Protecting cluster metadata

Overview

GKE uses instance metadata (https://cloud.google.com/compute/docs/storing-retrieving-metadata) to configure node VMs, but some of this metadata is potentially sensitive and should be protected from workloads running on the cluster.

Before you begin

To prepare for this task, perform the following steps:

- Ensure that you have enabled the Google Kubernetes Engine API.

  ENABLE GOOGLE KUBERNETES ENGINE API (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APIS/LIBRA

- Ensure that you have installed the Cloud SDK (https://cloud.google.com/sdk/downloads).
- Set your default project ID (https://support.google.com/cloud/answer/6158840):

```
gcloud config set project [PROJECT_ID]
```

- If you are working with zonal clusters, set your default compute zone (https://cloud.google.com/compute/docs/zones#available):

```
gcloud config set compute/zone [COMPUTE_ZONE]
```

- If you are working with regional clusters, set your default compute region (https://cloud.google.com/compute/docs/zones#available):

```
gcloud config set compute/region [COMPUTE_REGION]
```

- Update `gcloud` to the latest version:

```
gcloud components update
```

★ **Note:** You can override these default settings in `gcloud` commands using the `--project`, `--zone`, and `--region` operational flags.

Configure node service account

Because each node's service account credentials will continue to be exposed to workloads, you should ensure that you have configured a service account (https://cloud.google.com/compute/docs/access/create-enable-service-accounts-for-instances#createanewserviceaccount) with the minimal permissions it needs. Then, attach this service account to your nodes, so that an attacker cannot circumvent GKE's metadata protections by using the Compute Engine API to access the node instances directly.

Do not use a service account that has `compute.instances.get` permission, the Compute Instance Admin role, or other similar permissions, as they allow potential attackers to obtain instance metadata using the Compute Engine API. The best practice is to restrict the permissions of a node VM by using service account permissions, *not access scopes*. For more information, see Compute Engine's service accounts (https://cloud.google.com/compute/docs/access/service-accounts) documentation.

If you don't have a node service account, you can create one using the following commands:

```
export NODE_SA_NAME=gke-node-sa
gcloud iam service-accounts create $NODE_SA_NAME \
  --display-name "Node Service Account"
export NODE_SA_EMAIL=$(gcloud iam service-accounts list --format='value(email)' \
  --filter='displayName:Node Service Account')
```

To configure your service account with the necessary roles and permissions, run the following commands. PROJECT is your project ID (https://support.google.com/cloud/answer/6158840?hl=en):

```
export PROJECT=$(gcloud config get-value project)

gcloud projects add-iam-policy-binding $PROJECT \
  --member serviceAccount:$NODE_SA_EMAIL \
  --role roles/monitoring.metricWriter
gcloud projects add-iam-policy-binding $PROJECT \
```

```
  --member serviceAccount:$NODE_SA_EMAIL \
  --role roles/monitoring.viewer
gcloud projects add-iam-policy-binding $PROJECT \
  --member serviceAccount:$NODE_SA_EMAIL \
  --role roles/logging.logWriter
```

Additionally, if your cluster pulls private images from Container Registry (https://cloud.google.com/container-registry/docs/pushing-and-pulling), add the `storage.objectViewer` role:

```
gcloud projects add-iam-policy-binding $PROJECT \
  --member serviceAccount:$NODE_SA_EMAIL \
  --role roles/storage.objectViewer
```

Disabling and transitioning from legacy metadata APIs

Compute Engine v1beta1 and v0.1 metadata server endpoints are deprecated and scheduled for shutdown. Ensure that you update all requests to use the v1 endpoint.

The Compute Engine instance metadata (https://cloud.google.com/compute/docs/storing-retrieving-metadata) server exposes legacy v0.1 and v1beta1 endpoints, which do not enforce metadata query headers (https://cloud.google.com/compute/docs/storing-retrieving-metadata#querying). This is a feature in the v1 APIs that makes it more difficult for a potential attacker to retrieve instance metadata. Unless specifically required, we recommend you disable these legacy APIs.

Note: Starting in GKE 1.12, legacy Compute Engine metadata endpoints are disabled by default on new clusters.

Note: You can currently disable legacy metadata APIs *only when creating a new cluster, or when adding a new node pool to an existing cluster*.

The follow section explains how to:

- Identify which nodes are accessing the deprecated endpoints. If any nodes are using the deprecated endpoints, you need to migrate these nodes.

- Create a new cluster or node pool with the legacy metadata server endpoints disabled for these identified nodes.

Identifying nodes using the legacy metadata server endpoints

You can use the check-legacy-endpoint-access tool (https://github.com/GoogleCloudPlatform/k8s-node-tools/tree/master/check-legacy-endpoint-access) to determine which of your Kubernetes Engine nodes is using the legacy metadata server endpoints. When applied in your cluster, this tool logs all requests to the v0.1 and v1beta1 endpoints by your nodes every 5 minutes. This tool can also be used for identifying, debugging, and verifying the usage of the legacy endpoints in Kubernetes Engine.

To set up the check-legacy-endpoint-access tool, complete the following steps:

1. In each of your clusters, run the following command:

```
kubectl apply -f \
https://raw.githubusercontent.com/GoogleCloudPlatform\
/k8s-node-tools/master/check-legacy-endpoint-access/check-legacy-endpoint-acces
```

2. Query the logs collected with legacy endpoint usage information. To query the logs, in each cluster, run the following command:

```
kubectl -n kube-system logs -l \
app=check-legacy-endpoint-access | grep "access count"
```

You can also view the logs collected in Stackdriver Logging.

1. Go to the **Stackdriver Logging > Logs** (Logs Viewer) page in the Cloud Console:

   GO TO THE LOGS VIEWER PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/LOGS/VIEWER)

2. Apply the following filter:

```
resource.type="container"
resource.labels.namespace_id="kube-system"
logName:"/check-legacy-endpoint-access"
```

   GO TO THE FILTERED VIEW (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/LOGS/VIEWER?FILTERS=TEX

3. After you have identified the nodes, you need to identify the processes that are using these endpoints. For instructions to identify these processes, see Identifying the processes (https://cloud.google.com/compute/docs/migrating-to-v1-metadata-server#find-process).
4. Migrate these processes to use the v1 metadata server endpoint. For instructions to migrate your Compute Engine nodes and differences in the endpoints, see Migrating to v1 metadata server endpoint (https://cloud.google.com/compute/docs/migrating-to-v1-metadata-server).
5. Remove the check-legacy-endpoint-access daemonset.

```
kubectl delete check-legacy-endpoint-access
```

Creating a new node pool with legacy metadata APIs disabled

After you create a service account (#service-account), you can create a new node pool (or a default node pool in a new cluster) with legacy metadata APIs disabled by using the `gcloud` command-line tool.

To create a new node pool with legacy metadata APIs disabled, use the `--metadata disable-legacy-endpoints=true` flag. For example:

```
gcloud container node-pools create [POOL_NAME] \
  --service-account=$NODE_SA_EMAIL \
  --metadata disable-legacy-endpoints=true
```

A new cluster can be created with legacy metadata APIs disabled in *the default node pool* using the same flag. For example:

```
gcloud container clusters create [CLUSTER_NAME] \
  --service-account=$NODE_SA_EMAIL \
  --metadata disable-legacy-endpoints=true
```

Updating an existing cluster to disable legacy metadata APIs

After creating a new node pool with legacy metadata APIs disabled, you can update an existing cluster to use it by following the node pool migration guide (https://cloud.google.com/kubernetes-engine/docs/tutorials/migrating-node-pool).

Verifying that legacy metadata APIs are disabled

When the legacy instance metadata APIs are disabled, requests to `/0.1/` and `/v1beta1/` metadata server endpoints will return `403 Forbidden`.

To verify that the legacy metadata APIs have been disabled, you can run a `curl` command from within a Pod (https://kubernetes.io/docs/tasks/debug-application-cluster/get-shell-running-container):

```
root@pod-name# curl -H 'Metadata-Flavor: Google' \
'http://metadata.google.internal/computeMetadata/v1/instance/attributes/disable-lega
true
root@pod-name# curl 'http://metadata.google.internal/computeMetadata/v1beta1/instanc
... Error 403 (Forbidden) ... Legacy metadata endpoint accessed: /computeMetadata/v1
```

Metadata concealment

Warning: Workload Identity (https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity) replaces the need to use metadata concealment and the two approaches are incompatible. We recommend that you use Workload Identity instead of metadata concealment. Metadata concealment is scheduled to be deprecated and removed in the future.

Beta

This product or feature is in a pre-release state and might change or have limited support. For more information, see the product launch stages (https://cloud.google.com/products/#product-launch-stages).

GKE's **metadata concealment** protects some potentially sensitive system metadata from user workloads running on your cluster.

In Kubernetes v1.9.3 and higher, you can enable **metadata concealment** to prevent user Pods (https://cloud.google.com/kubernetes-engine/docs/concepts/pod) from accessing certain VM metadata for your cluster's nodes (https://cloud.google.com/kubernetes-engine/docs/concepts/cluster-architecture#nodes), such as Kubelet credentials and VM instance information. Specifically, metadata concealment protects access to `kube-env` (which contains Kubelet credentials) and the VM's instance identity token (https://cloud.google.com/compute/docs/instances/verifying-instance-identity).

Metadata concealment firewalls traffic from user Pods (Pods *not* running on `HostNetwork`) to the cluster metadata server, only allowing safe queries. The firewall prevents user Pods from using Kubelet credentials for privilege escalation attacks, or from using VM identity for instance escalation attacks.

Note: You can currently enable metadata concealment *only when creating a new cluster, or when adding a new node pool to an existing cluster*.

Limitations

- Metadata concealment only protects access to `kube-env` and the node's instance identity token (https://cloud.google.com/compute/docs/instances/verifying-instance-identity).
- Metadata concealment does not restrict access to the node's service account (https://cloud.google.com/compute/docs/access/service-accounts).
- Metadata concealment does not restrict access to other related instance metadata.
- Metadata concealment does not restrict access to other legacy metadata APIs.

Creating a new cluster or node pool with metadata concealment

After creating a service account (#service-account), you can create a new cluster with metadata concealment enabled by using the `gcloud` command-line tool.

To create a cluster with metadata concealment enabled, run the following command in your shell or terminal window:

```
gcloud beta container clusters create [CLUSTER_NAME] \
  --workload-metadata-from-node=SECURE \
  --service-account=$NODE_SA_EMAIL \
  --metadata disable-legacy-endpoints=true \
  [additional parameters and flags omitted]
```

where:

- ***[CLUSTER_NAME]*** is the name of the cluster to be created.
- `--workload-metadata-from-node` is set to `SECURE`; setting the flag to `EXPOSED` or `UNSPECIFIED` disables metadata concealment.

Verifying identity token metadata concealed from cluster's workload

When you conceal metadata, it should not be possible to request a signature (https://cloud.google.com/compute/docs/instances/verifying-instance-identity#request_signature) via the node's instance identity token (https://cloud.google.com/compute/docs/instances/verifying-instance-identity). To verify that requests explicitly inform users of concealed metadata, you can run a `curl` command from within a Pod (https://kubernetes.io/docs/tasks/debug-application-cluster/get-shell-running-container):

```
root@pod-name# curl -H "Metadata-Flavor: Google" \
'http://metadata/computeMetadata/v1/instance/service-accounts/default/identity?audie
This metadata endpoint is concealed.
```


Last updated December 4, 2019.