KubeCon

North America 2017

Container Identity Working Group Update

Greg Castle (@mrgcastle), GKE/Kubernetes Security, *Google*
Clayton Coleman (@smarterclayton), Architect and Engineer on Kubernetes, OpenShift, *Red Hat*

Today

- Focus: identity for applications
- Describe current state and future plans (10-15min)
- Discussion, Q&A (20 min)

Current state of K8s identity

- **Users** can auth a few ways (e.g. OAuth/OIDC, cert, password)
- But for **applications** only one built-in identity option
- K8s service account
- Not recognized outside the cluster
- JWT isn’t bound to an audience
- JWT lives forever (or until service account deleted)
- Issues scaling (#4808)
- SA tokens == permissions, get all secrets implications

Use Cases

- Secrets: Password in Hashicorp Vault
- Enterprise: Manage ID in LDAP/AD/Proprietary. One src of truth
- Multi-cloud: Google/Amazon/Azure IDs for various services
- Service-Service: Intra + inter cluster
- Container: Monitoring sidecar needs different ID to workload

What we want

- Get ID to apps with ≈0 developer effort
- Provision ID for multiple external systems
- Segmentation: minimum blast radius for compromise
- Limited lifetime, auto-rotation
- Non-exportable where possible (e.g. TPM available)

What is happening in this space?

- SPIFFE and SPIRE (spiffe.io)
  - Application x509 ID and standard naming scheme, API for workloads to access ID, runtime env for attestation, rotation
- Istio (istio.io)
  - Lots of service management features. Identity: SPIFFE-named x509 certs to identify services
- Vault integration (goo.gl/ZuAPtn)
  - Complete, but work underway (next slide, bound SA tokens) to scope K8s SA tokens

What is coming: tokenrequests API

Initial design PR: kubernetes/community/pull/1460

Improvements:

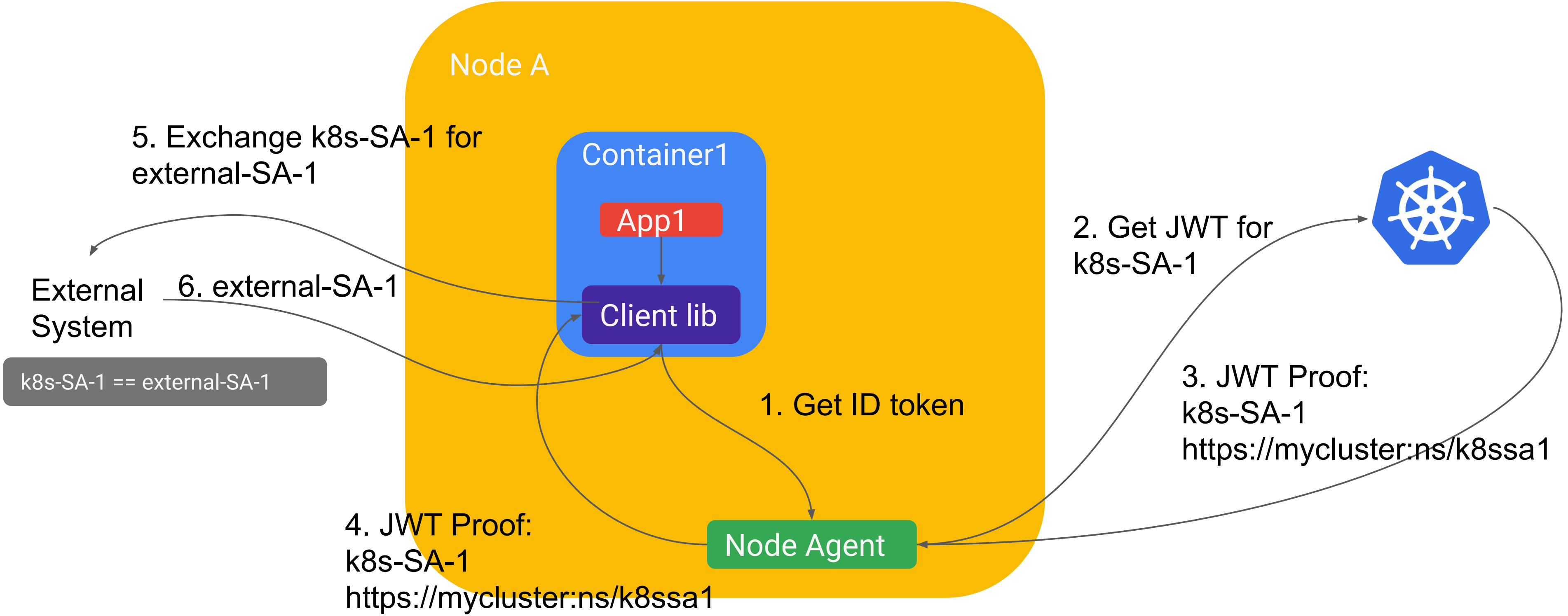
- Audience to address Vault (and similar) use case
- Expiration
- Scalability: not reliant on central DB
- Verification by external systems
- K8s doesn’t have to understand external ID systems

What does that get us?

- Next steps to get “≈0 developer effort”
- Node agent interacts with tokenrequests API
- Delivers creds to workloads with flex volumes
- Client libs use JWT directly; or
- Exchange for different identity (e.g. cloud provider)

- Flex volume pattern useful for other ID provisioning, don’t need to use K8s JWTs at all

Something like this



Node A

Container1

App1

Client lib

Node Agent

1. Get ID token

2. Get JWT for k8s-SA-1

3. JWT Proof:
k8s-SA-1
https://mycluster:ns/k8ssa1

4. JWT Proof:
k8s-SA-1
https://mycluster:ns/k8ssa1

5. Exchange k8s-SA-1 for external-SA-1

6. external-SA-1

External System

k8s-SA-1 == external-SA-1

Future work

- Improve flex volume usability for attaching identities to pods
- Consider identity API type instead of volume
- Support for ACL’ing with RBAC

Join the discussion

- See WG recording (goo.gl/b52mhR) where this system was discussed

- Meet Fridays every 2 weeks: goo.gl/cbq1Ca
- Join mailing list for cal invite

- Now: Group discussion and Q&A

Discussion topics

- Use cases for container identity (more granular than pod): e.g. monitoring sidecar - others?
- What identity systems do people want to integrate?
- How do you use k8s service accounts in applications? Mostly default per namespace or more sophisticated?
- What is your organization src of truth for robot accounts?