"Protecting Your Secrets: How to Encrypt Kubernetes Secrets in etcd"

Kubernetes: Encrypting Secrets in etcd

In Kubernetes, secrets are stored in the `etcd` key-value store, which serves as the cluster's primary data storage. By default, secrets are only Base64-encoded, not encrypted, which makes them potentially vulnerable if `etcd` is compromised. Encrypting secrets at rest provides an additional layer of security to protect sensitive data.

Step 1: Create a Kubernetes Secret

1. **Create a Secret Manifest File** (e.g., `secret.yaml`):

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: my-secret
     namespace: default
   type: Opaque
   data:
     username: YWRtaW4=    # Base64 encoded value for 'admin'
     password: cGFzc3dvcmQ= # Base64 encoded value for 'password'
   ```

2. **Apply the Secret Manifest**:

   ```
   kubectl apply -f secret.yaml
   ```

3. **Verify the Secret**:

   ```
   kubectl get secrets my-secret -o yaml
   ```

Step 2: Access the `etcd` Database

To verify the secret is stored in `etcd`, use the `etcdctl` command:

```
etcdctl get /registry/secrets/default/my-secret --
cert=/etc/kubernetes/pki/etcd/peer.crt \
--key=/etc/kubernetes/pki/etcd/peer.key --
cacert=/etc/kubernetes/pki/etcd/ca.crt
```

Decoded Output:

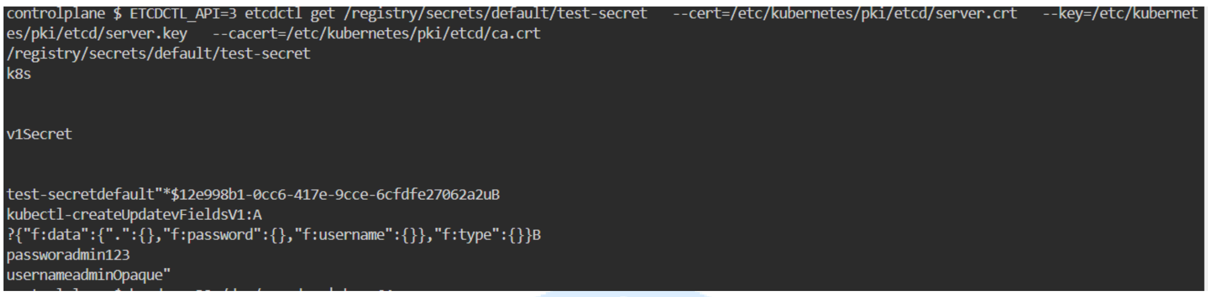
The secret will appear in Base64-encoded format. Decode it to view the actual data:

```
{
  "f:data": {
    ".": {},
```

```
    "f:password": {},
    "f:username": {}
  },
  "f:type": {}
}
```

Decoded data:

- **username**: admin
- **password**: password



```
controlplane $ ETCDCTL_API=3 etcdctl get /registry/secrets/default/test-secret   --cert=/etc/kubernetes/pki/etcd/server.crt   --key=/etc/kubernet
es/pki/etcd/server.key   --cacert=/etc/kubernetes/pki/etcd/ca.crt
/registry/secrets/default/test-secret
k8s


v1Secret


test-secretdefault"*$12e998b1-0cc6-417e-9cce-6cfdfe27062a2uB
kubectl-createUpdatevFieldsV1:A
?{"f:data":{".":{},"f:password":{},"f:username":{}},"f:type":{}}B
passworadmin123
usernameadminOpaque"
```

Steps to Enable Secrets Encryption in etcd

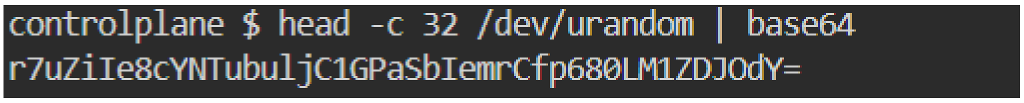
Step 1: Generate an Encryption Key

To generate a secure Base64-encoded encryption key, use:

```
head -c 32 /dev/urandom | base64
```

Example generated key:

```
o556O5o4A2mSFccVEJcQdRiJ+YiYT23H8uGZYqPt+JM=
```



```
controlplane $ head -c 32 /dev/urandom | base64
r7uZiIe8cYNTubuljC1GPaSbIemrCfp680LM1ZDJOdY=
```

Step 2: Create an Encryption Configuration File

Define the encryption provider and specify the encryption key in a new file called `encryption-config.yaml`. Example:

```
apiVersion: apiserver.config.k8s.io/v1
kind: EncryptionConfiguration
resources:
  - resources:
      - secrets
    providers:
      - aescbc:  # Encryption provider (AES-CBC)
          keys:
            - name: key1
              secret: r7uZiIe8cYNTubuljC1GPaSbIemrCfp680LM1ZDJOdY=
      - identity: {}  # Fallback for non-encrypted data
```

Step 3: Copy the Configuration File to the Correct Location

Copy the `encryption-config.yaml` file to the `/etc/kubernetes/pki/` directory:

```
sudo cp encryption-config.yaml /etc/kubernetes/pki/
```

Step 4: Verify the File Location

Check if the file has been copied successfully:

```
ls -l /etc/kubernetes/pki/encryption-config.yaml
```

Step 5: Update the API Server Configuration

Modify the `kube-apiserver` manifest to include the encryption configuration. The manifest file is typically located at `/etc/kubernetes/manifests/kube-apiserver.yaml`.

Add the following argument:

```
- --encryption-provider-config=/etc/kubernetes/pki/encryption-config.yaml
```

Step 6: Restart the API Server

After saving the changes, the `kube-apiserver` pod will automatically restart. You can verify this by checking the pods in the `kube-system` namespace:

```
kubectl get pods -n kube-system
```

Step 7: Re-encrypt Existing Secrets

Newly created secrets will be encrypted automatically. However, existing secrets will remain unencrypted. To re-encrypt them, you can:

- **Backup and Restore** secrets using a script or tool:

  ```
  kubectl get secrets --all-namespaces -o yaml | kubectl replace -f -
  ```

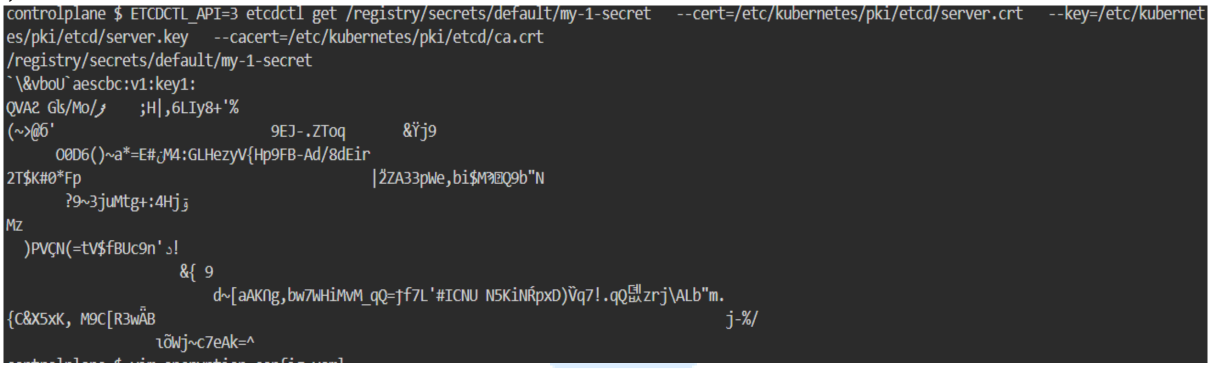
Step 8: Verify Encryption

After enabling encryption at rest, verify that secrets are indeed encrypted in `etcd`:

```
ETCDCTL_API=3 etcdctl get /registry/secrets/default/my-secret \
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt
```

Output Before Enabling Encryption at Rest:

```
/registry/secrets/default/my-secret
{
    "kind": "Secret",
    "apiVersion": "v1",
    "metadata": {
        "name": "my-secret",
        "namespace": "default",
        "uid": "12345",
        "creationTimestamp": "2024-01-01T12:00:00Z"
    },
    "data": {
        "password": "dGVzdHBhc3N3b3Jk"  # Base64 encoded data (e.g.,
"testpassword")
    },
    "type": "Opaque"
}
```



```
controlplane $ ETCDCTL_API=3 etcdctl get /registry/secrets/default/my-1-secret   --cert=/etc/kubernetes/pki/etcd/server.crt   --key=/etc/kubernet
es/pki/etcd/server.key   --cacert=/etc/kubernetes/pki/etcd/ca.crt
/registry/secrets/default/my-1-secret
`\&vboU`aescbc:v1:key1:
QVA2 Gls/Mo/ز    ;H|,6LIy8+'%
(~>@б'                          9EJ-.ZToq       &Ÿj9
      O0D6()~a*=E#¿M4:GLHezyV{Hp9FB-Ad/8dEir
2T$K#0*Fp                                |ŽZA33pWe,bi$M?Q9b"N
       ?9~3juMtg+:4Hjق
Mz
  )PVÇN(=tV$fBUc9n'د!
                     &{ 9
                         d~[aAKng,bw7WHiMvM_qQ=ƫf7L'#ICNU N5KiNŔpxD)V̈q7!.qQzrj\ALb"m.
{C&X5xK, M9C[R3wȀB                                                            j-%/
                   ıõWj~c7eAk=^
```