Exercise: Kubernetes Pod Security Policy Defense

Steps

1. Let's try another defense on the cluster takeover scenario: pod security policies. Now we want to stop any non-admin account from staging the attack pod. Our other focus will be to make sure that any pod launched has to have an AppArmor profile.

2. SSH into the Kubernetes control-plane node, using the `bustakube` user's password `bustakube`:

   ```
   ssh bustakube@bustakube-controlplane
   bustakube
   ```

3. `sudo` to `root`:

   ```
   sudo su -
   ```

4. Let's remove the RBAC controls we had added at the end of the Own the Nodes exercise.

   ```
   kubectl delete rolebinding get-only-on-pods-frontend-binding
   kubectl delete rolebinding get-only-on-pods-redis-binding
   cd /usr/share/bustakube/Scenario1-OwnTheNodes/
   kubectl apply -f Namespace-Default/binding-get-list-exec-pods-to-frontend.yaml
   kubectl apply -f Namespace-Default/binding-full-rw-and-exec-on-pods-to-redis.yaml
   ```

5. We'll apply a pod security policy (PSP) that will block any `hostPath` volumes from being mounted. Take a look:

   ```
   cd Defense/PodSecurityPolicies
   more psp-30-root-allowed-no-apparmor-required.yaml
   ```

6. Now, apply the pod security policy you just reviewed:

   ```
   kubectl apply -f psp-30-root-allowed-no-apparmor-required.yaml
   ```

7. We need a role, a list of actions that can be performed by any bound service accounts:

   ```
   kubectl apply -f role-cluster-use-psp-30-root-allowed-no-apparmor-required.yaml
   ```

8. We'll need a binding that binds all authenticated users to this role:

```
kubectl apply -f binding-cluster-all-to-psp-30-root-allowed-no-apparmor-required.yaml
```

9. Before we go on, we'll need to activate the PodSecurityPolicy controller.

   ```
   /usr/local/bin/toggle-psp-controller.sh activate
   ```

10. Now, let's see that the `redis-master` pod's service account can't stage such a pod. First, copy `kubectl` and the attack pod definition into the `redis-master` pod:

    ```
    pod=`kubectl get pods | grep redis-master | awk '{print $1}'`
    kubectl cp /usr/bin/kubectl $pod:/usr/bin
    kubectl cp /usr/share/bustakube/Scenario1-OwnTheNodes/Attack/attack-pod.yaml $pod:/tmp
    ```

11. Now, exec into the `redis-master` pod. to see whether it can stage a hostPath-mounting pod:

    ```
    kubectl exec -it $pod -- /bin/bash
    export PS1="\u@\h # "
    kubectl apply -f /tmp/attack-pod.yaml
    ```

12. For extra credit, see if you can modify the pod security policy to allow the `hostPath` pod mounting, but to force an AppArmor profile that won't permit the pod to read the host's `/etc/shadow` file.

    ```
    echo On my own here
    ```