

Hi, I'm Jose



Jose Blanquicet
Software Engineer
Microsoft



Agenda

- ☐ Introduction to Seccomp
- ☐ Seccomp in Kubernetes
- ☐ Inspektor Gadget and its seccomp gadget
- ☐ Live Demo! 💂









Seccomp



What is it?

> Seccomp stands for secure computing

➤ It is a Linux kernel feature that allows us to restrict the system calls a process can do



How does it work?

> Strict mode:

- The first developed seccomp mode.
- Allowed specific syscalls: read(), write(), exit(), and sigreturn().
- Any other syscall made by the process, seccomp sends a SIGKILL signal.

> Filter mode:

- We can specify what we want to do for each system calls.
- Every syscall passes through a filter that is done using BPF (Berkeley Packet Filter).
- Result of the filter: Allow, kill, errno, log and others.
- Also known as seccomp mode 2 or seccomp-bpf.



Seccomp in Kubernetes



Hardening container's security

- ➤ In the securityContext section of the Container or Pod, we can define the Seccomp profile to be used:
 - O <u>Kubernetes API Reference Docs</u>
 - Restrict a Container's Syscalls with seccomp Kubernetes [stable from v1.19]
- ➤ Seccomp profile types:
 - Unconfined: Seccomp disabled (Default configuration)
 - RuntimeDefault: Container runtimes default profile:
 - O <u>Default Docker profile</u> is what most runtimes use
 - Enable seccomp for all workloads with a new v1.22 alpha feature | Kubernetes
 - O Localhost: Custom profile:
 - localhostProfile: my-profiles/profile.json
 - Available in the node at <kubelet-root-dir>/seccomp/my-profiles/profile.json



How a Seccomp profile looks like?

```
"defaultAction": "SCMP_ACT_ERRNO",
"architectures": [
"syscalls": [
        "names": [
            "read",
            "write",
        "action": "SCMP_ACT_ALLOW"
    },
```



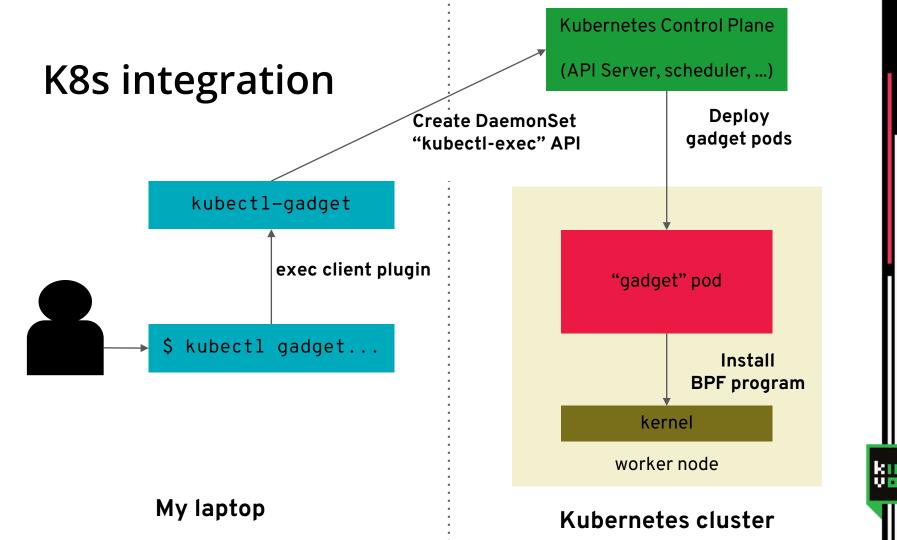
Inspektor Gadget



Tracing Cloud Native applications

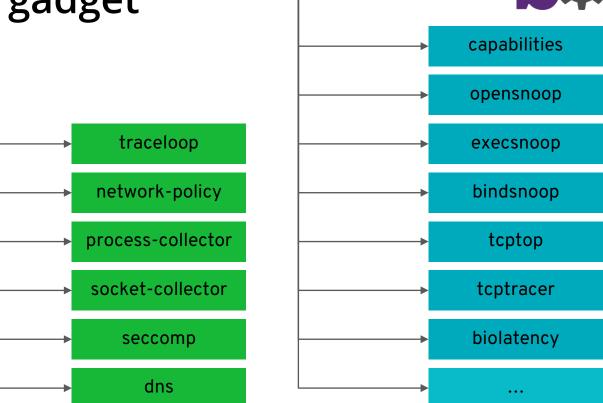
- ➤ Granularity of tracing: pods
 - O PIDs are not useful when we don't know which pod it is
 - We don't want to trace all the system processes on a node
- ➤ Aggregation
 - O Using Kubernetes labels, namespace, etc
- ➤ kubectl-like user experience
 - Developers should not need to SSH
 - O Developers should not need to deploy a pod + kubectl-exec for each tracing
- > Integrated experience with existing tools





"kubectl gadget"







Selectors

```
$ kubectl gadget execsnoop \
    --selector k8s-app=myapp,tier=bar \
    --namespace default \
    --podname myapp1-l9ttj \
    --node ip-10-0-12-31 \
    --containername my-container
```



Trace custom resource

```
apiVersion: gadget.kinvolk.io/v1alpha1
kind: Trace
metadata:
  name: process-collector
  namespace: gadget
spec:
  node: aks-agentpool-1234-vmss01
  gadget: process-collector
  filter:
    labels:
      component: etcd
    namespace: kube-system
  outputMode: Status
  runMode: Manual
```



Seccomp advisor gadget



How does it work?

- > Uses eBPF to records the syscalls that are issued by each container.
- ➤ Then, uses this information to generate a seccomp profile with SCMP_ACT_ERRNO as default action and an allowed list with the recorded syscalls.





How to use the gadget?

- > We can use the kubectl-gadget CLI or the Trace custom resource (integration with Web UIs).
- > Profiles are generated when we stop the gadget or, automatically, when pods terminate.
- ➤ Integration with the Kubernetes Security Profiles Operator or manually deploying the generated profiles in the nodes.



Live Demo



Questions?



Get involved!



Contribute

Inspektor Gadget:

- □ https://github.com/kinvolk/inspektor-gadget/blob/main/docs/
 CONTRIBUTING.md
- ☐ #inspektor-gadget channel in the Kubernetes Slack.



Thank you!

Jose Blanquicet

Github: blanquicet

LinkedIn: joseblanquicet Twitter: @jose_blanquicet Email: josebl@microsoft.com

