

Container Security

»Das einzig sichere System müsste ausgeschaltet, in einem versiegelten und von Stahlbeton ummantelten Raum und von bewaffneten Schutztruppen umstellt sein.«

US-Sicherheitsexperte Gene Spafford

Lernziele

- ★ Sie haben einen Überblick welche Sicherheits-Aspekte bei Containern zu beachten sind.

Zeitlicher Ablauf

- ★ Limitierte Container-Instanzen
- ★ Docker Security
- ★ Docker Security: TUF / Content Trust / Notary
- ★ Docker Security: Vulnerability Scanner
- ★ Weitere
- ★ Reflexion
- ★ Lernzielkontrolle

Limitierte Container-Instanzen

- ★ Die zu erzeugende/zu startende/gestartete Container-Instanz kann auch direkt limitiert werden.
- ★ Dazu brauchen wir zunächst einen Überblick über die aktuell verbrauchten Ressourcen.
 - `docker [container] stats`
- ★ Siehe Übung [03-3-Docker](#) und für Kubernetes: <http://blog.kubecost.com/blog/requests-and-limits/>
- ★ Mögliche Limitierungen
 - `--cpu-shares=512` – CPU shares (relative Gewichtung)
 - `--cpuset-cpus="2-3"` – CPUs Pinning (z. B.: 0-3, oder 0,1)
 - `--cpuset-mems="2,4"` – Speicherknoten (Memory Nodes – MEMs), in denen die Ausführung erlaubt ist, z. B.: 0-3 oder 0,1. Achtung: nur für NUMA-Systeme (eine Computer-Speicher-Architektur für Multiprozessorsysteme)
 - `--cpu-quota=25 #(%)` – Limitierung der CPU Quota via CFS (Completely Fair Scheduler): Limitiert so den maximalen CPU-Verbrauch des Containers
 - `--memory=""` – Maximal zulässiger Speicherverbrauch der Containers (Format: <number>[<unit>]). Number ist dabei ein positiver Integer-Wert, Units wie üblich: **b**, **k**, **m** oder **g**. Minimum sind **4M**.

Docker Security: TLS/SSL

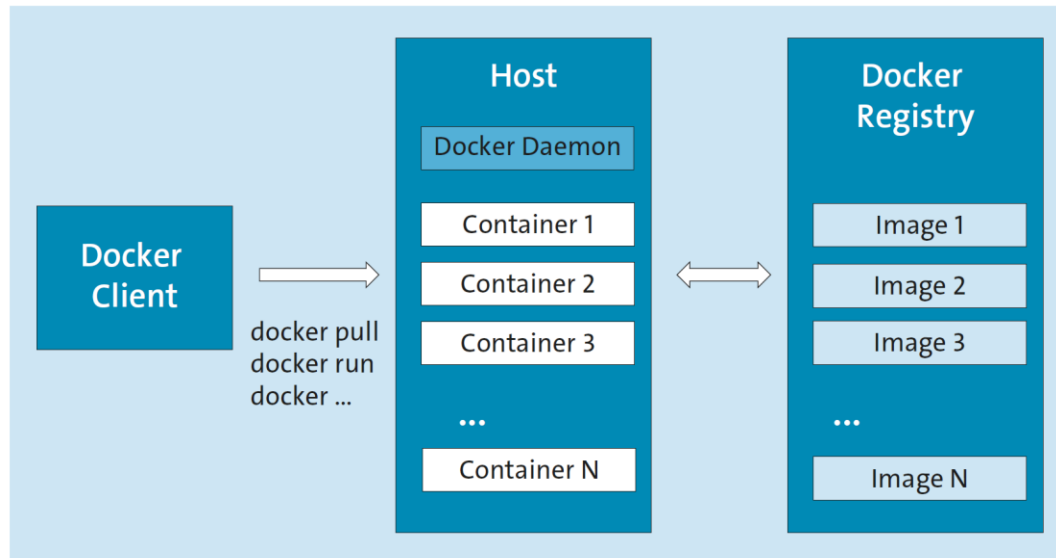


Abbildung 4.1 Docker Client, Daemon und Registry

- ★ Der Docker Client kommuniziert, defaultmässig, per Unix-Socket mit unserem Docker Daemon.
- ★ Ein (remote) Docker Host, kann aber auch via http-Socket, angesprochen werden.
- ★ **Empfehlungen:**
 - Die Kommunikation ist zwingend zu verschlüsseln (**Port 2376**, nicht 2375) und Verwendung eines Hosts Zertifikates (siehe Kubernetes Installation).
 - Bei Verwendung einer eigenen Registry ist das HTTPS Protokoll zu verwenden!
 - Mindestens TLSv1 [SSLv3.1] Verwenden!

Docker Security: TUF / Content Trust / Notary (1)

★ Vorbetrachtungen

- Mangelhafte Sicherheit im Vergleich zu realen Maschinen oder VMs, was die Isolation der Namespaces angeht.
- Keine Transparenz, was den Image Build angeht, sofern die Images aus externen Quellen stammen.
- Vertrauenswürdigkeit von externen Quellen/Repos/Registries

★ Per Default aktiv: TUF (The Update Framework)

- TUF stellt keine konkrete Software oder ein Tool dar, sondern versteht sich auch als [Spezifikation](#).
- **TUF** kann im einfachsten Fall als Helfer für die Softwareverwaltung verstanden werden, welche z. B. auf Basis von digitalen Signaturen, Meta-Informationen und korrespondierenden Schlüsseln prüft, ob eine neuere Version einer Software verfügbar ist und diese auch als valide bzw. »**sauber**« anzusehen

Docker Security: TUF / Content Trust / Notary (2)

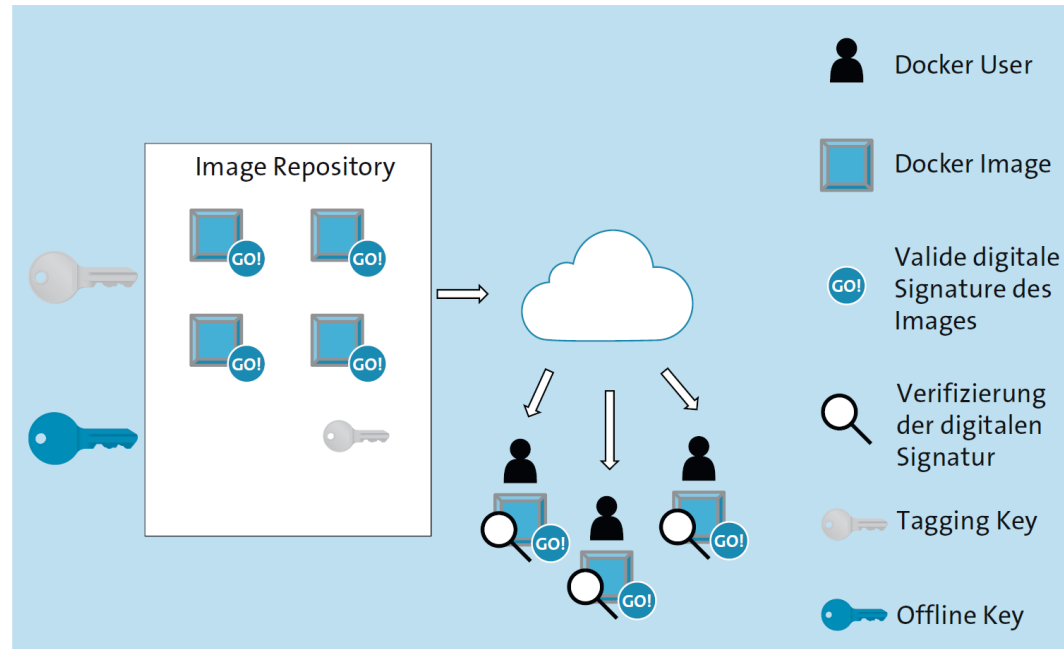


Abbildung 5.1 Docker und TUF

★ Docker Content Trust (DCT)

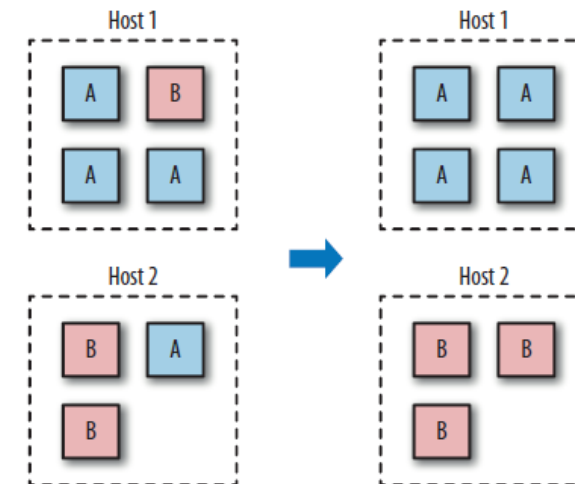
- Um DCT zu verwenden muss explizit entweder bei pull, push, create, run oder build die Zusatzkommandooption `--disable-content-trust=false` angegeben, oder die Docker-Umgebungsvariable `DOCKER_CONTENT_TRUST=1` gesetzt werden.
- **Bei aktivierten DCT funktioniert docker CLI nur mit Signierten Images!**
- **Notary** kann als Tool verstanden werden, das auf TUF aufsetzt und den DCT (Docker Content Trust) implementiert.

Docker Security: Vulnerability Scanner

- ★ Eine ebenfalls sehr wichtige ergänzende Maßnahme kann sicher die kontinuierliche Überwachung bereits existierender Images auf potenzielle Schwachstellen sein.
- ★ Wie üblich existieren mehrere Ansätze, einer davon ist Clair, ein anderer Bench. Beides sind Vulnerability- d. h. Schwachstellen-Scanner:
 - [Clair](#) für Images
 - [Bench](#) legt den Fokus eher auf den Docker Host.

Weitere

- ★ [Docker Logging](#)
- ★ [cAdvisor](#) (eine Abkürzung von Container Advisor) von Google ist das am häufigsten eingesetzte Monitoring-Tool für Docker.
- ★ Docker-Berechtigungen == Root-Berechtigungen (siehe auch [A Look at Rootless Containers](#))
 - `docker run -v /:/homeroot -it ubuntu bash`
- ★ Empfehlungen:
 - Einen User setzen
 - Netzwerkzugriffe von Containern beschränken
 - `setuid/setgid`-Binaries entfernen
 - Neustarts begrenzen (z.B. `--restart=on-failure:10`)
 - Zugriffe auf die Dateisysteme begrenzen (`--read-only`)
 - Container nach Host trennen



Microservice Patterns: The Twelve Factors App - <https://12factor.net/de/>

★ I. Codebase

- Eine im Versionsmanagementsystem verwaltete Codebase, viele Deployments

★ II. Abhängigkeiten

- Abhängigkeiten explizit deklarieren und isolieren

★ III. Konfiguration

- Die Konfiguration in Umgebungsvariablen ablegen

★ IV. Unterstützende Dienste

- Unterstützende Dienste als angehängte Ressourcen behandeln

★ V. Build, release, run

- Build- und Run-Phase strikt trennen

★ VI. Prozesse

- Die App als einen oder mehrere Prozesse ausführen

★ VII. Bindung an Ports

- Dienste durch das Binden von Ports exportieren

★ VIII. Nebenläufigkeit

- Mit dem Prozess-Modell skalieren

★ IX. Einweggebrauch

- Robuster mit schnellem Start und problemlosen Stopp

★ X. Dev-Prod-Vergleichbarkeit

- Entwicklung, Staging und Produktion so ähnlich wie möglich halten

★ XI. Logs

- Logs als Strom von Ereignissen behandeln

★ XII. Admin-Prozesse

- Admin/Management-Aufgaben als einmalige Vorgänge behandeln

open software security community (OWASP)

Welcome to OWASP

the free and open software security
community

• [Dependency
Check](#)

[Donate](#)

[About](#) • [Searching](#) • [Editing](#) • [New Article](#) • [OWASP Categories](#) • [CONTACT-US](#)



Every vibrant technology marketplace needs an unbiased source of information on best practices as well as an active body advocating open standards. In the Application Security space, one of those groups is the Open Web Application Security Project (or OWASP for short).

The Open Web Application Security Project (OWASP) is a [501\(c\)\(3\)](#) worldwide not-for-profit charitable organization focused on improving the security of software. Our mission is to make software security [visible](#), so that [individuals and organizations](#) are able to make informed decisions. OWASP is in a unique position to provide impartial, practical information about AppSec to individuals, corporations, universities, government agencies and other organizations worldwide. Operating as a community of like-minded professionals, OWASP issues software tools and knowledge-based documentation on application security.

Everyone is free to participate in OWASP and [all of our materials](#) are available under a free and open software license. You'll find everything [about OWASP](#) here on or linked from our wiki and current information on our [OWASP Blog](#). OWASP **does not endorse or recommend commercial products or services**, allowing our community to remain vendor neutral with the collective wisdom of the best minds in software security worldwide.

IT-Grundschutz-Baustein SYS.1.6 Container



 LEICHTE SPRACHE  GEBÄRDENSPRACHE [ENGLISH](#) [KONTAKT](#) [LOGIN](#)



[Themen](#) | [Das BSI](#) | [Presse](#) | [Publikationen](#) | [Service](#)

IT-Grundschutz

Community Draft zum IT-Grundschutz-Baustein SYS.1.6 Container

Datum 15.05.2018

Ziel dieses Bausteins ist der Schutz von Informationen, die in Containern verarbeitet, angeboten oder darüber übertragen werden. Der Baustein behandelt daher, wie Container grundsätzlich abgesichert, wie sie orchestriert und wie die verwendeten Images mit bordeigenen Mitteln verwaltet werden können, unabhängig vom Einsatzzweck des im Container betriebenen Dienstes bzw. der Anwendung. Dabei wird zwischen dem eigentlichen Container-Dienst, also der Software, die für Betrieb und Verwaltung der Container zuständig ist, und den Anwendungsdiensten, die in den Containern ausgeführt werden, unterschieden.

Inhaltsverzeichnis

[Veröffentlichungsworkflow](#)

[Final Drafts](#)

[Benutzerdefinierte Bausteine](#)

https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/IT-Grundschutz-Modernisierung/BS_Container.html



Übung: Container Security (04-1-ContainerSecurity)

Vertrauenswürdigkeit von externen Quellen/Repos/Registries

Verwendung von Docker Content Trust

- schlägt bei den unsigned Images fehl
- Funktioniert mit den Offiziellen Images von Docker Hub, wie z.B. postgres

```
! docker pull --disable-content-trust=false marcel1691/fhem
```

```
! (export DOCKER_CONTENT_TRUST=1 && docker pull marcel1691/fhem)
```

```
! docker pull --disable-content-trust=false postgres  
! docker image ls
```

Docker-Berechtigungen == Root-Berechtigungen

- Startet `dockerps.bat` im Verzeichnis `misegr`.
- Wechselt in die VM `vagrant ssh`
- Startet `docker run -v /:/homeroot -it ubuntu bash` und legt ein paar Verzeichnisse im `/` an.
- Beendet den Container mittels `exit`
- Überprüft die angelegten Verzeichnisse mittels `ls -l /`



Übung: Container Security – Unsichere Web Anwendung (Optional)

★ **WebGoat** ist eine bewusst unsichere Web-Anwendung. Sie wurde entwickelt für den Unterricht.

★ **Starten**

- `kubectl apply -f duk/owasp/webgoat-deployment.yaml`
- `kubectl apply -f duk/owasp/webgoat-service.yaml`
- `startsvc webgoat`
- Hinter dem geöffneten URL `/WebGoat/start.mvc` anfügen.

★ **Links**

- Doku zum Beispiel: <https://github.com/mc-b/duk/tree/master/owasp>
- WebGoat Webseite: https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project



Reflexion

- ★ Um Docker sicher einzusetzen, müssen Ihnen die potenziellen Sicherheitslücken bewusst sein, und Sie sollten die wichtigsten Tools und Techniken kennen, mit denen Sie containerbasierte Systeme absichern können.

? Lernzielkontrolle

- ★ Sie haben einen Überblick welche Sicherheits-Aspekte bei Containern zu beachten sind