Base64 is not encryption

A better story for Kubernetes secrets
What's a secret?
Secret (noun)

Credentials, configurations, API keys, or other pieces of information needed by an application at build time or run time
Why protect secrets?

- Attractive target for hackers
- Often leaked in repos or storage buckets
- Frequently includes overly broad permissions
Protecting secrets

Audit
Verify and log the use of individual secrets to a central system

Encrypt
Always encrypt secrets in transit with TLS and at rest

Rotate
Change a secret regularly or in case of suspected compromise

Isolate
Separate where secrets are used from where secrets are managed
Protecting secrets

**Audit**
Verify and log the use of individual secrets to a central system.

**Encrypt**
Always encrypt secrets in transit with TLS and at rest.

**Rotate**
Change a secret regularly or in case of suspected compromise.

**Isolate**
Separate where secrets are used from where secrets are managed.
Layers of encryption

- Application-layer encryption
- Service-level encryption
- Filesystem encryption
- Machine-level encryption
App-layer encryption

- Applied at earliest possible step
- Provides protection a very granular level
- Protects data as it moves through the system
Kubernetes defaults
Insecure by default

Secrets are stored in plaintext in etcd. They are base64-encoded, but not encrypted.
Insecure by default*

Secrets are stored in plaintext in etcd. They are base64-encoded, but not encrypted.

* Many providers alter this default behavior.
Encryption

shodan.io/search?query=etcd
Demo
Envelope encryption
Envelope encryption

Data

DEK
Data encryption key

KEK
Key encryption key
Encrypted data

Encrypted DEK
Envelope encryption

- Generate unique DEKs for each data entry
- Crypto-shred - revoke KEK and data is gone
- Easy versioning and rotation
Kubernetes 1.7

Envelope encryption
kind: EncryptionConfiguration
apiVersion: apiserver.config.k8s.io/v1
resources:
- resources:
  - secrets
providers:
- aescbc:
  keys:
  - name: key1
    secret: 9RlIhvmh1e6+Ixv0CjyUkA==
  - name: key2
    secret: u+aswHTypAyoRKH5/P0r5A==
- secretbox:
  keys:
  - name: key1
    secret: 9aHuiH/wrlmWEXZp9br4og==
./kube-apiserver \
   --encryption-provider-config=/etc/encryption-config.yaml \ 
   --other-options...
EncryptionConfiguration

kube-apiserver

etcd

Master
EncryptionConfiguration

kube-apiserver

etcd

Master
Drawbacks

• Need to generate keys yourself
• Key management is your responsibility
• Rotation is a manual process (and tedious)
• No HSM integration
Drawbacks

The underlying encryption keys are still stored in plaintext on the filesystem!
Kubernetes 1.10
KMS encryption providers
kind: EncryptionConfiguration
apiVersion: apisher.config.k8s.io/v1
resources:
- resources:
  - secrets
providers:
- kms:
  name: myKmsPlugin
  endpoint: unix:///tmp/kms-socketfile.sock
  cachesize: 100
EncryptionConfiguration

kube-apiserver

etcd

KMS

01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 01100101
01101011 00100000
01100100 01100101
01101011 00100000

Master
Existing plugins (GitHub)

- GoogleCloudPlatform/k8s-cloudkms-plugin
- Azure/kubernetes-kms
- kubernetes-sigs/aws-encryption-provider
- oracle/kubernetes-vault-kms-plugin
GKE Integration (beta)

gcloud beta container clusters create my-cluster
   --database-encryption-key-location us-east1
   --database-encryption-key-keyring my-keyring
   --database-encryption-key my-crypto-key
Initial secret problem?

- IAM can solve the "first secret" problem
- Delegate PAM to the cloud provider via IAM
- Separate concerns: etcd nodes don't need IAM permissions to talk to KMS
EncryptionConfiguration

kube-apiserver

Master

KMS

etcd

01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 01100101
01101011 00100000
Summary
Summary

- Use at least two layers of encryption
- Rotate keys regularly
- Leverage envelope encryption
- Protect K8S secrets using an external KMS
Thanks!

@sethvargo
Developer Relations Engineer