence, it protects secrets in transit as well as at rest.
→Vault provides several authentication and audit logging mechanisms.
→Dynamic secret generation feature of Vault, allows it to avoid providing clients with root privileges to underlying systems and therefore, it's possible to perform key rolling and revocation.

**Amazon Key Management Service (KMS):**
→A service provided in the AWS ecosystem for encryption key management.

**Submitted By:Ruchira Pokhriyal**
**Student ID: 801085619**

→Uses HSM for physical security.
→It is focused on securely storing encryption keys and supporting cryptographic operations using those keys. It also supports access controls and auditing.

→In contrast, Vault provides a comprehensive secret management solution, provides capabilities similar as KMS and goes much further than just key management.
→The flexible secrets engines allows Vault to handle any type of secret data, including API keys, PKI keys, encryption keys and even database credentials.
→Vault supports: dynamic secrets, non-repudiation, generating credentials on-demand for fine-grained security controls, auditing.
→All the secrets read from Vault have a mandatory lease associated to it, which would enable operations to audit key usage, perform key rolling, and also ensure automatic revocation.
→Vault provides multiple revocation mechanisms procedure after compromise.