Kernel hardening (for Kubernetes)

LUG Frankfurt

Laura Liparulo

24.03.2026

Agenda

- quick introduction to Kubernetes
- Kernel security
- Seccomp
- Apparmor
- Kubelet security

What is Kubernetes?

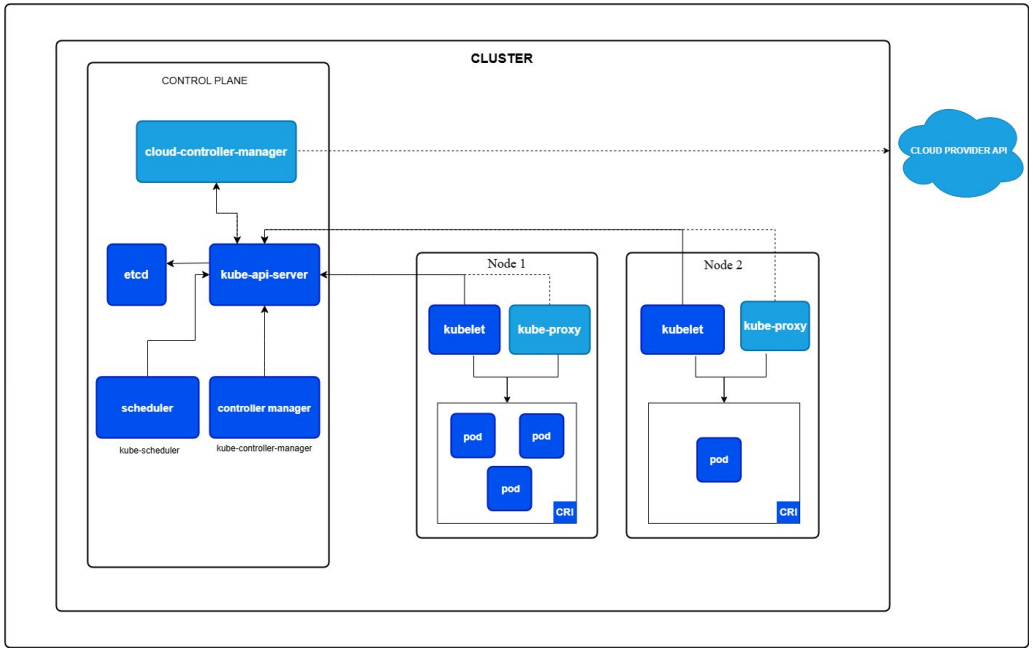
Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services.

It provides with:

- **Service discovery and load balancing**
- **Storage orchestration**
- **Automated rollouts and rollbacks**
- **Automatic bin packing**
- **Self-healing**
- **Secret and configuration management**
- **Batch execution**
- **Horizontal scaling**
- **IPv4/IPv6 dual-stack**
- **Designed for extensibility**

https://kubernetes.io/docs/concepts/overview/

Kubernetes architecture



https://kubernetes.io/docs/concepts/architecture/

OWASP KUBERNETES TOP 10

Top 10 Kubernetes Risks - 2025

2025 Top 10 Risks now available Feedback welcome. Please open issues or PRs for changes

- K01: Insecure Workload Configurations
- K02: Overly Permissive Authorization Configurations
- K03: Secrets Management Failures
- K04: Lack Of Cluster Level Policy Enforcement
- K05: Missing Network Segmentation Controls
- K06: Overly Exposed Kubernetes Components
- K07: Misconfigured And Vulnerable Cluster Components
- K08: Cluster To Cloud Lateral Movement
- K09: Broken Authentication Mechanisms
- K10: Inadequate Logging And Monitoring

K01: Insecure Workload Configurations

Pods are by default insecure!

It’s vital to implement the securityContext

To avoid:

- running as root
- **default linux capabilities**
- service account tokens
- **removing seccomp filters (default in Red Hat)**
- missing resource limits
- etc.

https://owasp.org/www-project-kubernetes-top-ten/2025/en/src/K01-Insecure-Workload-Configurations

Vulnerability example - CRI-O "cr8escape" vulnerability

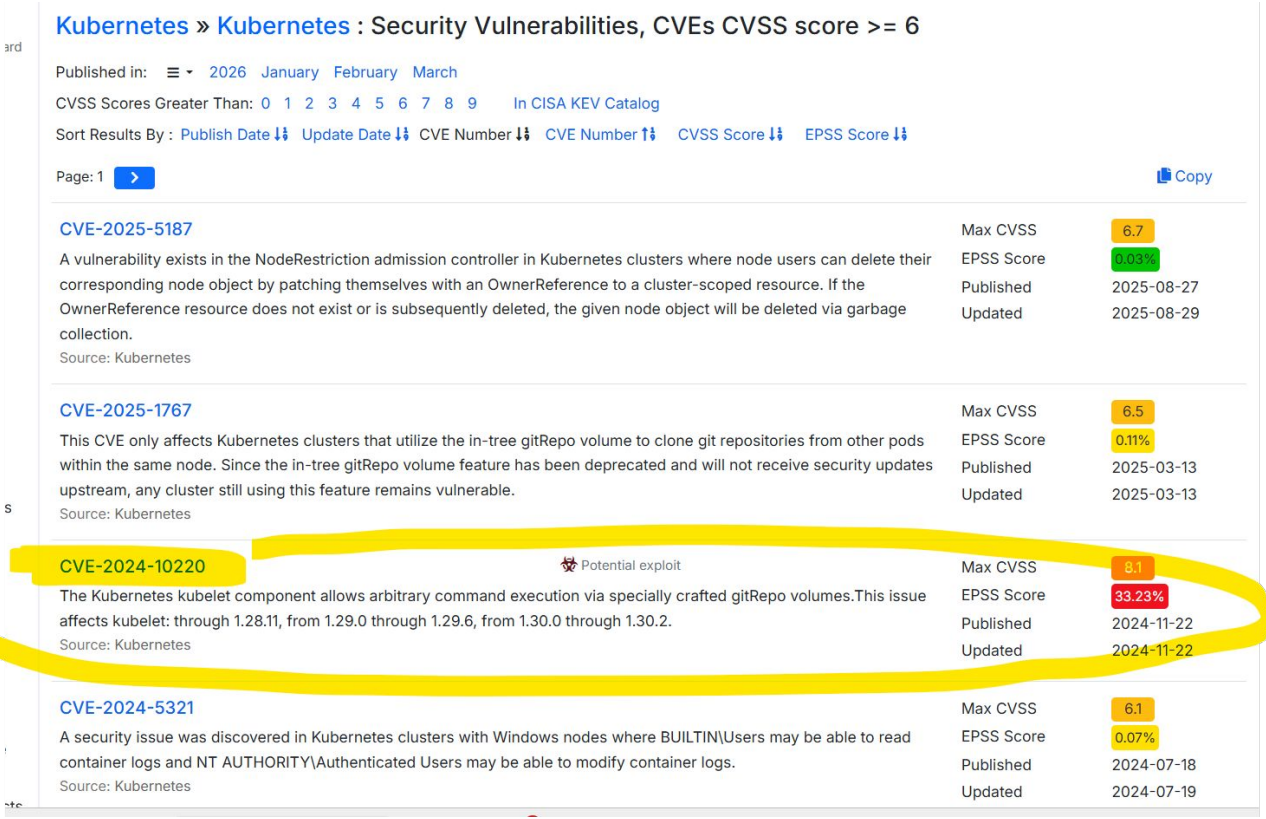
CVE-2022-0811 – Container Escape Vulnerability in CRI-O Runtime

- critical vulnerability in the **CRI-O** container runtime (**CVSS score 9.0**), discovered by CrowdStrike
- `kernel.core_pattern` parameter exploited during pod creation

An attacker can bypass container isolation to set arbitrary kernel parameters.

The container gets full **root access** to the host, facilitating malware deployment, data exfiltration, and lateral movement across the Kubernetes cluster.

Recently discovered K8s vulnerabilities



Kubernetes » Kubernetes : Security Vulnerabilities, CVEs CVSS score >= 6

Published in: 2026 January February March

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9 In CISA KEV Catalog

Sort Results By : Publish Date Update Date CVE Number CVE Number CVSS Score EPSS Score

Page: 1

Copy

CVE-2025-5187

A vulnerability exists in the NodeRestriction admission controller in Kubernetes clusters where node users can delete their corresponding node object by patching themselves with an OwnerReference to a cluster-scoped resource. If the OwnerReference resource does not exist or is subsequently deleted, the given node object will be deleted via garbage collection.

Source: Kubernetes

Max CVSS 6.7
EPSS Score 0.03%
Published 2025-08-27
Updated 2025-08-29

CVE-2025-1767

This CVE only affects Kubernetes clusters that utilize the in-tree gitRepo volume to clone git repositories from other pods within the same node. Since the in-tree gitRepo volume feature has been deprecated and will not receive security updates upstream, any cluster still using this feature remains vulnerable.

Source: Kubernetes

Max CVSS 6.5
EPSS Score 0.11%
Published 2025-03-13
Updated 2025-03-13

CVE-2024-10220

Potential exploit

The Kubernetes kubelet component allows arbitrary command execution via specially crafted gitRepo volumes.This issue affects kubelet: through 1.28.11, from 1.29.0 through 1.29.6, from 1.30.0 through 1.30.2.

Source: Kubernetes

Max CVSS 8.1
EPSS Score 33.23%
Published 2024-11-22
Updated 2024-11-22

CVE-2024-5321

A security issue was discovered in Kubernetes clusters with Windows nodes where BUILTIN\Users may be able to read container logs and NT AUTHORITY\Authenticated Users may be able to modify container logs.

Source: Kubernetes

Max CVSS 6.1
EPSS Score 0.07%
Published 2024-07-18
Updated 2024-07-19

Tools to defend from Kernel attacks

Kubernetes lets you configure and use Linux kernel features to improve isolation and harden your containerized workloads.

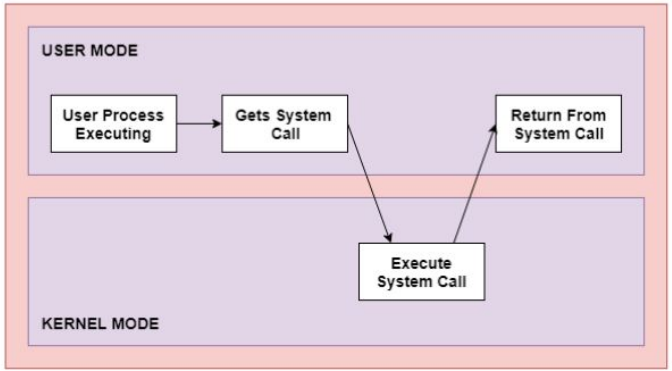
Common features include the following:

- **Secure computing mode (seccomp)**: Filter which system calls a process can make
- **AppArmor**: Restrict the access privileges of individual programs
- **Security Enhanced Linux (SELinux)**: Assign security labels to objects for more manageable security policy enforcement

Seccomp

Syscall workflow

1. The applications make the syscall request
2. The processor switches from *user mode* to *kernel mode*
3. The kernel performs the requested operation
4. The processor switches back to user mode



https://www.tutorialspoint.com/what-are-system-calls-in-operating-system

What is Seccomp?

- stands for “secure computing mode”
- sandboxing security mechanism
- first devised by Andrea Arcangeli in January 2005 for use in public grid computing
- feature of the Linux kernel since version 2.6.12 (March 2005)
- restricts the syscalls the process can make from user space into the kernel.
- extension called *seccomp-bpf* that uses the BPF - *Berkeley packet filter* - available

Seccomp in Kubernetes

Workload improved security by reducing the Linux kernel syscall attack surface available inside containers.

- containers use syscalls to interact with the host kernel
- each container's syscall can be monitored and restricted!
- seccomp profiles automatically loaded onto a node to your pods and containers.
- profiles can be applied to Kubernetes containers directly

Since Kubernetes 1.27, you can enable the use of RuntimeDefault as the default seccomp profile for all workloads.

Kubelet can be run with "**--seccomp-default**" to apply the runtime default profile

What Seccomp can do

Seccomp will:

- limits syscalls - allow only the syscalls required for your application to function
- prevents the exploitation of syscalls that are not approved.
- terminates the process in case of exploit
- lets you define three types of rules:
    - allow
    - deny
    - BPF based (Berkeley packet filter)

Configuring seccomp

Check the configuration within the kernel available:

```
controlplane:~$ grep SECCOMP /boot/config-6.8.0-90-generic
CONFIG_HAVE_ARCH_SECCOMP=y
CONFIG_HAVE_ARCH_SECCOMP_FILTER=y
CONFIG_SECCOMP=y
CONFIG_SECCOMP_FILTER=y
# CONFIG_SECCOMP_CACHE_DEBUG is not set
```

Ways to specify a seccomp profile

There are four ways to specify seccomp in the pod / deployment manifest

- for the whole Pod using spec.securityContext.seccompProfile
- for a single container using spec.containers[*].securityContext.seccompProfile
- for an (restartable / sidecar) init container using spec.initContainers[*].securityContext.seccompProfile
- for an ephemeral container using spec.ephemeralContainers[*].securityContext.seccompProfile

https://kubernetes.io/docs/reference/node/seccomp/

Seccomp profile types

- **Unconfined**

  The workload runs without any seccomp restrictions.

- **RuntimeDefault**

  A default seccomp profile defined by the container runtime is applied.

- **Localhost**

  The localhostProfile will be applied, which has to be available on the node disk (on Linux it's /var/lib/kubelet/seccomp)

  If the profile does not exist, then the container creation will fail with a *CreateContainerError*

https://kubernetes.io/docs/reference/node/seccomp/

Pod without seccomp - Privilege elevation example

Create a simple pod and escalate privileges by using the command “unshare” with “map-root-user” . As default, unshare syscall is available! you can operator as root!

It allow a process to control its shared execution context without creating a new process.

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: no-seccomp-pod
  name: no-seccomp-pod
spec:
  containers:
  - command: ["sleep","1h"]
    image: nginx
    name: no-seccomp-pod
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

```
root@controlplane:~$ kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
no-seccomp-pod   1/1     Running   0          79s
root@controlplane:~$ kubectl exec -it no-seccomp-pod -- unshare --map-root-user --user sh -c "whoami & id"
root
uid=0(root) gid=0(root) groups=0(root)
```

Pod with seccomp - Using the runtime default

RuntimeDefault as seccompProfile set in the security Context of the Pod:

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: seccomp-pod
  name: seccomp-pod
spec:
  securityContext:
    seccompProfile:
      type: RuntimeDefault
  containers:
  - command: ["sleep","1h"]
    image: nginx
    name: no-seccomp-pod
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

```
root@controlplane:~$ k get pods
NAME             READY   STATUS    RESTARTS   AGE
no-seccomp-pod   1/1     Running   0          4m9s
seccomp-pod      1/1     Running   0          3s
root@controlplane:~$ kubectl exec -it seccomp-pod -- unshare --map-root-user --user sh -c "whoami & id"
unshare: unshare failed: Operation not permitted
command terminated with exit code 1
```

We can get the container ID

```
root@controlplane:~$ k get pods -o wide
NAME              READY   STATUS    RESTARTS   AGE     IP             NODE     NOMINATED NODE   READINESS GATES
no-seccomp-pod    1/1     Running   0          6m29s   192.168.1.6    node01   <none>           <none>
seccomp-pod       1/1     Running   0          2m23s   192.168.1.7    node01   <none>           <none>
```

```
root@controlplane:~$ ssh node01
Last login: Mon Feb 10 22:06:42 2025 from 10.244.0.131
```

```
root@node01:~$ sudo crictl ps
CONTAINER           IMAGE               CREATED             STATE               NAME                ATTEMPT             POD ID              POD                 NAMESPACE
3a9e0a7d84a71       1a1e631364204       2 minutes ago       Running             no-seccomp-pod      0                   22c074529bdbc       seccomp-pod         default
0bd68b509c790       1a1e631364204       7 minutes ago       Running             no-seccomp-pod      0                   fc7234d9dc641       no-seccomp-pod      default
cec3e0a5dfdc1       52546a367cc9e       About an hour ago   Running             coredns             1                   f32225d682fde       coredns-76bb9b6fb5-vq26m   kube-system
4181f98702184       52546a367cc9e       About an hour ago   Running             coredns             1                   e3c8f5c77b180       coredns-76bb9b6fb5-l75b6   kube-system
fbe1fff441710       e6ea68648f0cd       About an hour ago   Running             kube-flannel        1                   dfbb48ecd97f7       canal-x84st         kube-system
0b6b2b785bb75       75392e3500e36       About an hour ago   Running             calico-node         1                   dfbb48ecd97f7       canal-x84st         kube-system
ea12d78545487       36eef8e07bdd6       About an hour ago   Running             kube-proxy          1                   60324c682e289       kube-proxy-wrxgl    kube-system
```

Take a look at the configuration

```
root@node01:~$ sudo crictl inspect 3a9e0a7d84a71 | jq .info.runtimeSpec.linux.seccomp
{
  "architectures": [
    "SCMP_ARCH_X86_64",
    "SCMP_ARCH_X86",
    "SCMP_ARCH_X32"
  ],
  "defaultAction": "SCMP_ACT_ERRNO",
  "syscalls": [
    {
      "action": "SCMP_ACT_ALLOW",
      "names": [
        "accept",
        "accept4",
        "access",
        "adjtimex",
        "alarm",
        "bind",
```

Simple Seccomp profile for auditing

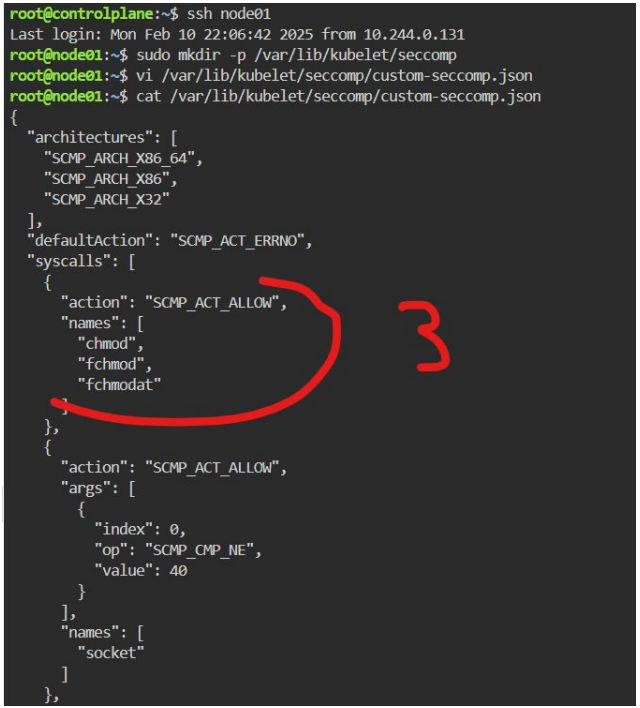
```
{
    "defaultAction": "SCMP_ACT_LOG"
}
```

Custom seccomp profile

You can place the json profile under **/var/lib/kubelet/seccomp**

SCMP_ACT_ALLOW : what is in the list will be allowed

```
root@controlplane:~$ ssh node01
Last login: Mon Feb 10 22:06:42 2025 from 10.244.0.131
root@node01:~$ sudo mkdir -p /var/lib/kubelet/seccomp
root@node01:~$ vi /var/lib/kubelet/seccomp/custom-seccomp.json
root@node01:~$ cat /var/lib/kubelet/seccomp/custom-seccomp.json
{
  "architectures": [
    "SCMP_ARCH_X86_64",
    "SCMP_ARCH_X86",
    "SCMP_ARCH_X32"
  ],
  "defaultAction": "SCMP_ACT_ERRNO",
  "syscalls": [
    {
      "action": "SCMP_ACT_ALLOW",
      "names": [
        "chmod",
        "fchmod",
        "fchmodat"
      ]
    },
    {
      "action": "SCMP_ACT_ALLOW",
      "args": [
        {
          "index": 0,
          "op": "SCMP_CMP_NE",
          "value": 40
        }
      ],
      "names": [
        "socket"
      ]
    },
```



Example - pod with custom seccomp profile

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-seccomp-pod
spec:
  securityContext:
    seccompProfile:
      type: Localhost
      localhostProfile: custom-seccomp.json
  containers:
  - name: nginx-container
    image: nginx
    command: ["/bin/sh", "-c", "sleep 1h"]
```

Set seccomp default

Edit the /var/lib/kubelet/config.yaml on the node and add:

```
seccompDefault: true
```

This will ensure that all the pods in the same node will use the standard Seccomp profile of the container runtime (CRI).

The profile defined in the pod manifest will take the precedence, if provided.

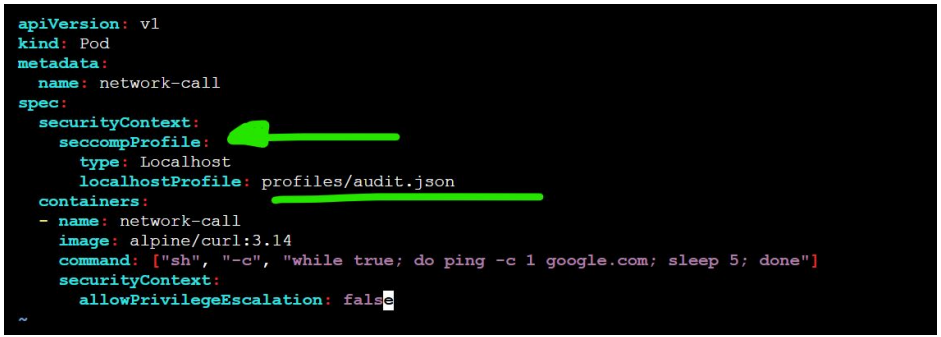
If no profile is specified, the runtime default will be used.

Demo - Logging all the syscalls - part 1

1) ssh node01
2) `mkdir -p /var/lib/kubelet/seccomp/profiles`
3) Create the seccomp profile - audit.json in the profiles folder with the default action

   `"defaultAction": "SCMP_ACT_LOG"`

4) exit from the node
5) on the controlpane create pod with the security contenxt



Demo - Logging all the syscalls - part 2

Inspect the logs

ssh

```
root@node01:~$ cat /var/log/syslog | grep sleep
Feb 22 13:08:59 node01 kernel: [ 5837.834446] audit: type=1326 audit(1771765739.539:233): auid=429496
7295 uid=0 gid=0 ses=4294967295 subj=cri-containerd.apparmor.d pid=11222 comm="sleep" exe="/bin/busyb
ox" sig=0 arch=c000003e syscall=231 compat=0 ip=0x7f2a7532bf0b code=0x7ffc0000
Feb 22 13:09:04 node01 kernel: [ 5842.849795] audit: type=1326 audit(1771765744.555:344): auid=429496
7295 uid=0 gid=0 ses=4294967295 subj=cri-containerd.apparmor.d pid=11247 comm="sleep" exe="/bin/busyb
ox" sig=0 arch=c000003e syscall=231 compat=0 ip=0x7f2ade499f0b code=0x7ffc0000
Feb 22 13:09:09 node01 kernel: [ 5847.860669] audit: type=1326 audit(1771765749.567:459): auid=429496
7295 uid=0 gid=0 ses=4294967295 subj=cri-containerd.apparmor.d pid=11314 comm="sleep" exe="/bin/busyb
ox" sig=0 arch=c000003e syscall=231 compat=0 ip=0x7fba0bc01f0b code=0x7ffc0000
Feb 22 13:09:14 node01 kernel: [ 5852.872874] audit: type=1326 audit(1771765754.579:576): auid=429496
7295 uid=0 gid=0 ses=4294967295 subj=cri-containerd.apparmor.d pid=11339 comm="sleep" exe="/bin/busyb
ox" sig=0 arch=c000003e syscall=231 compat=0 ip=0x7fea54c9bf0b code=0x7ffc0000
Feb 22 13:09:19 node01 kernel: [ 5857.883834] audit: type=1326 audit(1771765759.591:691): auid=429496
7295 uid=0 gid=0 ses=4294967295 subj=cri-containerd.apparmor.d pid=11392 comm="sleep" exe="/bin/busyb
ox" sig=0 arch=c000003e syscall=231 compat=0 ip=0x7f3404551f0b code=0x7ffc0000
Feb 22 13:09:24 node01 kernel: [ 5862.893980] audit: type=1326 audit(1771765764.599:802): auid=429496
7295 uid=0 gid=0 ses=4294967295 subj=cri-containerd.apparmor.d pid=11437 comm="sleep" exe="/bin/busyb
ox" sig=0 arch=c000003e syscall=231 compat=0 ip=0x7f545772df0b code=0x7ffc0000
Feb 22 13:09:29 node01 kernel: [ 5867.911715] audit: type=1326 audit(1771765769.619:915): auid=429496
7295 uid=0 gid=0 ses=4294967295 subj=cri-containerd.apparmor.d pid=11462 comm="sleep" exe="/bin/busyb
```

Apparmor

What is AppArmor?

- security kernel module that provides mandatory access control (MAC) for applications
- Tool limiting what a process can do on the host based on a profile (process isolation)
- uses program profiles to restrict the capabilities of individual programs
- initially released in 1998 by Immunix Inc., then SUSE since 2005
- included on Distributions like Debian, Arch Linux, openSUSE
- since Kernel version 2.6.36
- 2 modes
  - **enforcing mode** (prevent actions)
  - **complain mode**
- enhanced logging capabilities
- more focused on defending the application / resources

How does Apparmor work?

- **Profile-based** restrictions
- **Deny-by-Default** approach (define what is allowed)
- **Exploits mitigation** - malicious activities prevention

Install the extra Apparmor utilities

Apparmor can be often be found already installed, but you might also needs some other utilities.

On Ubuntu:

```
> sudo apt install -y apparmor apparmor-utils
```

```
> sudo apt install apparmor-notify
```

Check the status of the apparmor service

```
controlplane:~$ sudo systemctl status apparmor
● apparmor.service - Load AppArmor profiles
     Loaded: loaded (/usr/lib/systemd/system/apparmor.service; enabled; preset: enabled)
     Active: active (exited) since Tue 2025-12-23 20:28:47 UTC; 1h 7min ago
       Docs: man:apparmor(7)
             https://gitlab.com/apparmor/apparmor/wikis/home/
   Main PID: 451 (code=exited, status=0/SUCCESS)
        CPU: 347ms

Dec 23 20:28:46 controlplane systemd[1]: Starting apparmor.service - Load AppArmor profiles...
Dec 23 20:28:46 controlplane apparmor.systemd[451]: Restarting AppArmor
Dec 23 20:28:46 controlplane apparmor.systemd[451]: /lib/apparmor/apparmor.systemd: 148: [: Illegal number: yes
Dec 23 20:28:46 controlplane apparmor.systemd[451]: Reloading AppArmor profiles
Dec 23 20:28:47 controlplane systemd[1]: Finished apparmor.service - Load AppArmor profiles.
controlplane:~$
```

Apparmor configuration under /etc/apparmor.d

```
root@controlplane:~$ cd /etc/apparmor.d/
root@controlplane:/etc/apparmor.d$ ls
1password          ch-run          geary              lsb_release     nvidia_modprobe  rootlesskit          sbuild-destroychroot  stress-ng          unix-chkpw
d            vivaldi-bin
Discord            chrome          github-desktop     lxc-attach      obsidian         rpm                  sbuild-distupgrade    surfshark          unprivileg
ed_userns    vpnns
MongoDB_Compass    code            goldendict         lxc-create      opam             rssguard             sbuild-hold           systemd-coredump   userbindmo
unt          wike
QtWebEngineProcess crun            ipa_verify         lxc-destroy     opera            rsyslog.d            sbuild-shell          thunderbird        usr.bin.ma
n            wpcom
abi                devhelp         kchmviewer         lxc-execute     pageedit         runc                 sbuild-unhold         toybox             usr.bin.pa
sst
abstractions       element-desktop keybase            lxc-stop        plasmashell      sbuild               sbuild-update         transmission       usr.bin.tc
pdump
balena-etcher      epiphany        lc-compliance      lxc-unshare     podman           sbuild-abort         sbuild-upgrade        trinity            usr.sbin.c
ups-browsed
brave              evolution       libcamerify        lxc-usernsexec  polypane         sbuild-adduser       scide                 tunables           usr.sbin.c
upsd
buildah            firefox         lightdm-guest-session  mmdebstrap  privacybrowser   sbuild-apt           signal-desktop        tup                usr.sbin.r
syslogd
busybox            flatpak         linux-sandbox      msedge          qcam             sbuild-checkpackages slack                 tuxedo-control-center  uwsgi-core
cam                foliate         local              nautilus        qmapshack        sbuild-clean         slirp4netns           ubuntu_pro_apt_news    vdens
ch-checkns         force-complain  loupe              notepadqq       qutebrowser      sbuild-createchroot  steam                 ubuntu_pro_esm_cache   virtiofsd
```

Check apparmor status

> sudo aa-status / apparmor_status

```
Dec 23 20:28:47 controlplane systemd[1]: Finished apparmor.service - Loa
controlplane:~$ sudo aa-status
apparmor module is loaded.
126 profiles are loaded.
31 profiles are in enforce mode.
   /usr/bin/man
   /usr/lib/cups/backend/cups-pdf
   /usr/lib/lightdm/lightdm-guest-session
   /usr/lib/lightdm/lightdm-guest-session//chromium
   /usr/sbin/cups-browsed
   /usr/sbin/cupsd
   /usr/sbin/cupsd//third_party
   cri-containerd.apparmor.d
   docker-default
   lsb_release
   man_filter
   man_groff
   nvidia_modprobe
   nvidia_modprobe//kmod
   passt
   plasmashell
   plasmashell//QtWebEngineProcess
   rsyslogd
   tcpdump
   ubuntu_pro_apt_news
   ubuntu_pro_esm_cache
   ubuntu_pro_esm_cache//apt_methods
   ubuntu_pro_esm_cache//apt_methods_gpgv
   ubuntu_pro_esm_cache//cloud_id
   ubuntu_pro_esm_cache//dpkg
   ubuntu_pro_esm_cache//ps
   ubuntu_pro_esm_cache//ubuntu_distro_info
   ubuntu_pro_esm_cache_systemctl
   ubuntu_pro_esm_cache_systemd_detect_virt
```

Example - deny-write-tmp profile (in /etc/apparmor.d)

```
#include <tunables/global>


profile k8s-apparmor-example-deny-write flags=(attach_disconnected) {
  #include <abstractions/base>


  file,


  # Deny all file writes.
  deny /** w,
}
```

apparmor_parser

The apparmor_parser command loads the profile into the current state of apparmor:

```
vagrant@worker-node01:~$ sudo apparmor_parser -r -W /etc/apparmor.d/deny-write-tmp
```

The profile parsing must be repeated on each node!!!

Load the Apparmor profile on every node

```
# This example assumes that node names match host names, and are reachable via SSH.
NODES=($( kubectl get node -o jsonpath='{.items[*].status.addresses[?(.type == "Hostname")].address}' ))

for NODE in ${NODES[*]}; do ssh $NODE 'sudo apparmor_parser -q <<EOF
#include <tunables/global>

profile k8s-apparmor-example-deny-write flags=(attach_disconnected) {
  #include <abstractions/base>

  file,

  # Deny all file writes.
  deny /** w,
}
EOF'
done
```

Securing a pod

AppArmor profiles can be specified at the pod level or container level. The container AppArmor profile takes precedence over the pod profile.

```
securityContext:

 appArmorProfile:

   type: <profile_type>
```

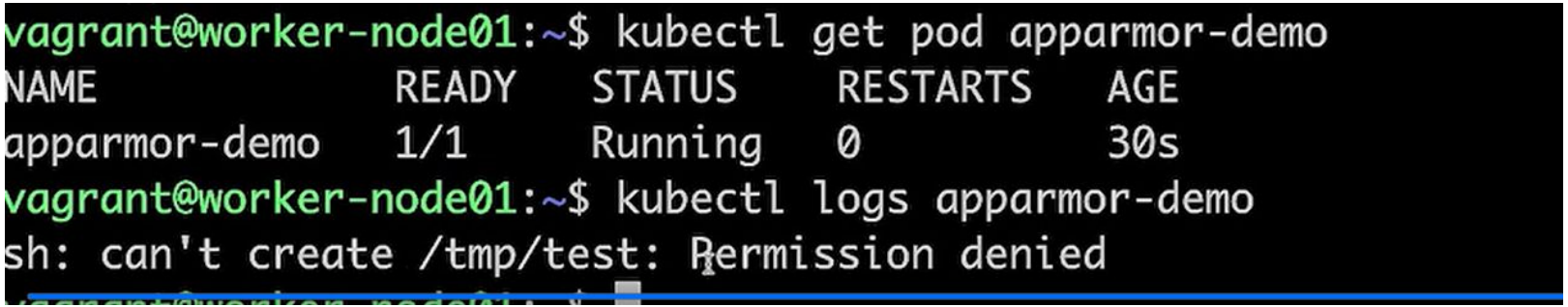
Where <profile_type> is one of:

- **RuntimeDefault** - to use the runtime's default profile
- **Localhost** - to use a profile loaded on the host
- **Unconfined** - to run without AppArmor

Add the profile to the resource

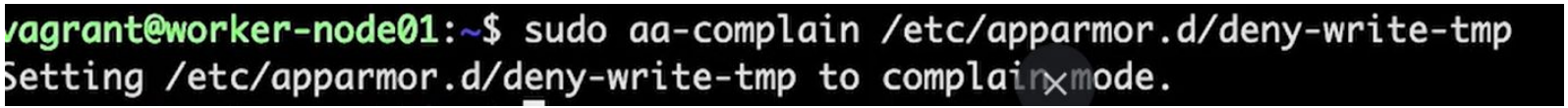
```
vagrant@worker-node01:~$ cat <<EOF | kubectl apply -f -
> apiVersion: v1
> kind: Pod
> metadata:
>   name: apparmor-demo
> spec:
>   containers:
>   - name: test-container
>     image: busybox
>     command: ["sh", "-c", "echo 'Hello' > /tmp/test; if [ $? -ne 0 ]; then echo 'Write failed
'; fi; sleep 1h"]
>     securityContext:
>       appArmorProfile:
>         type: Localhost
>         localhostProfile: deny-write-tmp
> EOF
```

Check the logs



From enforced to complain mode

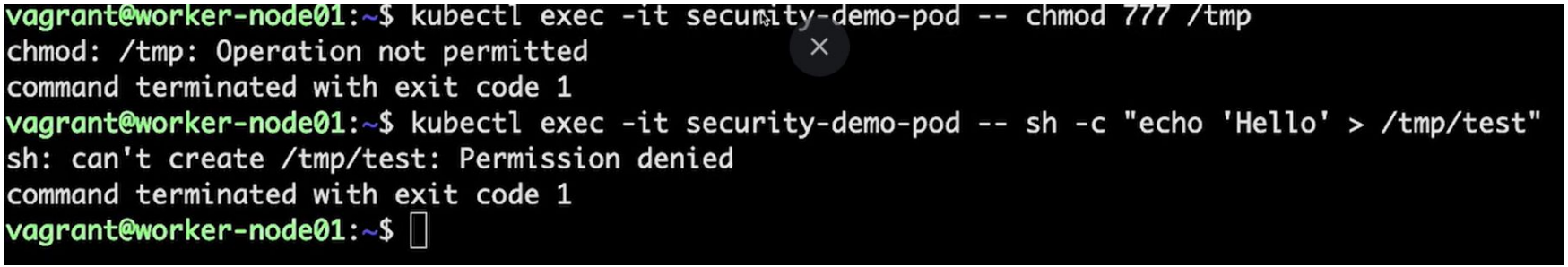
Using **aa-complain** command to set the profile in complain mode:



Using both Apparmor and Seccomp together

```
vagrant@worker-node01:~$ cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
   name: amicontained
spec:
   containers:
      - image: jess/amicontained:v0.4.9
        name: amicontained
        command: [ "/bin/sh", "-c", "--" ]
        args: [ "amicontained" ]
        securityContext:
          seccompProfile:
            type: Localhost
            localhostProfile: custom-profile.json
          appArmorProfile:
            type: Localhost
            localhostProfile: deny-write-tmp
EOF
pod/amicontained created
```

chmod syscall + resource access blocked! both in place!



Assigning the profile as annotation

You can assign the profile to a pod, also by using the annotation

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: network-call
  annotations:
    container.apparmor.security.beta.kubernetes.io/network-call: localhost/network-deny
spec:
  containers:
  - name: network-call
    image: alpine/curl:3.14
    command: ["sh", "-c", "while true; do ping -c 1 google.com; sleep 5; done"]
~
```

Apparmor profiles authoring

Getting AppArmor profiles specified correctly can be a tricky business.

Fortunately there are some tools to help with that:

- aa-genprof and aa-logprof generate profile rules by monitoring an application's activity and logs, and admitting the actions it takes. Further instructions are provided by the AppArmor documentation.
- bane is an AppArmor profile generator for Docker that uses a simplified profile language.

Kubelet security

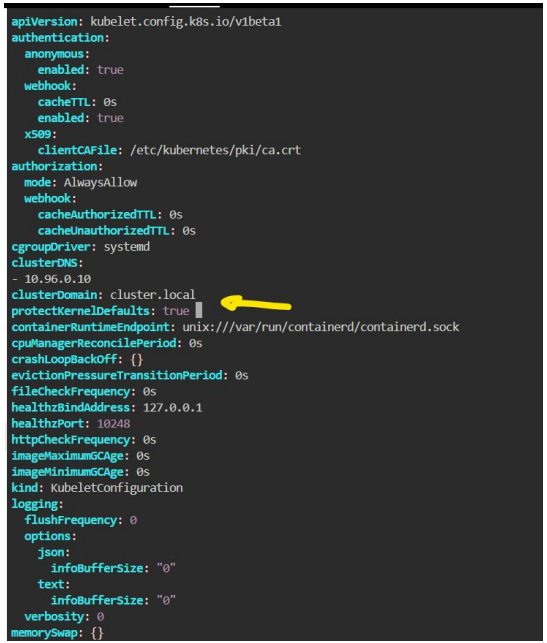
What is a kubelet?

A kubelet is the primary "node agent" that runs on every worker node in a Kubernetes cluster. It’s a local manager or "captain" of the node, ensuring that the assigned containers are running and healthy.

Kubelet security

You can use the property ***protectKernelDefaults*** in the kubelet configuration to prevent the kubelet from modifying the kernel defaults.

Main configuration: /var/lib/kubelet/config.yaml



Thanks for you attention!

Bibliography

- https://owasp.org/www-project-kubernetes-top-ten/
- https://owasp.org/www-project-kubernetes-top-ten/2025/en/src/K01-Insecure-Workload-Configurations
- https://kubernetes.io/docs/reference/issues-security/official-cve-feed/
- https://kubernetes.io/docs/tutorials/security/seccomp/
- https://kubernetes.io/docs/tutorials/security/apparmor
- https://gitlab.com/apparmor/apparmor
- https://www.cvedetails.com/vulnerability-list/vendor_id-15867/product_id-34016/Kubernetes-Kubernetes.html?page=1&cvssscoremin=6&order=1
- Certified Kubernetes Security Specialist (CKS) - Video Course, Chris Jackson, Pearson
- Certified Kubernetes Security Specialist (CKS) Study Guide, Benjamin Muschko, O'reilly
- https://www.kernel.org/doc/Documentation/prctl/seccomp_filter.txt
- https://gitlab.com/apparmor
- https://apparmor.net/
-

Questions?