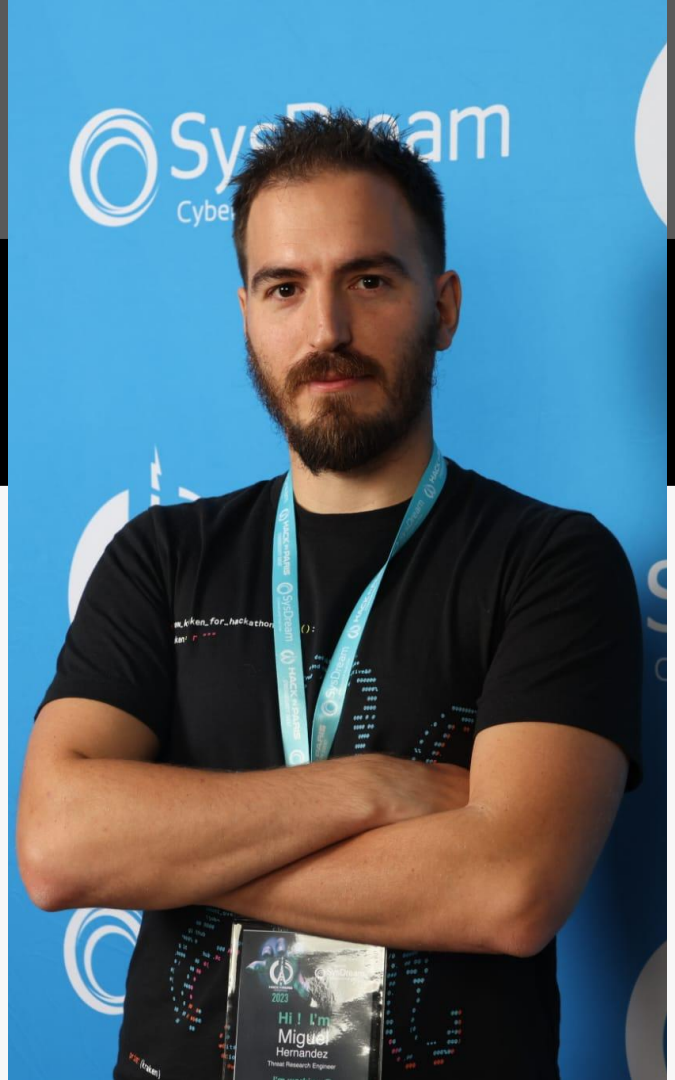
Taller: Falco 101
Detecting threats in real time

Falco

SECADMIN2023
Academy Edition



Miguel Hernández

Staff Threat Security Engineer at Sysdig

Miguel Hernández, is a lifelong learner with a passion for innovation. Over the past decade, Miguel has honed his expertise in security research, leaving his mark at prominent tech companies and fostering a spirit of collaboration through personal open-source initiatives.

Miguel has been a featured speaker at cybersecurity conferences such as HITB, HIP, CCN-CERT, RootedCon, TheStandoff, and Codemotion.

SECADMIN2023
Academy Edition

Vicente J. Jiménez

Open source evangelist

Vicente J. Jiménez Miras has a background as an infrastructure engineer and open source trainer. Being a digital nomad since early in his career, he's lived or worked in almost a hundred different locations. He also shares a passion for food, especially Indian cuisine, although, as a Spaniard, he's very proud of his tortilla española.

In 2021, after 4 years working as an instructor for Red Hat, he joined Sysdig to continue his work educating technical teams and creating awareness of open source cybersecurity.

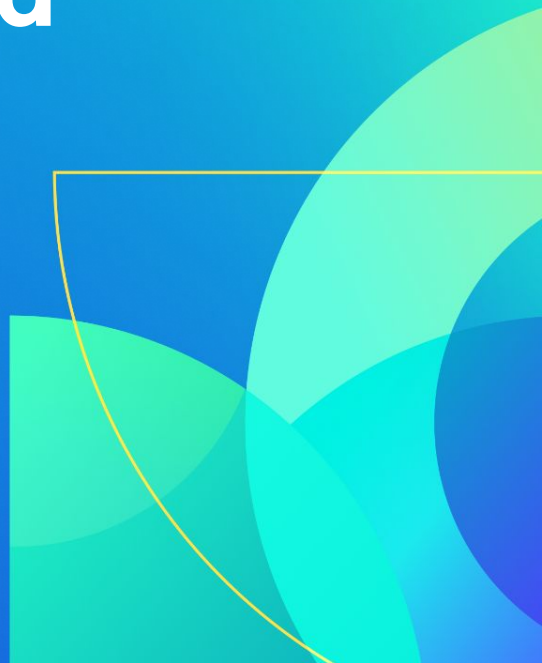
Logistics

- Workshop: https://play.instruqt.com/sysdig/invite/dkgkhwknsisr
- Length: 120min
- QR-Code:

Agenda

- Cloud Threat Detection and Runtime Security
- Falco engine
- Falco ecosystem
- Closing remarks

Cloud Threat Detection and Runtime Security



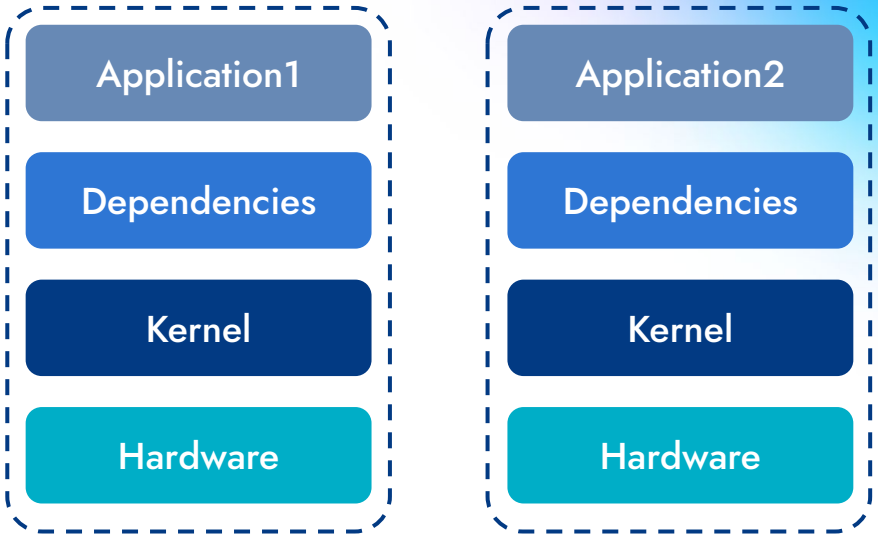
sysdig

Microservices, Containers, and Kubernetes

sysdig

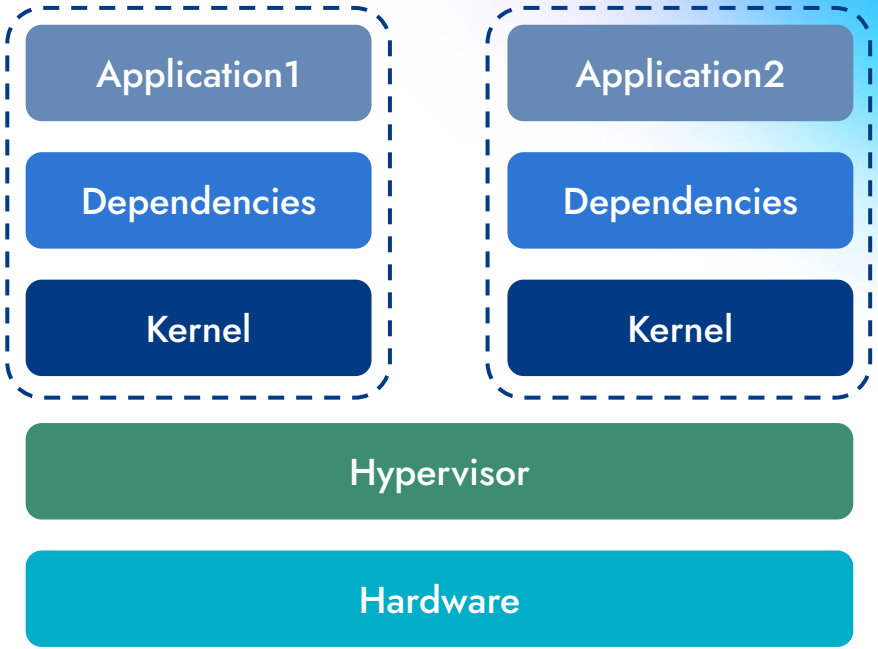
Once upon a time... dedicated servers

- Deployments took months
- Low utilization
- Not portable
- Hard to replicate



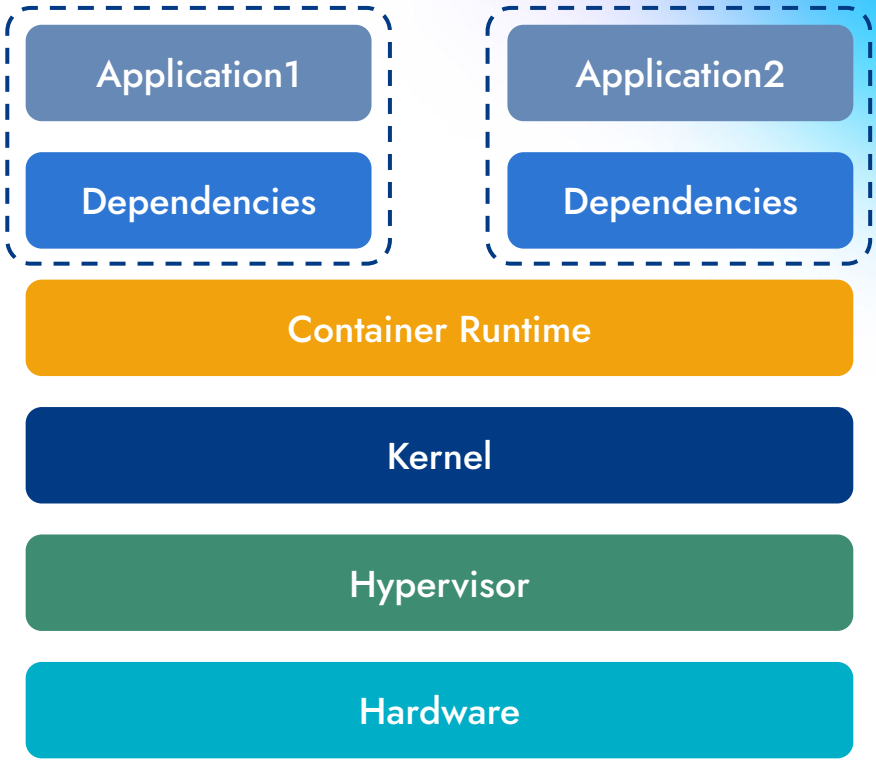
Virtualization

- Improved utilization
- Deployment took days
- Redundant kernel
- All bundled together
- Low app <–> kernel isolation



Containers

- Lightweight
- Stand-alone
- Resource efficient
- Portable execution packages
- Deploy within minutes/seconds



Rise of microservices and containers

- Need for greater flexibility, agility, scalability, and reliability
  - loosely coupled, fine-grained components.
- Microservices make it easier to:
  - build and maintain complex applications
  - scale individual services as needed, not scaling the entire system
  - adopt new technologies and tools as they become available.
  - implement changes and updates, without affecting the entire system
- Containers are a perfect fit.
- Cloud providers heavily support containers and microservices
  - easier and more cost-effective to build and scale applications.
- Containers are everywhere, but managing them at scale is challenging!

Kubernetes wins

- Kubernetes is an open-source container orchestration system
  - for automating software deployment, scaling, and management.
- Over 60% of orgs have adopted it already in 2022 (Statista research).
- Originally designed by Google,
  - the project is now maintained by the Cloud Native Computing Foundation.

CNCF - Cloud Native Computing Foundation

- Founded in 2015 to advance container technology and align the tech industry around its evolution.
- Announced alongside Kubernetes 1.0
- Part of the Linux Foundation.
- Projects have a maturity level: Sandbox, Incubated, and Graduated

What is Cloud Native?

- Cloud computing environments
- Highly scalable, flexible, and resilient applications (quick updates)
- Modern tools and techniques that support development on cloud infrastructure.
- Fast and frequent changes to applications without impacting service delivery

Cloud Threat Detection and Runtime Security

sysdig

Once, there was a perimeter

You had a perimeter **guarded by a firewall**

Detecting intrusions was your breach indicator

Now, there is no perimeter in the cloud

- Cloud providers own external connections
- Cloud is exposed to the outside world
- You need to control access to services your team uses
- You need to detect unusual activity

Why Runtime Security?

Runtime Security vs

- Runtime behav
- Strict inventori
- “Expected atta
- It is not vs, but +

, CSPM)

Prepare for the unexpected:

- Runtime Nature of Attacks
- Zero-day Vulnerabilities

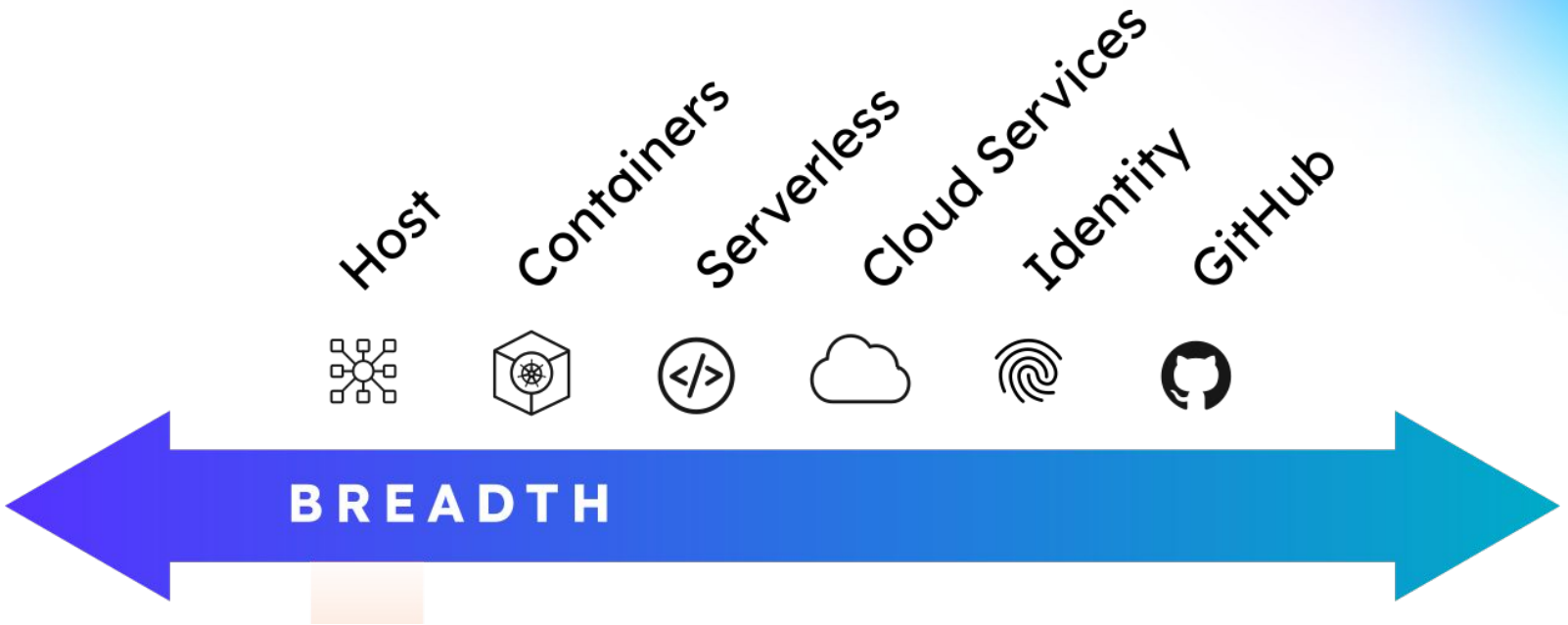
Example

- Log4j vulnerability was introduced in 2013.
- Reported and fixed: 2022.

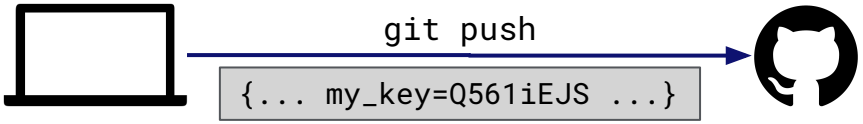
Without a perimeter, a security camera is more important than a good lock

- Watch for changes that create security gaps
- Identify intruders and suspicious insider behavior
- Send an alert and take immediate action

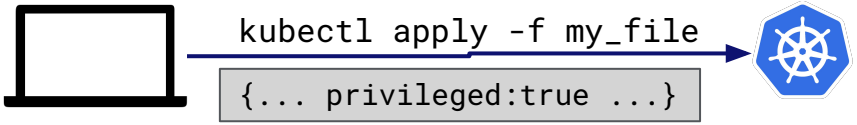
End-to-End Detection



Suspicious Activities

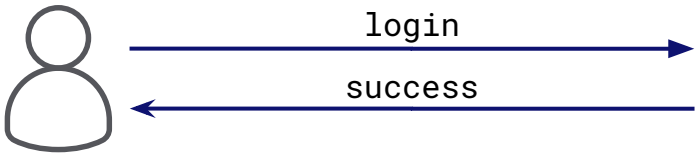


Secret pushed into a public repository.

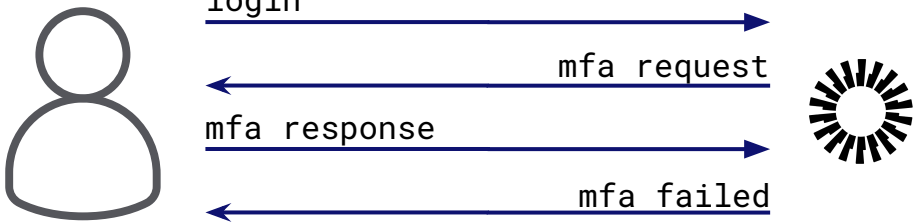


Create privileged pod.

Suspicious Activities

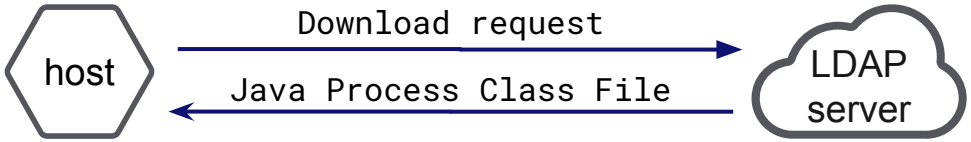


Console login without MFA



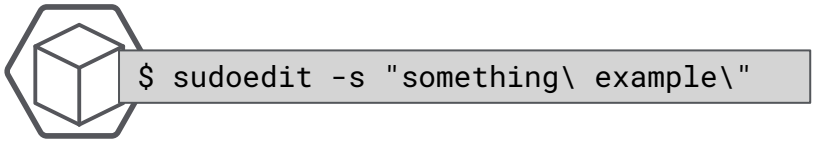
Too many failed MFA in the last 5m

Suspicious Activities



Java Process Class File Download

CVE-2021-4422



Sudo Potential Privilege Escalation

CVE-2021-3156

Falco

sysdig

Falco

Falco is an **open source** runtime security solution for **threat detection** across **Kubernetes**, containers, hosts and **the cloud**.

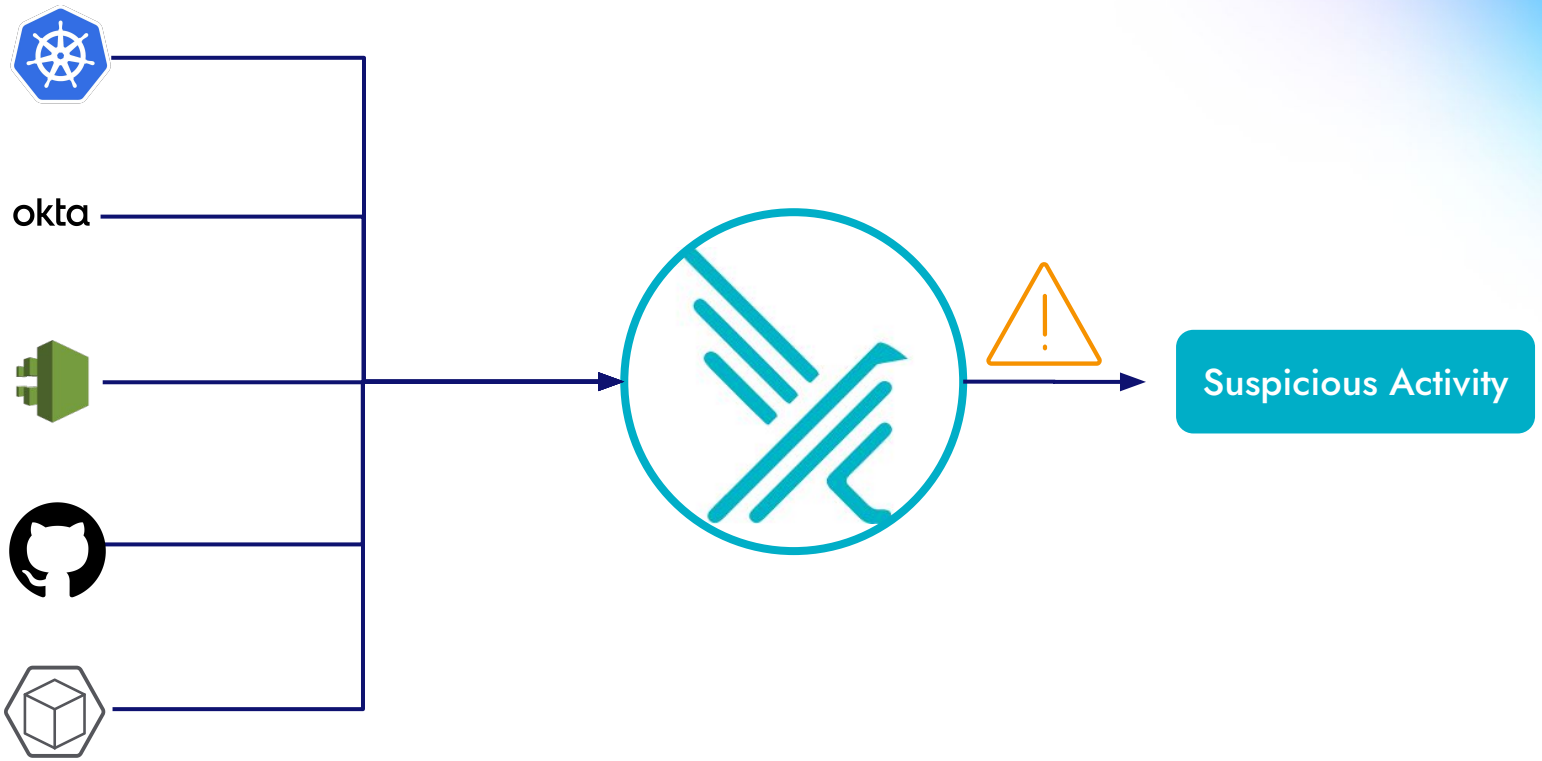
CNCF Incubation-Level Project
(the graduation will be official in December 2023)


6.7k


60M+ pulls


CLOUD NATIVE
COMPUTING FOUNDATION

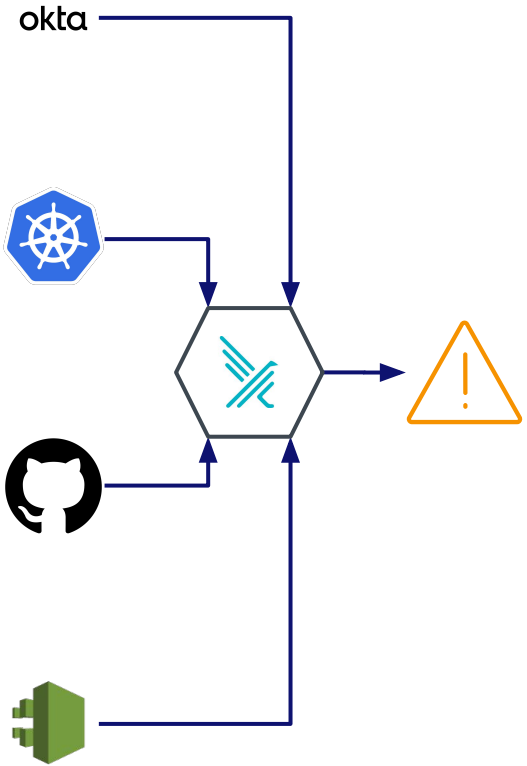
Falco overview

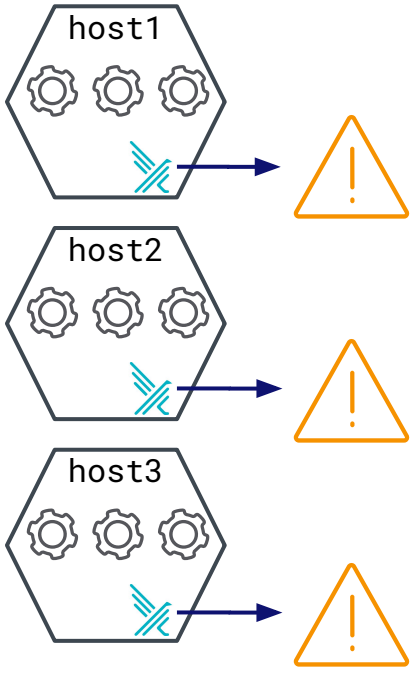


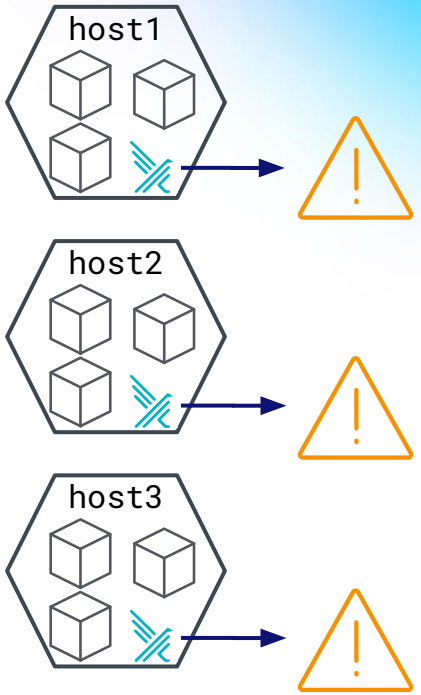
Falco

```
2022-04-07T12:51:08: Notice A shell was spawned in a container with an attached terminal (user=root
user_loginuid=-1 elastic_borg (id=a10bd3b1b2a8) shell=bash parent=<NA> cmdline=bash terminal=34816
container_id=a10bd3b1b2a8 image=ubuntu)
2022-04-07T12:51:41: Warning Netcat runs inside container that allows remote code execution
(user=root user_loginuid=-1 command=nc -e container_id=a10bd3b1b2a8 container_name=elastic_borg
image=ubuntu:latest)
```

Falco deployment examples



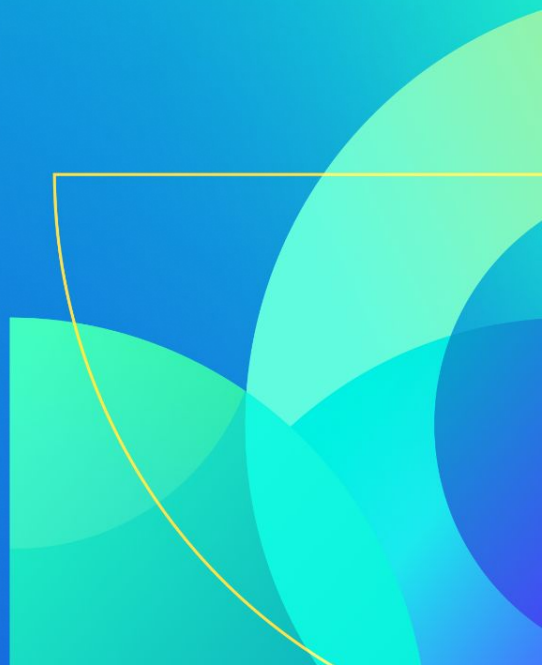




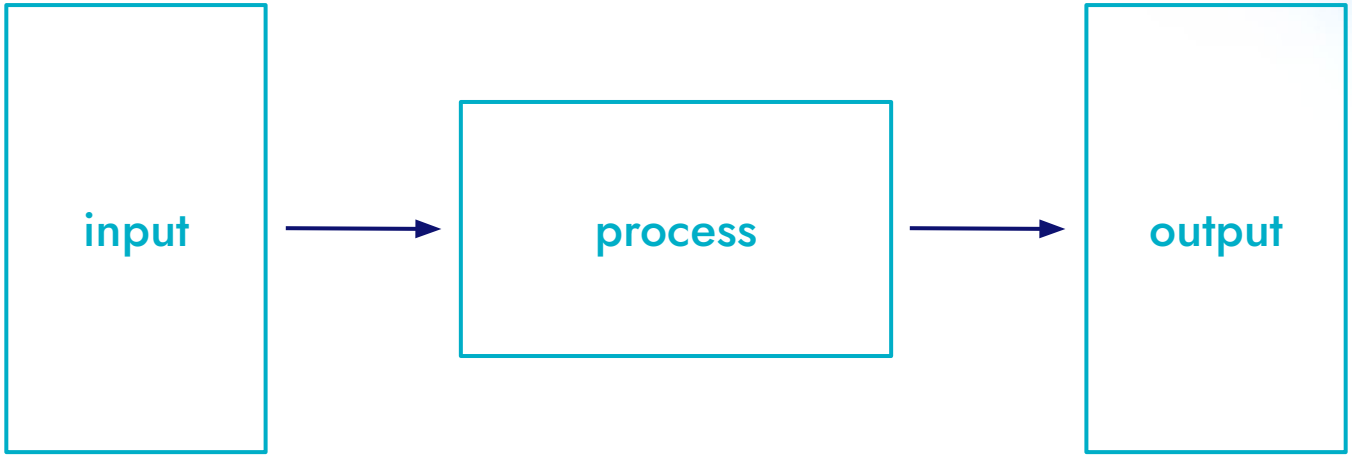
Set up Falco

Lab time!

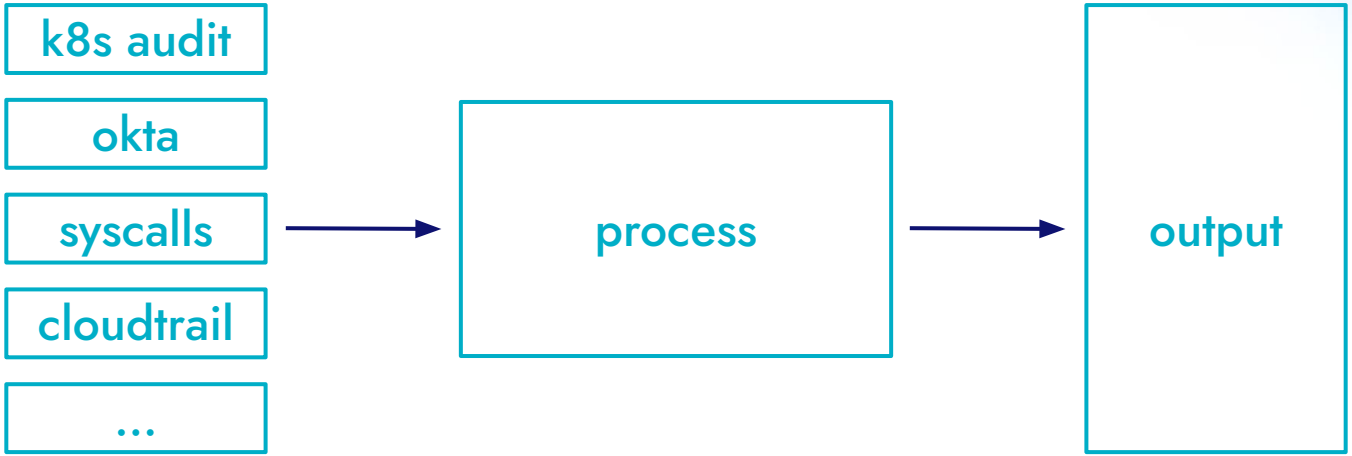
Falco engine



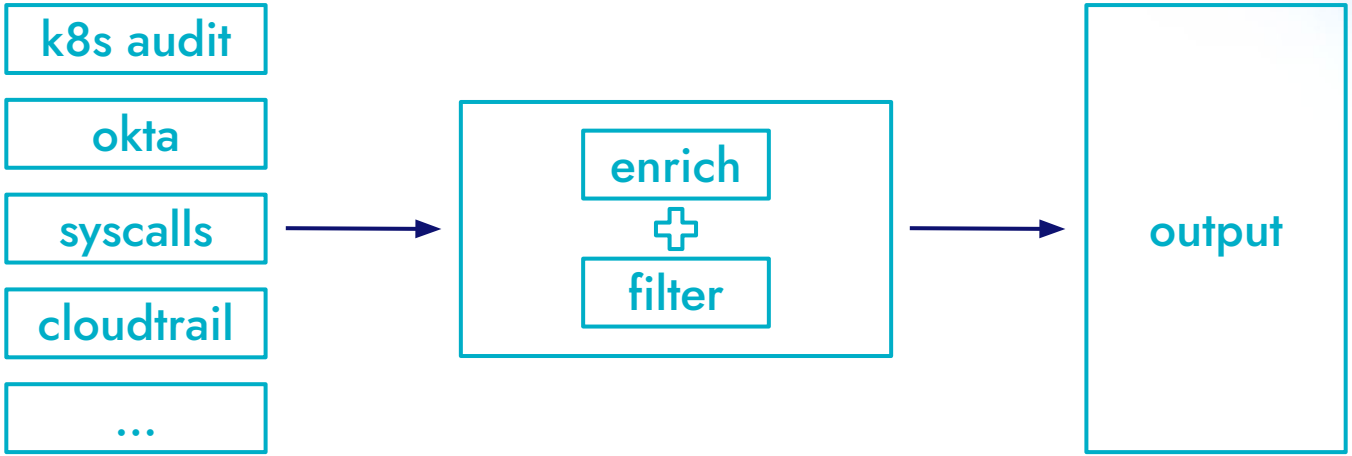
Falco architecture



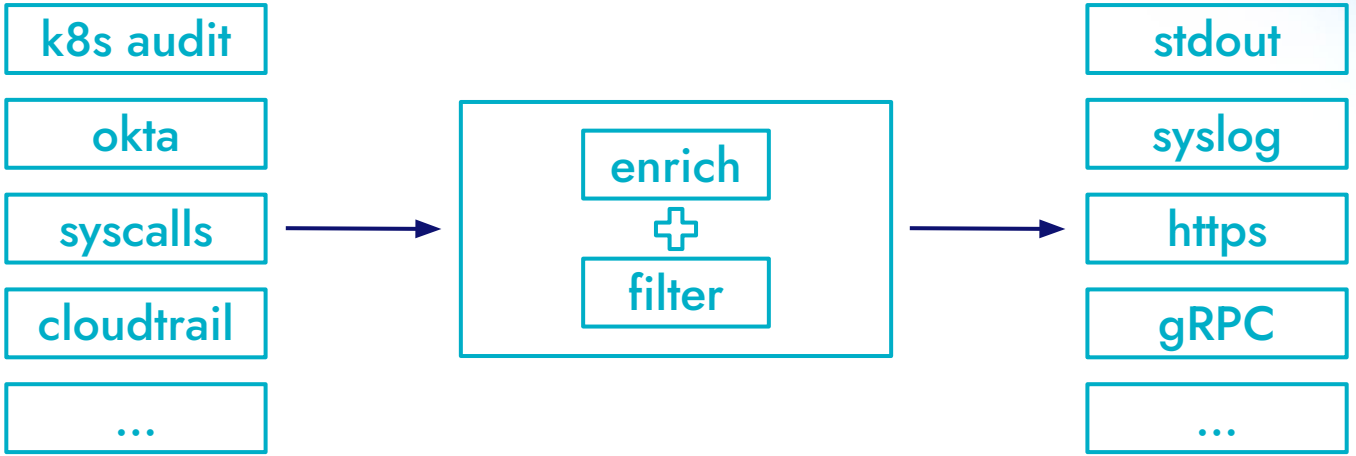
Falco architecture



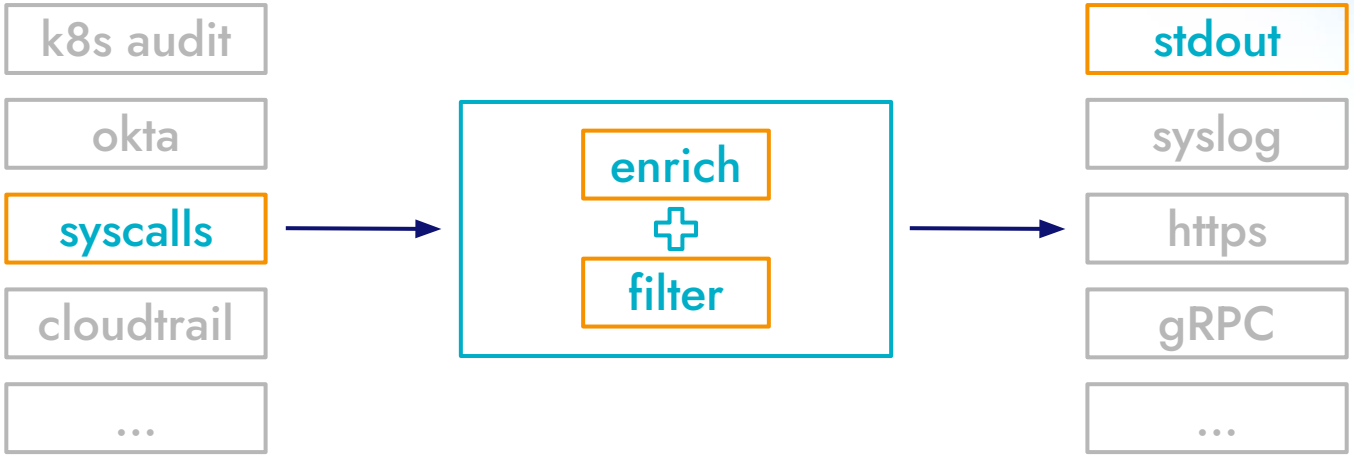
Falco architecture



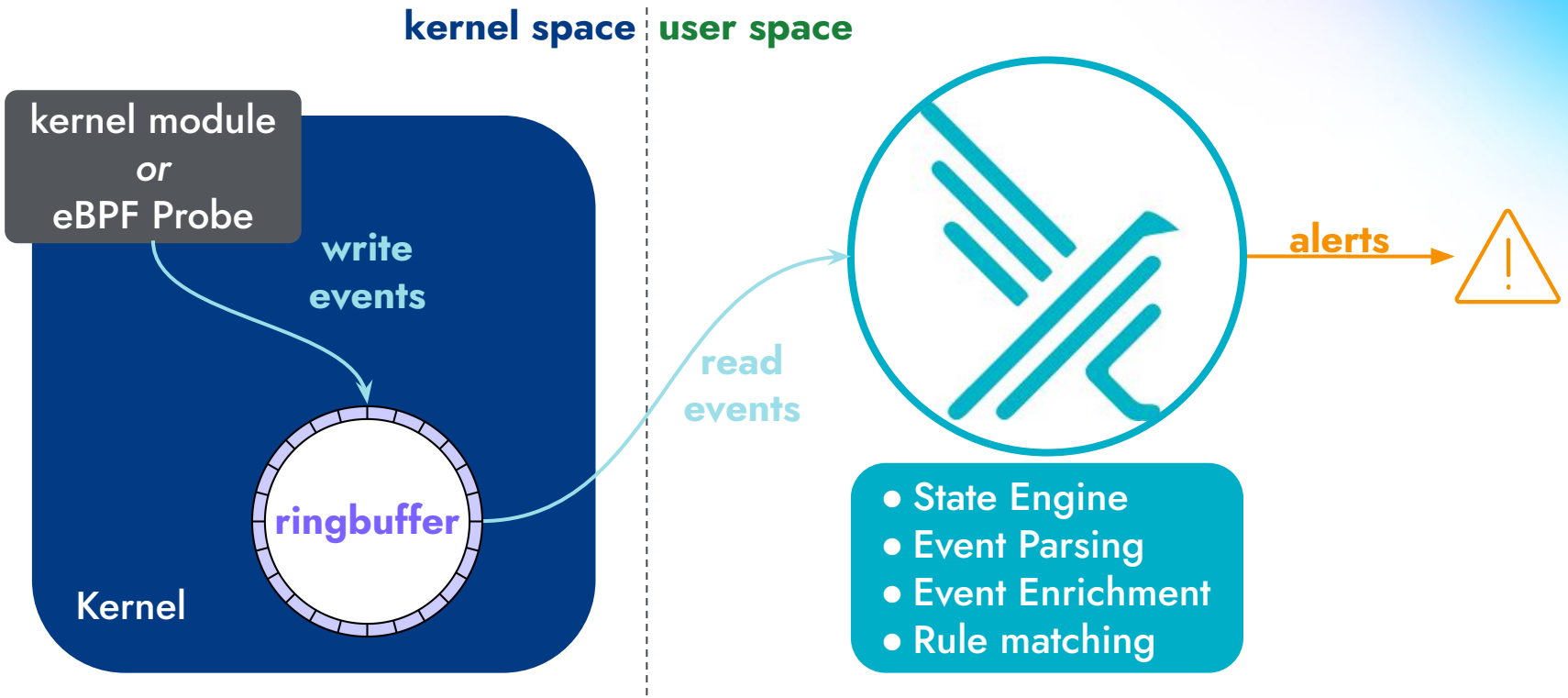
Falco architecture



Falco architecture



Falco Syscall Architecture Overview



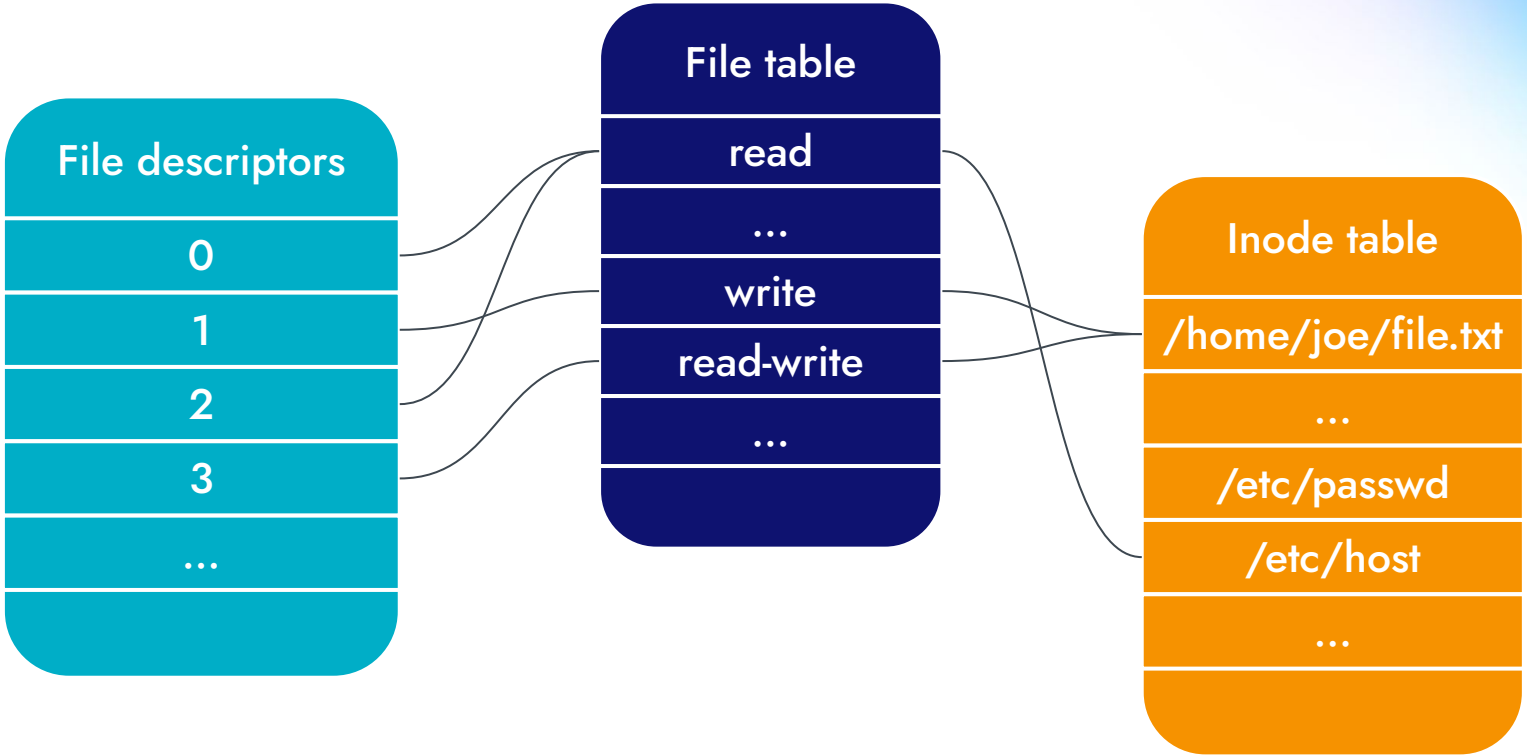
eBPF

- Extended Berkeley Packet Filter.
- Extend the kernel capabilities safely
  - without changing kernel source co
  - without loading kernel modules.

Kernel instrumentation made simple!

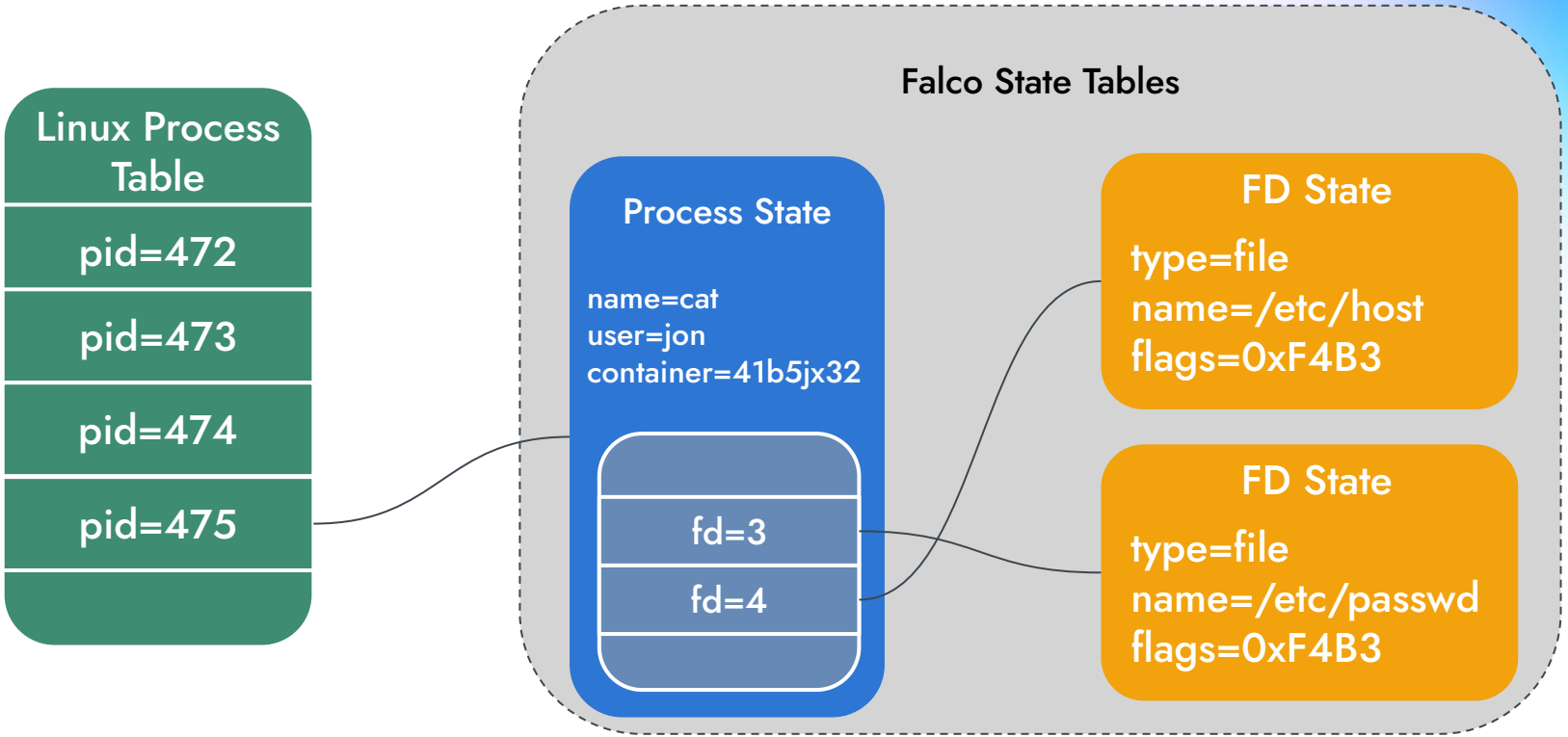
Modern, low-overhead, production-ready observability for monitoring and security in containers and Linux systems.

Navigating the Linux File System



Adapted from https://en.wikipedia.org/wiki/File_descriptor

Falco State



Adapted from "Practical Cloud Native Security with Falco"

Falco default rules

- Falco ships with more than 80 default rules:
  - Privilege escalation
  - R/W to sensitive directories
  - Executing shell
  - Execute SSH binaries
  - Mutating binaries
  - Creating symlinks
  - …

Falco syntax

```
- rule: Terminal shell in container
  desc: A shell has been spawned in a container.
  condition: >
    spawned_process and container and shell_procs
  output: >
    A shell was spawned in a container (user=%user.name
user_loginuid=%user.loginuid %container.info shell=%proc.name
parent=%proc.pname cmdline=%proc.cmdline container_id=%container.id)
```

https://falco.org/docs/reference/rules/rule-fields/

Falco syntax

```
- rule: Terminal shell in container
  desc: A shell has been spawned in
  condition: >
    spawned_process and container ar
  output: >
    A shell was spawned in a contain
user_loginuid=%user.loginuid %contai
parent=%proc.pname cmdline=%proc.cmd
```

```
- list: shell_binaries
  items: [ash, bash, csh, ksh, sh,
tcsh, zsh, dash]

- macro: shell_procs
  condition: proc.name in
(shell_binaries)

- macro: container
  condition: (container.id != host)

- macro: spawned_process
  condition: >
        evt.type in (execve, execveat)
        and evt.dir=<
```

https://falco.org/docs/reference/rules/supported-fields/

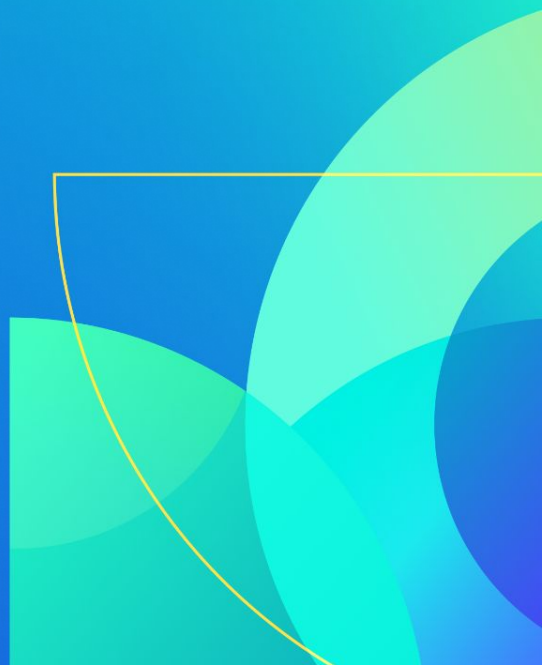
Macros and Lists

- **Macros** allow you to define conditions and reuse them wherever you want.
- **Lists** help you organize your rules files with naming and segmentation.
- They have four main benefits:
  - code reuse
  - avoid long strings of conditions
  - rules are easier to understand
  - easier to extend

Falco default rules

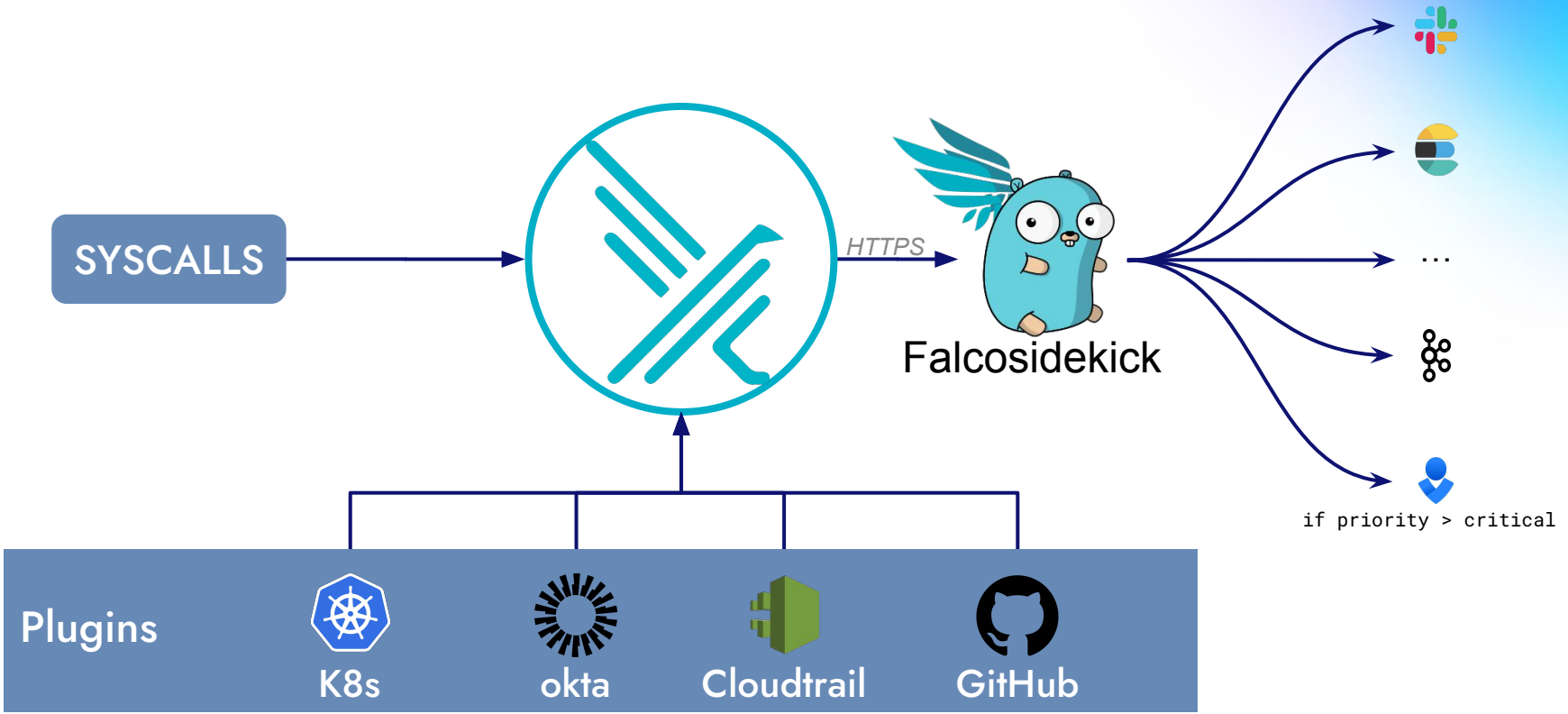
Lab time!

Falco ecosystem

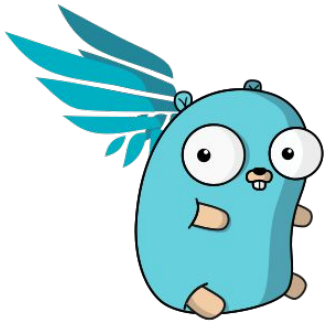


sysdig

Falco ecosystem



Falcosidekick



chat

logs

queue/streaming

faas

metrics

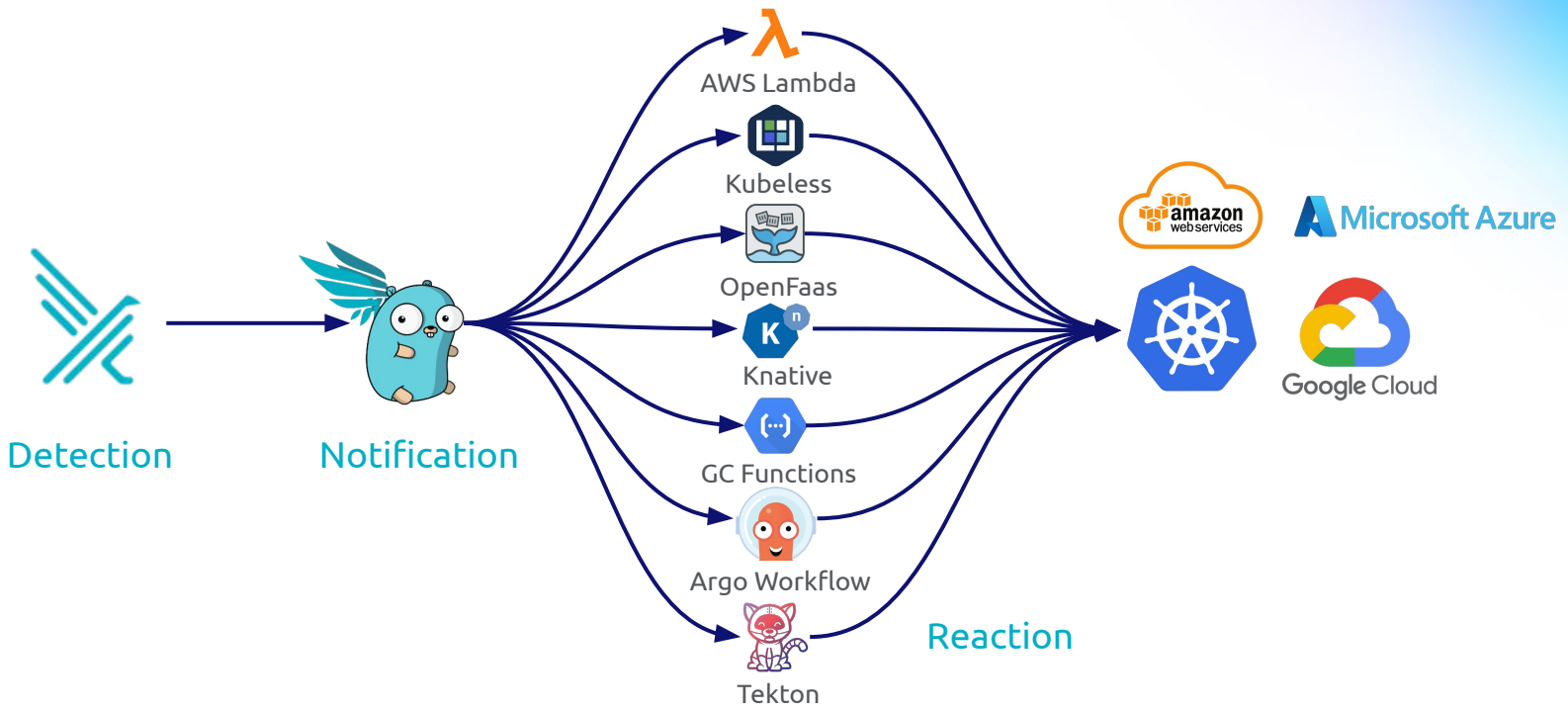
alerting

storage

and more ...

https://github.com/falcosecurity/falcosidekick

React to events



Falcosidekick install

Standalone:

```
helm repo add falcosecurity https://falcosecurity.github.io/charts
helm repo update

helm install falcosidekick -n falco --set config.debug=true falcosecurity/falcosidekick
```

With Falco:

```
helm install falco -n falco falcosecurity/falco \
  --set falcosidekick.enabled=true \
  --set falcosidekick.webui.enabled=true
```

Event generator

- Generates a variety of suspect actions that are detected by Falco rulesets.
- Good to test Falco rulesets:
  - syscalls
  - kubernetes audit
- Run it within Docker or Kubernetes,
  - as some commands might alter your system.

```
docker run -it --rm falcosecurity/event-generator run
```

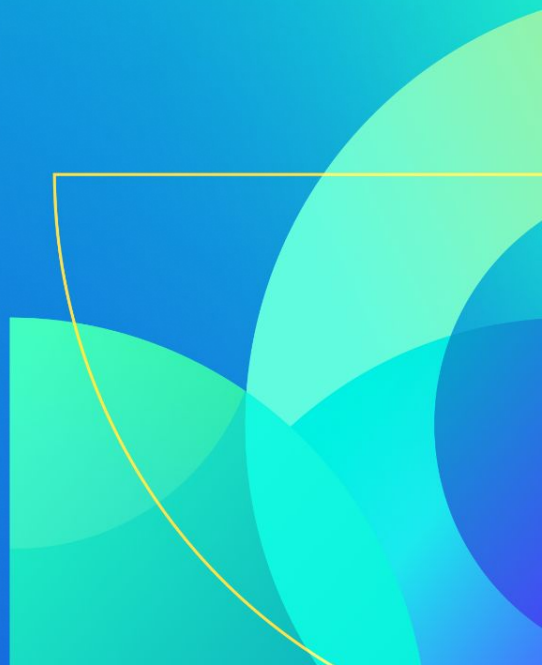
```
helm install event-generator falcosecurity/event-generator \
  --namespace event-generator \
  --create-namespace \
  --set config.actions=""
```

https://github.com/falcosecurity/event-generator

Falco ecosystem

Lab time!

Closing remarks



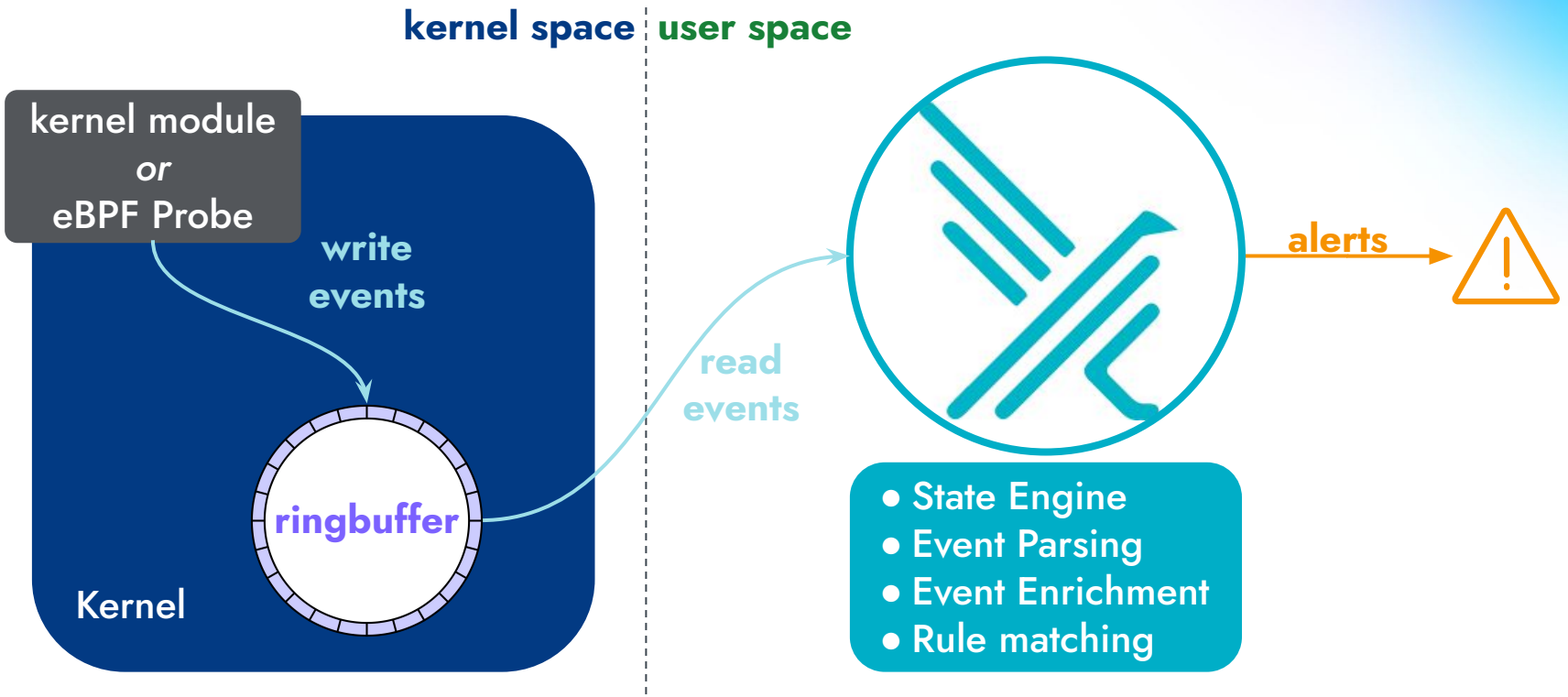
sysdig

Runtime security

Guard perimeter

Detect unusual activities

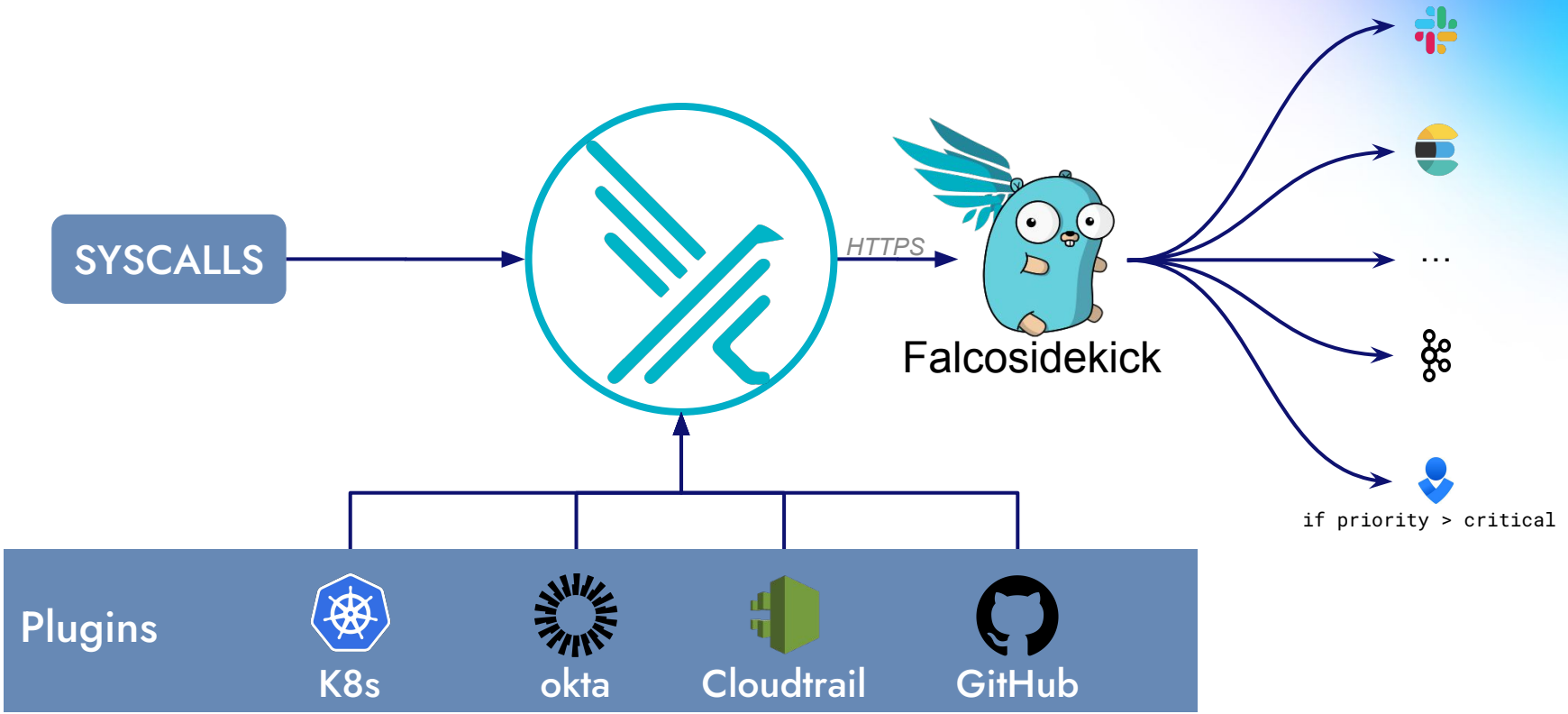
Falco Syscall Architecture Overview



Falco rules

```
- rule: Terminal shell in container
  desc: A shell has been spawned in a container.
  condition: >
    spawned_process and container and shell_procs
  output: >
    A shell was spawned in a container (user=%user.name
user_loginuid=%user.loginuid %container.info shell=%proc.name
parent=%proc.pname cmdline=%proc.cmdline container_id=%container.id)
  priority: WARNING
  tags: [container, shell, mitre_execution]
```

Falco ecosystem



Reference

"Practical Cloud Native Security with Falco", O'Reilly book

https://falco.org/docs

https://falco.org/training

https://falco.org/blog/extend-falco-outputs-with-falcosidekick/

Survey

https://bit.ly/falcosurveypdx