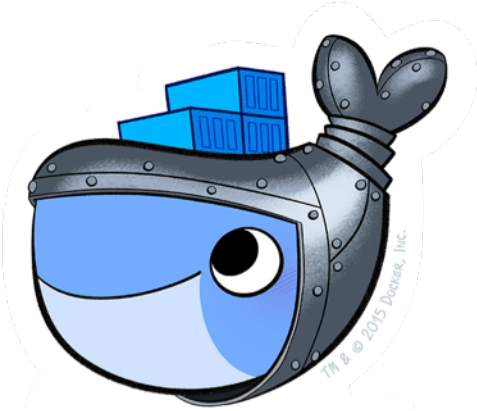
Docker Security #2

Security Best Practices for Docker Host, Docker Daemon, Images, Container, Operations



whoami

- Martin Pizala

- 2014 System Engineer at Micromata
- 2015 Focused on IT-Security & InfoSec
  - Vulnerability assessment
  - Security Best Practices & Hardening
  - Infrastructure & Web Application Penetration Testing, OSCP
  - Secure Architecture / Security in CI/CD / SecDevOps
- 2017 IT-Security ~~Engineer / Analyst / Consultant~~ Enthusiast

Agenda

- No Introduction into Docker
- Docker Security #1 (SMKS #17 – 13.09.2017)
  - Architecture and Design (-Flaws)
  - Docker Security Basics
  - Docker Penetration Testing
- Docker Security #2 (SMKS #19 – 14.11.2017)
  - Security Best Practices for
    - OS/Docker Host
    - Docker Daemon
    - Images / Registries
    - Container
    - Operations

OS Host - Basics

- OS Hardening
  - (Security-) Updates (Patch early, Patch often)
  - Use Kernel-Security (secomp, AppArmor / SELinux)
  - Principle of minimal ...
    - Privilege, Packages, Services, etc

- Harden all your daemons and use good crypto
  - google for
    - $product security best practices
    - $product hardening
  - Transport Layer Security / OpenSSH (BetterCrypto, Mozilla SSL-Config Generator)
  - Web-Server & Web-Browser Security (Mozilla Observatory)

  => Read && understand || (read more || pay for it)

- Perform Vulnerability Assessment

OS Host - Advanced

- Perform Audits / Pentests
- Perform backups and practice restores regularly
- Perform Monitoring
  - Operations (Load, Memory, Disk Usage, Services, ...)
  - Security (HIDS, Audit-Logs, Network, Firewall ...)
  - Log-Management (ship to an external system)

Docker Best Practices

- Docker Security
  https://docs.docker.com/engine/security/security/

- Understanding Docker Security and Best Practices
  https://blog.docker.com/2015/05/understanding-docker-security-and-best-practices/

- Dockerfile Best Practices
  https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices/

- Aquasec Docker Security Best Practices 2017
  https://blog.aquasec.com/docker-security-best-practices

- 10 things to avoid in docker containers
  https://developers.redhat.com/blog/2016/02/24/10-things-to-avoid-in-docker-containers/

Docker Best Practices Audit

- CIS Benchmark for Docker
  1. Host Configuration
  2. Docker Daemon Configuration
  3. Docker Daemon Configuration Files
  4. Container Images and Build Files
  5. Container Runtime
  6. Docker Security Operations

- Center for Internet Security / NGO
- Kostenloser Download nach Registrierung

- Audit Tool Docker Bench for Security

Docker Bench for Security

- https://github.com/docker/docker-bench-security

```
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.3
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Community Edition Benchmark v1.1.0.
# ------------------------------------------------------------------------------

Initializing Fri Jul 14 09:18:42 UTC 2017


[INFO] 1 - Host Configuration
[WARN] 1.1  - Ensure a separate partition for containers has been created
[NOTE] 1.2  - Ensure the container host has been Hardened
[PASS] 1.3  - Ensure Docker is up to date
[INFO]      * Using 17.06.0 which is current
[INFO]      * Check with your operating system vendor for support and security maintenance for Docker
[INFO] 1.4  - Ensure only trusted users are allowed to control Docker daemon
[INFO]      * docker:x:992:vagrant
[WARN] 1.5  - Ensure auditing is configured for the Docker daemon
[WARN] 1.6  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.7  - Ensure auditing is configured for Docker files and directories - /etc/docker
[WARN] 1.8  - Ensure auditing is configured for Docker files and directories - docker.service
[INFO] 1.9  - Ensure auditing is configured for Docker files and directories - docker.socket
[INFO]      * File not found
[INFO] 1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO]      * File not found
[INFO] 1.11 - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO]      * File not found
[WARN] 1.12 - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-containerd
[WARN] 1.13 - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-runc


[INFO] 2 - Docker daemon configuration
[WARN] 2.1  - Ensure network traffic is restricted between containers on the default bridge
[PASS] 2.2  - Ensure the logging level is set to 'info'
[PASS] 2.3  - Ensure Docker is allowed to make changes to iptables
[PASS] 2.4  - Ensure insecure registries are not used
[PASS] 2.5  - Ensure aufs storage driver is not used
```

Docker Host - Basics

- Create a separate partition for docker data
  - /var/lib/docker && /wherever/your/persistent/container/data/is
  - Ops know why
- Ensure Docker files and directories are protected
  - Pentesters know why
- Keep Docker up to date
  - Everybody know why
- Allow only trusted users (and definded pipelines) to control a Docker daemon
  - You know why?

Docker Host - Access to dockerd

- Allow only trusted users (and definded pipelines) to control a Docker daemon

- User in Docker Group = root@host

```
user:docker@host = root@container = root@host = dockerd
```

- Docker Daemon Socket Options

File

```
dockerd -H fd:// OR -H unix:///var/run/docker.sock
=> srw-rw---- root docker /var/run/docker.sock
```

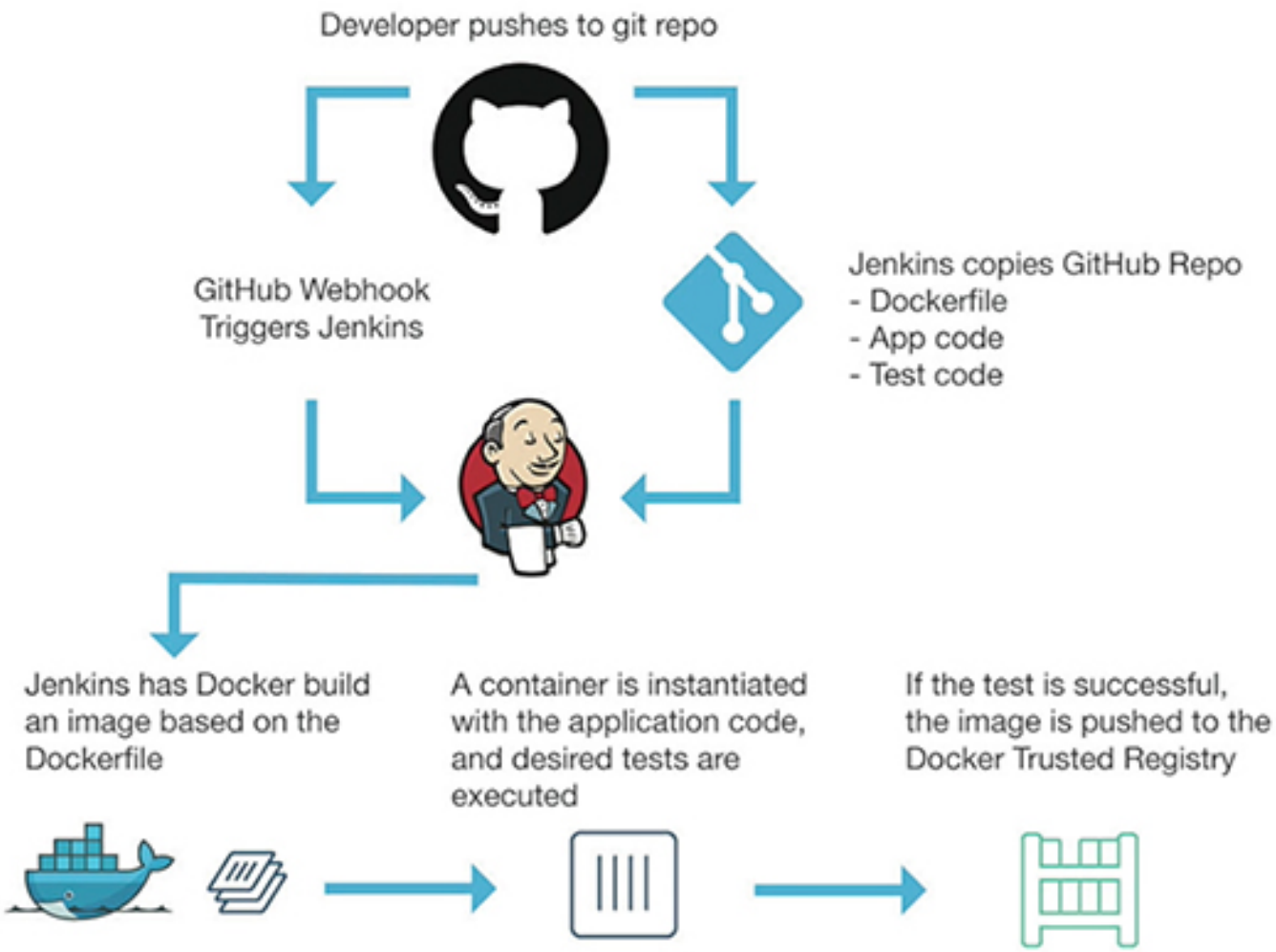
Only root and docker group members has access

TCP

```
dockerd -H tcp://0.0.0.0:2375
=> TCP *:2375 (LISTEN)
```

Misconfiguration, when not used with TLS and TLS-AUTH

Docker Host - Definded pipeline



EXPLOIT DATABASE

Home Exploits Shellcode Papers Google Hacking Database Submit Search

Docker Daemon - Unprotected TCP Socket

EDB-ID: 42356	**Author**: Martin Pizala	**Published**: 2017-07-20
CVE: N/A	**Type**: Local	**Platform**: Linux
E-DB Verified:	**Exploit**: Download / View Raw	**Vulnerable App**: N/A

« Previous Exploit Next Exploit »

```
# Exploit Title: Docker Daemon - Unprotected TCP Socket
# Date: 20-07-2017
# Exploit Author: Martin Pizala
# Vendor Homepage: https://www.docker.com
# Software Link: https://www.docker.com/get-docker
# Version: Since 0.4.7 (2013-06-28) (feature: mount host directories)
# Tested on: Docker CE 17.06.0-ce and Docker Engine 1.13.1

1. Description

Utilizing Docker via unprotected tcp socket (2375/tcp, maybe 2376/tcp with tls but without tls-auth), an attacker can create a
docker container with the '/' path mounted with read/write permissions on the host server that is running the docker container and
use chroot to escape the container-jail.

2. Proof of Concept

docker -H tcp://<ip>:<port> run --rm -ti -v /:/mnt alpine chroot /mnt /bin/sh

3. Solution:

Protect the tcp socket
https://docs.docker.com/engine/reference/commandline/dockerd/#bind-docker-to-another-hostport-or-a-unix-socket
https://docs.docker.com/engine/security/https/
```

EXPLOIT DATABASE

Home Exploits Shellcode Papers Google Hacking Database Submit Search

Docker Daemon - Unprotected TCP Socket (Metasploit)

EDB-ID: 42650	**Author**: Metasploit	**Published**: 2017-09-11
CVE: N/A	**Type**: Remote	**Platform**: Python
Aliases: N/A	**Advisory/Source**: Link	**Tags**: Metasploit Framework (MSF)
E-DB Verified: ✔	**Exploit**: Download / View Raw	**Vulnerable App**: N/A

« Previous Exploit Next Exploit »

```
##
# This module requires Metasploit: https://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

class MetasploitModule < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::HttpClient
  include Msf::Exploit::FileDropper

  def initialize(info = {})
    super(update_info(info,
      'Name'           => 'Docker Daemon - Unprotected TCP Socket Exploit',
      'Description'    => %q{
        Utilizing Docker via unprotected tcp socket (2375/tcp, maybe 2376/tcp
        with tls but without tls-auth), an attacker can create a Docker
        container with the '/' path mounted with read/write permissions on the
        host server that is running the Docker container. As the Docker
        container executes command as uid 0 it is honored by the host operating
        system allowing the attacker to edit/create files owned by root. This
        exploit abuses this to creates a cron job in the '/etc/cron.d/' path of
        the host server.
```

Jonathan Cran @jcran · 10. Sep.

Handy! Metasploit exploit for unauth'd Docker daemon

bit.ly/2gTsObS

Original (Englisch) übersetzen

3 6 9

MARK MANNING

@antitree

Folgen

Antwort an @jcran

Probably the best reason to enable namespace support on your Docker daemon

Original (Englisch) übersetzen

04:50 - 11. Sep. 2017

1 „Gefällt mir"-Angabe

1

Docker Daemon - User namespaces

- User namespaces

  is a mechanism for remapping UIDs inside a container

  => Container **UID =** Host **UID + Subordinate UID**

```
docker run -v /:/mnt debian:8 touch /mnt/tmp/file
Container:     -rw-r--r-- root root /mnt/tmp/file
Host:          -rw-r--r-- 100000 100000 /tmp/file
```

```
docker run -v /:/mnt --user 1000:1000 debian:8 touch /mnt/tmp/file
Container:     -rw-r--r-- root root /mnt/tmp/file
Host:          -rw-r--r-- 101000 101000 /tmp/file
```

Docker Daemon - User namespaces

- How to setup User namespaces

```
# Add a unprivileged user
useradd someuser

# Specify subordinate UID and GID ranges
cat /etc/subuid
someuser:100000:65536
cat /etc/subgid
someuser:100000:65536

# Enable User Namespaces in dockerd
dockerd --userns-remap=someuser
```

Docker Daemon - User namespaces

- User namespaces - Good protection

- root@container has access like
  - nobody@host
  - + persistent docker files with owned by uid/gids in subordinate Range

```
docker run -v /:/mnt debian:8 cat /mnt/etc/shadow
cat: /mnt/etc/shadow: Permission denied
```

Docker Daemon - User namespaces

- User namespace known limitations

  The following standard Docker features are incompatible with running a Docker daemon with user namespaces enabled:

  - **sharing PID or NET namespaces** with the host (--pid=host or --network=host).
  - **external (volume or storage) drivers** which are unaware or incapable of using daemon user mappings.

Docker Daemon - User namespaces

- User namespace known limitations

  The following standard Docker features are incompatible with running a Docker daemon with user namespaces enabled:

  - **sharing PID or NET namespaces** with the host (--pid=host or --network=host).
  - **external (volume or storage) drivers** which are unaware or incapable of using daemon user mappings.

- **Could be disabled!**

  - Using the **--privileged** mode flag on docker run without also specifying **--userns=host**.

    ```
    docker run --privileged --userns=host -v /:/mnt debian:8 cat /mnt/etc/shadow
    ```

    Protect your Daemon. Nothing else!

Docker Daemon - Network

- Restricted network traffic between containers on the default bridge

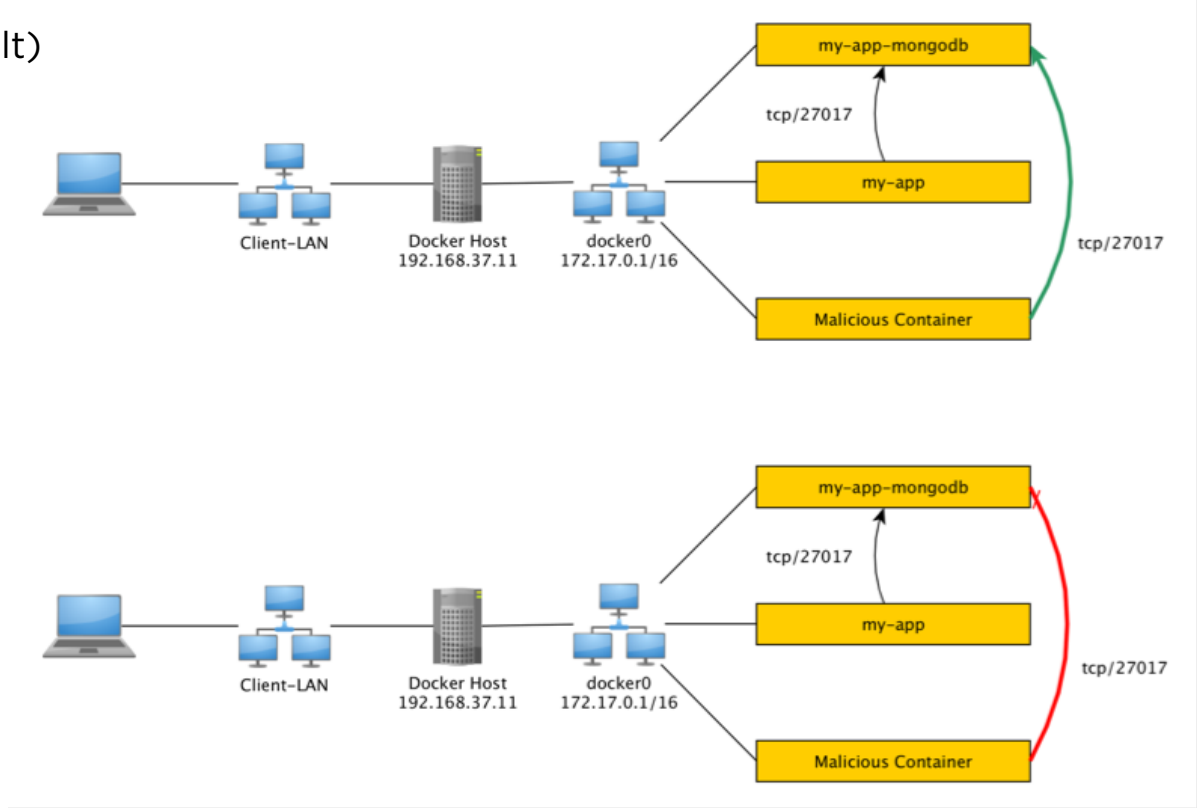
  Inter-Container-Communication

  ```
  # inspect
  docker network inspect --format '{{ .Name }}: {{ .Options }}' \
  $(docker network ls -q)

  # dockerd options
  ps aux | grep [d]ockerd

  dockerd --icc=false # not default
  ```

- Only use default bridge when icc=false and your container is standalone

Docker Daemon - Network

- Inter-Container-Communication
  - icc=true (default)



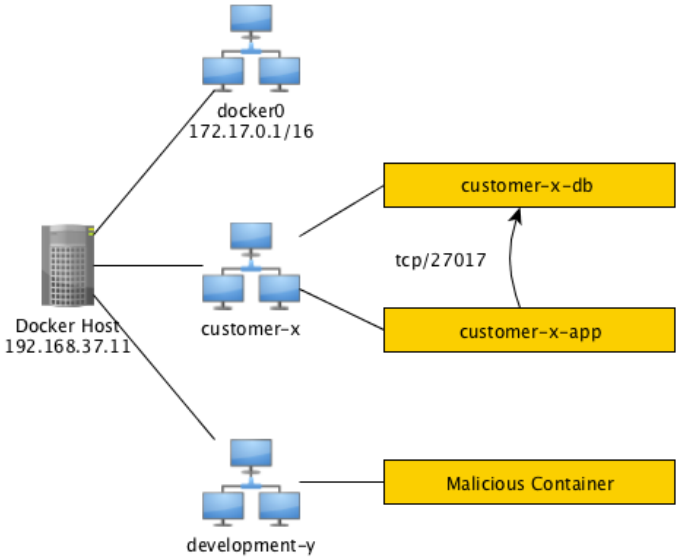
  - icc=false

Docker Daemon - Network

- Only use default bridge when icc=false and your container is standalone

- Create a network for each stack

```
docker network create customer-x
docker run --network=customer-x --name customer-x-db image-db
docker run --network=customer-x --name customer-x-app image-app
```



Docker Daemon - Network

- High Protection / in Production
  - Disable Legacy Registry
  - Use SCM, Build-Pipeline and private Registry instead
  - Know your network traffic (inbound / forward / outbond)
  - Proxy your traffic when needed

Docker Images

- Docker Images
- Docker Build
- Docker Registries and 3rd-party

Docker Images

offical (docker hub)

```
docker run/pull image[:tag]
                debian:latest
                centos
                mongo
```

public / 3rd-party (docker hub)

```
                username/imagename[:tag]
                xxx/debian:latest
```

3rd-party (self hosted registry)

```
                hostname[:port]:username/imagename[:tag]
                server.name:trustme/debian:latest
```

Docker Images

- Docker Hub, Docker Store, Docker Cloud, Docker Notary (Docker, Inc.)
  - offical: maintained, reviewes images with process transparency
  - public: Have you ever heard **HUB**

  - trustworthy: transparent, reputation, sources
  - untrustworthy: not transparent, no reputation, blobs / no sources

  - secure: used not docker run flags
  - unsecure: privileged, default creds, no crypto

  - Same for 3rd-party Repo like github

=> **understand and reconstruct 3rd-party images**

Anything else is something like blind code execution from github with root access

Docker Images - Base Images

- What is debian:latest

- Go to docker hub
- Search debian
- Open debian offical
- Search latest under "Supported tags and respective Dockerfile links"
- Open *stretch/Dockerfile*

- *Ah FROM SCRATCH?*

Docker Images - Base Images

- FROM parent image
  A parent image is the image that a image is based on

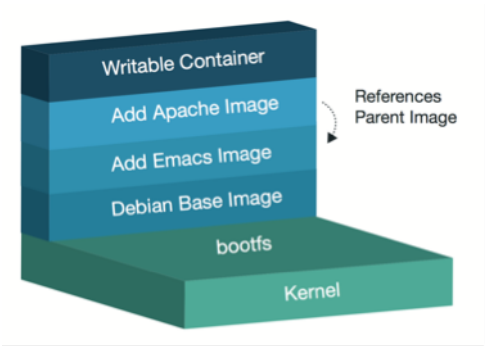
- FROM scratch
  A base image has no parent

```
FROM scratch
ADD rootfs.tar.xz /
CMD ["bash"]

debootstrap jessie rootfs.tar.xz
```

Docker Images

- Dockerfiles

```
cd ~/my-apache2-image/
cat Dockerfile
FROM debian:latest
RUN apt-get && \
    apt-get install emacs-nox
RUN apt-get install apache2
CMD httpd -DFOREGROUND
```



- Docker Build

```
docker build --no-cache -t my-apache2-image .
```

- Docker Run (create and start container)

```
docker run my-apache2-image
```

Docker Images - Secure Images

- Write readable and comment, if needed

- Implement update mechanism and install software from primary source

```
RUN apt-get update && \
    apt-get full-upgrade -y
ADD openjdk8.tar.gz
RUN apt-get install -y openjdk8
```

- Drops privileged

```
RUN useradd -m user
USER user
WORKDIR /home/user
```

- Write healtchecks

```
HEALTHCHECK --interval=1m CMD curl -f http://127.0.0.1:8080/ || exit 1
```

Docker Images - Paranoid Secure Image

- Install all Patches
- Remove all setuid-bits
- Remove world-writable permissions.
- Remove unnecessary user accounts.
- Remove interactive login shell for everybody but user.

Docker Images - Paranoid Secure Image

```
FROM debian
RUN apt-get update && \
    apt-get -y dist-upgrade
# Remove all setuid-bits
RUN find / -xdev -perm /g=s -type f -exec chmod g-s {} \; && \
    find / -xdev -perm /u=s -type f -exec chmod u-s {} \;
# Remove world-writable permissions.
# This breaks apps that need to write to /tmp
RUN find / -xdev -type d -perm /o+w -exec chmod o-w {} \; && \
    find / -xdev -type f -perm /o+w -exec chmod o-w {} \;
RUN useradd -m -s /bin/bash user
# Remove unnecessary user accounts.
RUN sed -i -r '/^(user|root)/!d' /etc/group && \
    sed -i -r '/^(user|root)/!d' /etc/passwd && \
    sed -i -r '/^(user|root)/!d' /etc/shadow
# Remove interactive login shell for everybody but user.
RUN sed -i -r '/^user:/! s#^(.*):[^:]*$#\1:/usr/sbin/nologin#' /etc/passwd
USER user
WORKDIR /home/user
CMD ["/bin/bash"]

# docker run -ti -d --cap-drop ALL --privileged=false --user 1000:1000 debian-secured bash
```

Docker Images

- Ensure
  - that containers use trusted base images
  - unnecessary packages are not installed in the container
  - images are scanned and rebuilt to include security patches
  - Content trust for Docker is Enabled
  - HEALTHCHECK instructions have been added to the container image
  - update instructions are not use alone in the Dockerfile
  - setuid and setgid permissions are removed in the images
  - COPY is used instead of ADD in Dockerfile
  - secrets are not stored in Dockerfiles
  - verified packages are only Installed

=> Read CIS Docker Benchmark

Docker Registries

- Docker Hub, Docker Store, Docker Cloud, Docker Notary (Docker, Inc.)

- Self-hosted
  - Private Registry

- 3rd-party
  - Private Registry (cloud providers, with awesome features)
    - like vulnerability scanner

Docker Registries

- Setup Docker Private Registry v2

```
# server
docker run -d -p 5000:5000 --name registry registry:2

# developer-1
docker pull debian:8
docker tag debian:8 192.168.161.5:5000/company/debian:8
docker push 192.168.161.5:5000/company/debian:8

# server
docker run 192.168.161.5:5000/company/debian:8
```

Docker Registries for Penetration Tester

Docker Registries for Penetration Tester

- Attacking a unproteced Docker Private Registry v2

```
# attacker
docker pull 192.168.161.5:5000/company/debian:8
# inspect
docker history $imageid
# extend (new layers, or from scratch)
cat Dockerfile
  FROM 192.168.161.5:5000/company/debian:8
  ADD backdoor
docker build -t my-new-image .
docker tag my-new-image 192.168.161.5:5000/company/debian:8
# push
docker push 192.168.161.5:5000/company/debian:8
```

Docker Registries - Secure Private Registry

- Secure Docker Private Registry v2
  - Use TLS and TLS-AUTH
  - No User Directory
  - No ACLs

- Use SUSE Portus
  - Use TLS
  - Authorization service and frontend for Docker registry (v2)
  - User Directory (with eg LDAP Support)
  - Roles and Permissions

Docker Registries

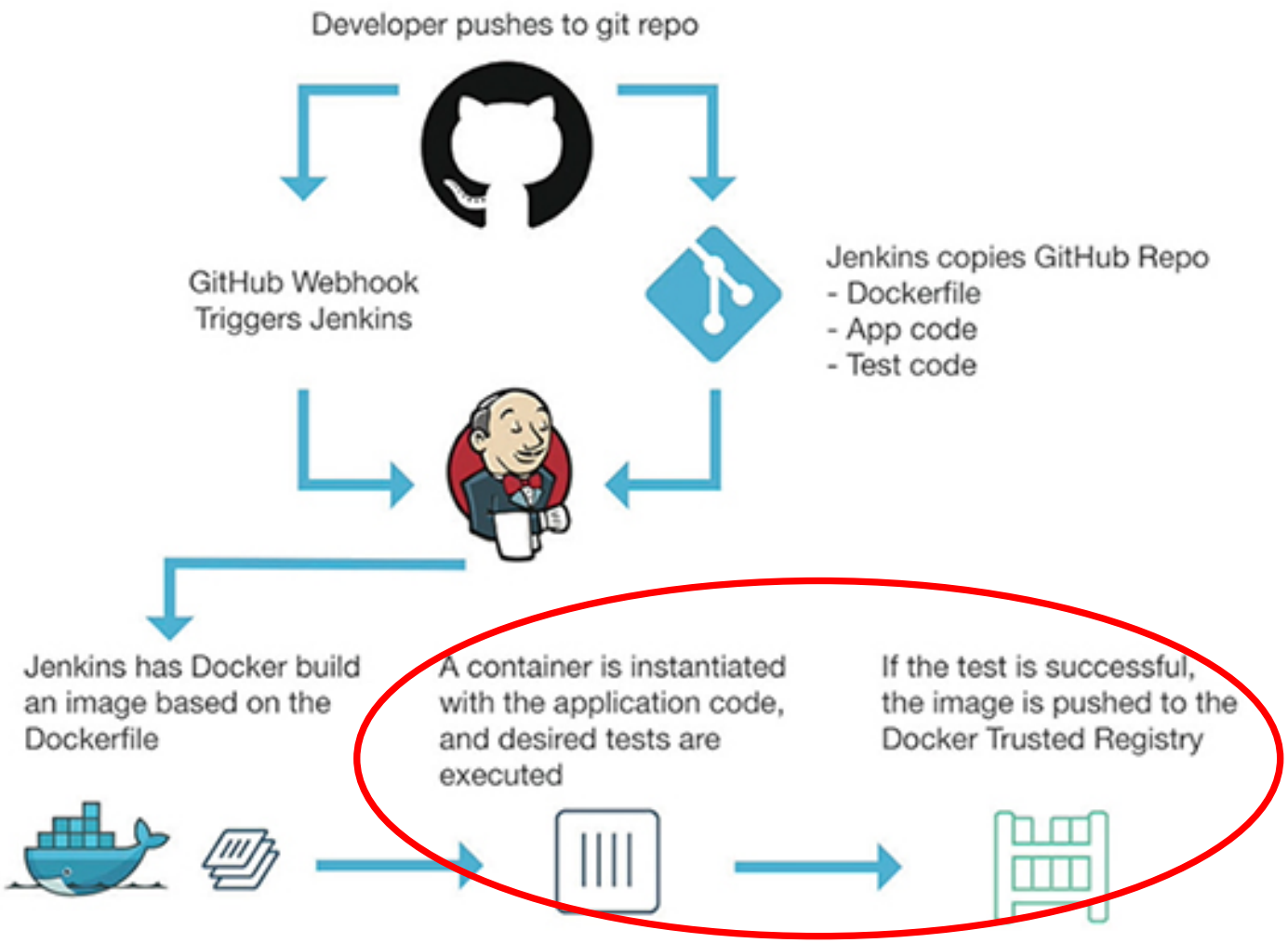
- 3rd-party companies
  - Private Registry (cloud providers, with awesome features)
    - like vulnerability scanner

Docker Registries - Image Scanner

- Cloud
  - Docker, Inc.
    - Login to docker hub and take a look at https://hub.docker.com/r/library/debian/tags/latest/
  - Tenable Flawcheck
  - CoreOS Quay

- Self-Hosted
  - CoreOS Clair

Docker Registries - Image Scanner



OFFICIAL REPOSITORY

debian ☆

Last pushed: 11 days ago

Repo Info Tags

Scan results for **debian:latest**

3 of 37 components are vulnerable

Scanned 2 days ago

Provide Feedback

CVE-2017-15670
The GNU C Library (aka glibc or libc6) before 2.27 contains an off-by-one error leading to a heap-based buffer overflow in the glob function in glob.c, related to the processing of home directories using the ~ operator followed by a long string.

Layers

1 ADD file:a71e077a

Compressed size:

COMPONENT		SEVERITY
glibc 2.24-11+deb9u1 LGPL:Lgpl License	CVE-2017-15670	Critical
	CVE-2017-15804	Critical
	CVE-2017-15671	Major
berkeleydb 5.3.28-12+deb9u1 sleepycat:Copyleft License	CVE-2016-0692	Major
	CVE-2016-0694	Major
	CVE-2016-3418	Major
	CVE-2016-0682	Major
	CVE-2016-0689	Major
systemd 232-25+deb9u1 LGPL:Lgpl License	CVE-2017-15908	Major

Components

Dashboard | FlawCheck Pri

https://registry.flawcheck.com

Anthony Bettini
FlawCheck

- Dashboard
- Repositories
- Inspection Reports
- Vulnerability DB
- Teams
- People
- Invite Team Members
- Billing
- Settings
- Documentation
- API
- Getting Started

Welcome to FlawCheck Private Registry | Log out

Repositories (Total)	Container Images (Current)	Pushes (Total)	Pulls (Total)
92	1148	1149	15

Container Risk (Average)	Vulnerabilities Discovered (Total)	Vulnerabilities per Image (Average)	Malware Discovered (Total)
6	33362	29	0
Average risk is moderate	High for this many repos	High number discovered	No malware found

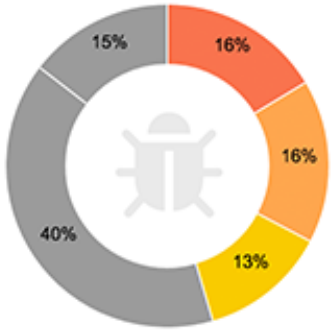
Vulnerability Distribution By CVSS Scores Distribution of vulnerabilities by Common Vulnerability Scoring System (CVSS) scores across all container images in all repositories

Privacy Policy | Terms of Service | Security | Support

QUAY Repositories Tutorial Docs Blog Quay User

exmaple/nginx

99009dfc5e95



Quay Security Scanner has detected **55** vulnerabilities.

Patches are available for **15** vulnerabilities.

- **9** High-level vulnerabilities.
- **9** Medium-level vulnerabilities.
- **7** Low-level vulnerabilities.
- **22** Negligible-level vulnerabilities.
- **8** Unknown-level vulnerabilities.

Image Vulnerabilities

Filter Vulnerabilities... Only show fixable

CVE	SEVERITY	PACKAGE	CURRENT VERSION	FIXED IN VERSION	INTRODUCED IN IMAGE
CVE-2016-2108	10 / 10	openssl	1.0.1k-3+deb8u4	1.0.1k-3+deb8u5	RUN apt-key adv --keyserver hkp...
CVE-2016-3191	High	pcre3	2:8.35-3.3+deb8u2	2:8.35-3.3+deb8u3	ADD file:b5391cb13172fb513dbfca...

Docker Registries - Secure Private Registry

- Secure Docker Private Registry v2
  - Use TLS and TLS-AUTH
  - No User Directory
  - No ACLs

- Use SUSE Portus
  - Use TLS
  - Authorization service and frontend for Docker registry (v2)
  - User Directory (with eg LDAP Support)
  - Roles and Permissions

Docker Image Scanner - Clair

- Integrate Clair into Portus

coreos/dex

0 Delete repository

Tags

Tag	Author	Image	Pushed at	Security
unrelated	mssola	baa5d63471ea	2017-07-14T15:06:31.000Z	2 vulnerabilities
another v2.2.1	mssola	020daa9b782f	2017-07-14T15:06:30.000Z	2 vulnerabilities

Tag: unrelated (repository: dex)

clair

- CVE-2016-8859 (severity: High)
- CVE-2016-6301 (severity: High)

Docker Container

- Donts
- Avoids
- Dos

Docker Container - Dont's

=> Especially when untrusted 3rd-party

- Do not mount /var/run/docker.sock or /var/lib/docker

```
docker run -v /var/run/docker.sock:/var/run/docker.sock debian:8
apt install netcat-openbsd
nc -U /var/run/docker.sock
```

- Avoid privileged containers

```
docker run --privileged debian:8
mkdir /ROOT; mount /dev/sda1 /ROOT/
touch /ROOT/touched-from-container.txt
```

- Don't play with Namespaces

```
--pid=host, --net=host, --ipc=host, --uts=host
kill, tcpdump, etc
```

Docker Container - 10 Things to avoid

1. Don’t store data in containers
2. Don’t ship your application in two pieces
3. Don’t create large images
4. Don’t use a single layer image
5. Don’t create images from running containers
6. Don’t use only the “latest” tag
7. Don’t run more than one process in a single container
8. Don’t store credentials in the image. Use environment variables
9. Don’t run processes as a root user
10. Don’t rely on IP addresses

=> Read 10 things to avoid in docker containers

Docker Container - Do's

- No privileged Containers, drop capabilites
- Isolate networks
- Set user - Container UID Management
- Use cgroups to limit resources

```
docker run -ti \
  -p 8080:8080 \
  --privileged=false \
  --cap-drop all \
  --network=my-image \
  --user 1001:1001 \
  --cpu-shares = 128 # 1024/SUM(CORES) = 1 CORE = 128
  --memory="512M" \
  my-image
```

=> Read Docker run reference

Docker Operations

- Audit-Logs
- Avoid image and container sprawl

Q&A

Next round

Docker Security #3

Container management systems

Swarm, DCOS, Rancher, Kubernetes, OpenShift