
KubeCon
CloudNativeCon
North America 2023


KubeCon | CloudNativeCon

North America 2023

Securing Kubernetes: Migrating From Long-Lived to Time-Bound Tokens Without Disrupting Existing Applications

Yuan Chen, James Munnelly

Apple Inc.

Some things about us

Yuan Chen, Software Engineer at Apple
Active contributor of sig-scheduling, sig-node
yuanchen8911@github
YuanChenByte@twitter
https://www.linkedin.com/in/yuanchen

James Munnelly, Field Engineer at Apple
Active member of sig-apimachinery, sig-testing,sig-auth
munnerz@github
JamesMunnelly@twitter
https://www.linkedin.com/in/munnellyjames/

Agenda

Introduction

- Service account tokens in Kubernetes
- Parameters and feature gates

Impact on different use cases

- Inside pods and Kubernetes
- External system integration

Tracking and monitoring

- Annotations
- Metrics

Migration and upgrade

- When and how
- Integration of external systems

Service account tokens in Kubernetes

Auto-generated secret-based long-lived tokens

- Automatically generated when creating service accounts
- Less secure
- No longer recommended

Time-bound tokens

- Obtained through the *TokenRequest API*
- Refresh periodically
- E.g., projected volume pod tokens

Manually created secrets with long-lived tokens

- Through a special service account annotation in a secret

Goals

Replace legacy long-lived secret-based tokens with time-bound tokens

- More secure
- Audience bound
- Removed when a pod is deleted

Upgrade and integrate tokens seamlessly without disrupting current usage

- Service account tokens for pods
- Tokens used by external systems/applications

Auto-generated secret-based tokens


```
[yuanchen:~]$ kubectl get serviceaccount default -o yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  creationTimestamp: "2021-10-18T14:59:35Z"
  name: default
  namespace: default
  resourceVersion: "352830"
  uid: 727f75de-b86f-4855-8f26-37842d0eb1a2
secrets:
- name: default-token-5dnt5
```

```
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: default
    kubernetes.io/service-account.uid: 727f75de-b86f-4855-8f26-37842d0eb1a2
  creationTimestamp: "2021-10-18T14:59:35Z"
  name: default-token-5dnt5
  namespace: default
  resourceVersion: "352829"
  uid: 7306a81a-6421-45f6-b3fd-e524a4fead24
type: kubernetes.io/service-account-token
```

- Bound to a pod and mounted at */var/run/secrets/kubernetes.io/serviceaccount*
- Non pod-bound: stored and used by external services
- Do not expire
- Shared by multiple pods/apps.
- Bi-directionally referenced by a service account and a secret

Time-bound tokens via TokenRequest API

Simplify the token creation process and enhance security

- Request a token for a given service account through the token request API
- Audience-bound
- Time limited with expiration time

PAYLOAD: DATA


```
{
  "aud": [
    "https://kubernetes.default.svc.cluster.local"
  ],
  "exp": 1657621213,
  "iat": 1657617613,
  "iss":
"https://kubernetes.default.svc.cluster.local",
  "kubernetes.io": {
    "namespace": "default",
    "serviceaccount": {
      "name": "default",
      "uid": "ef07d610-ef21-4cd7-81cf-06a075f40fe2"
    }
  },
  "nbf": 1657617613,
  "sub": "system:serviceaccount:default:default"
}
```

Bound service account tokens for pods


```yaml
volumes:
- name: kube-api-access-cd8n7
  projected:
    defaultMode: 420
    sources:
    - serviceAccountToken:
        expirationSeconds: 3607
        path: token
```

- Obtained directly using the *TokenRequest* API
- Per pod
- In-cluster only audience
- Limited lifetime
- Refresh periodically
- *service-account-extend-token-expiration* extends the expiration period to 1 year

```yaml
  volumeMounts:
  - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
    name: kube-api-access-cd8n7
    readOnly: true
```

Manually created long-lived tokens

- Create a new Secret with a special annotation: *kubernetes.io/service-account.name*
- The control plane automatically generates a token for that ServiceAccount, and stores it into the associated Secret
- Delete a ServiceAccount automatically cleans up the long-lived token from that Secret.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  creationTimestamp: "2023-10-20T21:14:20Z"
  name: test-sa
  namespace: default
  resourceVersion: "463"
  uid: a984eddb-dbf6-4414-816e-97defdf012c0
```

```
Name:         test-secret
Namespace:    default
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: test-sa
              kubernetes.io/service-account.uid: a984eddb-dbf6-4414-816e-97defdf012c0

Type:  kubernetes.io/service-account-token

Data
====
namespace:  7 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IlRXNHBPSTI
```

Parameters and feature gates

- `BoundServiceAccountTokenVolume` : enable or disable the use of projected volume for a pod's service account token. Default in k8s 1.21 (Beta) and GA in k8s .122.
- `service-account-extend-token-expiration` : control whether or not to extend the expirations of projected pod tokens to 1 year (JWT requested by kubelet and issued by TokenRequest API).
- `service-account-max-token-expiration`: control the maximum expiration of a service account token requested through TokenRequest API. Note that the value has no impact on a JWT issued for a pod's projected volume if the auto extension of projected pod token expiration is enabled. 10 min <= expiration <= `service-account-max-token-expiration` <= 2^32 seconds.
- `LegacyServiceAccountTokenNoAutoGeneration` : control if stop creating auto-creating secret for service accounts. GA k8s 1.26. (remove in k8s 1.29)
- `LegacyServiceAccountTokenTracking`: a simple controller in the kube-apiserver to issue a warning, track a legacy token last used in `kubernetes.io/legacy-token-last-used` on the secret at date granularity, and record in a metric. GA k8s 1.28. (remove in k8s 1.30)
- `LegacyServiceAccountTokenCleanUp`: Token Controller starts to remove unused auto-generated secrets (secrets bi-directionally referenced by the service account) and not mounted by pods. When this feature is enabled, delete secrets if it is over a sufficient period of time (one year by default) since last used. The period can be configured by cluster admins. GA k8s 1.30.

Impact on different use cases

Legacy auto-generated secret-based long-lived tokens

- No impact on existing tokens until k8s 1.30+
- Stop auto-generation of secret-based tokens in k8s 1.29.
- Plan the migration to time-bound tokens using *TokenRequest* API
- After k8s 1.30, unused auto-generated long-lived tokens will be removed

Impact on different use cases (cont'd)

Bound service account pod tokens

- Most applications should work without changes with a year expiration in JWT (k8s 1.22)
- The token refresh frequency is based on the token expiration time (extended to 1 year).
- Set *service-account-extend-token-expiration=false* (k8s 1.26), JWT's expiration will become 1 hour
- Tokens have to be reloaded periodically from the disk by applications, to handle expiration
- Most updated client libraries should do it for applications

Impact on different use cases (cont'd)

Time-bound tokens used by external systems

- The expiration time can be specified in a *TokenRequest*, but its maximum value must be less than *service-account-max-token-expiration*
- Clients should periodically refresh their tokens and update their configurations through *TokenRequest API*
- One of the challenges is to update the configuration automatically, e.g., *kubeconfig*

Summary of impact on use cases

	Secret-based	Expiration	Pod-bound volume mount	BoundServiceAccountToken Volume	service-account-extend-token-expiration	service-account-max-token-expiration	LegacyServiceAccountTokenNoAutoGeneration	LegacyServiceAccountTokenTracking	LegacyServiceAccountTokenCleanUp
Auto-generated	Yes	Long lived	Yes/No	No	No	No	Yes	Yes	Yes
Manually-created secret based	Yes	long lived	Yes/No	No	No	No	No	No	No
Projected volume for pod	No	Time-bound	Yes	Yes	Yes	May affect token refresh frequency	No	No	No
Time-bound TokenRequest API (non pod bound)	No	Time-bound	No	No	No	Yes	No	No	No

Tracking and monitoring token use: audit

Annotations in api-server audit log

- *authentication.k8s.io/stale-token*
- *authentication.k8s.io/legacy-token-autogenerated-secret*
- *authentication.k8s.io/legacy-token-manual-secret*

Examples

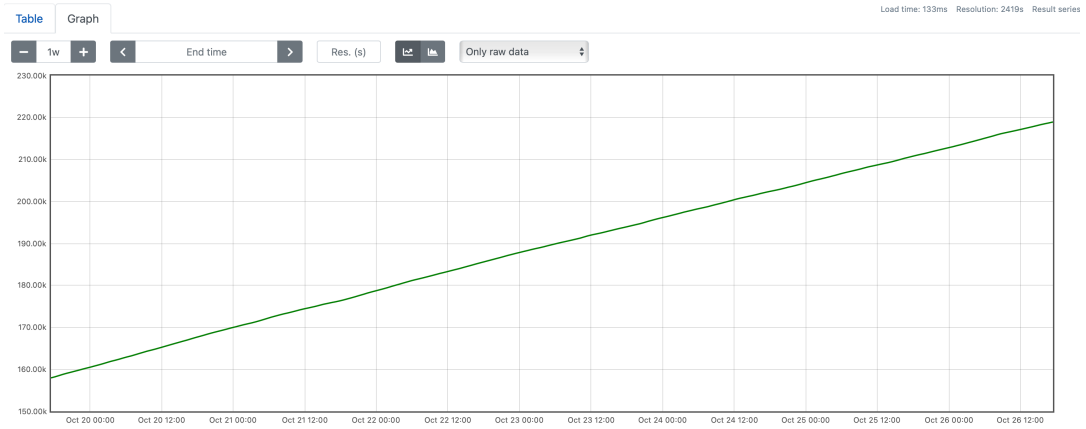
{"authentication.k8s.io/stale-token":"subject: system:serviceaccount:default:default-sa, seconds after warning threshold: 111787", …}

{"authentication.k8s.io/legacy-token":"system:serviceaccount:default:test-sa", "authentication.k8s.io/legacy-token-manual-secret":"test-secret", ...}

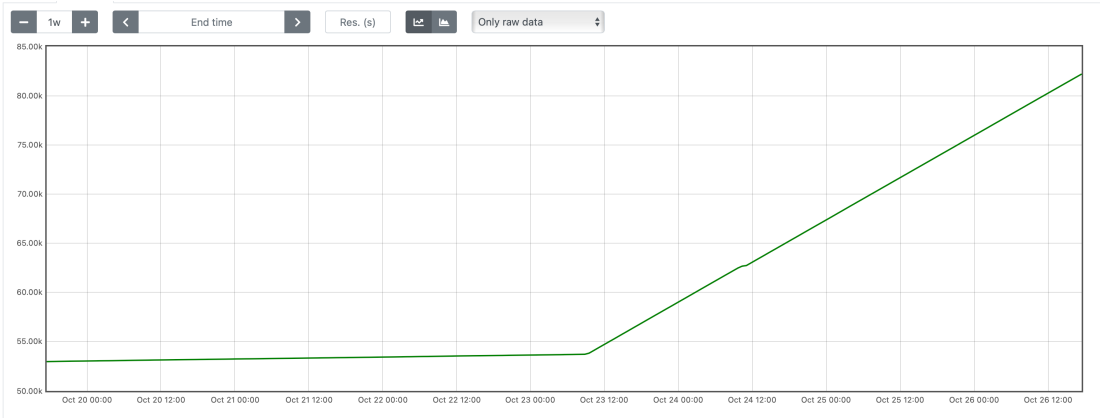
{"authentication.k8s.io/legacy.-token":"system:serviceaccount:default:default","authentication.k8s.io/legacy-token-autogenerated-secret":"default-token-nwq6w", …}

Tracking and monitoring token use: metrics

- *serviceaccount_stale_tokens_total*
- *serviceaccount_legacy_tokens_total*
- *serviceaccount_auto_generated_tokens_total*
- *serviceaccount_manually_created_tokens_total*



serviceaccount_legacy_tokens_total



serviceaccount_stale_tokens_total

ServiceAccount token transition

Use the monitoring and tracking capabilities to track the current use of different service account tokens

Based on the collected data, ask customers to take actions and decide when to make a change (feature gates, parameters)

- Disable auto-generation
- Disable expiration extension
- Enable legacy token cleanup
- Integrate external systems using *TokenRequest* API

A sample plan for token upgrade

k8s 1.24

- Continue the use of a long maximum expiration time: service-account-max-token-expiration=1 year
- Continue auto extension of expiration for bound service acount tokens: service-account-extend-token-expiration=true
- Continue the support of auto-generation of secret-based long lived tokens, but not recommend, LegacyServiceAccountTokenNoAutoGeneration=false

A sample plan for token upgrade (cont'd)

k8s 1.26

- Discontinue auto-generation of long-lived secret-based tokens
- Use a short expiration for time-bound tokens: service-account-max-token-expiration=1 month
- Discontinue auto-generation of secret-based long-lived tokens: LegacyServiceAccountTokenNoAutoGeneration=true

A sample plan for token upgrade (cont'd)

k8s 1.28

- Use a short maximum expiration
- Discontinue auto-extension of token expiration for projected volume tokens: service-account-extend-token-expiration=false
- Use the default short expiration for time-bound token: service-account-max-token-expiration=1 hour
- Enable legacy token tracking: LegacyServiceAccountTokenTracking=true

A sample plan for token upgrade (cont'd)

k8s 1.30+

- Removed unused auto-generated legacy secret-based tokens and complete the migration
- All auto-generated long-lived tokens should be replaced by time-bound tokens
- Auto-generated secret-based token will stop working

Integrating external systems

Lots of external systems have relied upon being able to create long-lived tokens

We have a few different options to support these:

- Switch to OIDC and use an existing identity document to ‘pivot’ into JWTs
- Long-lived credentials that only have permission to request short-lived tokens

Integrating external systems

Pivot from external identity into a JWT from OIDC provider

- Relies on having some existing identity document to pivot from
- Very effective and avoids service account tokens altogether

Integrating external systems

Using long-lived token to pivot to short-lived tokens

- Long lived credential with fewer permissions
- Only has permission to fetch short-lived tokens
- Each credential then gets a unique identifier

Integrating external systems

Credential identifiers (alpha k8s 1.29+)

- Unique identifier for each issued token
- Allows cross-referencing requests to precise token

Summary

- Newer Kubernetes versions use time-bound API tokens via the *TokenRequest* API
- Long-lived auto-generated tokens based on secrets will no longer function, necessitating the adoption of time-bound tokens
- The key challenge is to upgrade tokens seamlessly without disrupting current usage
- Tokens can be configured using parameters and feature gates
- The ability to track token usage is crucial during the token upgrade process

Acknowledgements

KubeCon CloudNativeCon North America 2023

Deep Debroy

Katie Gamanji

Alena Prokharchyk

Jenny Thayer

Adam Dema

Jordan Liggitt

Q&A

Thank you!

Please scan the QR Code above
to leave feedback on this session