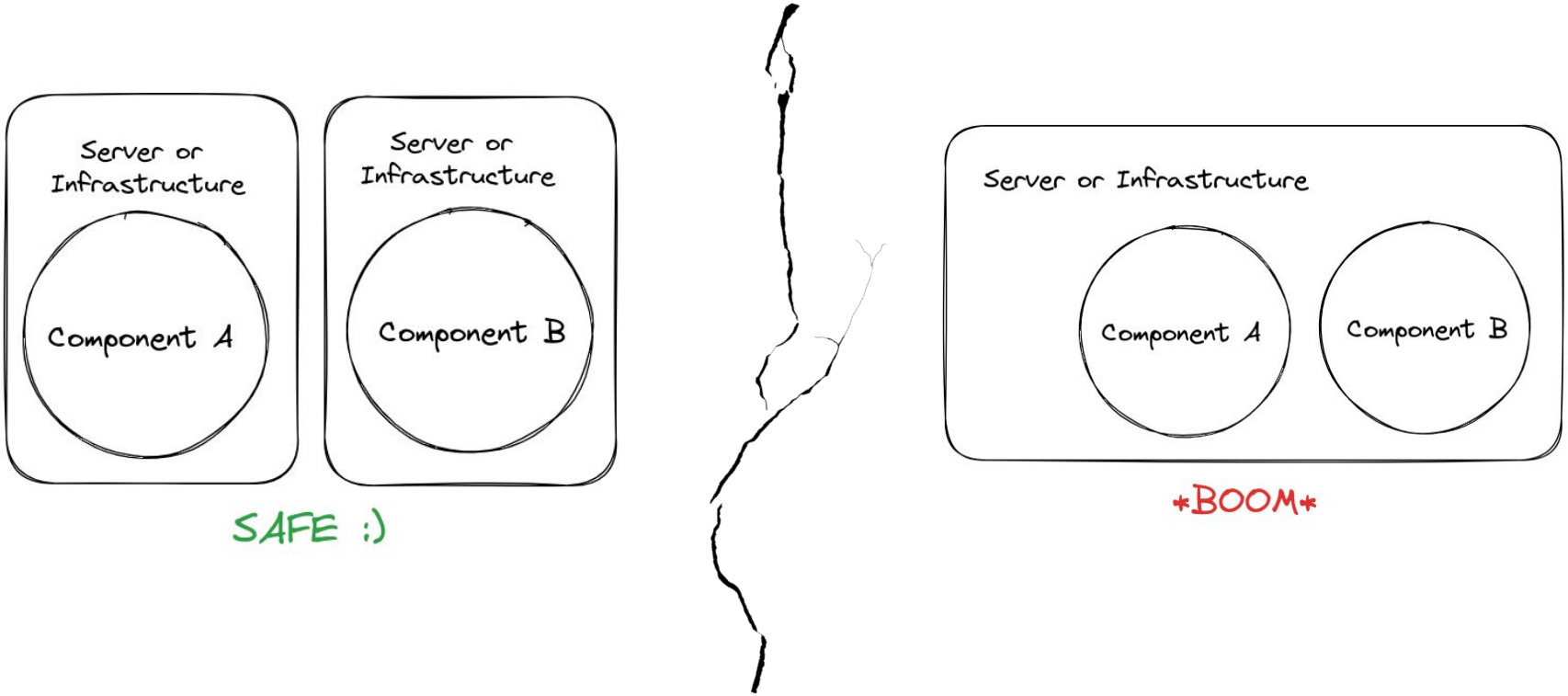
System Introspection for Micro-Services Pentests in K8S

@TheLaluka
thinkloveshare.com
offenskill.com

1. Introduction to the “in-between” world
2. Capturing Component System Interactions
3. Micro-Services Pentesting in K8S 101
4. How-To Falco & My Favorite SRE
5. Pros, Cons, & WhatNot
6. Conclusion & Kudos

Introduction to the “in-between” world

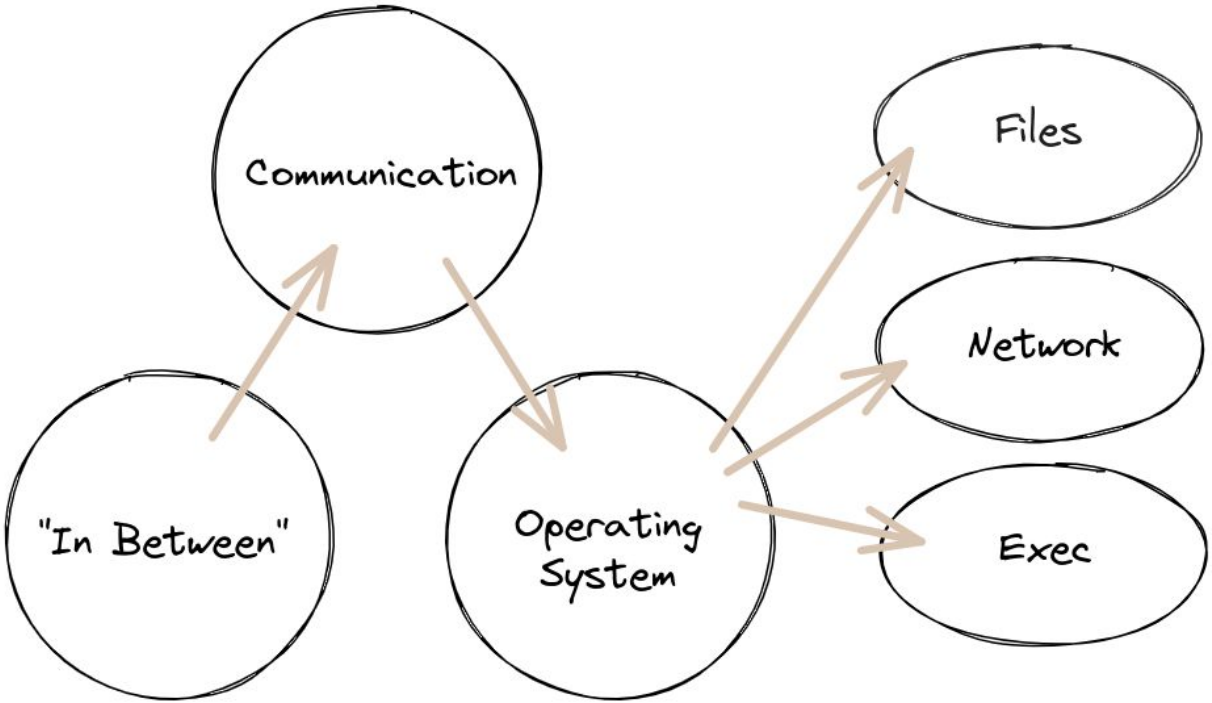
The root of evil: “You & I are safe, so we are safe”



The “in between”: Examples

- File upload + SSH server with key
- FTP service + PHP server
- Fetch by URL + Cloud Infra
- URL parser front != URL parser back
- Admin iface on loopback + Proxy service
- Customizable username + PDF export
- PNG file upload + PHAR format

The “in between”: Primitives & Detection



The “in between”: Primitives & Detection

getpgrp
futimesat
get_kernel_syms
getresgid
epoll_wait
getrusage
delete_module
epoll_pwait
exit_group
fremovexattr
get_mempolicy
getppid
renameat
clone
clock_settime
signalfd
fsync
ftruncate
creat
bind
chmod
clock_getres
fcntl
getsockname
add_key
_sysctl
eventfd
get_robust_list
alarm
fchmodat
execve
getpid
acct
pwritev
getdents
clock_gettime
fanotify_mark
getpmsg
getcpu
flistxattr
clock_adjtime
getegid
flock
mlock
epoll_create
fdatasync
fchownat
connect
pipe
chown
access
fork
bpf
get_thread_area
getsockopt
create_module
futex
fchdir
capget
preadv
arch_prctl
fstatfs
fanotify_init
brk
chdir
fgetxattr
getcwd
accept
dup
fchown
inotify_init
finit_module
get
capset
adjtimex
exit
fallocate
getresuid
fsetxattr
epoll_ctl
chroot
afs_syscall
getpgid
clock_nanosleep
epoll_ctl_old
fstat
getgroups
fadvise
close
getpriority
epoll_wait_old
copy_file_range
faccessat
getpeername
getrandom
getitimer
fchmod
getgid
getsid

The “in between”: Pick your BOX, but stay blind.

Capturing Component System Interactions

wikiHow rechercher comment...

Bienvenue sur wikiHow, le site de tutoriels le plus fiable d'internet.

Qu'allez-vous découvrir sur wikiHow aujourd'hui ?

How to hook syscalls please ?

wikiHow

How to hook syscalls

Hook within the binary in User-Land?

28 351 vues

Actualisé il y a 7 ans

Hook with a Kernel Module?

21 579 vues

Actualisé il y a 2 ans

Rely on eBPF probes?

159 989 vues

Actualisé il y a 2 ans

wikiHow

How to hook syscalls

Comment survivre dans 7 Days **to** Die

28 351 vues

Actualisé il y a 7 ans

Comment cultiver dans 7 Days **to** Die

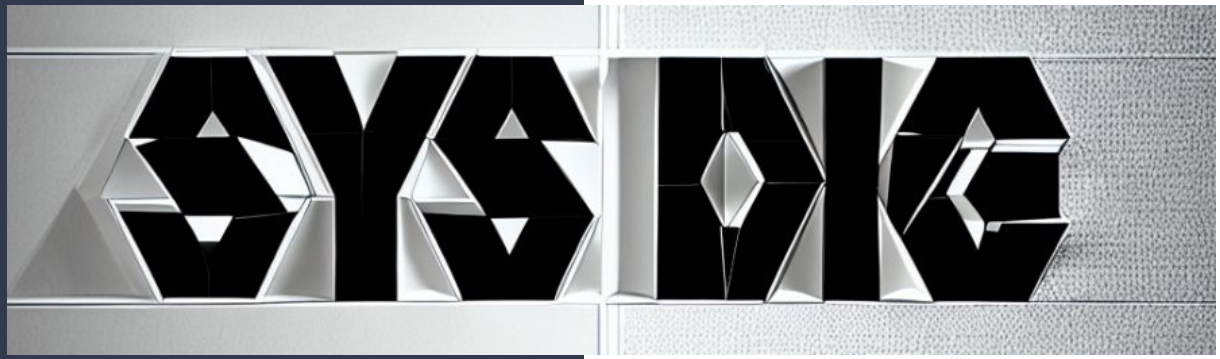
21 579 vues

Actualisé il y a 2 ans

Comment éviter d'avoir un camel toe (l'orteil de chameau)

159 989 vues

Actualisé il y a 2 ans



Getting Started

Run Sysdig in a container:

```
sudo docker run --rm -i -t --privileged --net=host \
    -v /var/run/docker.sock:/host/var/run/docker.sock \
    -v /dev:/host/dev \
    -v /proc:/host/proc:ro \
    -v /boot:/host/boot:ro \
    -v /src:/src \
    -v /lib/modules:/host/lib/modules:ro \
    -v /usr:/host/usr:ro \
    -v /etc:/host/etc:ro \
    docker.io/sysdig/sysdig
```

And then run the `sysdig` or `csysdig` tool from the container shell!

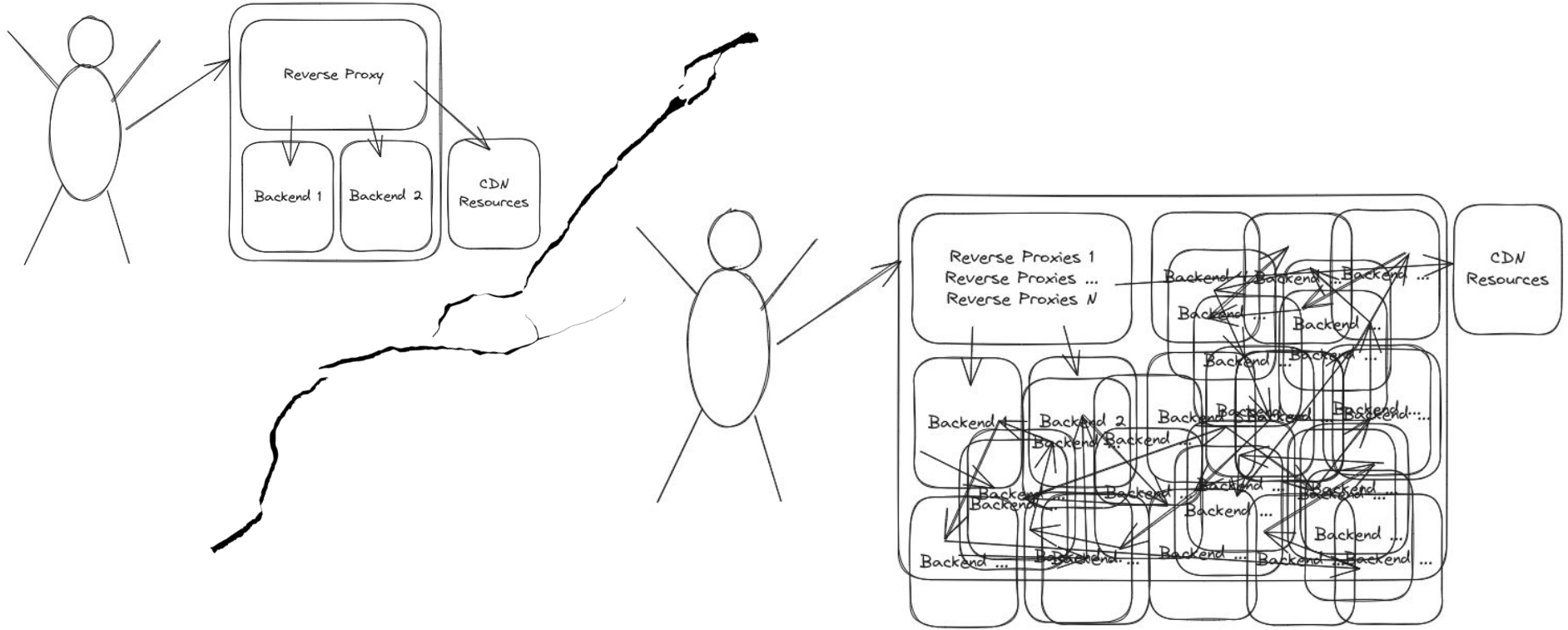
Or install the latest release with a `deb` or `rpm` package for your distribution.

```
root@ML-PF2W5DKR:/# whoami
root
root@ML-PF2W5DKR:/#
```

```
[root@ML-PF2W5DKR /]# sysdig -A container.name=ubuntu | grep -P 'exec|open|write|read'
6712 16:48:20.882391241 5 bash (106256) > read fd=0(<f>/dev/pts/0) size=1
6714 16:48:20.882400157 5 bash (106256) < read res=1 data=
6716 16:48:20.882406794 5 bash (106256) > write fd=2(<f>/dev/pts/0) size=1
6718 16:48:20.882413531 5 bash (106256) < write res=1 data=
6719 16:48:20.882421681 5 bash (106256) > write fd=2(<f>/dev/pts/0) size=9
6722 16:48:20.882424145 5 bash (106256) < write res=9 data=
7021 16:48:20.883043224 4 bash (108244) > read fd=3(<p>) size=1
7023 16:48:20.883046538 4 bash (108244) < read res=0 data=NULL
7100 16:48:20.883149165 4 bash (108244) > execve filename=/usr/bin/whoami
7198 16:48:20.883552137 4 whoami (108244) < execve res=0 exe=whoami args=NULL tid=108244(whoami) pid=108244(whoami) ptid=106256(bash) cwd= fdlimit=1048
m_rss=4 vm_swap=0 comm=whoami cgroups=
7219 16:48:20.883742774 4 whoami (108244) > openat dirfd=-100(AT_FDCWD) name=/etc/ld.so.cache flags=4097(O_RDONLY|O_CLOEXEC) mode=0
7228 16:48:20.883760911 4 whoami (108244) < openat fd=3(<f>/etc/ld.so.cache) dirfd=-100(AT_FDCWD) name=/etc/ld.so.cache flags=4097(O_RDONLY|O_CLOEXEC)
7250 16:48:20.883791145 4 whoami (108244) > openat dirfd=-100(AT_FDCWD) name=/lib/x86_64-linux-gnu/libc.so.6 flags=4097(O_RDONLY|O_CLOEXEC) mode=0
7254 16:48:20.883806160 4 whoami (108244) < openat fd=3(<f>/lib/x86_64-linux-gnu/libc.so.6) dirfd=-100(AT_FDCWD) name=/lib/x86_64-linux-gnu/libc.so.6 f
=38 ino=10003473
7255 16:48:20.883808525 4 whoami (108244) > read fd=3(<f>/lib/x86_64-linux-gnu/libc.so.6) size=832
7261 16:48:20.883813228 4 whoami (108244) < read res=832 data=
7267 16:48:20.883820820 4 whoami (108244) > pread fd=3(<f>/lib/x86_64-linux-gnu/libc.so.6) size=784 pos=64
7269 16:48:20.883823209 4 whoami (108244) < pread res=784 data=
7270 16:48:20.883824525 4 whoami (108244) > pread fd=3(<f>/lib/x86_64-linux-gnu/libc.so.6) size=48 pos=848
7272 16:48:20.883825936 4 whoami (108244) < pread res=48 data=
7273 16:48:20.883827018 4 whoami (108244) > pread fd=3(<f>/lib/x86_64-linux-gnu/libc.so.6) size=68 pos=896
7274 16:48:20.883828432 4 whoami (108244) < pread res=68 data=
7277 16:48:20.883837008 4 whoami (108244) > pread fd=3(<f>/lib/x86_64-linux-gnu/libc.so.6) size=784 pos=64
7278 16:48:20.883838433 4 whoami (108244) < pread res=784 data=
7427 16:48:20.884627998 4 whoami (108244) > openat dirfd=-100(AT_FDCWD) name=/etc/nsswitch.conf flags=4097(O_RDONLY|O_CLOEXEC) mode=0
7431 16:48:20.884642651 4 whoami (108244) < openat fd=3(<f>/etc/nsswitch.conf) dirfd=-100(AT_FDCWD) name=/etc/nsswitch.conf flags=4097(O_RDONLY|O_CLOE
7435 16:48:20.884665915 4 whoami (108244) > read fd=3(<f>/etc/nsswitch.conf) size=4096
7436 16:48:20.884671829 4 whoami (108244) < read res=494 data=
7437 16:48:20.884692309 4 whoami (108244) > read fd=3(<f>/etc/nsswitch.conf) size=4096
7438 16:48:20.884693817 4 whoami (108244) < read res=0 data=NULL
7444 16:48:20.884719309 4 whoami (108244) > openat dirfd=-100(AT_FDCWD) name=/etc/passwd flags=4097(O_RDONLY|O_CLOEXEC) mode=0
7448 16:48:20.884731758 4 whoami (108244) < openat fd=3(<f>/etc/passwd) dirfd=-100(AT_FDCWD) name=/etc/passwd flags=4097(O_RDONLY|O_CLOEXEC) mode=0 dev
```

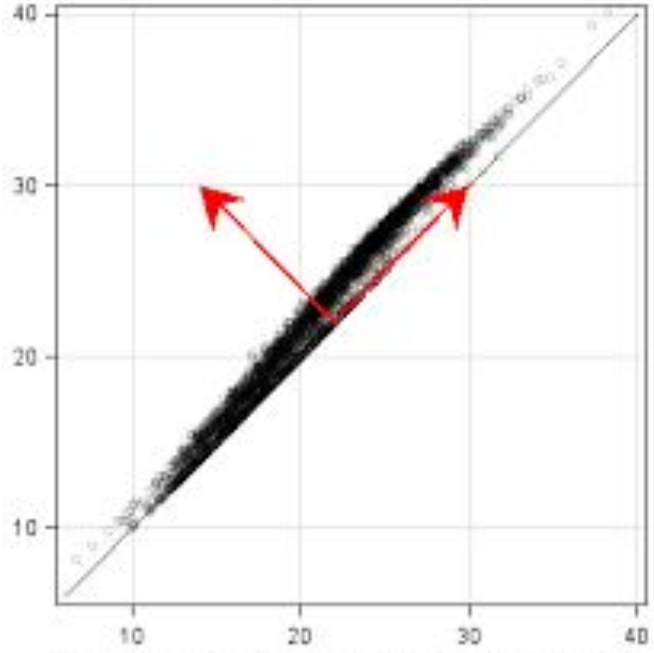
Micro-Services Pentesting in K8S 101

The pentest I Know Vs The one I Fear

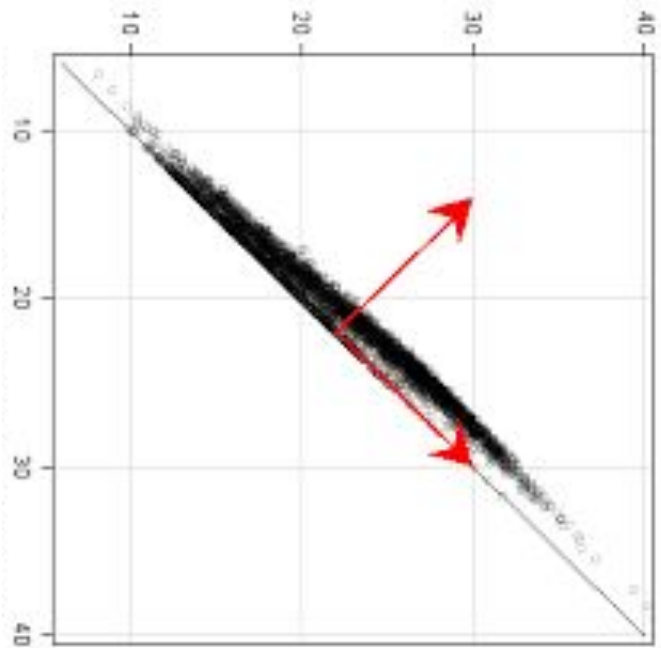


Totally stolen yet accurate graphs

More Complexity → More Bugs



More Complexity → WTF's happening ?



Huge structures have the more -security- bugs

Huge structures offer no “easy assumptions”

How can we bring our magic-glasses to K8S?


WHAT IF WE'VE ALWAYS HAD THEM???

sysdig

Falco

Rule Engine
- Rule language compiler
- Rules evaluation
- Rules management
- Inputs + outputs

libsinsp
- Event parsing
- State engine
- Filtering
- Output formatting

libscap
- Capture Control
- Dump files R/W
- OS state collection

User

Kernel

System calls → Kernel Module

eBPF Probe
- Non-blocking event collection
- Type-based event packing
- Memory mapped buffer handling

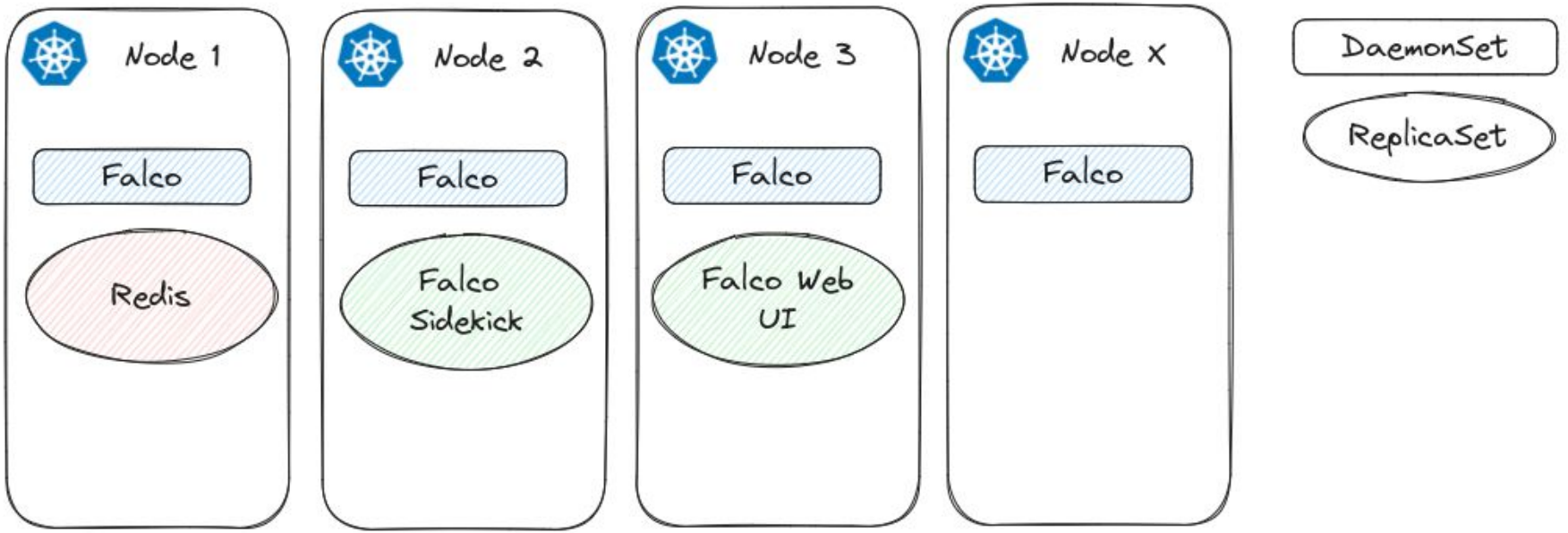
Falco

How-To Falco & My Favorite SRE

Falco containers services

- Falco agent
  - Monitor kernel events base on custom rules
- Falco sidekick
  - Take Falco events and forward to different output (Slack, Prometheus, Kafka …)
- Falco Web UI (Optional)
  - Simple web ui to displaying latest event
- Redis (Optional)
  - Store Falco events for Falco Web UI

Falco on Kubernetes



Hello Helm chart !

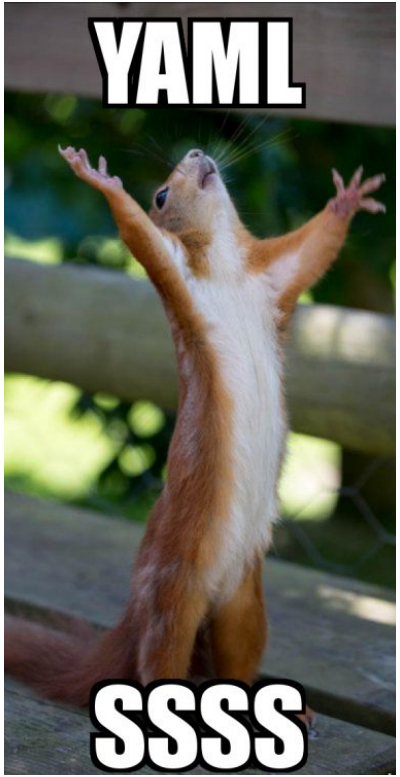
```
~$ kubectl -n falco get deployment
NAME                                    READY   UP-TO-DATE   AVAILABLE
infra-falco-sidekick-falcosidekick      1/1     1            1
infra-falco-sidekick-falcosidekick-ui   1/1     1            1

~$ kubectl -n falco get daemonset
NAME                     DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE
infra-falco-controller   28        28        28      28           28
```

```yaml
---
image:
  repository: "public/falcosecurity/falco-no-driver"

resources:
  requests:
    cpu: 200m
    memory: 512Mi
  limits:
    cpu: 1000m
    memory: 1024Mi

affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - fargate
        - key: karpenter.k8s.aws/instance-gpu-count
          operator: DoesNotExist
```




```
{{- if eq .Values.controller.kind "daemonset" }}
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: {{ include "falco.fullname" . }}
  namespace: {{ include "falco.namespace" . }}
  labels:
    {{- include "falco.labels" . | nindent 4 }}
  {{- if .Values.controller.annotations }}
  annotations:
  {{ toYaml .Values.controller.annotations | nindent 4 }}
  {{- end }}
spec:
  selector:
    matchLabels:
      {{- include "falco.selectorLabels" . | nindent 6 }}
  template:
    {{- include "falco.podTemplate" . | nindent 4 }}
  {{- with .Values.controller.daemonset.updateStrategy }}
  updateStrategy:
    {{- toYaml . | nindent 4 }}
  {{- end }}
{{- end }}
```

A little bit of Magic !

From Falco’s yaml rules-file

```yaml
- macro: isLogLoopSafe
  condition: >
    ((k8s.pod.name exists)
    and (not k8s.pod.name icontains "falco")
    and (not k8s.pod.name icontains "fluent")
    and (not k8s.pod.name icontains "guardduty")
    and (not k8s.pod.name icontains "vector-forward"))

- rule: Privilege Escalation Script
  desc: A shell script has been used to attempt privilege escalation and enumeratoin.
  condition: >
    spawned_process and (
    proc.cmdline icontains LinEnum or
    proc.cmdline icontains lse.sh or
    proc.cmdline icontains smartenum or
    proc.cmdline icontains exploit or
    proc.cmdline icontains privcheck or
    proc.cmdline icontains linpeas)
    and isLogLoopSafe
  output: >
    A shell script has been used to attempt privilege escalation and enumeratoin. (evt.type=%evt.type
user=%user.name user_loginuid=%user.loginuid shell=%proc.name parent=%proc.pname cmdline=%proc.cmdline
pid=%proc.pid terminal=%proc.tty)
  priority: CRITICAL
  tags: [manomano, shell, privesc]
```

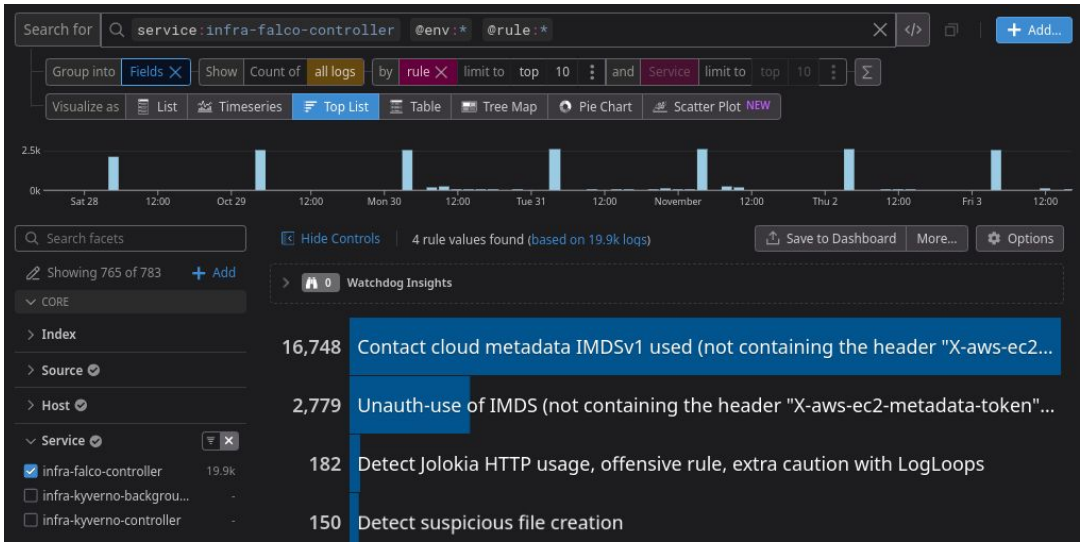
Kustomize

… to Kubernetes ConfigMap

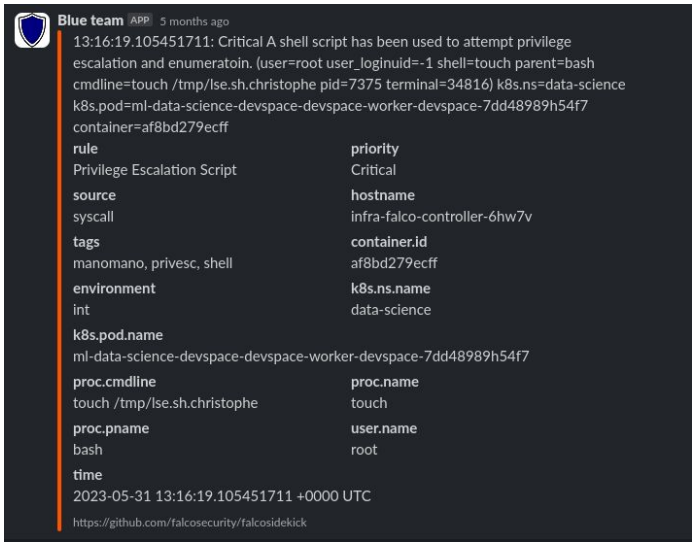
```yaml
apiVersion: v1
data:
  falco_rules.local.yaml: "# MM Custom rules\n\n- macro: isLogLoopSafe\n  condition:
    >\n    ((k8s.pod.name exists)\n    and (not k8s.pod.name icontains \"falco\")\n
    \   and (not k8s.pod.name icontains \"fluent\")\n    and (not k8s.pod.name icontains
    \"guardduty\")\n    and (not k8s.pod.name icontains \"vector-forward\"))\n\n-
    list: mysql-db-fqdns\n  items: [int-app-common-db.manomano.tech, int-app-common-db-
ro.manomano.tech,
    stg-app-common-db.manomano.tech, stg-app-common-db-ro.manomano.tech, stg-app-common-db-
bi.manomano.tech,
    prd-app-common-db.manomano.tech, prd-app-common-db-ro.manomano.tech, prd-app-common-db-
bi.manomano.tech]\n\n-
    list: allowed-files\n  items: [dd_init.php, App_KernelProdContainer.preload.php,
    flatted.php]\n\n- list: monitored-dangerous-sockets\n  items: [docker.sock, containerd.sock,
    docker-containerd.sock, crio.sock]\n\n- rule: Privilege Escalation Script\n  desc:
    A shell script has been used to attempt privilege escalation and enumeratoin.\n
    \ condition: >\n    spawned_process and (\n    proc.cmdline icontains LinEnum
    or\n    proc.cmdline icontains lse.sh or\n    proc.cmdline icontains smartenum
    or\n    proc.cmdline icontains exploit or\n    proc.cmdline icontains privcheck
    or\n    proc.cmdline icontains linpeas)\n    and isLogLoopSafe\n  output: >\n
    \   A shell script has been used to attempt privilege escalation and enumeratoin.
```

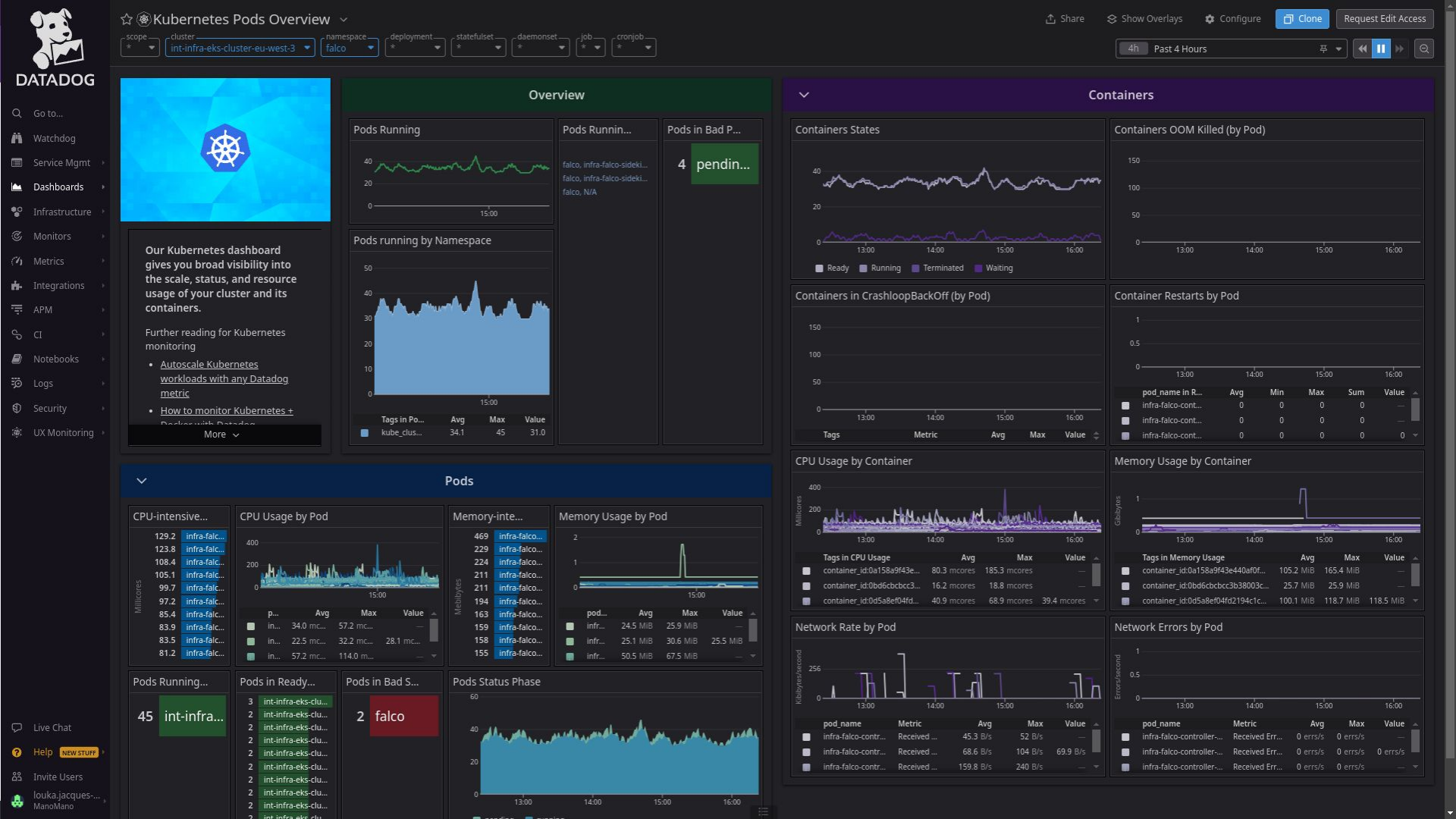
Houston we have a problem !

Datadog TopList & Logs

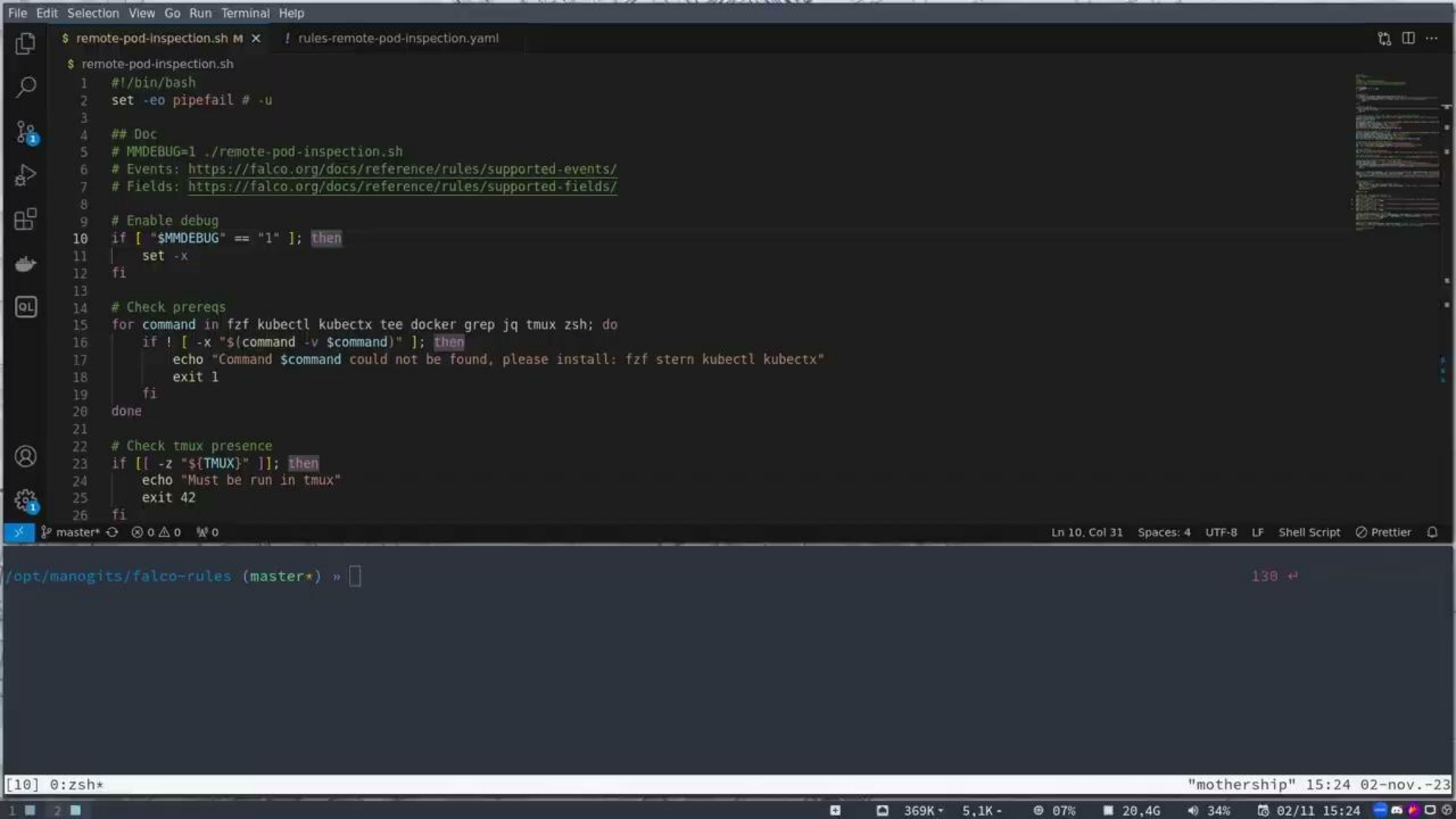


Slack




DATADOG
Kubernetes Pods Overview
scope
cluster
int-infra-eks-cluster-eu-west-3
namespace
falco
deployment
statefulset
daemonset
job
cronjob
Share
Show Overlays
Configure
Clone
Request Edit Access
4h
Past 4 Hours
Go to...
Watchdog
Service Mgmt
Dashboards
Infrastructure
Monitors
Metrics
Integrations
APM
CI
Notebooks
Logs
Security
UX Monitoring
Live Chat
Help
NEW STUFF
Invite Users
louka.jacques-...
ManoMano
Our Kubernetes dashboard gives you broad visibility into the scale, status, and resource usage of your cluster and its containers.
Further reading for Kubernetes monitoring
Autoscale Kubernetes workloads with any Datadog metric
How to monitor Kubernetes +
More
Overview
Pods Running
Pods Runnin...
Pods in Bad P...
4 pendin...
Pods running by Namespace
Containers
Containers States
Containers OOM Killed (by Pod)
Containers in CrashloopBackOff (by Pod)
Container Restarts by Pod
Pods
CPU-intensive...
CPU Usage by Pod
Memory-inte...
Memory Usage by Pod
CPU Usage by Container
Memory Usage by Container
Pods Running...
45 int-infra...
Pods in Ready...
Pods in Bad S...
2 falco
Pods Status Phase
Network Rate by Pod
Network Errors by Pod

DEMO


File Edit Selection View Go Run Terminal Help
$ remote-pod-inspection.sh M
! rules-remote-pod-inspection.yaml
$ remote-pod-inspection.sh
#!/bin/bash
set -eo pipefail # -u
Doc
# MMDEBUG=1 ./remote-pod-inspection.sh
# Events: https://falco.org/docs/reference/rules/supported-events/
# Fields: https://falco.org/docs/reference/rules/supported-fields/
# Enable debug
if ["$MMDEBUG" == "1"]; then
set -x
fi
# Check prereqs
for command in fzf kubectl kubectx tee docker grep jq tmux zsh; do
if ! [-x "$(command -v $command)"]; then
echo "Command $command could not be found, please install: fzf stern kubectl kubectx"
exit 1
fi
done
# Check tmux presence
if [[-z "${TMUX}"]]; then
echo "Must be run in tmux"
exit 42
fi
master*
Ln 10, Col 31
Spaces: 4
UTF-8
LF
Shell Script
Prettier
/opt/manogits/falco-rules (master*) »
130
[10] 0:zsh*
"mothership" 15:24 02-nov.-23
369K
5,1K
07%
20,4G
34%
02/11 15:24

Pros, Cons, & WhatNot

Costs

- **Resource overhead**: Use the Kubernetes cluster resources (CPU/Memory/Network)
- **Storage**: Need to store the logs and events generated by Falco
- **Deployment and management**: Keeping Falco up to date, setup alerts, rules
- **Training and skill development**: Create rules, interpret alerts, setup Falco
- **Integration**: Monitoring and other security solutions

Limitations

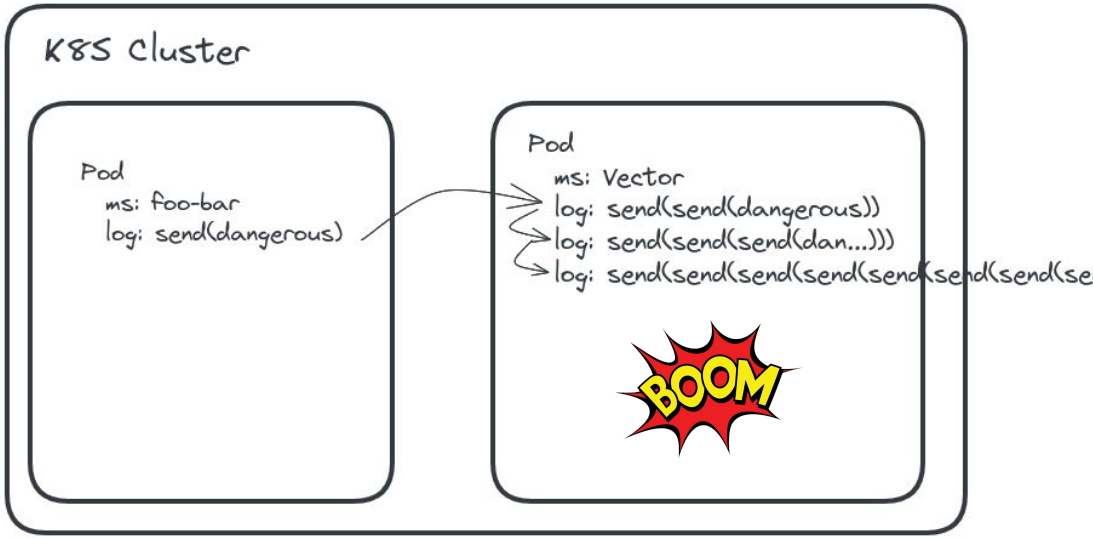
- **Kernel or eBPF features**
- **Limited scope**
- **Complexity**
- **Customization**
- **False positives**
- **Performance impact**
- **Community and support**

Results

- **Default rules:**
  - Crystal Clear cluster vision
- **Custom rules:**
  - Migration from IMDSv1 to IMDSv2
  - Containerd Socket Observability
  - Red-Team “Blind” Fuzzing
  - WebShells Trusted Alerting
  - Fast detection time for *any event*

Instabilities

- High memory usage of some Falco features
- Log tempest
- Load external kernel module
- Kernel < 5.8 need privileged access
- More complexity



On-Call & Incident Switch-Army-Knife

- Have an historic of weird events during an incident
  - Shell opened in a pod
  - Http request received/sent
  - Files used
- Reproduce and analyse
- Alerts for anomalous activity to on-call Devops
- Monitor service response during security fuzzing

Conclusion & Kudos

Increase Observability & Find More Bugs!

- If you are a pentester you will
  - Start loving breaking things again
  - Benefit from a huge time gain
  - Coder a wider attack surface
- If you are a company you will have
  - A small yet very powerful SoC
  - A powerful way to diagnose anything system-related
  - More findings if you provide this during pentests ! 🎁

System Introspection for Micro-Services Pentests in K8S

@TheLaluka
thinkloveshare.com
offenskill.com

Thank you!