Using Kernel Hardening Tools: seccomp

Linux sys calls

EveryLinux machine will have a kernal space on the hardware, then we have a user space where we keep our apps, every app perform some calls like (read, send, close, write, execute, accept).

Seccomp purpose is to restrict Container's (pod is container) syscalls so it cannot compromise the machine.

Seccomp restricts syscalls to resources.

(calls like: clock_adjtime, mount, unmount, open.....etc)

Objective is to secure/restrict the containers by what they are allowed to do?

Like AppArmor it is also configured through profiles

SecComp Profile are a list of SysCalls to allow/deny/audit.

SecComp Profiles are enabled at kubelet level config (disabled by default)

let say container run time is restricting system calls, we can enable by:

--feature-gates=SeccompDefault=true

seccomp profiles default placement:

/var/lib/kubelet/seccomp/profiles/<profile-name.json>

SECCOMP Profiles are 3 types

- Default Profile
- Audit – audit.json = all the syscalls will be logged at /var/log/syslog
- Violation- violation.json = all syscalls will be rejected.
- Fine-Grained (Custom) – fine-grained.json = I want to allow only 5 or 10 calls rest all are restricted.

SecComp profile has defaultAction and Action

SECCOMP can operate with 3 modes

Mode 0 – disabled mode

Mode 1 - strict mode

Mode 2 - filtered mode

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

```
{
    "defaultAction": "SCMP_ACT_ERRNO"
}
```

```
{
    "defaultAction": "SCMP_ACT_ERRNO",
    "architectures": [
        "SCMP_ARCH_X86_64",
        "SCMP_ARCH_X86",
        "SCMP_ARCH_X32"
    ],
    "syscalls": [
        {
            "names": [
                "accept4",
                "epoll_wait",
                "pselect6",
                "futex",
                "madvise",
                "epoll_ctl",
                "getsockname",
                "setsockopt",
                "vfork",
                "mmap",
                "read",
                "write",
                "close",
                "arch_prctl",
                "sched_getaffinity",
                "munmap",
                "brk",
                "rt_sigaction",
                "rt_sigprocmask",
                "sigaltstack",
                "gettid",
                "clone",
                "bind",
                "socket",
                "openat",
                "readlinkat",
                "exit_group",
                "epoll_create1",
                "listen",
                "rt_sigreturn",
                "sched_yield",
                "clock_gettime",
                "connect",
                "dup2",
                "epoll_pwait",
                "execve",
                "exit",
                "fcntl",
                "getpid",
                "getuid",
                "ioctl",
                "mprotect",
                "nanosleep",
                "open",
                "poll",
                "recvfrom",
                "sendto",
                "set_tid_address",
                "setitimer",
                "writev"
            ],
            "action": "SCMP_ACT_ALLOW"
        }
    ]
}
```

we will follow the procedure as follow after creating a cluster:

Create Profile

Copy SecComp profiles (to all nodes of cluster or the pod you want to ` deploy it in)

Apply profile to Pod SecContext

Containers & System are Secure

there are three types:

Localhost

RuntimeDefault

unconfined

```
ApiVersion: v1
kind: Pod
metadata:
  name: audit-pod
  labels:
    app: audit-pod
spec:
  securityContext:
    seccompProfile:
      type: Localhost
      localhostProfile: profiles/audit.json
  containers:
  - name: test-container
    image: hashicorp/http-echo:1.0
    args:
    - "-text=just made some syscalls!"
    securityContext:
      allowPrivilegeEscalation: false
```

Make sure Privilege Escalation is false in this container as one pod can have multiple containers.

DEMO

kubectl get nodes

kubectl get pods

systemctl status kubelet

clear

cat /var/lib/kubelet/config.yaml

cd /var/lib/kubelet/

mkdir -p /var/lib/kubelet/seccomp/profiles

cd seccomp

ls

```
curl -L -o profiles/audit.json
https://k8s.io/examples/pods/security/seccomp/profiles/audit.json
curl -L -o profiles/violation.json
https://k8s.io/examples/pods/security/seccomp/profiles/violation.json
curl -L -o profiles/fine-grained.json
https://k8s.io/examples/pods/security/seccomp/profiles/fine-grained.json
```

ls profiles

Shift Terminal

kubectl get pod

kubectl create deployment web --image nginx --replicas 1

kubectl get pod

kubectl apply -f https://k8s.io/examples/pods/security/seccomp/ga/audit-pod.yaml

kubectl get pod -o wide

Shift Terminal

tail -f /var/log/syslog # these are the syscalls audit pod is making.

Shift terminal

kubectl get svc

kubectl expose pod audit-pod --type NodePort --port 5678

kubectl get svc

kubectl get node -o wide

curl http://192.168.56.10:30040/

# it will log every syscall

```
kubectl delete service audit-pod --wait
kubectl delete pod audit-pod --wait --now
```

#lets move to violation pod, nothing will work as it will block all the calls

kubectl apply -f https://k8s.io/examples/pods/security/seccomp/ga/violation-pod.yaml

kubectl get pod -o wide

kubectl get svc

#change terminal and you will see Violation pod

kubectl describe pod violation-pod

#backoff starting the failed container it happens because we are using violation pod which is blocking every sys call.

kubectl delete pod violation-pod --wait --now

kubectl get pod -o wide

kubectl apply -f https://k8s.io/examples/pods/security/seccomp/ga/fine-pod.yaml

kubectl get pod -o wide

kubectl get svc

kubectl expose pod fine-pod --type NodePort --port 5678

kubectl get svc

```
curl http://192.168.56.10:31994/
```

```
cat /var/log/syslog | grep -i fine-pod
```

https://kubernetes.io/docs/tutorials/security/seccomp/