Kubernetes Security
Agenda

1. Introduction to containers and Kubernetes
2. Kubernetes Attack Surface
3. Securing Kubernetes Clusters
4. Tools for Automated Assessments
5. Conclusion and Road Ahead
Introduction to containers

- Containers are a solution to the problem of how to get software to run reliably when moved from one computing environment to another.

- Examples of container runtimes: Docker, CoreOS rkt, Apache Mesos, LXC etc.
Introduction to Docker

What is Docker?
• Docker is a tool for OS virtualization - Containers
• Docker is a tool to create, deploy, and run applications by using containers.

Core concepts for beginners
• Images
• Containers
• Docker Daemon
• Docker CLI
• Docker Registry
Container Orchestration

- Containers in enterprise environments are much more complex
- When more and more containers are introduced, managing them gets harder
- A container orchestration tool is needed to manage the containers
- Docker Swarm and Kubernetes are two popular container orchestration tools
Kubernetes Architecture

Cluster

MASTER

WORKER 1

WORKER n

NODE

POD

CONTAINER
Sample Kubernetes Cluster

- MASTER
  - API SERVER
  - SCHEDULER
  - CONTROLLER MANAGER
  - ETCD

- WORKER 1
  - KUBELET
  - KUBE PROXY
  - SERVICE (WEBAPP)
  - ATTACKER

- NODE
- POD
- CONTAINER
- SERVICE
# Kubernetes Attack Surface

<table>
<thead>
<tr>
<th>Initial Access</th>
<th>Execution</th>
<th>Persistence</th>
<th>Privilege Escalation</th>
<th>Defense Evasion</th>
<th>Credential Access</th>
<th>Discovery</th>
<th>Lateral Movement</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using Cloud credentials</td>
<td>Exec into container</td>
<td>Backdoor container</td>
<td>Privileged container</td>
<td>Clear container logs</td>
<td>List K8S secrets</td>
<td>Access the K8S API server</td>
<td>Access cloud resources</td>
<td>Data Destruction</td>
</tr>
<tr>
<td>Compromised images in registry</td>
<td>bash/cmd inside container</td>
<td>Writable hostPath mount</td>
<td>Cluster-admin binding</td>
<td>Delete K8S events</td>
<td>Mount service principal</td>
<td>Access Kubelet API</td>
<td>Container service account</td>
<td>Resource Hijacking</td>
</tr>
<tr>
<td>Kubeconfig file</td>
<td>New container</td>
<td>Kubernetes CronJob</td>
<td>hostPath mount</td>
<td>Pod / container name similarity</td>
<td>Access container service account</td>
<td>Network mapping</td>
<td>Cluster internal networking</td>
<td>Denial of service</td>
</tr>
<tr>
<td>Application vulnerability</td>
<td>Application exploit (RCE)</td>
<td>Access cloud resources</td>
<td>Connect from Proxy server</td>
<td>Applications credentials in configuration files</td>
<td>Access Kubernetes dashboard</td>
<td>Applications credentials in configuration files</td>
<td>Instance Metadata API</td>
<td>Writable volume mounts on the host</td>
</tr>
<tr>
<td>Exposed Dashboard</td>
<td>SSH server running inside container</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Access Kubernetes dashboard</td>
<td>Access tiller endpoint</td>
</tr>
</tbody>
</table>

- **Initial Access**: Various methods to gain initial access to the Kubernetes environment.
- **Execution**: Actions taken once initial access is achieved.
- **Persistence**: Maintaining access and establishing a foothold.
- **Privilege Escalation**: Elevating privileges to gain more control.
- **Defense Evasion**: Techniques to evade detection or blocklist.
- **Credential Access**: Gaining access to sensitive information.
- **Discovery**: Identifying targets or vulnerabilities.
- **Lateral Movement**: Moving laterally within the environment.
- **Impact**: Consequences of successful attacks.
Attack Demo
1. Application Security

- Application vulnerabilities are one of the common entry points
- Easiest way to gain initial access to a cluster
- Ensure that applications being deployed into the cluster are well secured
2. Network Exposure

• Network exposure of all security sensitive services must be restricted

- API Server
- Kubelet API
- Kubernetes Dashboard
- etcd

  Appropriate authentication controls should be enforced
3. Service Account Privileges

- Care must be taken when creating service accounts and binding them with roles
- Service accounts for the namespace are injected into the pod
- Granting unnecessary privileges is too risky
- Apply principle of least privilege (RBAC)
4. Use of Admission Controllers

- An additional layer of Access Controls
- Comes with plugins that govern and enforce how the cluster is used.
- Act as a gatekeeper that intercept API requests and may change the request object or deny the request altogether, but after the request is authenticated and authorized
- Examples:
  - AlwaysPullImages – Blocks if you spin up a container without pulling an image from registry
  - PodSecurityPolicy – We will discuss later
5. Vulnerable Container Images

- Images used to spin up containers can have vulnerabilities (Base images/Custom code)
- Scan container images for security vulnerabilities
- Several automated tools are available for scanning Docker images
- Examples – Trivy, Clair
6. Harden the containers

- Even if images are safe, the way containers are started can cause trouble.

- A container spun up using --privileged flag can give an attacker full access to the underlying host

- If hosts volume is mounted on to the container, it will be accessible from within the container

- Containers are started with root user accounts by default.

- Remove unnecessary capabilities from the containers when running them.

- Automated tools are available to spot missing best practices.
7. Use of Security Context

- A feature that allows us to enforce restrictions when creating a pod or deployment
- Run Pods with non root containers
- Run containers with read only file system
- Drop dangerous container capabilities
- Enforce apparmor profiles – granular control on files
- Enforce seccomp profiles – limit system calls
8. Use of Pod Security Policies

- A Pod Security Policy is a cluster level resource that controls the security sensitive aspects of the pod-specification.

- The PodSecurityPolicy objects define a set of conditions that a pod must run with in order to be accepted into the system.

- Examples:
  - Privileged Containers
  - Use of host file system
  - Read only root file system
  - Restricting escalation to root privileges
9. Secrets Management

- Applications running in the containers may need access to secrets
- Secrets should be accessible to pods
- Secrets are usually passed using environment variables and mountable volumes
- Kubernetes offers etcd to store the secrets
- Not encrypted by default
- Recommended to use a vault for secrets:
  - Encrypt data at rest
  - Token to access the secrets
  - Token rotation
10. Network Policies

- In our demo, we got a reverse shell using remote code execution vulnerability
- If we have a network rule that blocks outbound connection, reverse shell payload won't work
- Use a networking plugin that supports network policies
- Allows us to enforce ingress and egress rules
Automated Assessments

- Trivy is a tool that can be used for scanning Docker images for known vulnerabilities.
- Kube Bench can be used for assessing the cluster (host, images and containers).
- Kube sec can be used to scan the YAML files.
Conclusion

Key takeaways:
• Kubernetes is a great tool for container orchestration, but with great features come great security issues.
• Kubernetes is prone to several misconfigurations and we must be aware of the exact implications of features being used – A simple misconfiguration can cause serious damage.
• Several tools and techniques are available for hardening the clusters – use them.

Road ahead
• New vulnerabilities are discovered every now and then – keep an eye.
• Get familiar with security concepts of other container orchestration platforms such as Docker Swarm
Additional Resources

- https://kubernetes.io/docs/concepts/security/
- https://kubernetes.io/docs/tasks/administer-cluster/securing-a-cluster/
- https://www.microsoft.com/security/blog/2020/04/02/attack-matrix-kubernetes/
- https://securekubernetes.com/scenario_1_attack/
- https://github.com/ksoclabs/awesome-kubernetes-security
THANK YOU