Passwd Kubernetes Lab – Demo Guide

This guide explains each file in this repo, then walks you through a short live demo you can present to others. The lab shows insecure file permissions and container security context pitfalls using three Pods.

Repository files overview

- **Dockerfile**: Ubuntu-based image used by `vuln-pod.yaml` and `vuln-pod-no-capabilities.yaml` via the tag `passwd-lab:latest`.
  - Installs basic tools (`sudo`, `passwd`, `python3`, etc.).
  - Creates a user `attacker` with password `attacker`.
  - Deliberately sets dangerous permissions (`chmod 666 /etc/passwd` and `/etc/shadow`).
  - Copies `passwd.py` to `/home/attacker/`.
- **Dockerfile-alpine**: Alpine-based image used by `vuln-pod-alpine.yaml` via the tag `passwd-lab-alpine:latest`.
  - Installs `bash`, `python3`, `pip`, `shadow`, `py3-cryptography`.
  - Creates the same `attacker` user and sets world-writable permissions for `/etc/passwd` and `/etc/shadow`.
  - Copies `passwd_alpine.py`.
- **passwd.py**: Demo helper script for the Ubuntu image. Shows how weak file permissions enable unauthorized modifications of `/etc/passwd`//`etc/shadow` and related insecure behaviors.
- **passwd_alpine.py**: Same concept tailored for Alpine (BusyBox/Shadow differences). Used inside the Alpine container.
- **passwd_external.py**: Optional helper demonstrating interactions outside the container image (e.g., showing how external logic might attempt to read/modify passwd-like content). Not used directly by the Pods.
- **vuln-pod.yaml**: Pod manifest for Ubuntu-based vulnerable container with `allowPrivilegeEscalation: true`.
  - Uses image `passwd-lab:latest`.
  - `imagePullPolicy: Never` to use the locally-built image in Minikube.
- **vuln-pod-no-capabilities.yaml**: Ubuntu-based container with stricter `securityContext` (drops all Linux capabilities and sets `allowPrivilegeEscalation: false`).
  - Uses image `passwd-lab:latest`.
  - Lets you contrast behavior with and without capabilities.
- **vuln-pod-alpine.yaml**: Alpine-based vulnerable container.

  - Uses image `passwd-lab-alpine:latest`.
  - Mirrors the insecure file permissions scenario on Alpine.

- **setup_and_run.sh**: Convenience script to build images and run the demo end-to-end (optional; verify contents before use).
- **test_cases.py** and **TEST_CASES_README.md**: Sample test scaffolding and documentation to validate expected behaviors or reproduce scenarios.
- **README.md**: Project overview and general notes.

Prerequisites

- Minikube running locally.
- kubectl configured to talk to the Minikube cluster.
- Docker CLI.

Build images (inside Minikube Docker)

Use the Minikube Docker daemon so Pods with `imagePullPolicy: Never` can find images:

```
eval $(minikube docker-env)
docker build -t passwd-lab:latest -f Dockerfile .
docker build -t passwd-lab-alpine:latest -f Dockerfile-alpine .
```

Deploy the Pods

```
kubectl apply -f vuln-pod.yaml -f vuln-pod-alpine.yaml -f vuln-pod-no-capabilities.yaml
kubectl get pods -n passwd-lab
```

Expect three Pods: `vuln-passwd-pod`, `vuln-passwd-pod-alpine`, and `vuln-passwd-pod-no-caps`.

Live demo flow (5–8 minutes)

1) Show all Pods running

```
kubectl get pods -n passwd-lab
```

2) Exec into Ubuntu vulnerable Pod and demonstrate insecure permissions

```
kubectl exec -n passwd-lab -it vuln-passwd-pod -- /bin/bash
id
ls -l /etc/passwd /etc/shadow
python3 /home/attacker/passwd.py
```

Key talking points: - Files are world-writable (mode `666`), allowing non-root user `attacker` to tamper with system identity files. - This simulates a misconfiguration where container images ship with dangerous permissions.

3) Exec into Alpine vulnerable Pod and repeat

```
kubectl exec -n passwd-lab -it vuln-passwd-pod-alpine -- /bin/bash
id
ls -l /etc/passwd /etc/shadow
python3 /home/attacker/passwd_alpine.py
```

Talking points: - Same class of issue on a different base image. - Highlight differences in Alpine's tools (BusyBox vs GNU) if relevant.

4) Compare with the no-capabilities Pod

```
kubectl exec -n passwd-lab -it vuln-passwd-pod-no-caps -- /bin/bash
id
ls -l /etc/passwd /etc/shadow
```

Talking points: - `allowPrivilegeEscalation: false` and dropped capabilities reduce the blast radius of certain attacks. - Even with bad file modes, removing capabilities often prevents escalation via privileged operations (demonstrate any differences your script surfaces, or attempt typical escalation commands that fail here).

5) Wrap-up

- Bad image defaults (world-writable critical files) are dangerous.
- SecurityContext controls (no privilege escalation, drop caps) help—but do not excuse insecure base images.
- Always scan images, lock down permissions, and apply least privilege.

Clean up

```
kubectl delete -f vuln-pod.yaml -f vuln-pod-alpine.yaml -f vuln-pod-no-capabilities.yaml
```

Troubleshooting

- ErrImageNeverPull
  - Ensure you built images into Minikube's Docker: `eval $(minikube docker-env)` before `docker build`.
- Pod stuck ContainerCreating
  - `kubectl describe pod <name> -n passwd-lab` to see events (volume mounts, permissions, or image issues).
- Shell on Alpine
  - Container uses bash (installed in Dockerfile-alpine). If `/bin/bash` fails in your env, try `/bin/sh`.

Presenting tips

- Keep the focus on: insecure image -> easy tampering; safer securityContext -> fewer escalation paths.
- Prepare a few commands in a text file to paste quickly.

- Time-box the demo; aim for one clear win per Pod.