

































-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  **Kubernetes**
-  Objective C
-  PHP
-  PL/I
-  PL/SQL
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML



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...



Mounting sensitive file system paths is security-sensitive

 Security Hotspot

Using host operating system namespaces is security-sensitive

 Security Hotspot

Allowing process privilege escalations is security-sensitive

 Security Hotspot

Exposing Docker sockets is security-sensitive

 Security Hotspot


Running containers in privileged mode is security-sensitive

 Security Hotspot

Setting capabilities is security-sensitive

 Security Hotspot

Kubernetes parsing failure

 Code Smell

Using host operating system namespaces is security-sensitive

Analyze your code

 Security Hotspot  Major   cwe

Using host operating system namespaces can lead to compromise of the host systems.

These attacks would target:

- host processes
- host inter-process communication (IPC) mechanisms
- network services of the local host system

These three items likely include systems that support either the internal operation of the Kubernetes cluster or the enterprise's internal infrastructure.

Opening these points to containers opens new attack surfaces for attackers who have already successfully exploited services exposed by containers. Depending on how resilient the cluster is, attackers can extend their attack to the cluster by compromising the nodes from which the cluster started the process.

Host network sharing could provide a significant performance advantage for workloads that require critical network performance. However, the successful exploitation of this attack vector could have a catastrophic impact on confidentiality within the cluster.

Ask Yourself Whether

- The services of this Pod are accessible to people who are not administrators of the Kubernetes cluster.
- The cluster's services performances do **not** rely on operating system namespaces.

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

Recommended Secure Coding Practices

Do not use host operating system namespaces.

Sensitive Code Example

```
apiVersion: v1
kind: Pod
metadata:
  name: example
spec:
  containers:
    - name: web
      image: nginx
      ports:
        - name: web
          containerPort: 80
          protocol: TCP
  hostPID: true      # Sensitive
  hostIPC: true      # Sensitive
  hostNetwork: true # Sensitive
```

Compliant Solution

```
apiVersion: v1
```

```
kind: Pod
metadata:
  name: example
spec:
  containers:
    - name: web
      image: nginx
      ports:
        - name: web
          containerPort: 80
          protocol: TCP
  hostPID: false
  hostIPC: false
  hostNetwork: false
```

See

- [MITRE, CWE-284](#) - Improper Access Control

Available In:

