ABB Technikerschule

Technik. Informatik. Wirtschaft. Management →

Dominik Meyer
Software Engineering

Container Security

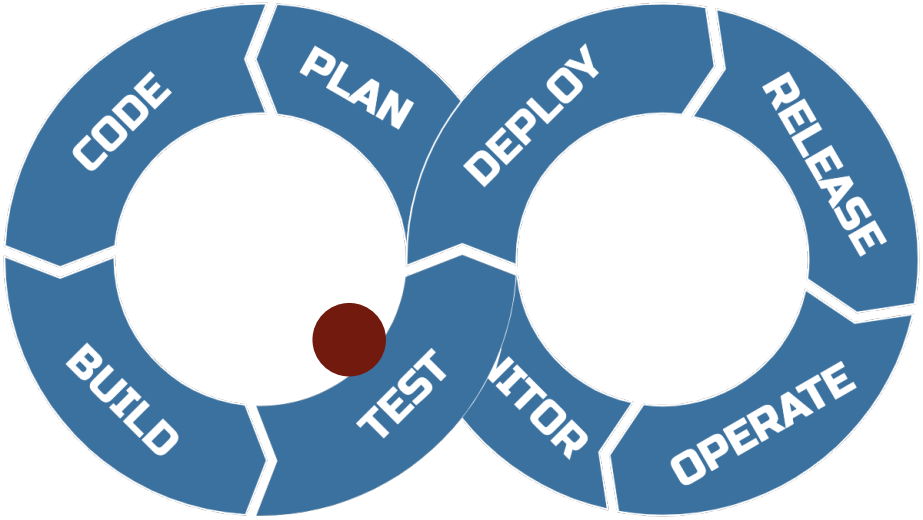
Ziel

Nach der Lektion haben die Studierenden ihren oder irgendeinen Docker Container auf Vulnerabilities gescannt und das Resultat interpretiert.

Docker, schon wieder 🤯

- Wir haben heute schon zwei neue Skills erworben
  - CI, mit der Absicht kontinuierlich sicher zu stellen das keine Lücken auftreten
  - Docker Images builden und starteb
- Wir sagten unser Output zu CCT sei ein Container
- Korrektur
  - Unser Output zu CCT ist ein **sicherer** 🔐 Container

- Dazu müssen wir diesen Scannen und gefundene Lücken mitigieren



Agenda

- Warum dieser Zirkus «Sicherheit» hier?
- Tooling
- Lokal eigenes oder anderes Docker scannen mit Snyk
  - Ab drei Scans braucht es ein Gratis Token
- Modulvorgaben (am Schluss weil Detail dieser Lektion relevant)
- Zielkontrolle

Schneller Demo Scan

```
⌥⌘1                    i511895@C02Y91DDJGH6:~/SAPDevelop/abb/docs
80% ──────────────────────────────  16% ──────────  12 GB ──────────
❯ docker scan --token "xxxx" openjdk:8-alpine

Testing openjdk:8-alpine...

✗ Low severity vulnerability found in openssl/libcrypto1.1
  Description: Inadequate Encryption Strength
  Info: https://snyk.io/vuln/SNYK-ALPINE39-OPENSSL-1089236
  Introduced through: openssl/libcrypto1.1@1.1.1b-r1, openssl/libssl1.1@1.1.1b-r1, apk-tools/apk-tools@2.10.3-r1, libtls-stan
dalone/libtls-standalone@2.7.4-r6, ca-certificates/ca-certificates@20190108-r0, krb5-conf/krb5-conf@1.0-r1
  From: openssl/libcrypto1.1@1.1.1b-r1
```

```
Organization:      mambax
Package manager:   apk
Project name:      docker-image|openjdk
Docker image:      openjdk:8-alpine
Platform:          linux/amd64
Licenses:          enabled

Tested 55 dependencies for known issues, found 87 issues.

Alpine 3.9.4 is no longer supported by the Alpine maintainers. Vulnerability detection may be affected by a lack of security updates.
```

Warum der Zirkus «Sicherheit»?

Als ich noch jung war…

- Security war das Langweiligste, die Tools waren auch «wüst»
- 2021 kann aber Security schlichtweg nicht ignoriert werden
- Muss priorisiert werden
  - Imageschaden
  - Immense Kosten
  - Bussen
  - Skandale, schlechtes Engineering
- Moderne Tools machen das Leben leichter
- Es macht sogar Freude

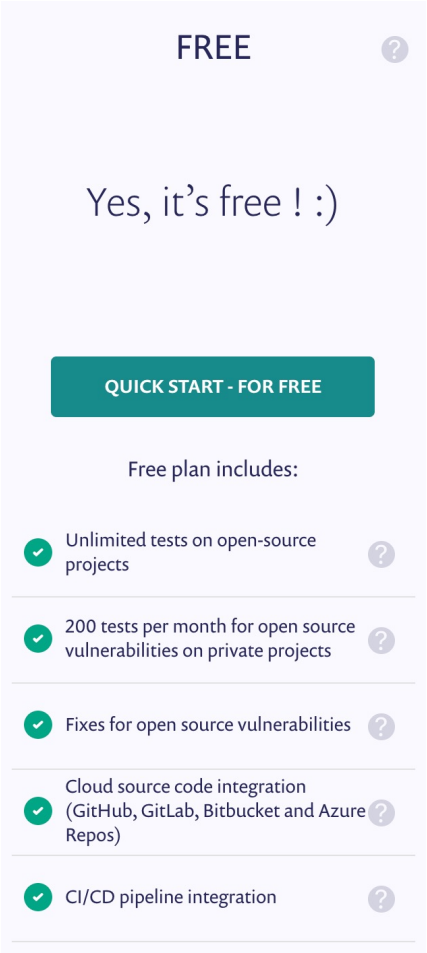
Lasst uns die einfachsten Mittel anschauen

Docker Desktop kommt mit Snyk

- Für Open Source Projekte gratis
  - API Token wird benötigt
- Kann Container und Anwendungen scannen

- Angenehme User Experience
- Einfach zu bedienen lokal
- Bringt GitHub Actions

Aber was ist mit «Closed Source»?



Closed Source Scanning

- z.B. Azure/container-scan
  - benutzt unter der Haube *Trivy*
    - *Gibt es auch lokal*
  - Findet zudem Bad Practises mit *Dockle*

- Die Findings sind nicht immer 100% gleich
  - Vendors haben andere Skalen
  - CVSS und CVE sind aber genormt

- Merke, all diese Tools befragen eine Datenbank, z.T. lokal, z.T. remote
  - Stolperfalle: In CI eine 600MB Datenbank ziehen ist generell eher eine doofe Idee

Warum nicht die Registry scannen? 🧐

– Gibt es, klaro!

– Kostet aber meistens, weil ist ein «Service»

– Beispiel: Azure Defender

Pricing:	**Azure Defender for container registries** is billed as shown on the pricing page

– Für uns ist es einfacher mit Container Scanning zu starten, wenn wir einer der vielen Lösungen nicht als Service beziehen

– Sobald Ihr mal 50+ Images in einer Registry managt, könnt ihr euch das überlegen

Task

Lokal scannen (mit Snyk)

AUFTRAG

(Achtung, drei Scans danach müssen wir ein Token lösen)

Navigieren Sie in das Packer Verzeichnis (geklont in Lektion 4)
– Sie brauchen noch den Tag den Sie verwendet haben
– Ansonsten builden Sie einfach ein neues Image

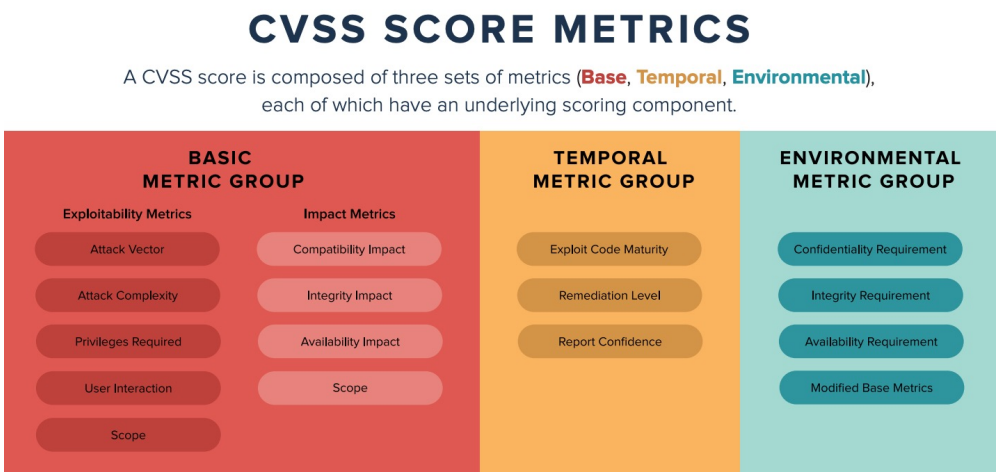
Scannen Sie das fertige Image
`docker scan <tag>`
Ändern Sie das BaseImage zu `FROM openjdk:8-alpine`,
builden und scannen Sie nochmal.

Was beobachten Sie?

Task

```
⌥⌘1                                    i511895@C02Y91DDJGH6:~/SAPDevelop/abb/exman-packer
 80%                                    7%                                    9.5 GB
❯ docker build -t packer .
[+] Building 21.2s (11/11) FINISHED
 => [internal] load build definition from Dockerfile                                    0.0s
 => => transferring dockerfile: 357B                                                    0.0s
 => [internal] load .dockerignore                                                       0.0s
 => => transferring context: 2B                                                         0.0s
 => [internal] load metadata for docker.io/library/openjdk:16-alpine                    2.1s
 => [auth] library/openjdk:pull token for registry-1.docker.io                          0.0s
 => [1/6] FROM docker.io/library/openjdk:16-alpine@sha256:49d822f4fa4deb5f9d0201ffeec9f4d113bcb4e7e49bd6bc063d3ba93aacbcae   11.5s
 => => resolve docker.io/library/openjdk:16-alpine@sha256:49d822f4fa4deb5f9d0201ffeec9f4d113bcb4e7e49bd6bc063d3ba93aacbcae   0.0s
 => => sha256:49d822f4fa4deb5f9d0201ffeec9f4d113bcb4e7e49bd6bc063d3ba93aacbcae 319B / 319B   0.0s
 => => sha256:f9be8e89a2bbf973dcd6c286f85bb0f68a8f9d5fa7c6241eb59f07add4a24789 951B / 951B   0.0s
 => => sha256:2aa8569968b8d76c8e121516f183b8abc2b639158973a5812a2a6c474a80c63a 3.48kB / 3.48kB   0.0s
 => => sha256:4c0d98bf9879488e0407f897d9dd4bf758555a78e39675e72b5124ccf12c2580 2.81MB / 2.81MB   0.4s
 => => sha256:6f1834e342ac63f74ab268f59b78219089f04f37c6c39e469afc0292192b9c2d 928.24kB / 928.24kB   0.7s
 => => sha256:78f4563ac5cfaa5b85c77a78809e4a4d118e39ea769eff648554e6b66d21ad3f 185.96MB / 185.96MB   5.0s
 => => extracting sha256:4c0d98bf9879488e0407f897d9dd4bf758555a78e39675e72b5124ccf12c2580   0.3s
 => => extracting sha256:6f1834e342ac63f74ab268f59b78219089f04f37c6c39e469afc0292192b9c2d   0.3s
 => => extracting sha256:78f4563ac5cfaa5b85c77a78809e4a4d118e39ea769eff648554e6b66d21ad3f   5.3s
 => [internal] load build context                                                       0.0s
 => => transferring context: 111B                                                       0.0s
 => [2/6] RUN apk update && apk upgrade && apk add --no-cache supervisor openssh nginx bash curl   6.4s
 => [3/6] COPY supervisord.conf /etc/supervisord.conf                                   0.1s
 => [4/6] ADD ExManRest.jar ExManRest.jar                                               0.0s
 => [5/6] ADD load-expeditions.sh load-expeditions.sh                                   0.0s
 => exporting to image                                                                  0.8s
 => => exporting layers                                                                 0.8s
 => => writing image sha256:7edad10e4e36a9afd96b00e8045fd3e78ee6bdbe3f60dfc916c3bda1ad358d72   0.0s
 => => naming to docker.io/library/packer                                               0.0s
❯ docker scan packer

Testing packer...

✗ Medium severity vulnerability found in musl/musl-utils
  Description: Out-of-bounds Write
  Info: https://snyk.io/vuln/SNYK-ALPINE313-MUSL-1067865
  Introduced through: musl/musl-utils@1.2.2-r0, libc-dev/libc-utils@0.7.2-r3, meta-common-packages@meta
  From: musl/musl-utils@1.2.2-r0
  From: libc-dev/libc-utils@0.7.2-r3 > musl/musl-utils@1.2.2-r0
  From: meta-common-packages@meta > musl/musl@1.2.2-r0
  Fixed in: 1.2.2_pre2-r0



Organization:      mambax
Package manager:   apk
Project name:      docker-image|packer
Docker image:      packer
Platform:          linux/amd64
Licenses:          enabled

Tested 52 dependencies for known issues, found 1 issue.
```

Vulnerabilities (VULNAS) - CVSS



CVSS Score	Qualitative Rating
0.0	None
0.1 – 3.9	Low
4.0 – 6.9	Medium
7.0 – 8.9	High
9.0 – 10.0	Critical

CVE-[4 Digit Year]-[Sequential Identifier]

Common Vulnerability Enumeration,

– Gute Daumenregel: >= Medium nix gut!

- What is the difference between CVE and CVSS?

  CVE stands for Common Vulnerability Enumeration, which is a unique identifier for each vulnerability listed in the NIST NVD. CVE is simply a list of all publicly disclosed vulnerabilities that includes the CVE ID, a description, dates, and comments.

  For example, the CVE for the Heartbleed vulnerability is: CVE-2014-0160

  CVSS (Common Vulnerability Scoring System) provides an indication of the severity of each CVE. The CVSS score is not reported in the CVE listing – you must use the NVD to find assigned CVSS scores.

Zielkontrolle

Nenne drei Möglichkeiten Container zu «Scannen».

Warum ist es ein Problem, wenn ein «lower layer» eine Sicherheitslücke hat, aber unser Code nicht?

Wo sollten wir den Scan einbauen?

- Wir werden das in Lektion 19 spätestens bauen (SWT)
- ExMan Packer wird bereits gescannt, falls Sie sich das anschauen wollen

Sie kennen GitHub, GitHub Actions und deren CI Hintergrund und Sie können auf der Kommandozeile einen Scan ausführen… Was ist der logische nächste Schritt?

Modulvorgaben

– Etwaige Unklarheiten jetzt schriftlich festhalten
– Danach keine Änderungen an den MA, nur noch Absprachen

– Da GitHub Actions verlangt wird, sollte ihr Code vorzugsweise auf Git**Hub** abgelegt sein (wenn NDA, dann Private Repo)


ianmarkow

WEITER WISSEN.

Wir begleiten Sie!