RUNTIME THREAT DETECTION WITH FALCO

For Kubernetes and CNCF Tampere Meetup on 23 May 2022

Prepared by Jarno Virtanen

jarno.virtanen@fraktal.fi

www.fraktal.fi

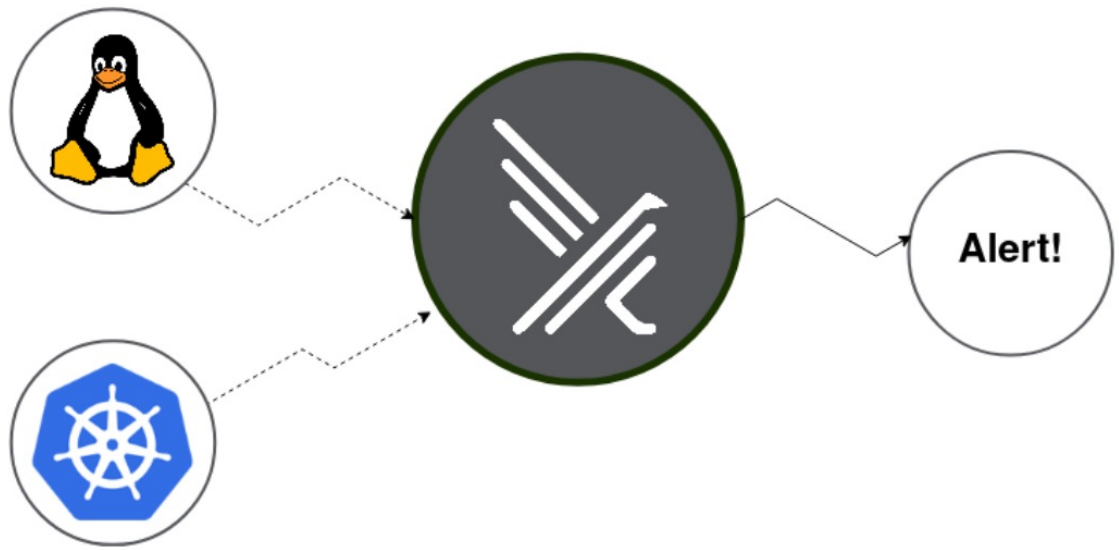
20+ years of dev, last year of CYBER at Fraktal

What the h*ck is Falco?

Falco, the cloud-native runtime security project, is the de facto **Kubernetes threat detection engine**

Falco is the first runtime security project to join CNCF as an incubation-level project. Falco acts as a security camera detecting unexpected behavior, intrusions, and data theft in real time.

Created by sysdig

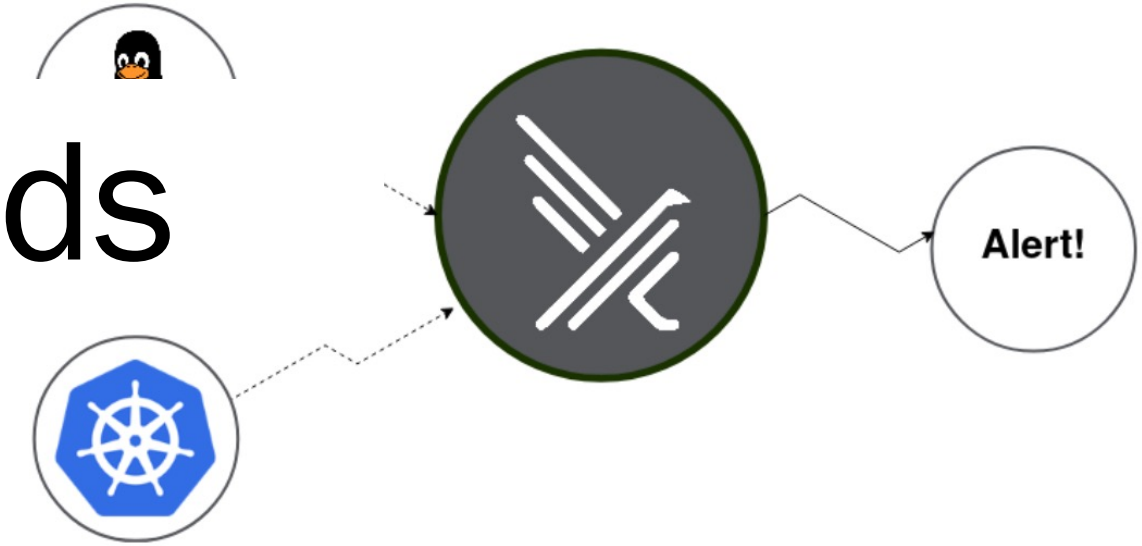


Falco, the cloud-native runtime security project, is
the d

Falco
incub
unexpected behavior, intrusions, and data theft in real time.

Created by sysdig

A lot of buzzwords



Why Falco?

Strengthen container security

The flexible rules engine allows you to describe any type of host or container behavior or activity.

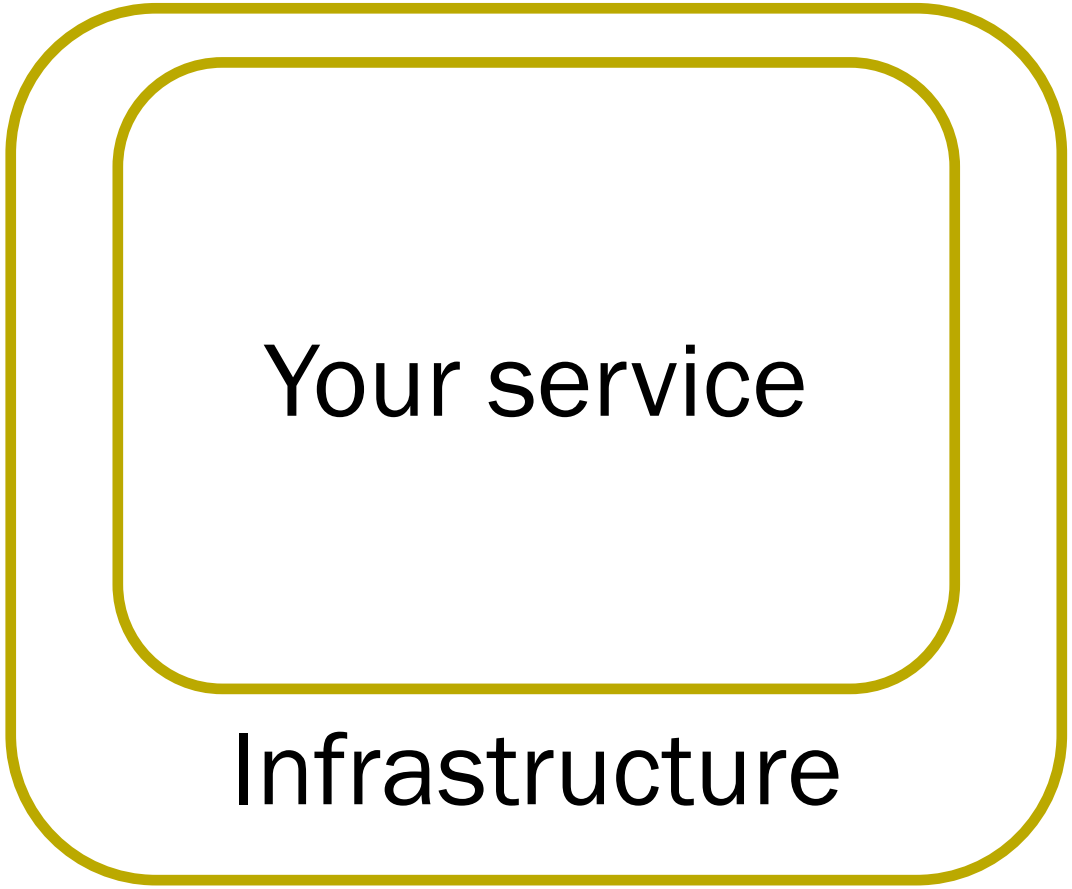
Reduce risk via immediate alerts

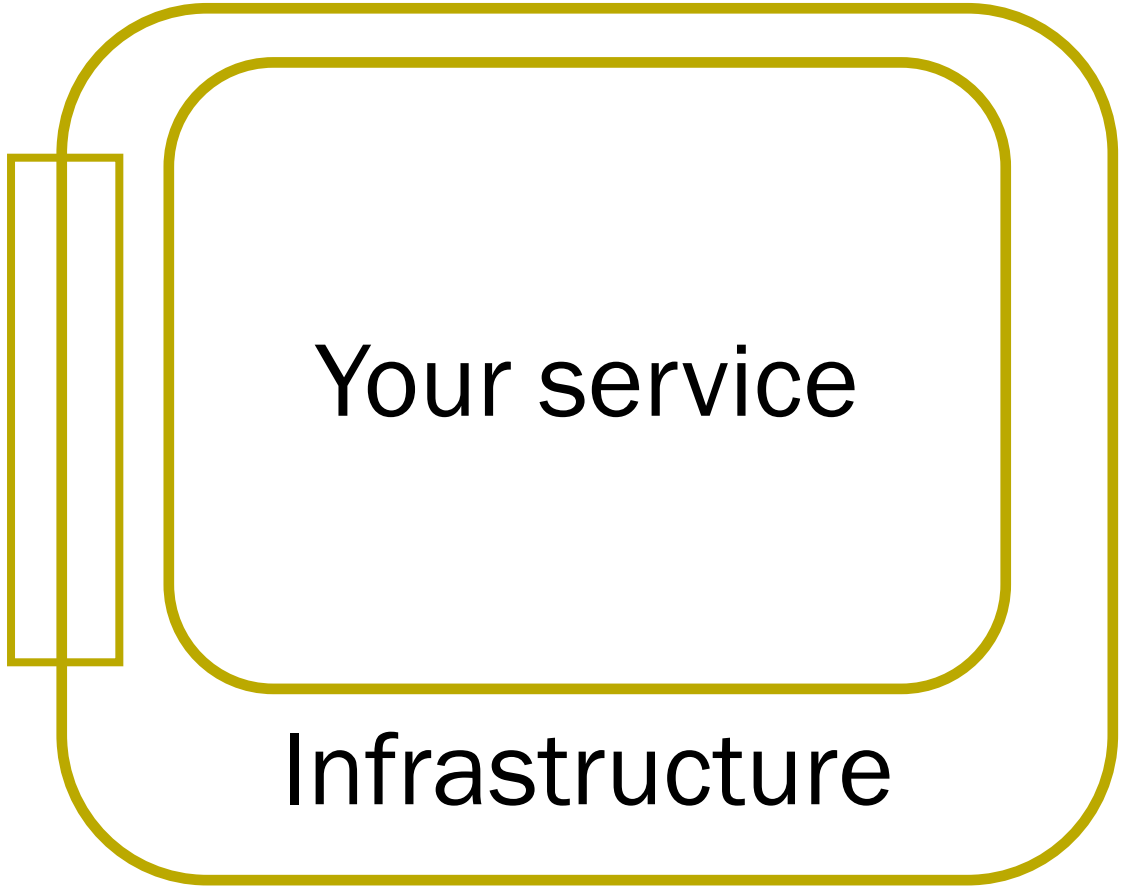
You can immediately respond to policy violation alerts and integrate Falco within your response workflows.

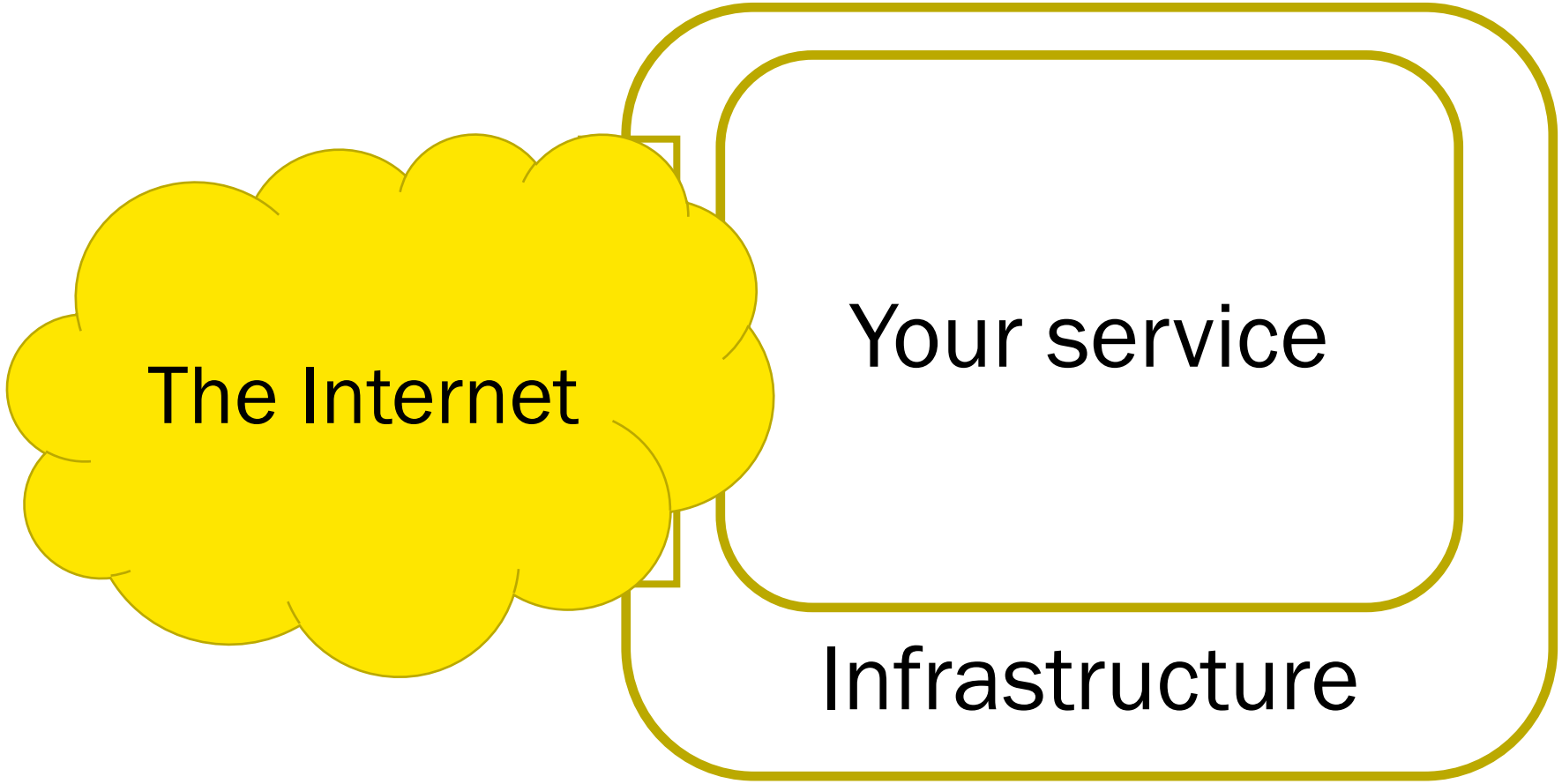
Leverage most current detection rules

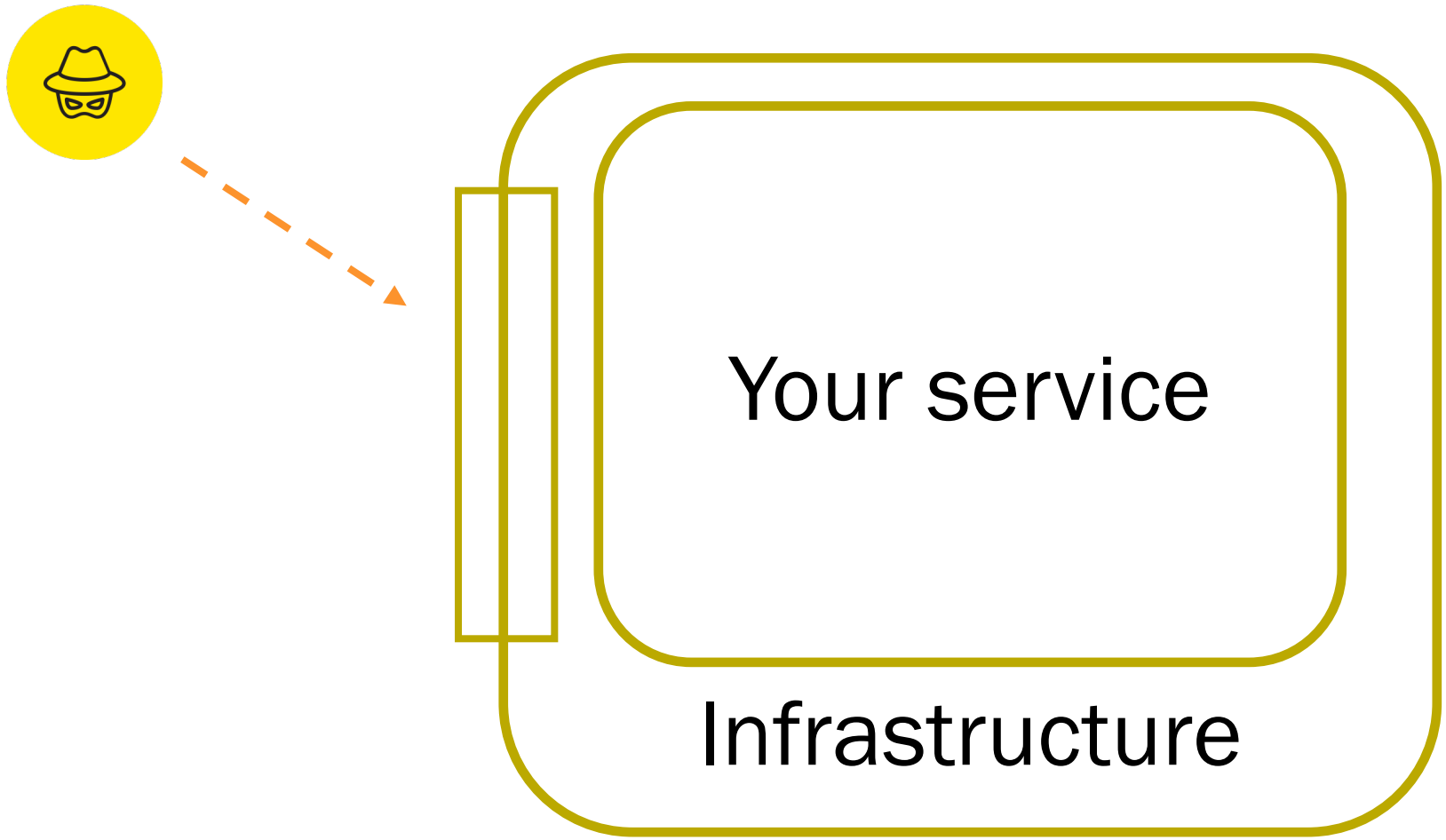
Falco out-of-the box rules alert on malicious activity and CVE exploits.

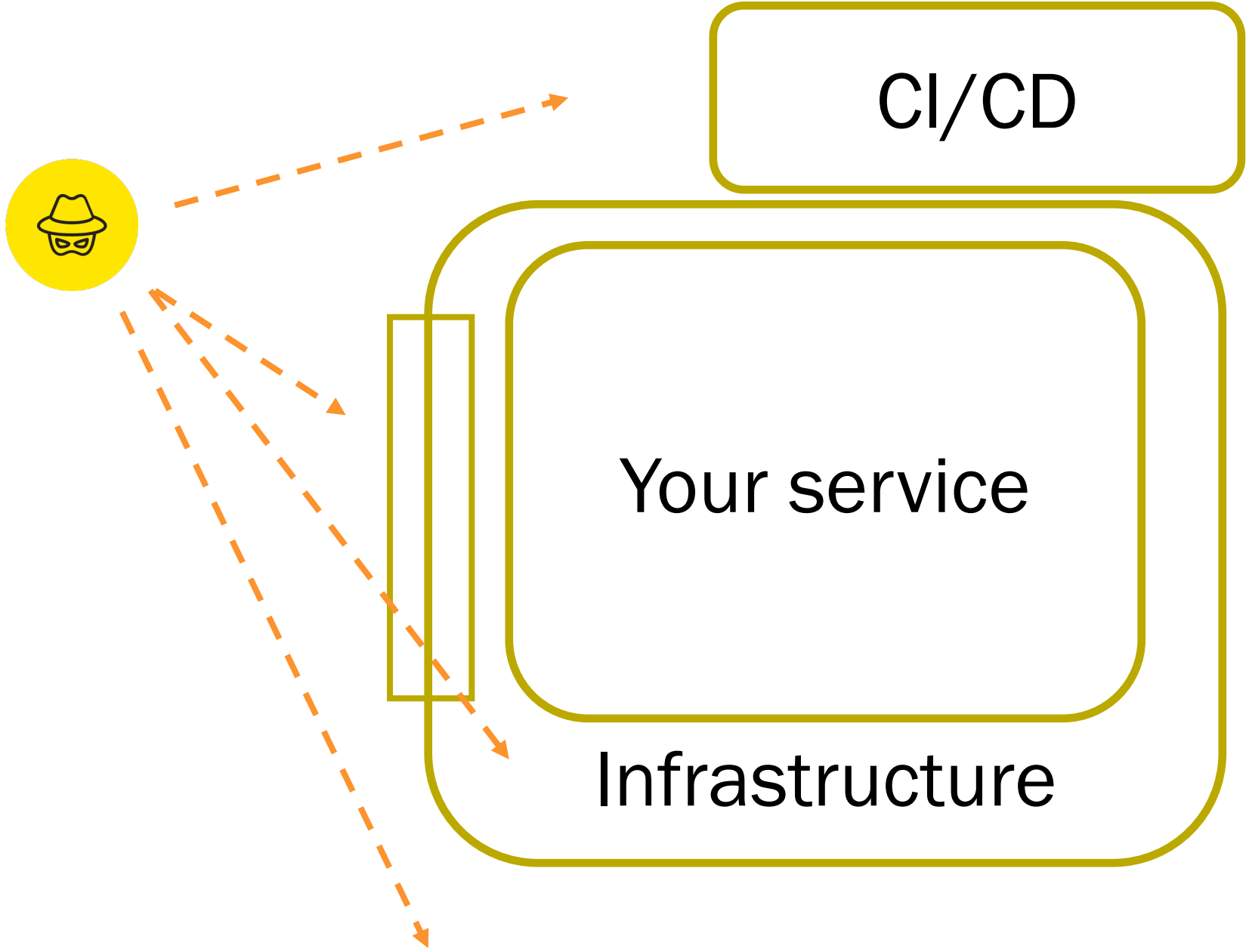
But seriously, *why*?

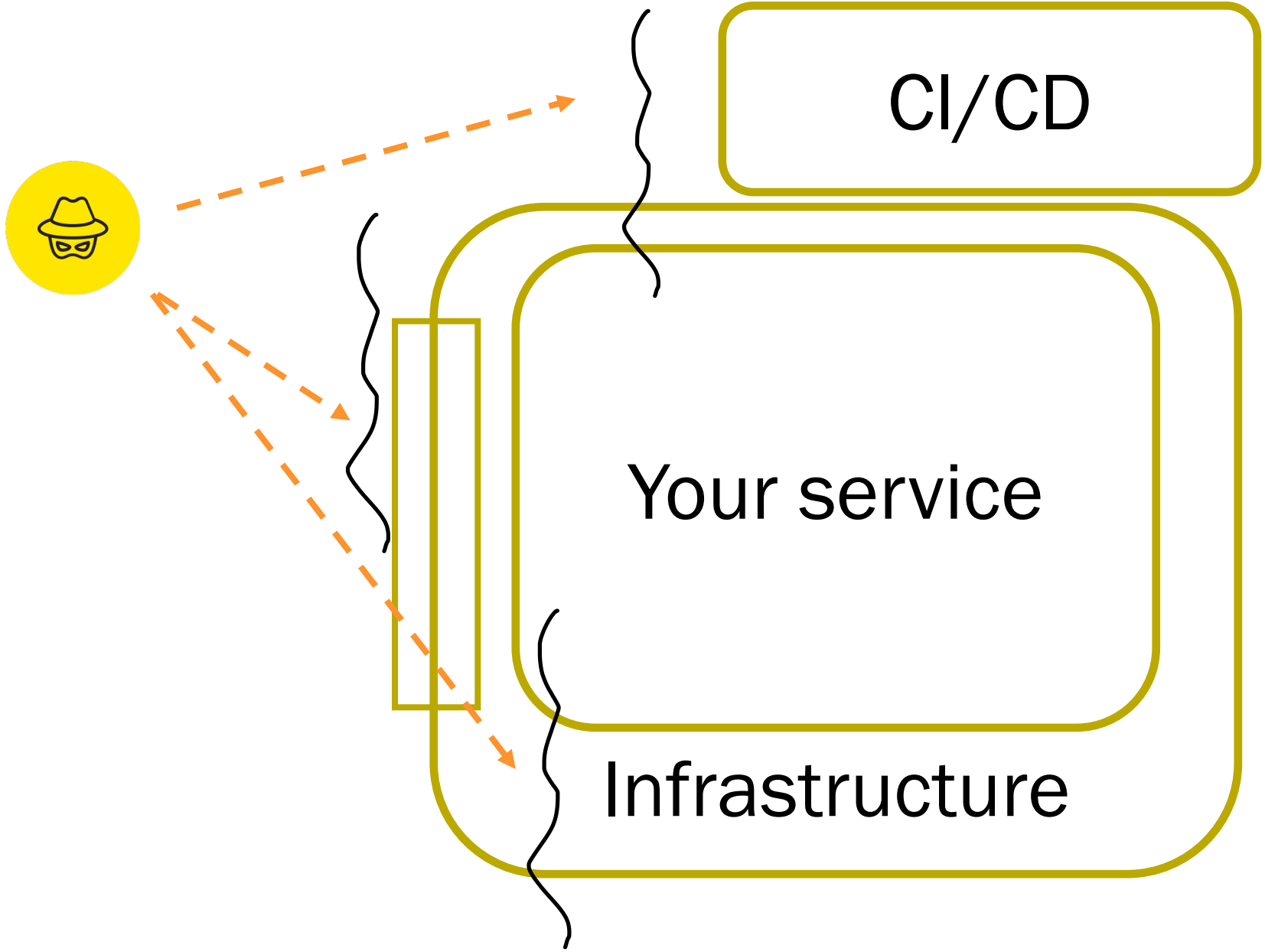

Your service
Infrastructure


Your service
Infrastructure


The Internet
Your service
Infrastructure


Your service
Infrastructure


CI/CD
Your service
Infrastructure


CI/CD
Your service
Infrastructure

How watertight are these defenses?

Well, it depends

The scale of the level of sophistication and determination of the attackers

Automated scripts

State level actors

The past few years have provided us with a number of high profile hacks and data breaches. In 2010 Google famously announced that they were hacked and put out details on the compromise (later dubbed the Aurora incident). In the months that followed, it became clear that google were not the only Aurora victims. Companies in almost every sector from DuPont to Disney were also breached (but were less forthcoming on the details).

Why wasn't yours?

Sadly two of the likeliest answers to this question are equally uncomfortable.

a) you haven't been compromised (yet) because people haven't bothered

b) your company has been compromised and you just don't know it

“… your cyber systems continue to function … not due to the expertise of your security staff but solely due to the sufferance of your opponents.” – Brian Snow

on sufferance formal

with unwilling permission:

0-day vulnerabilities, unpatched infra, phishing, novel supply chain attacks

Why wasn't yours?

Sadly two of the likeliest answers to this question are equally uncomfortable.

a) you haven't been compromised (yet) because people haven't bothered

b) your company has been compromised and you just don't know it

RADWARE CYBERSECURITY CYBERSAFETY

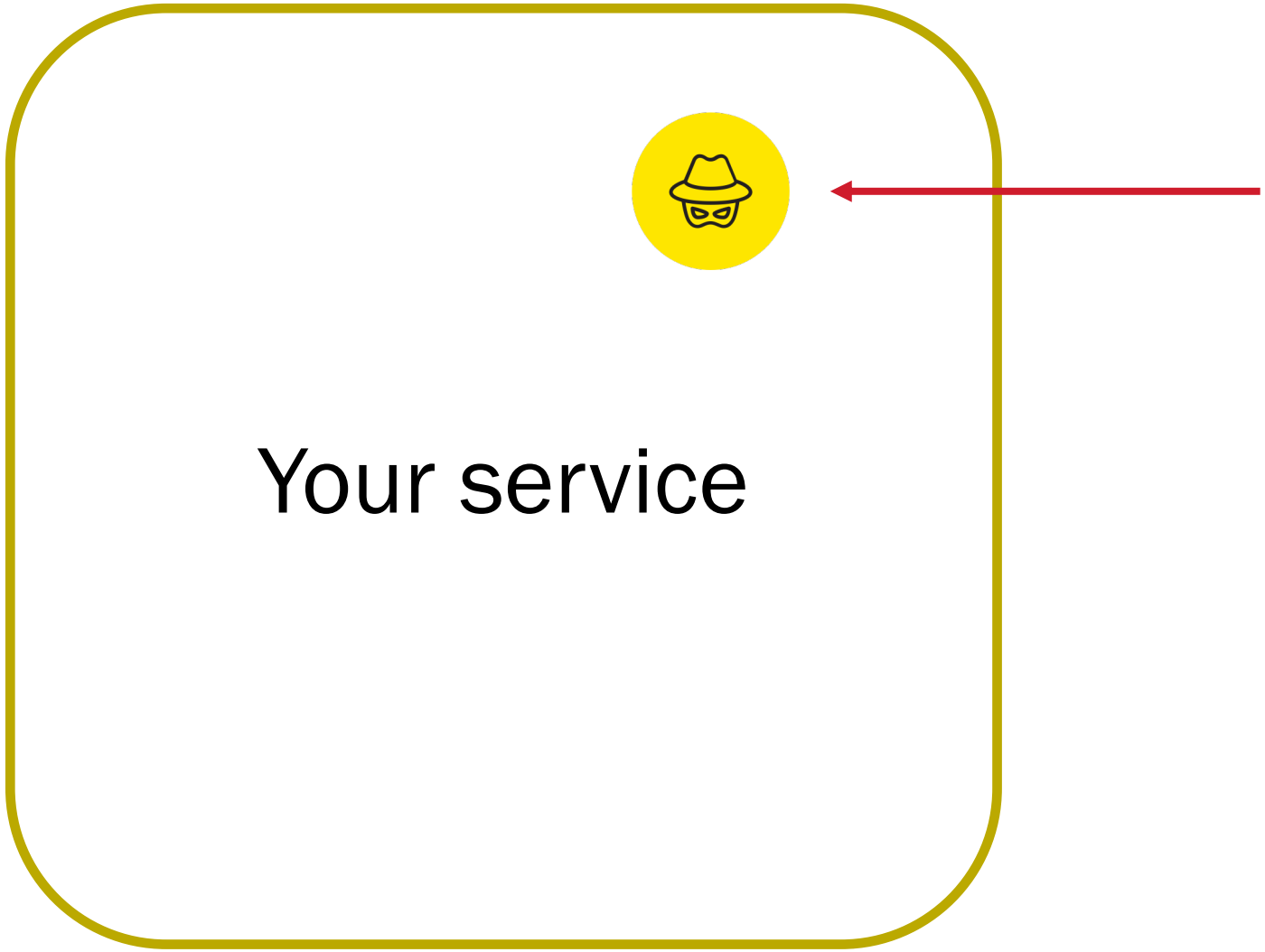
Why the biggest cyber-attacks go undetected

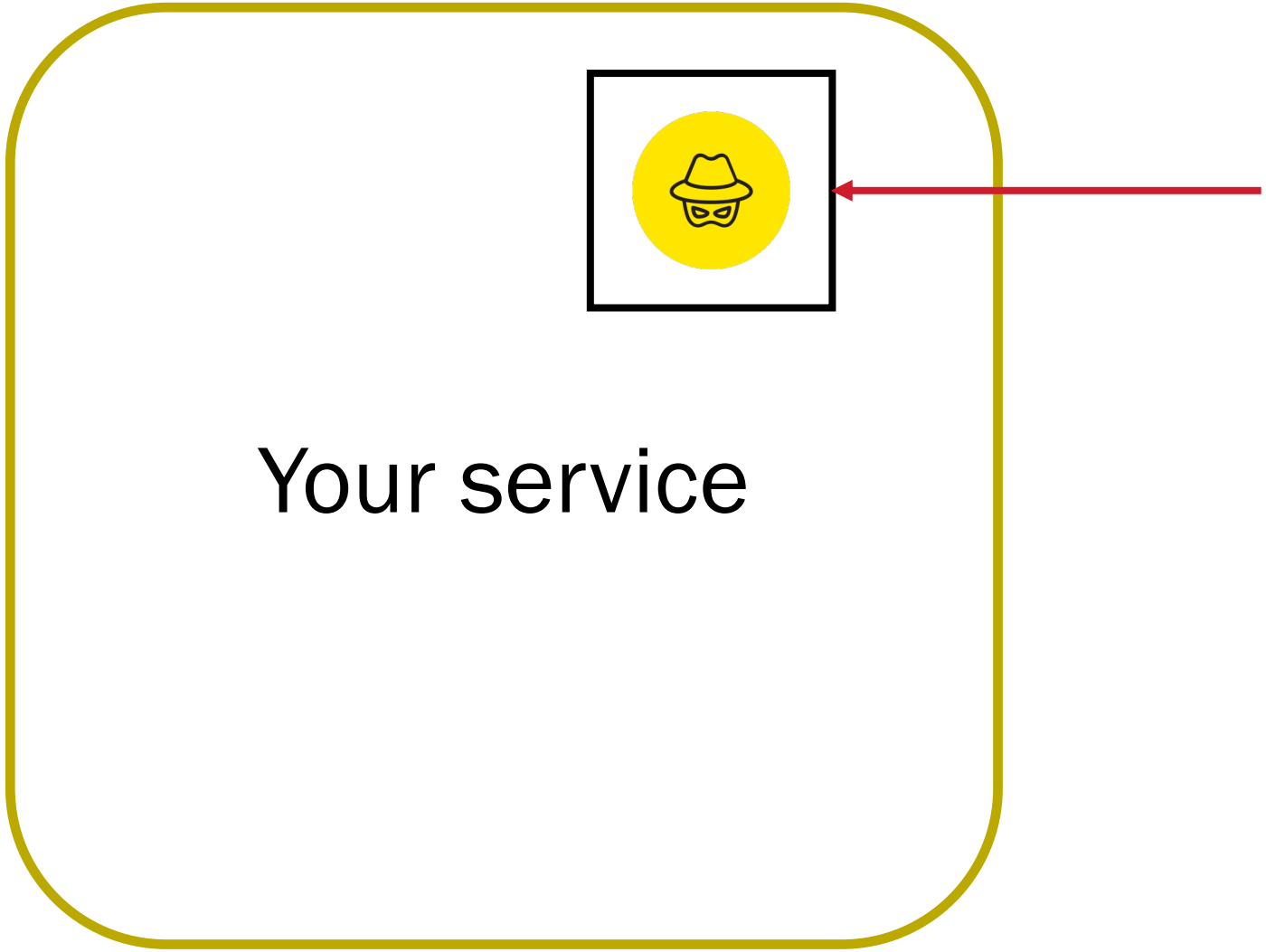
By Contributor Fri 12 Nov 2021

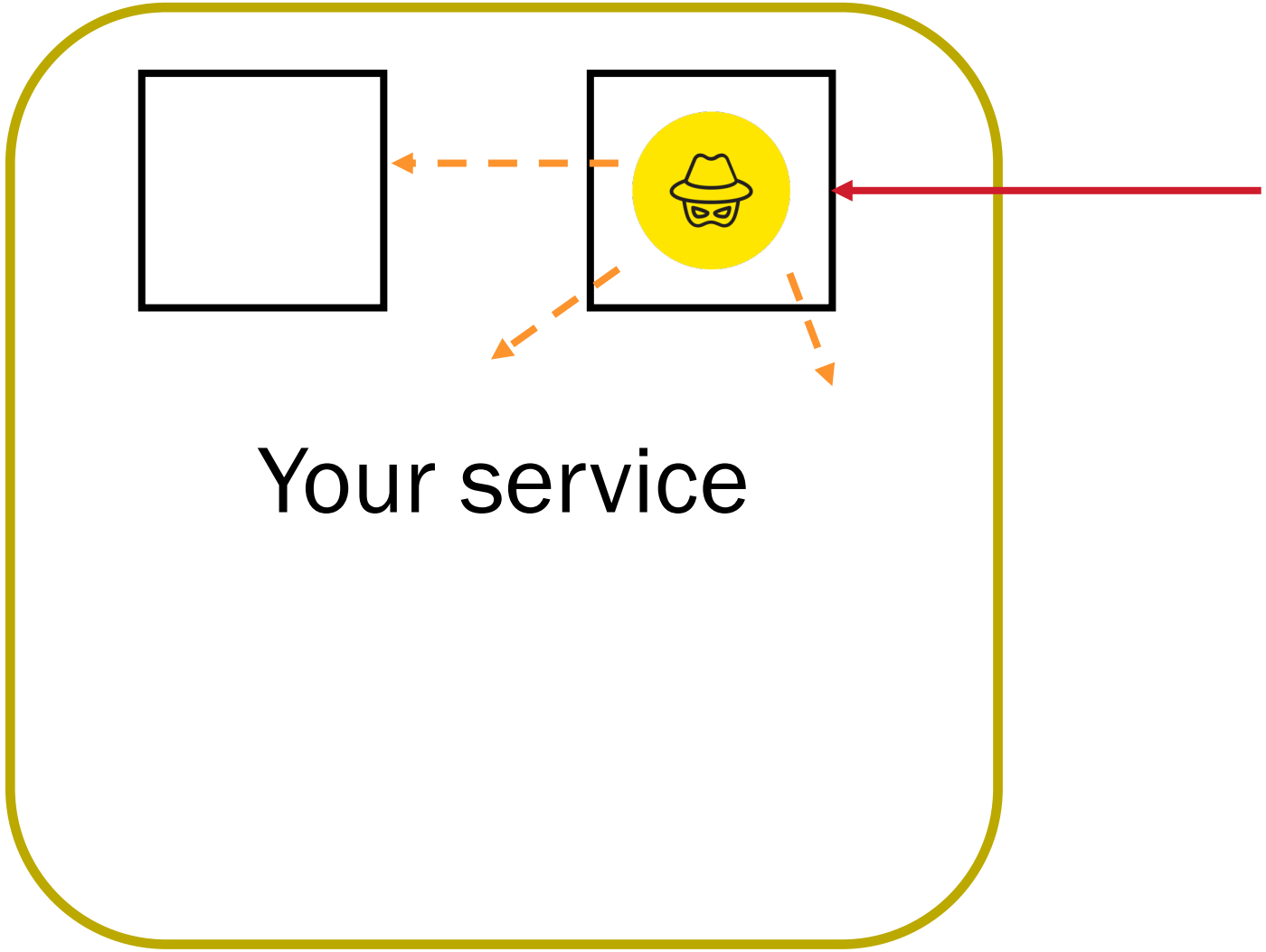
Article by Radware product marketing manager Eyal Arazi.

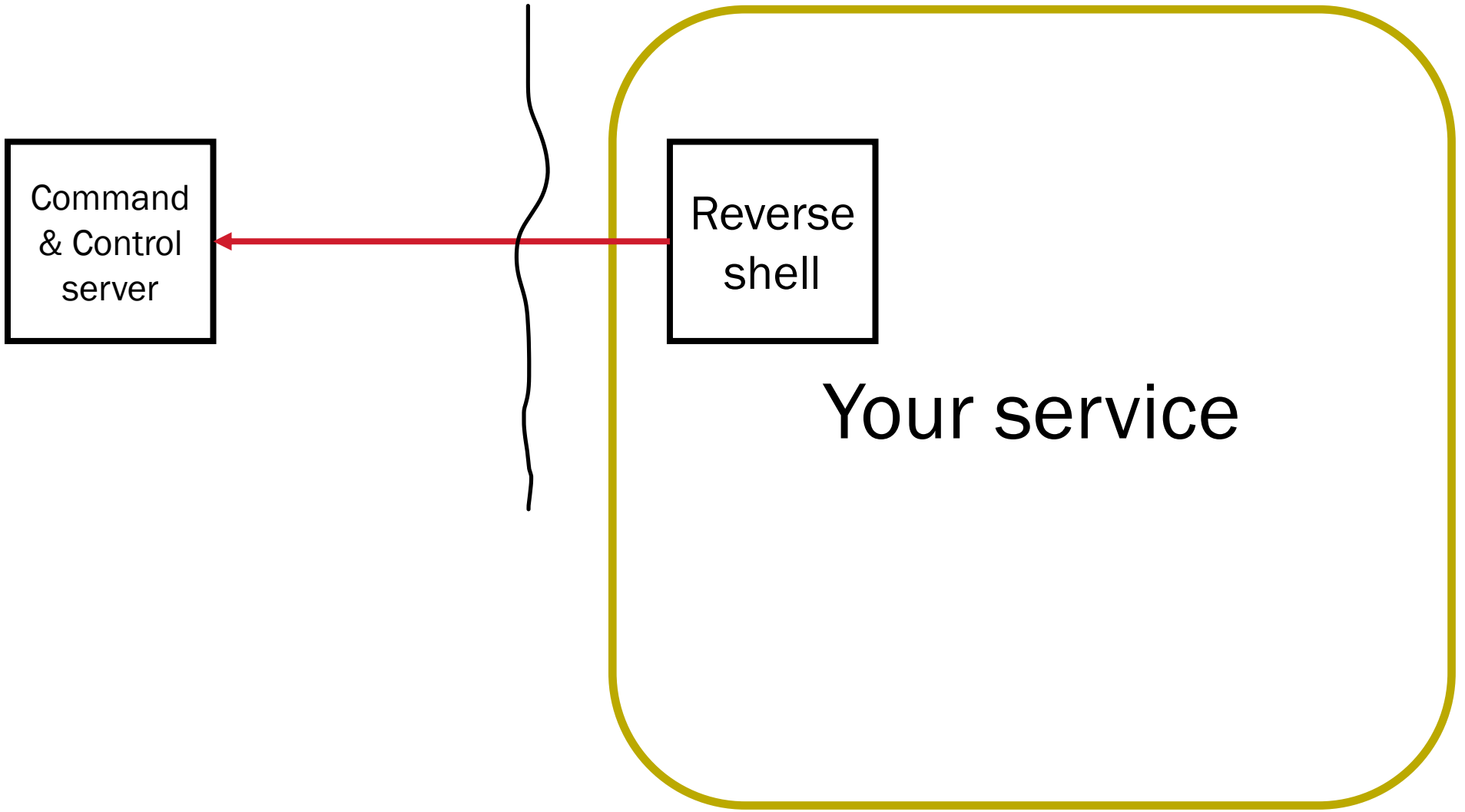
We've all seen this movie or TV scene: A hacker sits in a shadowy room busily typing on his keyboard. The camera slow
protagonist has deployed a cyber-attack into the highly secured target he was trying to penetrate. 'I'm in', he says.

This may make for great TV, but the reality of data breaches is not as exciting. In fact, the biggest and most damaging a
steps that unfold over months.


Your service


Your service

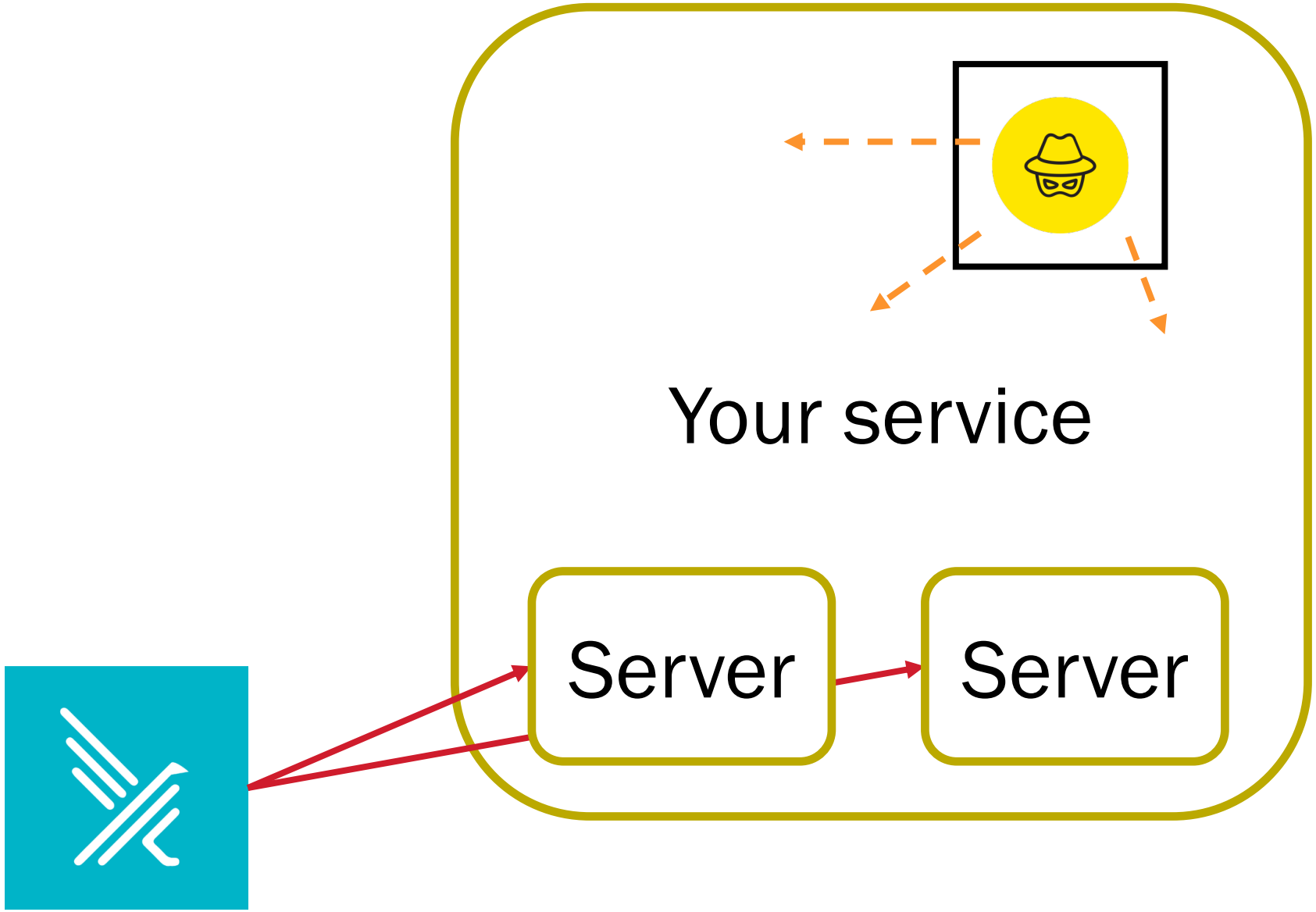

Your service

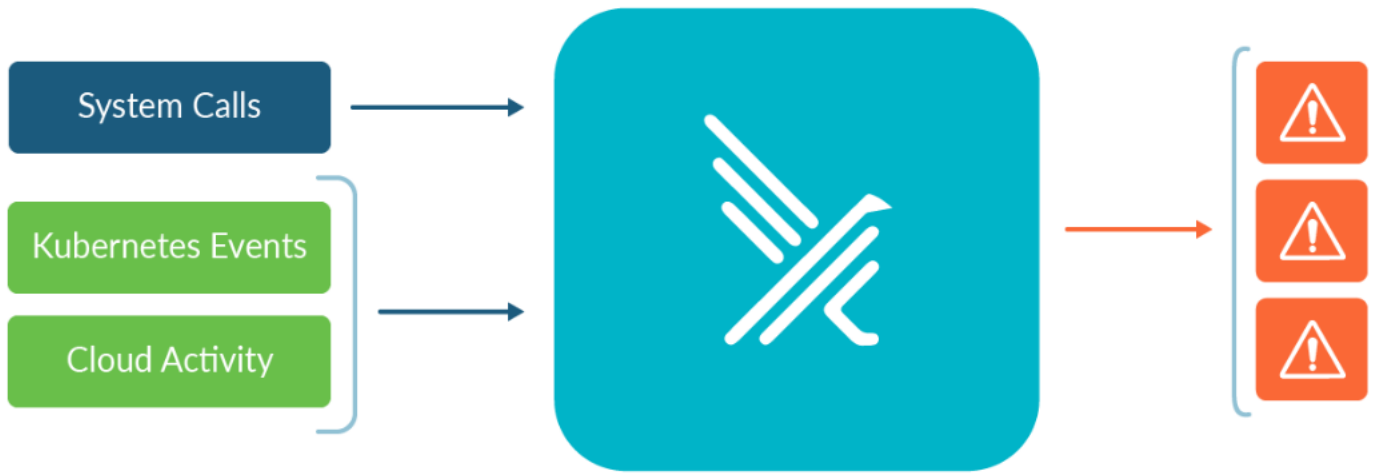

Command & Control server
Reverse shell
Your service

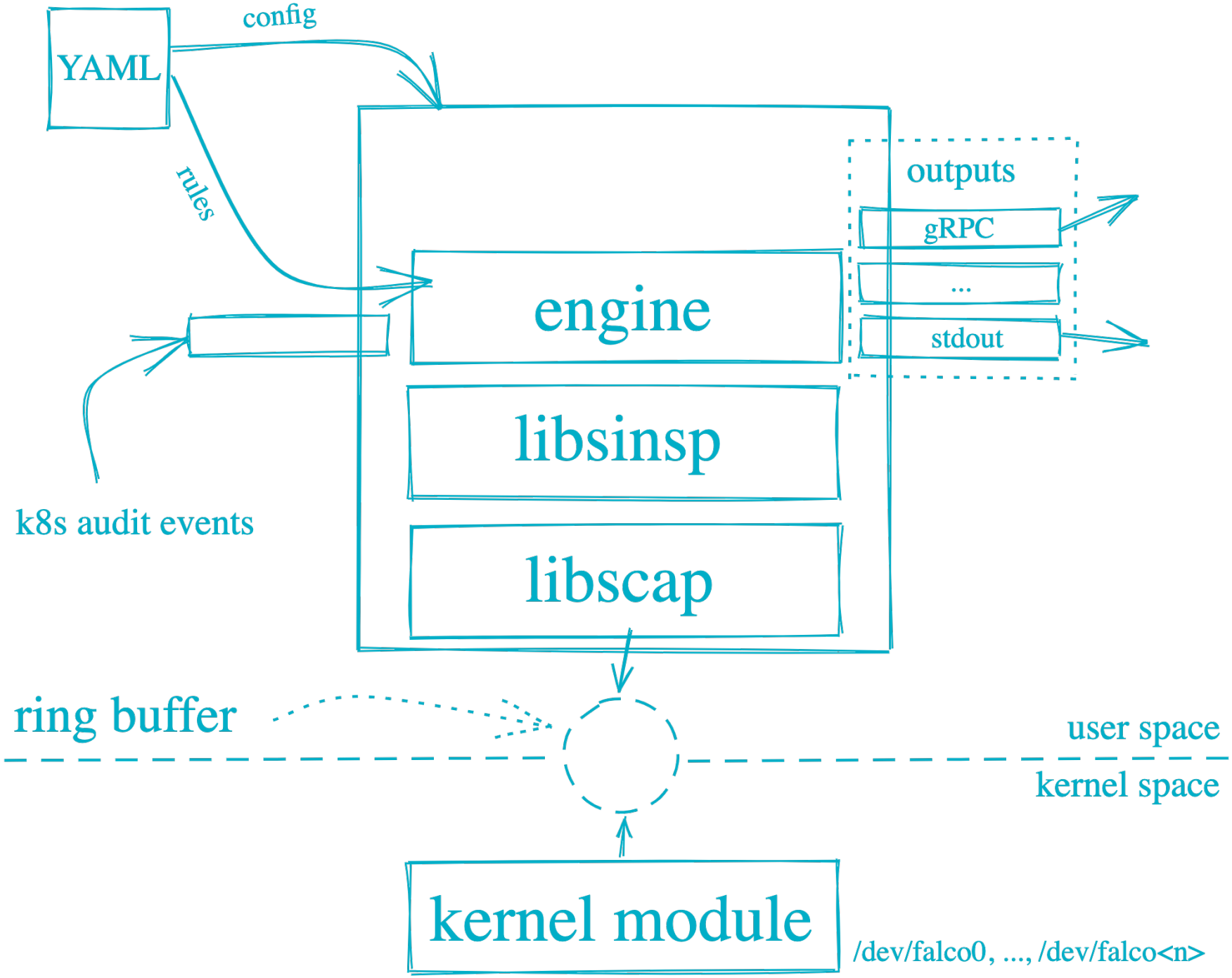
Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Impact
Using Cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8S secrets	Access the K8S API server	Access cloud resources	Images from a private registry	Data Destruction
Compromised images in registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account		Resource Hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access container service account	Network mapping	Cluster internal networking		Denial of service
Application vulnerability	Application exploit (RCE)	Malicious admission controller	Access cloud resources	Connect from Proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files		
Exposed Dashboard	SSH server running inside container				Access managed identity credential	Instance Metadata API	Writable volume mounts on the host		
Exposed sensitive interfaces	Sidecar injection				Malicious admission controller		Access Kubernetes dashboard		
							Access tiller endpoint		
							CoreDNS poisoning		
							ARP poisoning and IP spoofing		

= **New technique**

= **Deprecated technique**


Your service
Server
Server


System Calls
Kubernetes Events
Cloud Activity


YAML
config
rules
outputs
gRPC
...
stdout
engine
libsinsp
libscap
k8s audit events
ring buffer
user space
kernel space
kernel module
/dev/falco0, ..., /dev/falco<n>

“System calls don’t lie”

To achieve anything, you need to do *some* system calls

Examples: read/write a file, open a connection, spawn a process

```
- rule: Write below binary dir
  desc: an attempt to write to any file below a set of binary directories
  condition: >
    bin_dir and evt.dir = < and open_write
    and not package_mgmt_procs
    and not exe_running_docker_save
    and not python_running_get_pip
    and not python_running_ms_oms
    and not user_known_write_below_binary_dir_activities
  output: >
    File below a known binary directory opened for writing (user=%user.name user_loginuid=%user.loginuid
```

```
condition: >
  bin_dir and evt.dir = < and open_write
```

```
- macro: open_write
  condition: evt.type in (open,openat,openat2) and evt.is_open_write=true
```

```
# touch /usr/bin/helloworld
```

```
Error File below a known binary
directory opened for writing
(user=root, user_loginuid=-1,
command=touch /usr/bin/helloworld ...)
```

```
- rule: Contact K8S API Server From Container
  desc: Detect attempts to contact the K8S API Server from a container
  condition: >
    evt.type=connect and evt.dir=< and
    (fd.typechar=4 or fd.typechar=6) and
    container and
    not k8s_containers and
    k8s_api_server and
    not user_known_contact_k8s_api_server_activities
  output: Unexpected connection to K8s API Server from container (command=%proc
  priority: NOTICE
  tags: [network, k8s, container, mitre_discovery]
```

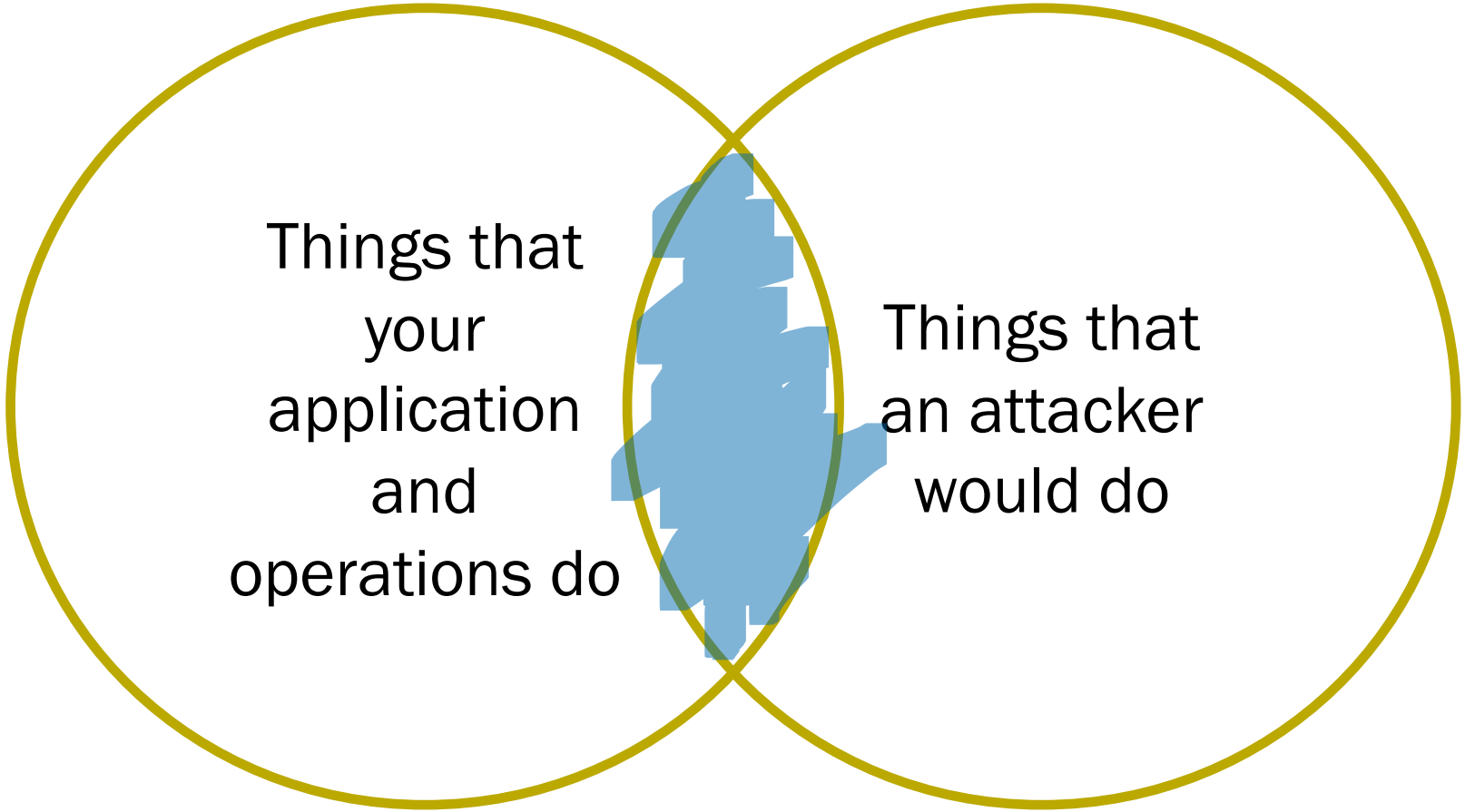
```
- rule: Redirect STDOUT/STDIN to Network Connection in Container
  desc: Detect redirecting stdout/stdin to network connection in container (potential reverse shell).
  condition: evt.type=dup and evt.dir=> and container and fd.num in (0, 1, 2) and fd.type in ("ipv4", "ipv6") and not user_k
  output: >
    Redirect stdout/stdin to network connection (user=%user.name user_loginuid=%user.loginuid %container.info process=%proc.
  priority: WARNING
```

Will there be false positives?

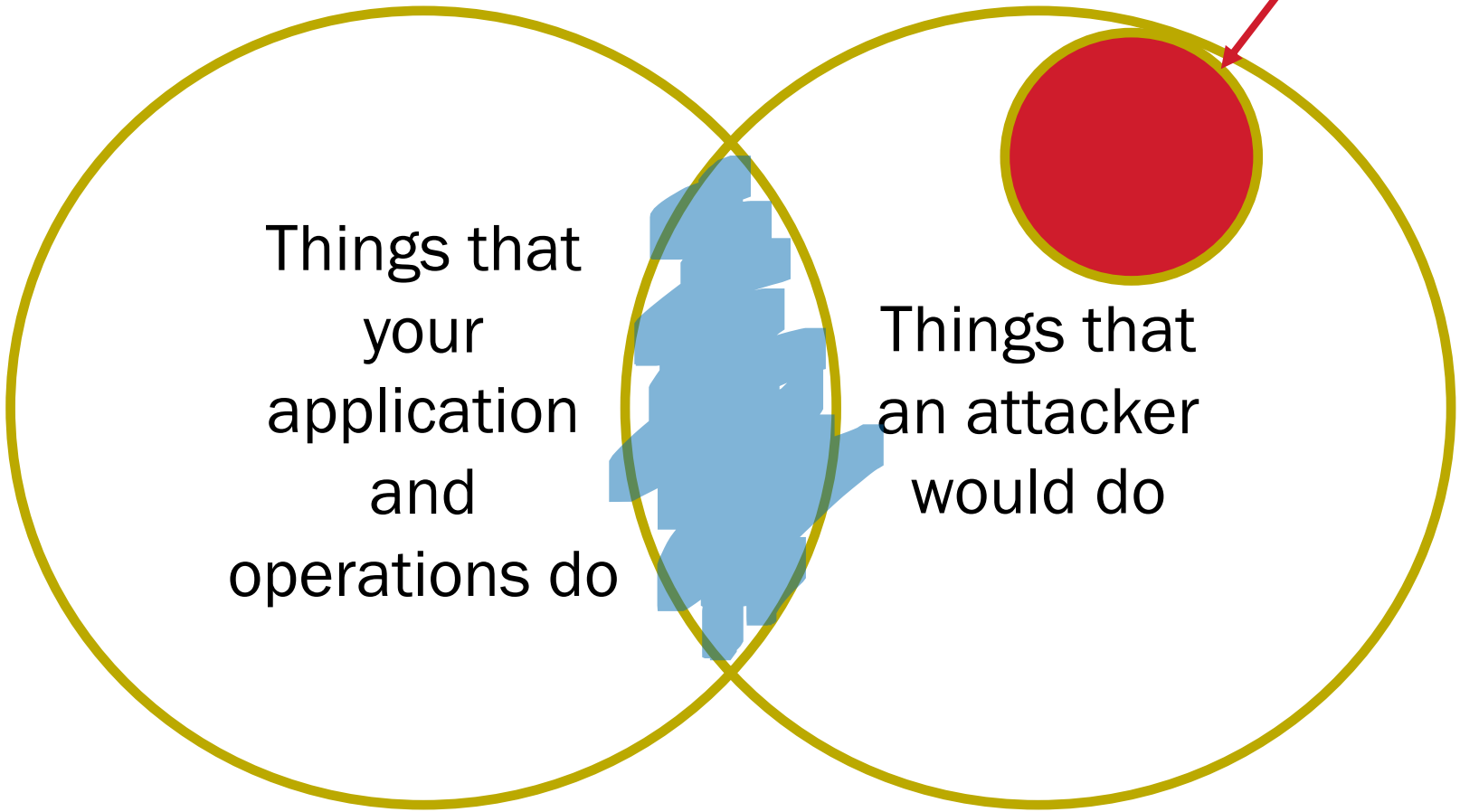
Legitimate activity that is detected

```
- rule: Contact K8S API Server From Container
  desc: Detect attempts to contact the K8S API Server from a container
  condition: >
    evt.type=connect and evt.dir=< and
    (fd.typechar=4 or fd.typechar=6) and
    container and
    not k8s_containers and
    k8s_api_server and
    not user_known_contact_k8s_api_server_activities
  output: Unexpected connection to K8s API Server from container (command=%proc
  priority: NOTICE
  tags: [network, k8s, container, mitre_discovery]
```

So probably, yes


Things that your application and operations do
Things that an attacker would do

An action that the attacker likely does, but an action your application *never* does



Bad news: you probably need to fine-tune the rules

```yaml
- macro: calico_node
  condition: (container.image.repository endswith calico/node and proc.name=calico-node)

- macro: weaveworks_scope
  condition: (container.image.repository endswith weaveworks/scope and proc.name=scope)
```

Good news: *you* can identify what looks legitimate

Good news: *you* can identify what looks legitimate

(and not the Security Operations Center)

Even better news: doing stuff Cloud *Natively* should make it easier

Example: running a package manager *inside* a container

You should never need to do this

The attacker might want to do it

Closer to the Cloud Native way, the less false positives

Use Falco to improve your “*cloudnativeness*”

Use Falco to improve your security posture

Why *specifically* Falco, then?

Open source

Pluggable

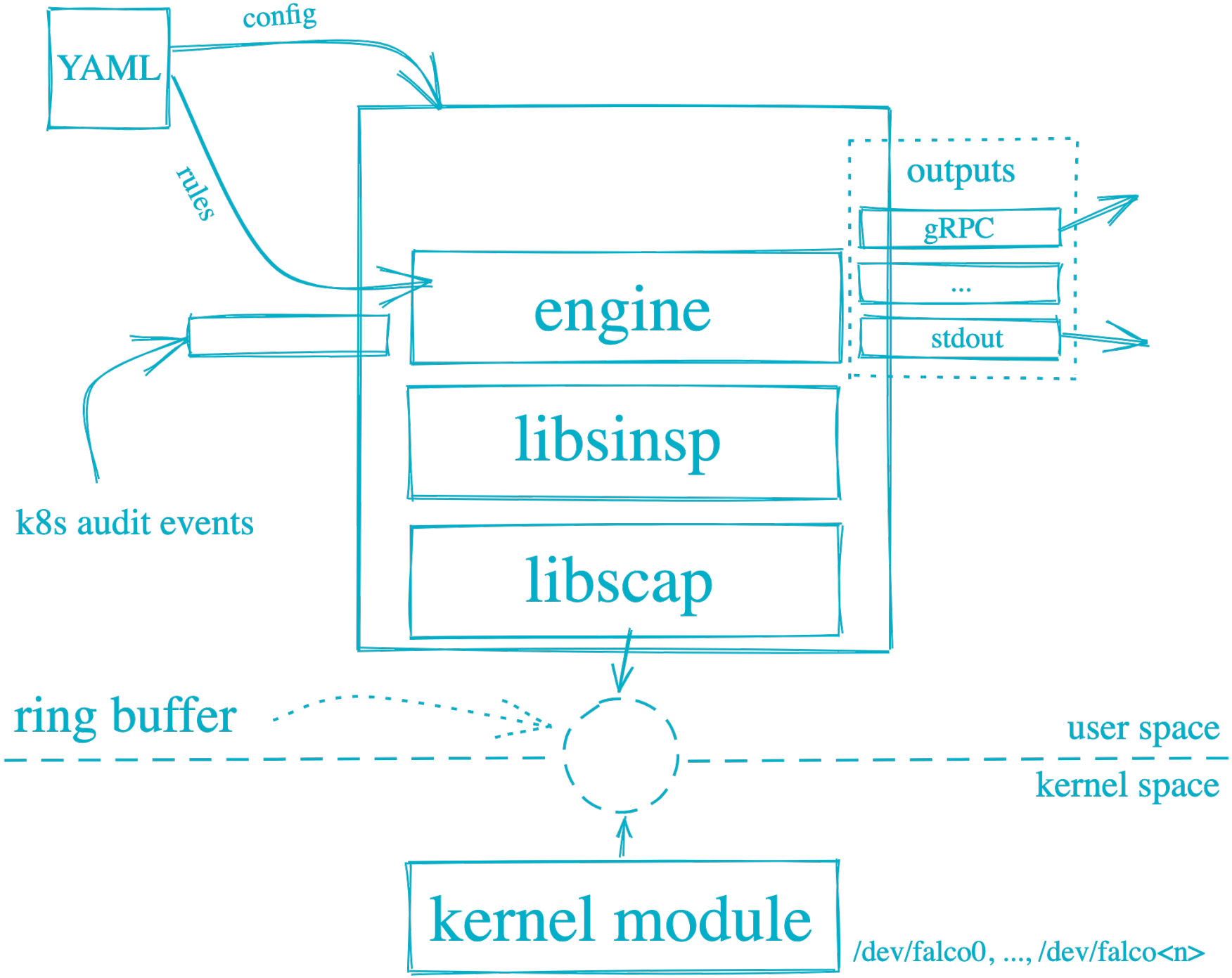
Single purpose

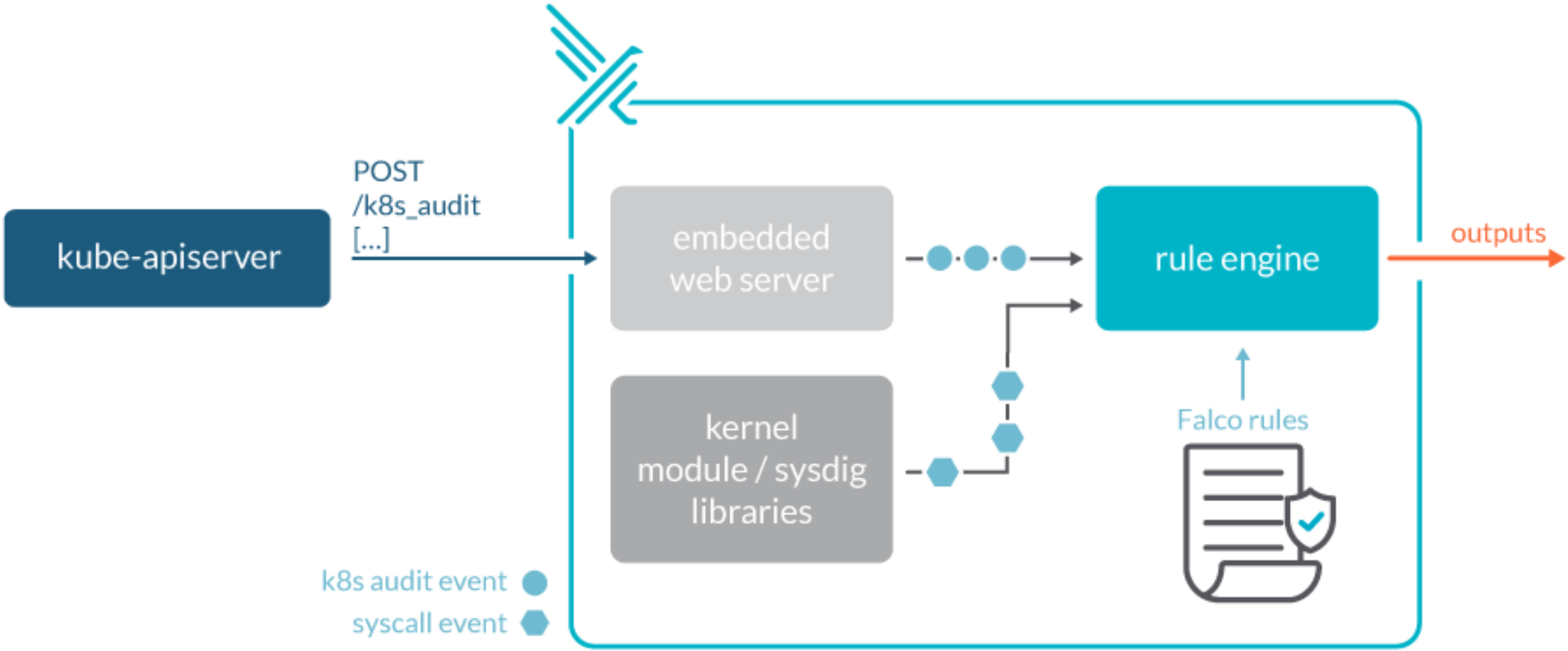
Easy to understand

Cloud Native & K8s support

Good built-in detections

Kubernetes Audit Events


YAML
config
rules
outputs
gRPC
...
stdout
engine
libsinsp
libscap
k8s audit events
ring buffer
user space
kernel space
kernel module
/dev/falco0, ..., /dev/falco<n>


kube-apiserver
POST
/k8s_audit
[...]
embedded
web server
kernel
module / sysdig
libraries
rule engine
outputs
Falco rules
k8s audit event
syscall event

```
- rule: Create Disallowed Pod
  desc: >
    Detect an attempt to start a pod with a container image outside of a list of allowed images.
  condition: kevt and pod and kcreate and not allowed_k8s_containers
  output: >
    Pod started with container not in allowed list (user=%ka.user.name
    pod=%ka.resp.name ns=%ka.target.namespace
    images=%ka.req.pod.containers.image)
  priority: WARNING
  source: k8s_audit
  tags: [k8s]
```

```
- macro: kcreate
  condition: ka.verb=create

- macro: kmodify
  condition: (ka.verb in (create,update,patch))

- macro: kdelete
  condition: ka.verb=delete

- macro: pod
  condition: ka.target.resource=pods and not ka.target.subresource exists
```

Thanks.
Questions or comments?

FRAKTAL
Cyber Positivity®

2019
YEAR OF FOUNDING

22
TEAM SIZE WINTER 2022

HELSINKI
HEAD OFFICE

TAMPERE
COPENHAGEN
LOCAL PRESENCE

WHAT WE KNOW

We are certified experts in cyber security management as well as AWS, Azure and Google cloud architectures and security technologies.

Software security

Cloud security

DevSecOps

Security compliance and regulation

Tailored security assessments

Zero trust architectures

TIBER-EU and TIBER-FI

WHAT WE DELIVER

WE ADVISE SECURITY
Security roadmaps, plans, design, training.

WE BUILD SECURITY
Risk analysis, threat models, developing security processes, secure software development, secure cloud adoption.

WE RUN SECURITY
Exercises, testing technical capabilities, SOC testing, incident response, security expertise as service.

FRAKTAL