Inspektor Gadget

Kernel observability using eBPF made easy and approachable with Inspektor Gadget

SCaLE 22x

Maya Singh & Jose Blanquicet

March 8th, 2025

CLOUD NATIVE COMPUTING FOUNDATION

Agenda

- Introductions
- Interactive poll
- Introduction to eBPF
- Introduction to Inspektor Gadget
  - Demo
  - How it works
  - How you can use it
- Demos!

Introductions

Maya Singh

Product Manager @ Microsoft

Jose Blanquicet

Sr Software Engineer @ Microsoft

System Level Challenges

Inspektor Gadget

Debugging

- OOM Kill events
- DNS queries not resolving
- Deadlocks (when a process doesn't proceed due to another process using resources)

Monitoring

- Identifying source of high latency
  - Visibility into CPU usage
  - Stress on the system from a specific container
  - Process from a container excessively reading/writing to a file

Security

- Identifying potential malicious activity
  - Shell open on a K8s cluster
  - Validating binary changes in container image
  - Monitoring processes accessing the filesystem
- Seccomp profiles generation

Inspektor Gadget

Menti Poll

Which of these issues do you have the hardest time solving?

Inspektor Gadget

Menti Poll

What comes to mind when you think about “eBPF”?

Inspektor Gadget

Intro to eBPF

What is eBPF?

Inspektor Gadget

eBPF is in-kernel bytecode runtime used for tracing, security, networking ...

Capabilities

+ Brings flexibility to the kernel
+ Low strain from a performance perspective
+ Won’t crash your kernel

Examples of eBPF use cases

Observability
eBPF can be used to measure CPU usage, memory allocation, and similar metrics which can be used for performance troubleshooting

Security
eBPF can be used to enforce access control policies, you can whitelist/blacklist specific system calls, network connection etc...

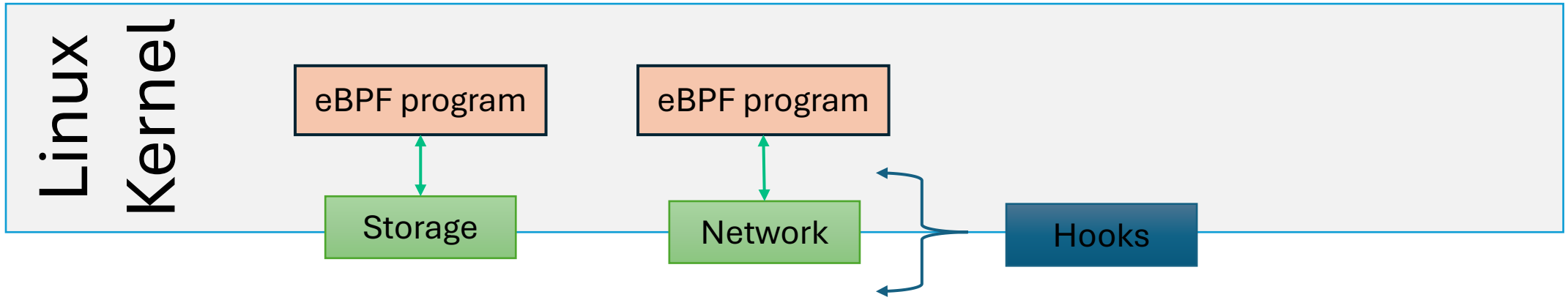
Networking
eBPF allows for packet filtering and modification within the Linux kernel (Firewall rules)

eBPF traffic monitoring | Android Open Source Project
Keynote: eBPF - Everything You Need to Know in 5 Minutes - Thomas Graf, CTO, Isovalent (youtube.com)

Inspektor Gadget

eBPF Hooks

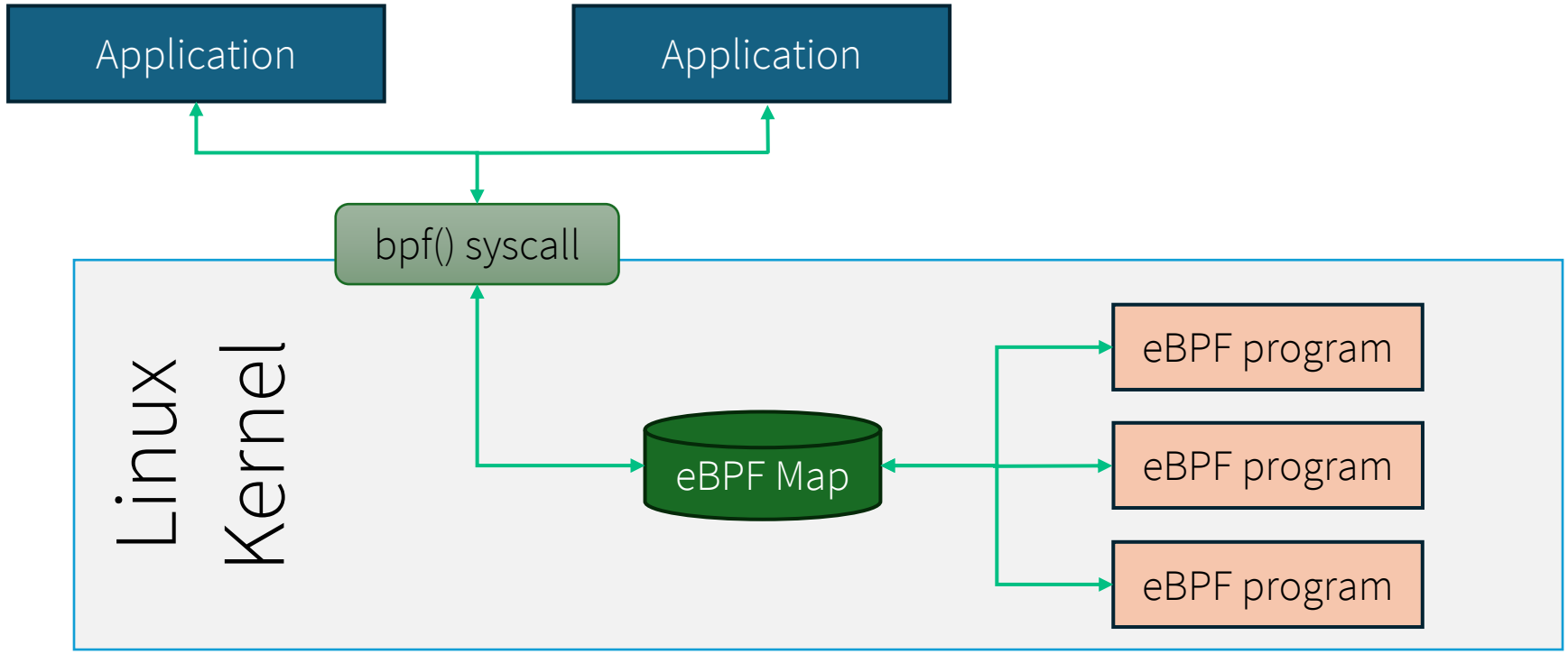
eBPF is event driven, when “hooks” are passed, eBPF programs are executed



Source: Mauricio Vasquez Bernal & Chris Kuehl

eBPF Maps

Key/Value structures to share information between eBPF programs and user space applications



Source: Mauricio Vasquez Bernal & Chris Kuehl

What is eBPF?

Inspektor Gadget

eBPF is in-kernel bytecode runtime used for tracing, security, networking ...

Capabilities

- Brings flexibility to the kernel
- Low strain from a performance perspective
- Won’t crash your kernel

Challenges

- Steep learning curve
- Requires deep knowledge of low-level systems troubleshooting
- Limited higher level context

Inspektor Gadget

Intro to Inspektor Gadget

Why Inspektor Gadget

Inspektor Gadget

- eBPF is an extremely powerful tool for gathering system information
- But eBPF is hard – technically and intuitively
- Once you have data, it's still not immediately useful
  - How does this kernel data relate to my system as I understand it?
  - Where do I send the data?
- Lots of additional tooling needed for…
  - Managing eBPF programs
  - Mapping kernel data to higher-level resources (K8s, container runtimes, etc.)
  - Doing userspace processing
  - Exporting data / providing data via API

open source tool and framework for data collection and systems inspection on Kubernetes and Linux hosts using eBPF

Gadgets

Gadgets encapsulate eBPF programs in OCI images for powerful and performant systems inspection in a secure way

Enrichment

Container and K8s aware - automatically map low-level systems information to high-level Kubernetes and container resources

Framework

An observability framework with everything you need to collect, filter, format and export valuable systems data

Inspektor Gadget

Demo

open source tools and framework for data collection and systems inspection on Kubernetes and Linux hosts using eBPF

Gadgets

Gadgets encapsulate eBPF programs in OCI images for powerful and performant systems inspection in a secure way

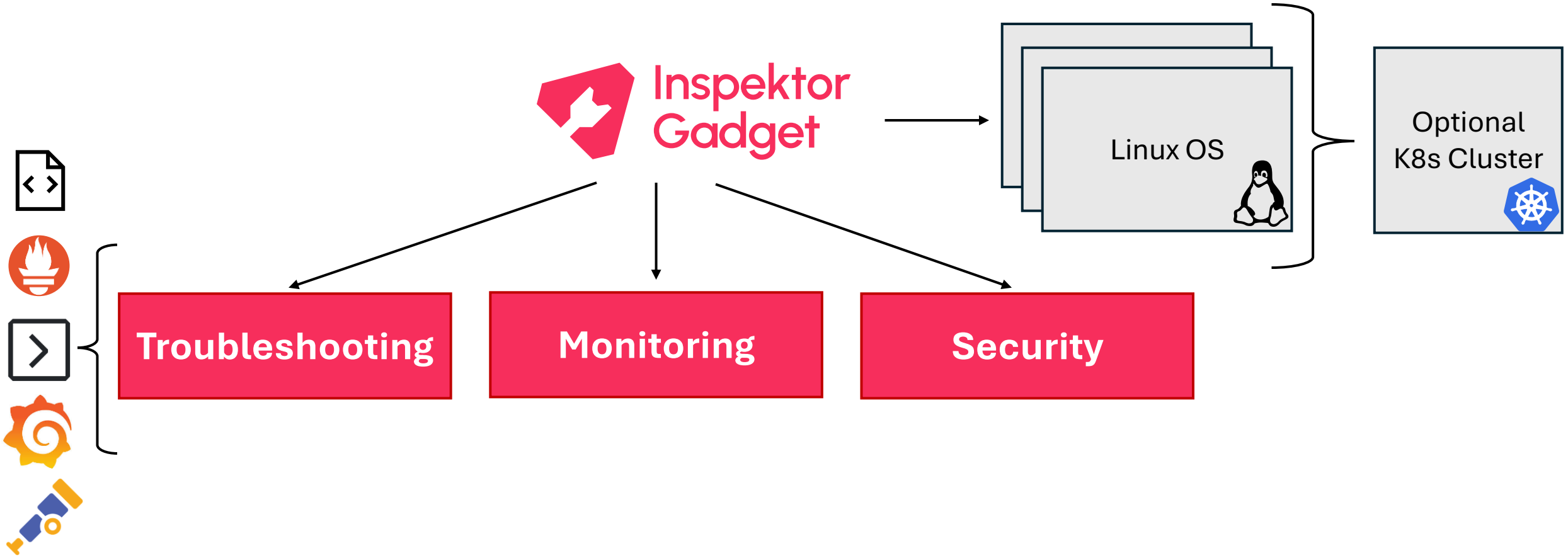
Enrichment

Container and K8s aware - automatically map low-level systems information to high-level Kubernetes and container resources

Framework

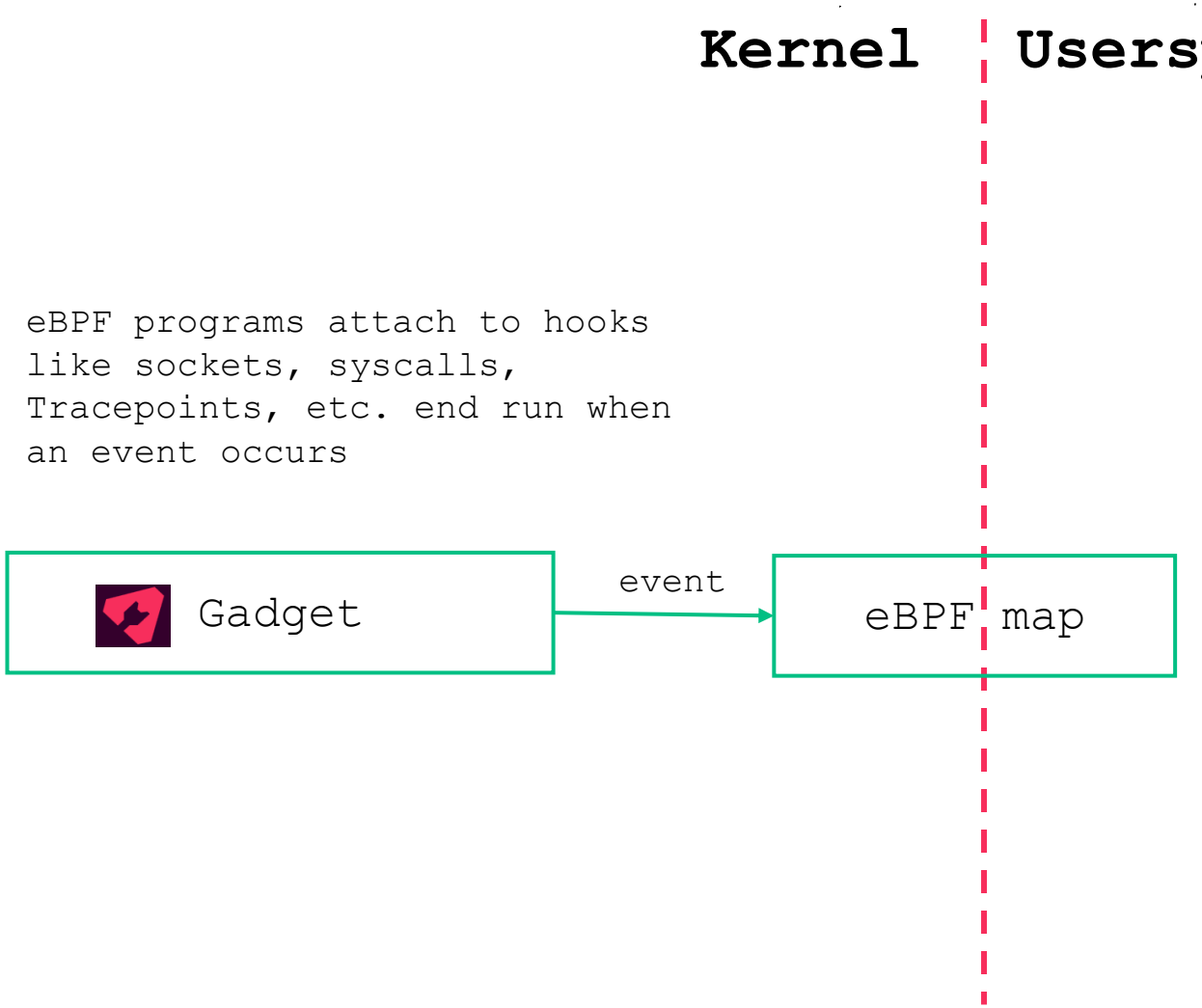
An observability framework with everything you need to collect, filter, format and export valuable systems data

Inspektor Gadget



eBPF with Inspektor Gadget

Inspektor Gadget



- Enrichment
- Filtering
- Userspace processing
- Data export
- Sharing & distribution
- Many modes of use

Enrichment & Filtering

Inspektor Gadget

Problem: events from eBPF give **low-level data**:

- Kernel namespaces
- cgroups

Solution: event enrichment **adds high-level data**:

- Kubernetes pods, containers
- Domain names or Kubernetes services from IP
- Container information

Event filtering: showing a subset of events

- From selected containers, Kubernetes pod, namespace, labels
- Filtered in eBPF for performance, but abstracted for gadget authors

Inspektor Gadget

Demo

Inspektor Gadget

Intro to Gadgets

Anatomy of a gadget

Inspektor Gadget

Gadget (OCI Image)

- Metadata (yaml file)
- eBPF programs+maps (ELF file)
- Userspace module (Wasm file)

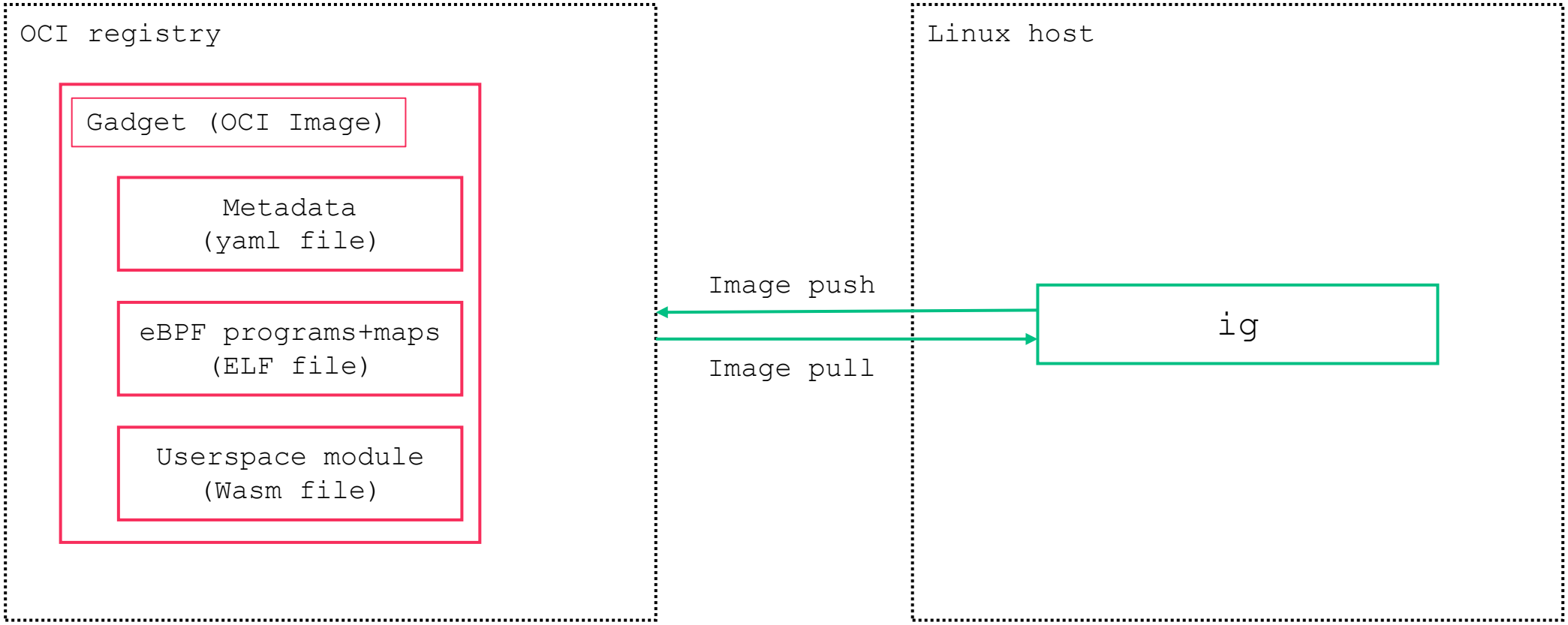
- Information about
  - The gadget
  - Capabilities
  - Output formatting
  - Build information

- One or more eBPF programs

- Userspace modules for post-processing of eBPF data.
- Can be in any language WASM supports

* Also looking to include
  - Documentation
  - Source code
  - Logo
  - etc.

How it works


OCI registry
Gadget (OCI Image)
Metadata
(yaml file)
eBPF programs+maps
(ELF file)
Userspace module
(Wasm file)
Image push
Image pull
Linux host
ig

Official Gadgets

Inspektor Gadget

Advise: Recommend system configurations based on collected information

seccomp-profile, network-policies

Audit: Audit a subsystem

seccomp

Profile: Profile different subsystems

block-io, cpu

Snapshot: Take a snapshot of a subsystem and print it

process, socket

Top: Gather, sort and periodically report events according to a given criteria

file, tcp

Trace: Trace and print system events

bind, dns, exec, mount, oomkill, tcp{drop, retrans}, open, few more...

Artifact Hub

Inspektor Gadget

Artifact HUB | Search packages | DOCS | STATS | SIGN UP | SIGN IN

1 - 34 of 34 results for "gadget"

Sort: Last updated | Show: 60

Filters: KIND: Inspektor gadgets

FILTERS Reset

- Official
- Verified publishers
- CNCF

KIND

- Helm charts (3)
- Inspektor gadgets (34)
- Krew kubectl plugins (1)

CATEGORY

- Monitoring and logging (35)
- Networking (1)

LICENSE

- Apache-2.0 (3)

OTHERS

- Only operators
- Include deprecated

deadlock
Inspektor Gadget | Official gadgets
★ 1 | Inspektor gadget
Updated 8 hours ago
Version 0.38.0
use uprobe to trace pthread_mutex_lock and pthread_mutex_unlock in libc.so and detect potential deadlocks
Monitoring and logging

audit seccomp
Inspektor Gadget | Official gadgets
★ 0 | Inspektor gadget
Updated 8 hours ago
Version 0.38.0
Audit syscalls according to the seccomp profile
Monitoring and logging

fdpass
Inspektor Gadget | Official gadgets
★ 0 | Inspektor gadget
Updated 8 hours ago
Version 0.38.0
Trace file descriptor passing via a unix socket (SCM_RIGHTS)
Monitoring and logging

Inspektor Gadget

Demo

Inspektor Gadget

Using Inspektor Gadget

Modes of Operation

Inspektor Gadget

Linux host

- Ig binary
- Ig inside a container

Client-server setup

- Ig runs as a service inside the host
- We use a client called gadgetctl to control the service (via API call)

Kubernetes

- Ig is deployed via daemon set
- Kubectl-gadget plugin used to control the daemon set

Go library API

Gadget Instance Manifests

Inspektor Gadget

```yaml
# Trace DNS requests that are not successful (rcode!=Success) and are responses (qr==R) in all namespaces.
# Gadgets used:
# - trace_dns (https://www.inspektor-gadget.io/docs/latest/gadgets/trace_dns)
apiVersion: 1
kind: instance-spec
image: trace_dns:latest
name: failed-dns-requests-all
paramValues:
  # The following parameter is used to trace DNS requests in all namespaces
  # See: https://www.inspektor-gadget.io/docs/latest/spec/operators/kubemanager
  operator.KubeManager.all-namespaces: true
  # The following parameter is used to filter the DNS requests that are:
  #   1. Not successful (rcode!=Success)
  #   2. Responses only (qr==R)
  # See: https://www.inspektor-gadget.io/docs/latest/spec/operators/filter
  operator.filter.filter: rcode!=Success,qr==R
  # The following parameter is used to display the following fields in the output:
  #   1. Namespace of the pod
  #   2. Name of the pod
  #   3. Source endpoint of the DNS request
  #   4. Destination endpoint of the DNS request
  #   5. Name in the DNS request
  #   6. Response code of the DNS request
  # See: https://www.inspektor-gadget.io/docs/latest/spec/operators/cli
  operator.cli.fields: k8s.namespace,k8s.podname,src,dst,name,rcode
```

Exporting Options

Inspektor Gadget

Raw Data

You can consume via the CLI or export to a json file

Prometheus

You can export to Prometheus metrics

OpenTelemetry

We support logs and metrics through OTel

Inspektor Gadget

Demo

System Level Challenges

Inspektor Gadget

Debugging

- OOM Kill events
  - Trace oomkill
- DNS queries not resolving
  - Trace DNS
- Deadlocks (when a process doesn't proceed due to another process using resources
  - Deadlock

Monitoring

- Identifying source of high latency
  - Visibility into CPU usage
    - Profile cpu
  - Stress on the system from a specific container
    - Profile blockio
  - Process from a container excessively reading/writing to a file
    - Top file

Security

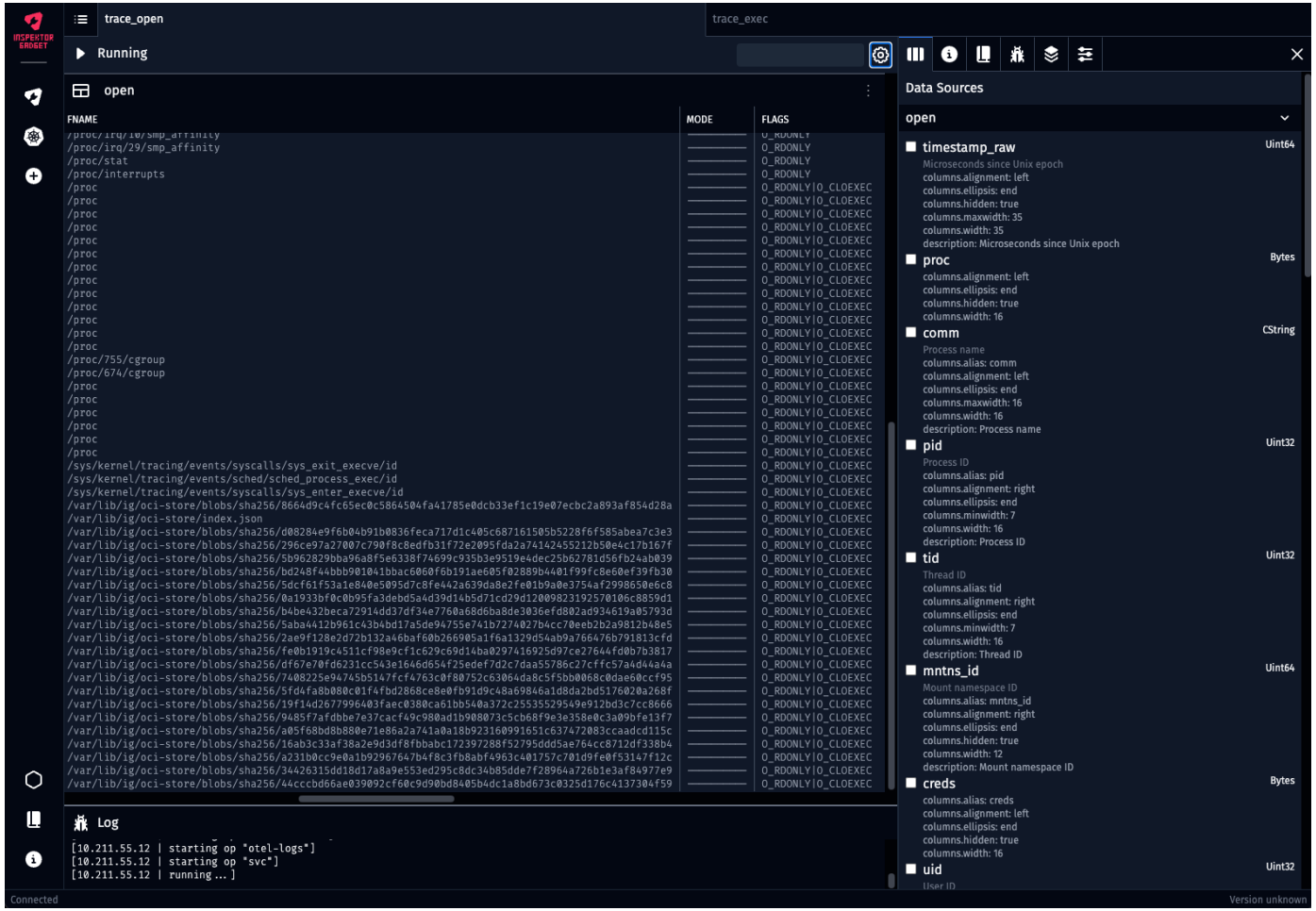
- Identifying potential malicious activity
  - Shell open on a K8s cluster
    - Trace exec
  - Validating binary changes in container image
    - Trace exec
  - Monitoring processes accessing the filesystem
    - Trace open
- Seccomp profiles generation
  - Advise seccomp profile

Inspektor Gadget

What’s Next & Close Out

What’s next?

- 1.0
- New Graphical User Interface (App)
- Gadgets around enhanced CPU profiling and off CPU
- Understand community priorities
- Proper documentation for all this ;-)



Call to Action

Inspektor Gadget

Think about how eBPF could be used to enhance the projects you're working on and see if we have a gadget that could help you!

Thank you!

Inspektor Gadget

Web: **inspektor-gadget.io**

Slack: **#inspektor-gadget on the Kubernetes Slack**

Github: **github.com/inspektor-gadget**

Fill out our survey!

Appendix

Inspektor Gadget

Some examples of IG use cases ☺

ARMO

ARMO and the Opensource project Kubescape use IG to enhance detecting vulnerabilities in containers

MS Defender

Inspektor Gadget is used in MS Defender for Containers to collect security events, generate insights and real-time threat detection alerts.

Amazon EKS

Amazon EKS Users leverage Inspektor Gadget to inspect their Kubernetes environment with eBPF tools

Source: Empowering Kubernetes Observability with eBPF on Amazon EKS | Containers
CNCF On demand webinar: eBPF-based Kubernetes Security | CNCF