Using K8s Audit Logs to Secure Your Cluster.

Mark Stemm, Falco Engineer

About me.

Mark Stemm

- Senior Software Engineer, Sysdig
- Maintainer, Falco

 @mstemm

K8s audit events.

- New in K8s v1.11, updated in K8s 1.13
- Provides chronological set of records documenting changes to cluster
- Each record is a JSON object
- Audit policy controls which events are included in event log
- Log backend controls where events are sent
  - Log file
  - Webhook

K8s audit events.

```
{
    "kind": "Event",
    "timestamp": "2018-10-26T13:00:25Z",
    "stage": "ResponseComplete",
    "verb": "delete",
    "requestURI": "/api/v1/namespaces/foo",
    "user": { "username": "minikube-user" },
    "responseStatus": { "code": 200 },
    "objectRef": { "resource": “namespaces", "namespace": “foo” },
    "level": "Request",
    "auditID": "693f4726-2430-450a-83e1-123c050fde98",
    "annotations": { "authorization.k8s.io/decision": "allow" }
}
```

K8s audit event examples.

- Create/destroy/modify deployment, namespace, pod, etc.
- Attach/exec into pod
- Create service account to access resources
- Create/modify role/cluster role to define access
- Create a role binding to link roles and accounts
- Listing/reading resources: **`kubectl get pods`**, etc.
- Actions by users (minikube) as well as other internal K8s services (kube-scheduler)

K8s audit configuration.

- Audit Policy
  - Controls what events are included
  - Yaml file containing a list of rules.
  - An event must match a rule to be included
- Audit Backend
  - Where audit events go
    - logfile, webhook (>= 1.11)
    - dynamic (>= 1.13)

K8s audit backends.

- Log backend
  - One JSON record per event, per line
  - Controls for location, log rotation (size and/or age)
- Webhook backend
  - Batch of records (JSON Array) per group of events
  - Controls for location (kubeconfig), batching algorithm, throttling, etc.
- Dynamic
  - Like above, but managed like other K8s resources

Enabling K8s audit in 1.11.

- Audit policy and backend config are *static*
  - Command line arguments to kube-apiserver
    - **`–audit-webhook-config-file`**
    - **`–audit-log-path`**
    - **`–audit-policy-file`**
    - etc.
- Must be specified at startup or api server must be restarted to pick up new config
- Can’t directly view config/policy other than examining files/ps output

Enabling K8s audit in 1.13.

- Policy and Webhook specified in AuditSink objects
- Managed like other K8s Resources
- Creating AuditSink objects requires **`cluster-admin`** privileges
- Can create multiple AuditSinks, each with different policy/ destination

Searching audit events (hard way).

- **`jq`**: command-line json parser
  - Reads input json objects, filters/transforms, writes output object

Select example

```
$ echo '{"key": "some-value"}' | jq '.key'
“some-value”
```

Filter example

```
$ echo [{"key": "v1"},{"key": "v2"}]' | jq '.[] | select(.key == "v1")'
{"key": “v1"}
```

Transform example

```
$ echo '{"key": "some-value"}' | jq '"Prop key has value= " + .key'
"Prop key has value=some-value"
```

JQ examples for K8s audit events.

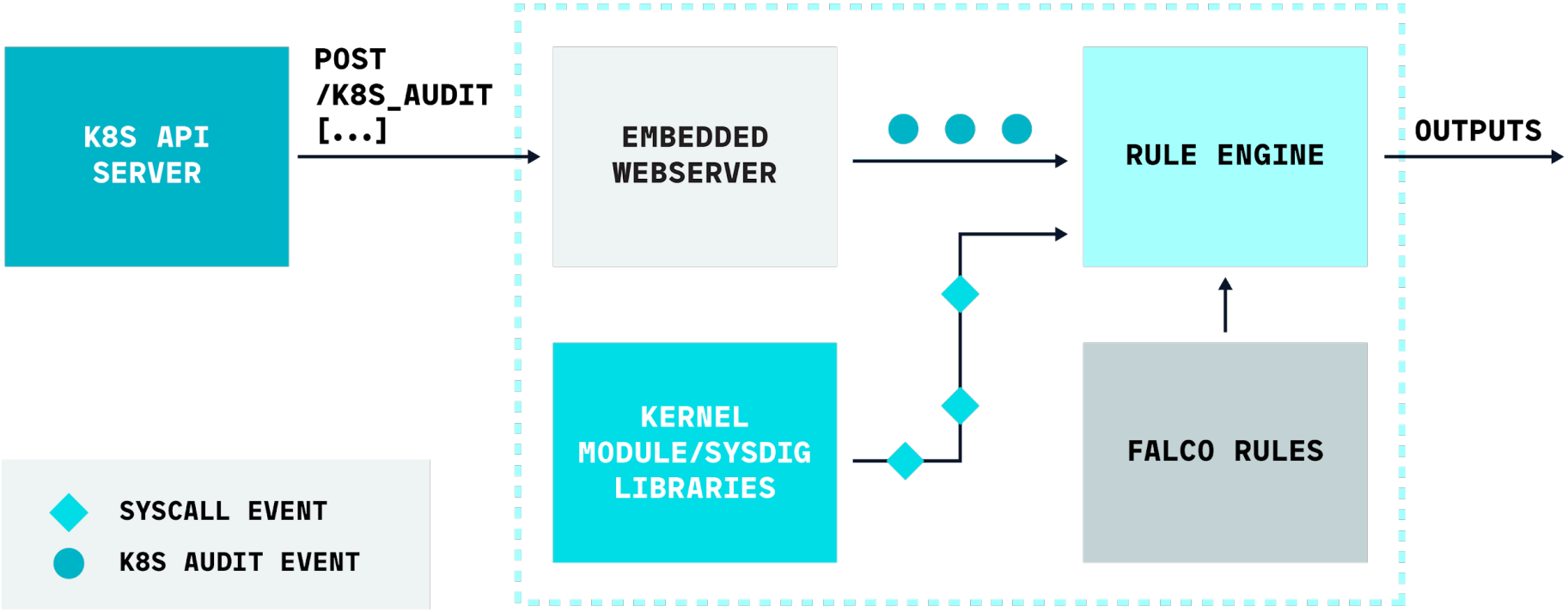
```
# Create Namespace
(select(.verb == "create" and .objectRef.resource=="namespaces") |
    "[" + .stageTimestamp + "] " + "Namespace Created: name=" + .objectRef.name),

# Delete Namespace
(select(.verb == "delete" and .objectRef.resource=="namespaces") |
    "[" + .stageTimestamp + "] " + "Namespace Deleted: name=" + .objectRef.name),

# Create Configmap containing password or AWS Private Key
(select(.verb == "create" and .objectRef.resource=="configmaps" and
        (.requestObject.data | tostring | contains("aws_access_key_id"))) |
    "[" + .stageTimestamp + "] " + "Configmap Created: name=" + .objectRef.name +
                                   " Configmap= "+ (.requestObject.data | tostring)),
```

- Full List: https://gist.github.com/mstemm, “JQ Filters for K8s Audit Events"

Searching audit events (easy way).

- Falco (CNCF Project) supports K8s audit events!
  - Embedded web server to receive events
  - Write rules in yaml with filter conditions and output expressions
  - ~30 built-in rules to detect:
    - Suspicious activity
    - Change-related activity
    - All activity (very verbose)

Falco + K8s audit events.



K8s audit rule example.

```
- macro: contains_private_credentials
  condition: >
    (ka.req.configmap.obj contains "aws_access_key_id" or
     ka.req.configmap.obj contains "aws_s3_access_key_id" or
     ka.req.configmap.obj contains “password")

- macro: configmap
  condition: ka.target.resource=configmaps

- macro: modify
  condition: (ka.verb in (create,update,patch))

- rule: Create/Modify Configmap With Private Credentials
  desc: Detect creating/modifying a configmap containing a private credential
    (aws key, password, etc.)
  condition: configmap and modify and contains_private_credentials
  output: K8s configmap with private credential (user=%ka.user.name
          verb=%ka.verb name=%ka.req.configmap.name
          configmap=%ka.req.configmap.name config=%ka.req.configmap.obj)
  priority: WARNING
  source: k8s_audit
  tags: [k8s]
```

K8s audit use cases.

- Watch changes to your cluster
- Falco **`k8s_audit_rules.yaml`** rule names
  - **`K8s Deployment {Created,Deleted}`**
  - **`K8s Service {Created,Deleted}`**
  - **`K8s Namespace {Created,Deleted}`**
  - **`K8s ConfigMap {Created,Deleted}`**
  - **`K8s Serviceaccount {Created,Deleted}`**
  - **`K8s Role/Clusterrole {Created,Deleted}`**
  - **`K8s Role/Clusterrolebinding {Created,Deleted}`**

K8s audit use cases.

- Enumerate what's allowed, look for exceptions
  - **`Disallowed K8s User`**
  - **`Create Disallowed Pod`**
  - **`Create Disallowed Namespace`**

K8s audit use cases

- Limit what pods can access
  - **`Create Privileged Pod`**:
    - Look for pod create where **`"securityContext":{"privileged":true}`**
  - **`Create Sensitive Mount Pod`**:
    - Look for pod create where pod mounts sensitive paths from host filesystem
  - **`Create HostNetwork Pod`**:
    - Look for pod create where pod uses host network namespace
  - **`Pod Created in Kube Namespace`**:
    - Look for pod create in **`kube-system`** or **`kube-public`** namespaces
- Limit access to pods
  - **`Attach/Exec Pod`**:
    - Look for any **`kubectl exec/attach pod …`**

K8s audit use cases

- Protect users/accounts
  - **`Service Account Created in Kube Namespace`**
    - Creating service account in **`kube-system/kube-public`** namespaces
  - **`System ClusterRole Modified/Deleted`**
    - Any delete/modify to roles starting with **`“system:”`**
  - **`Attach to cluster-admin Role`**
    - Creating a role binding linked to **`cluster-admin`** role
  - **`ClusterRole With Wildcard Created`**
    - Creating a role that does not explicitly enumerate resources or verbs (e.g. **`“resources":["*"]`**)
  - **`ClusterRole With Write Privileges Created`**
    - Creating a role that can perform deletes/writes
  - **`ClusterRole With Pod Exec Created`**
    - Creating a role that can exec/attach to pods

K8s audit use cases

- Other
  - **`Create/Modify Configmap With Private Credentials`**
    - Create configmap containing “password”, “aws_access_key”, etc.
  - **`Anonymous Request Allowed`**
    - Any request by **`system:anonymous`** that was allowed
  - **`Create NodePort Service`**
    - Create a service with a NodePort service type

Demo.

Join the community.

Website

- https://falco.org

Public Slack

- http://slack.sysdig.com/
- https://sysdig.slack.com/messages/falco

Blog

- https://sysdig.com/blog/tag/falco/

Github

- https://github.com/falcosecurity/falco

Documentation

- https://github.com/falcosecurity/falco/wiki

Docker Hub

- https://hub.docker.com/r/falcosecurity/falco/

Thank You!

Backup Slides

K8s audit events.

- Who
  - **`user`**, **`groups`**, **`username`**, **`sourceIPs`**
- What
  - **`verb`**, **`objectRef`**
- When
  - **`stageTimestamp`**
- Result
  - **`authorization.k8s.io/{decision,reason}`**
  - **`responseStatus`**

K8s audit policy.

```yaml
apiVersion: audit.k8s.io/v1 # This is required.
kind: Policy
# Don't generate audit events for all requests in RequestReceived stage.
omitStages:
  - "RequestReceived"
rules:
  # Log pod changes at RequestResponse level
  - level: RequestResponse
    resources:
    - group: ""
      # Resource "pods" doesn't match requests to any subresource of pods,
      # which is consistent with the RBAC policy.
      resources: ["pods"]
  # Log "pods/log", "pods/status" at Metadata level
  - level: Metadata
    resources:
    - group: ""
      resources: ["pods/log", "pods/status"]
```

K8s audit sink.

```yaml
apiVersion: auditregistration.k8s.io/v1alpha1
kind: AuditSink
metadata:
  name: mysink
spec:
  policy:
    level: Metadata
    stages:
    - ResponseComplete
  webhook:
    throttle:
      qps: 10
      burst: 15
    clientConfig:
      url: "https://audit.app"
```