Kubernetes static code analysis

Unique rules to find Security Hotspots in your KUBERNETES code

All rules 7 | Security Hotspot 6 | Code Smell 1

Tags | Search by name...

Setting capabilities is security-sensitive

Analyze your code

Security Hotspot   Major   cwe

Setting capabilities can lead to privilege escalation and container escapes.

Linux capabilities allow you to assign narrow slices of `root`'s permissions to processes. A thread with capabilities bypasses the normal kernel security checks to execute high-privilege actions such as mounting a device to a directory, without requiring additional root privileges.

In a container, capabilities might allow to access resources from the host system which can result in container escapes. For example, with the capability `SYS_ADMIN` an attacker might be able to mount devices from the host system inside of the container.

Ask Yourself Whether

Capabilities are granted:

- To a process that does not require all capabilities to do its job.
- To a not trusted process.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

Capabilities are high privileges, traditionally associated with superuser (root), thus make sure that the most restrictive and necessary capabilities are assigned.

Sensitive Code Example

```
apiVersion: v1
kind: Pod
metadata:
  name: example
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    securityContext:
      capabilities:
        add: ["SYS_ADMIN"] # Sensitive
```

Compliant Solution

```
apiVersion: v1
kind: Pod
metadata:
  name: example
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
```

See

- MITRE, CWE-250 - Execution with Unnecessary Privileges

- MITRE, CWE-266 - Incorrect Privilege Assignment
- Kubernetes Documentation - Configure a Security Context for a Pod or Container
- Linux manual page - capabilities(7)

Available In:

sonarcloud sonarqube