Automated security testing for software developers who don't know security!

Christian Kühn - @CYxChris

Agenda

whoami?

why?

what?

how?

whoami?

Christian Kühn

system developer

#java #kubernetes #devops

@DevOpsKA Meetup Organizer

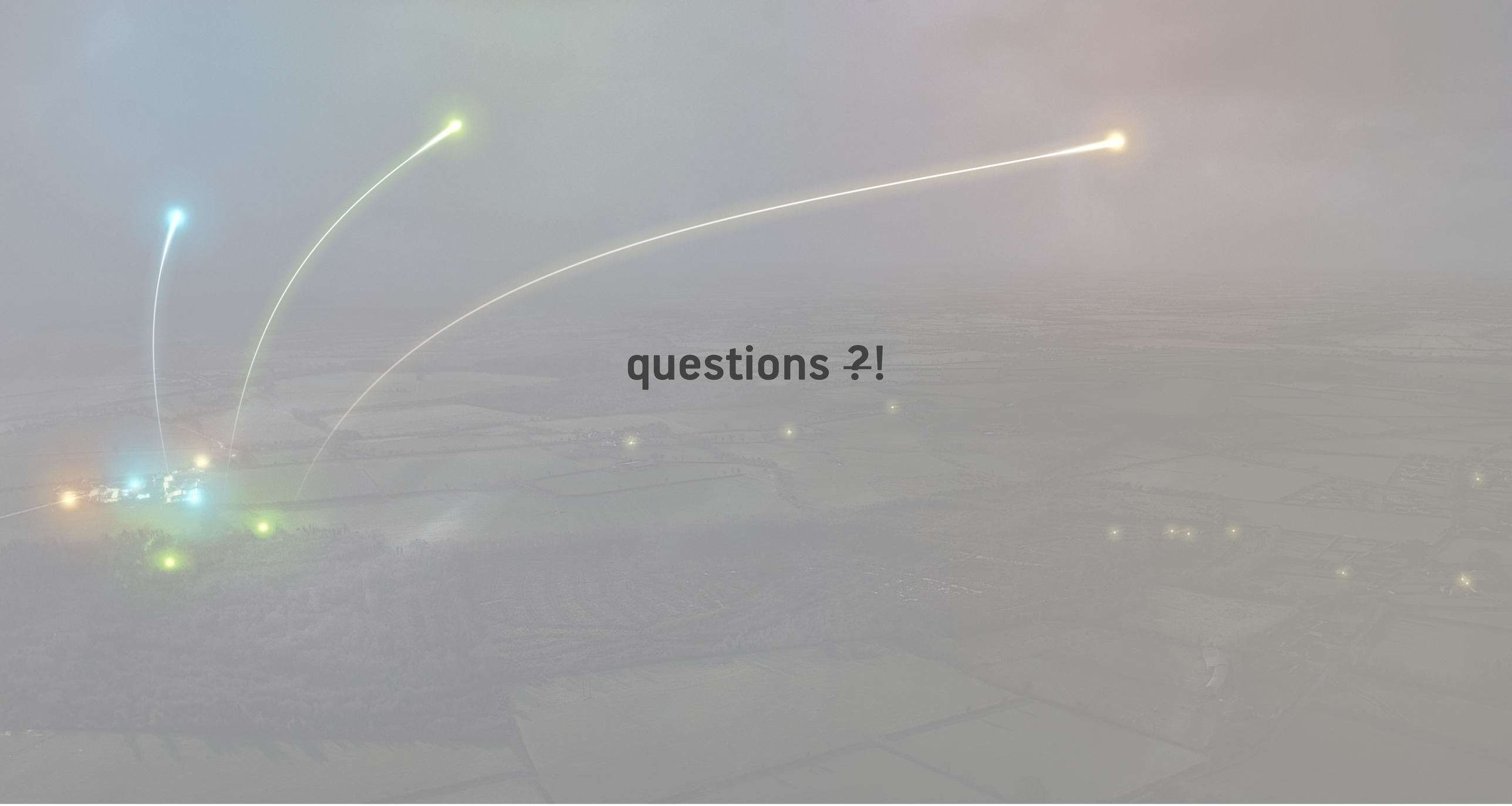
synyx GmbH Karlsruhe

code with attitude!

sw development

consulting

cccamp ticket = training budget :-)


questions ?!

software security issues:
what could possibly go wrong?

- leakage of business data
- leakage of user/customer data
- service interruption
- industry malfunction
- death (😱)

examples:

equifax - “Credit Monitoring”

hacked 2017
vulnerability in Apache Struts dependency

143,000,000 SSN
209,000 credit card numbers
182,000 “consumers” with PII

https://krebsonsecurity.com/2017/09/the-equifax-breach-what-you-should-know/

examples:

Mossack Fonseca - “Law Firm and coprorate service provider”

hacked 2015
vulnerability in Drupal

11.5 million leaked documents about
money laundering
tax avoidance
corruption

https://en.wikipedia.org/wiki/Panama_Papers

what stops developers from patching?

negligence

priorities / lack of time

skills / training

insight

“security - not my department” (or is it?)

A9:2017-Using Components with Known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

T10 OWASP Top 10 Application Security Risks – 2017 6

A1:2017-Injection	Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2:2017-Broken Authentication	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
A3:2017-Sensitive Data Exposure	Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
A4:2017-XML External Entities (XXE)	Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
A5:2017-Broken Access Control	Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.
A6:2017-Security Misconfiguration	Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.
A7:2017-Cross-Site Scripting (XSS)	XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A8:2017-Insecure Deserialization	Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.
A9:2017-Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
A10:2017-Insufficient Logging & Monitoring	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

OWASP

Open Web Application Security Project

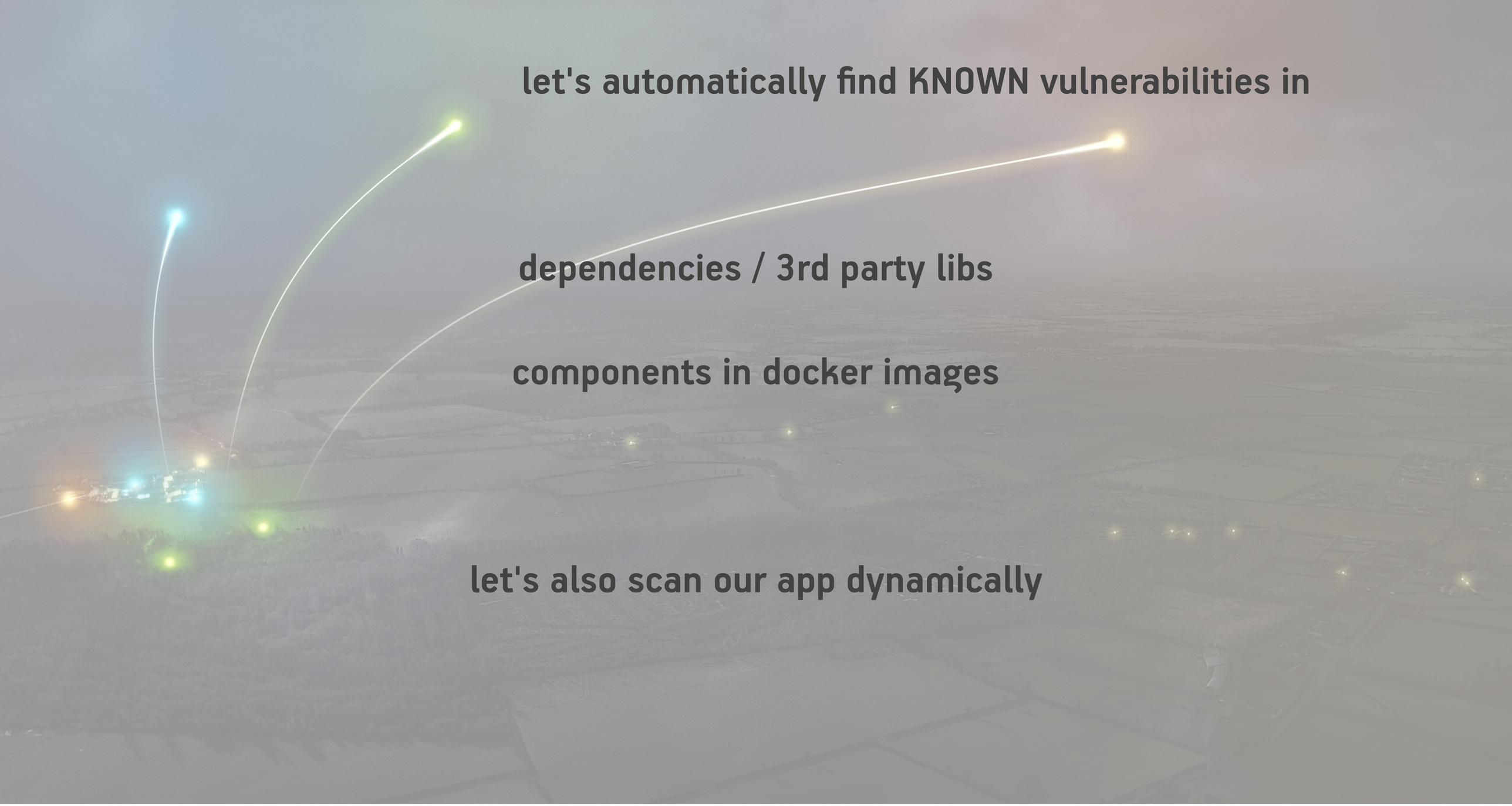
solution

search for known vulnerabilities

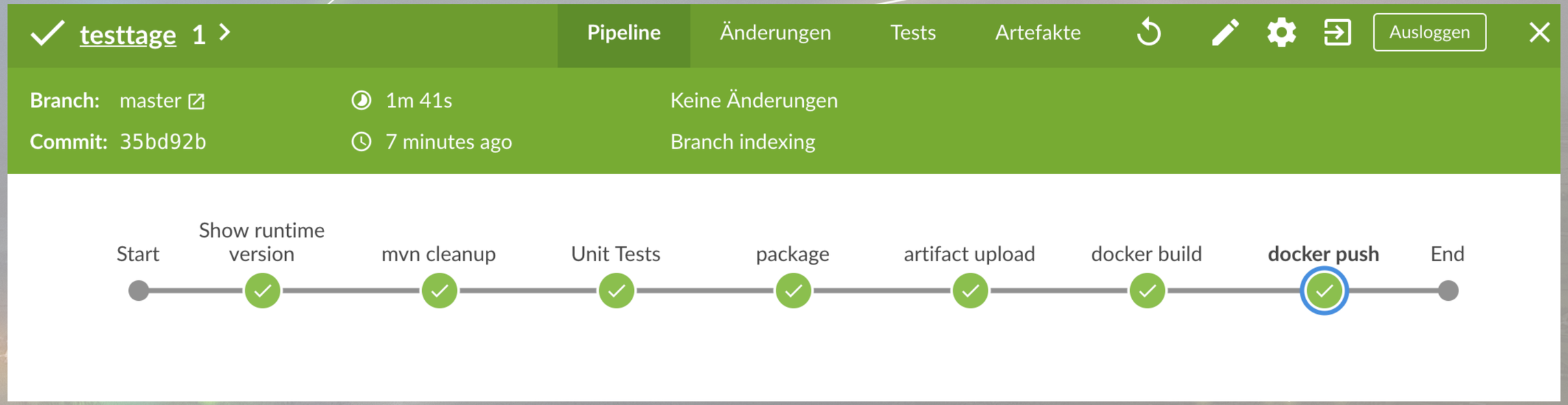
implement a process to fix ASAP (or whitelist 😇)

treat security issues like technical debt

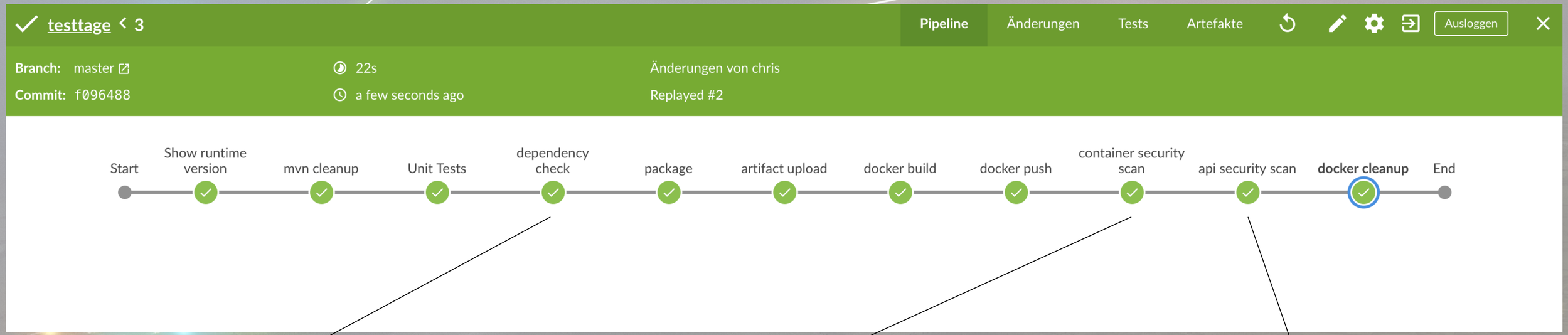
automate the sh!t out of it

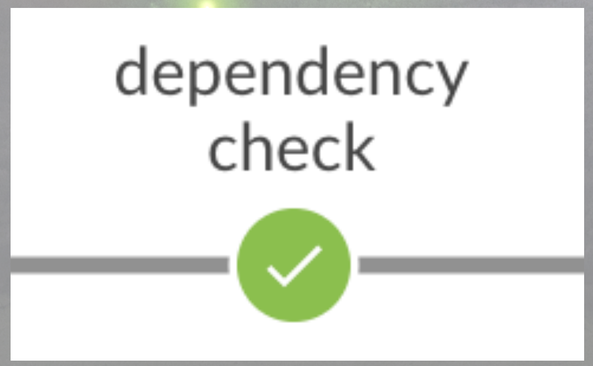

let's automatically find KNOWN vulnerabilities in
dependencies / 3rd party libs
components in docker images
let's also scan our app dynamically

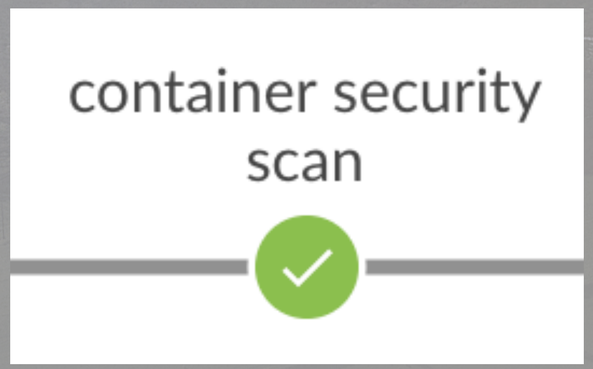
continuous delivery today



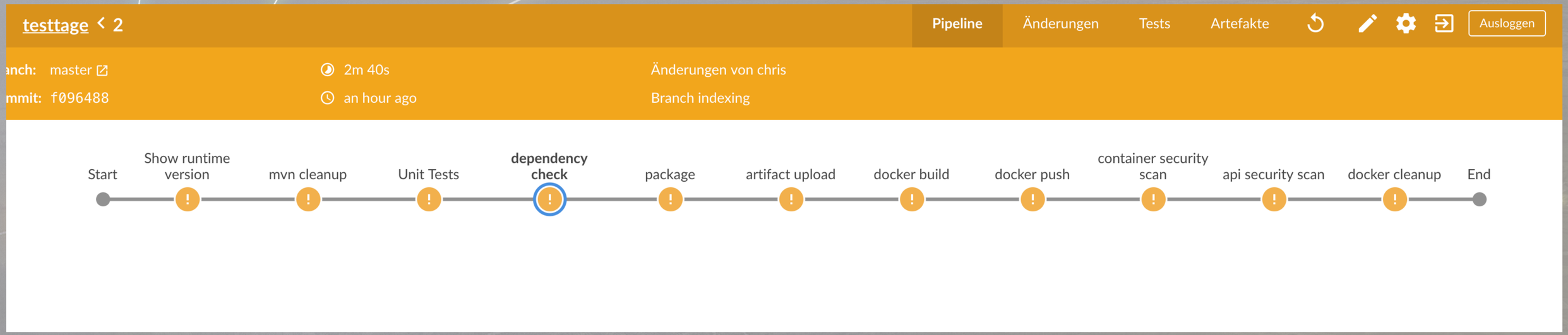
continuous delivery ++







continuous delivery++



CVE - Common Vulnerabilities and Exposures

vulnerability

/vʌln(ə)rəˈbɪlɪti/

noun

1. the quality or state of being exposed to the possibility of being attacked or harmed, either physically or emotionally.

…

CVE - reference for publicly known information-security vulnerabilities and exposures

CVE-2017-5638 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

The Jakarta Multipart parser in Apache Struts 2 2.3.x before 2.3.32 and 2.5.x before 2.5.10.1 has incorrect exception handling and error-message generation during file-upload attempts, which allows remote attackers to execute arbitrary commands via a crafted Content-Type, Content-Disposition, or Content-Length HTTP header, as exploited in the wild in March 2017 with a Content-Type header containing a #cmd= string.

Source: MITRE
Description Last Modified: 09/22/2017
View Analysis Description

Impact

CVSS v3.0 Severity and Metrics:

Base Score: 10.0 CRITICAL
Vector: AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H (V3 legend)
Impact Score: 6.0
Exploitability Score: 3.9

Attack Vector (AV): Network
Attack Complexity (AC): Low
Privileges Required (PR): None
User Interaction (UI): None
Scope (S): Changed

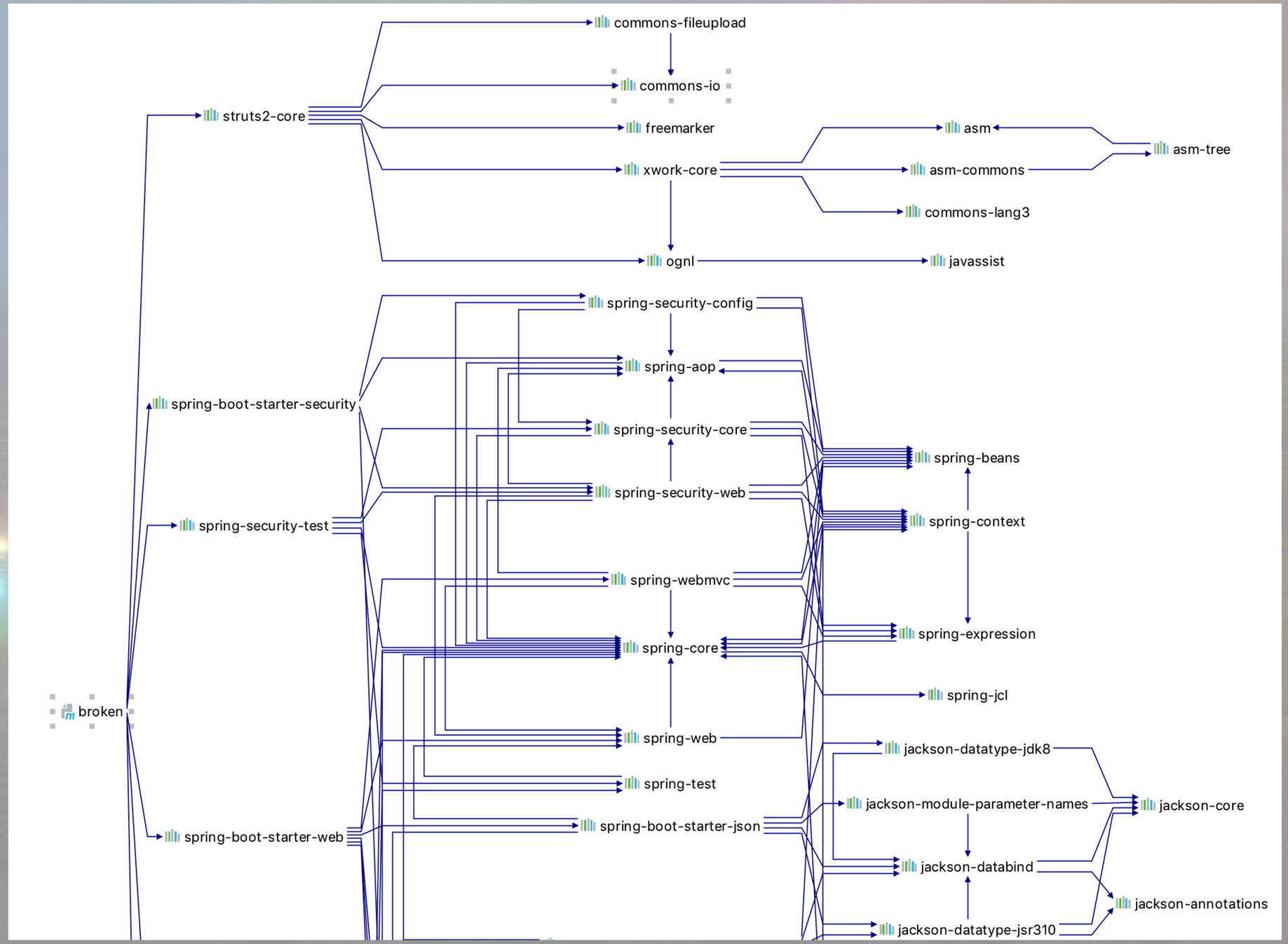
CVSS v2.0 Severity and Metrics:

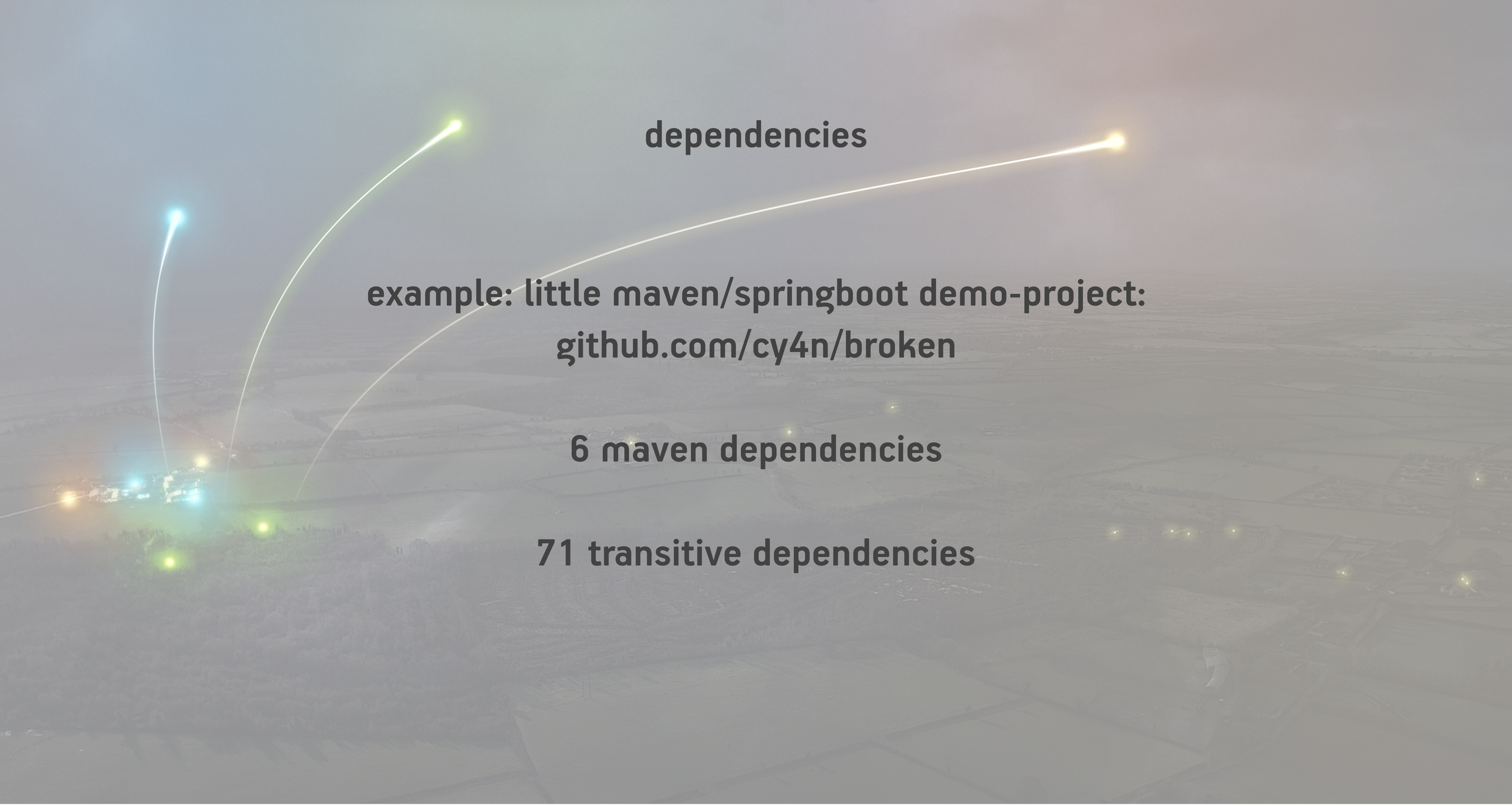
Base Score: 10.0 HIGH
Vector: (AV:N/AC:L/Au:N/C:C/I:C/A:C) (V2 legend)
Impact Subscore: 10.0
Exploitability Subscore: 10.0

Access Vector (AV): Network
Access Complexity (AC): Low
Authentication (AU): None
Confidentiality (C): Complete
Integrity (I): Complete
Availability (A): Complete

QUICK INFO

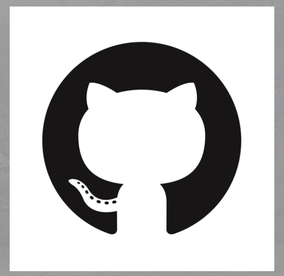
CVE Dictionary Entry:
CVE-2017-5638
NVD Published Date:
03/10/2017
NVD Last Modified:
03/03/2018

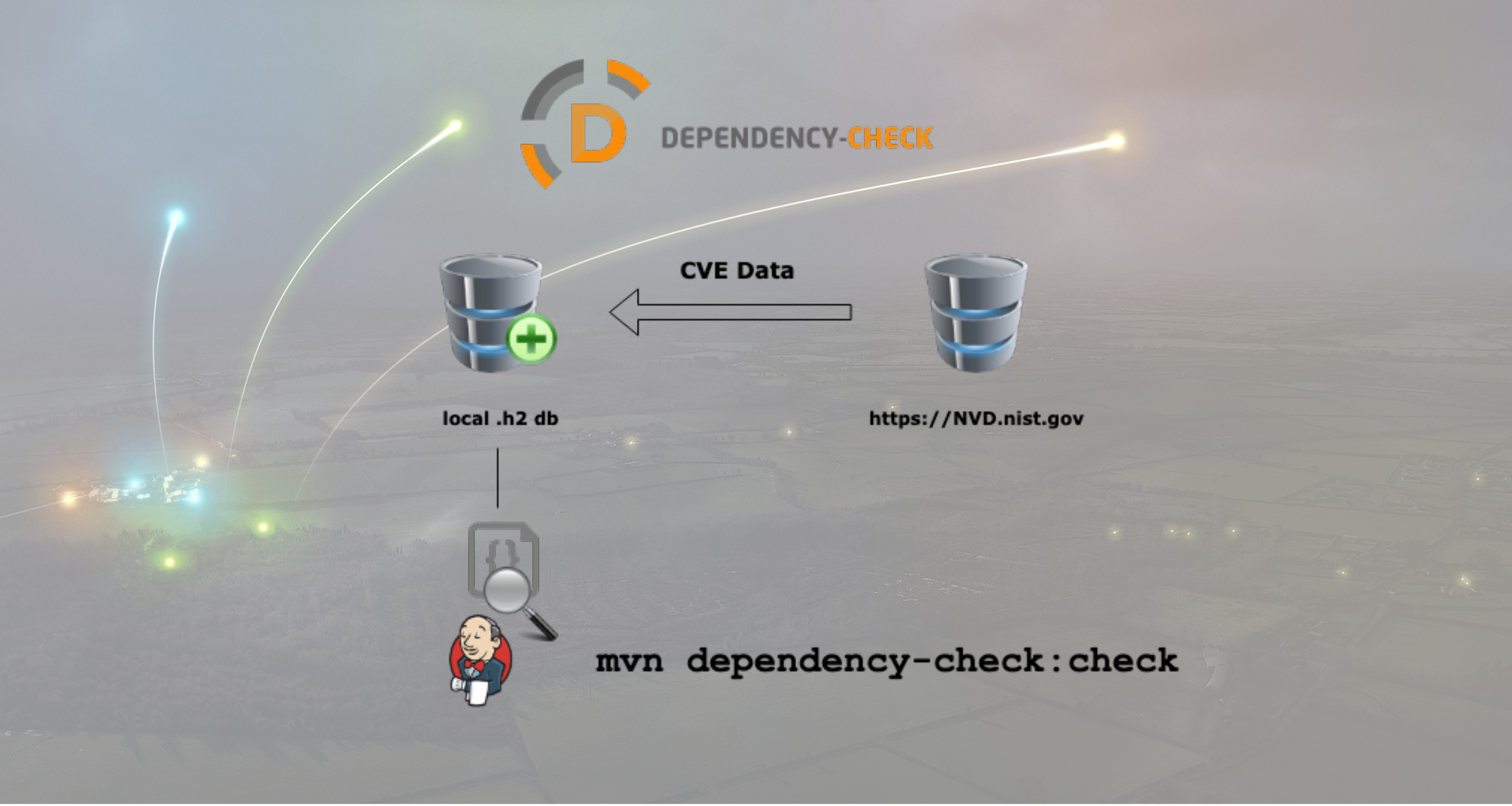

commons-fileupload
commons-io
struts2-core
freemarker
asm
asm-tree
xwork-core
asm-commons
commons-lang3
ognl
javassist
spring-security-config
spring-aop
spring-boot-starter-security
spring-security-core
spring-beans
spring-security-web
spring-context
spring-security-test
spring-webmvc
spring-expression
spring-core
spring-jcl
broken
spring-web
jackson-datatype-jdk8
spring-test
jackson-module-parameter-names
jackson-core
spring-boot-starter-json
spring-boot-starter-web
jackson-databind
jackson-annotations
jackson-datatype-jsr310

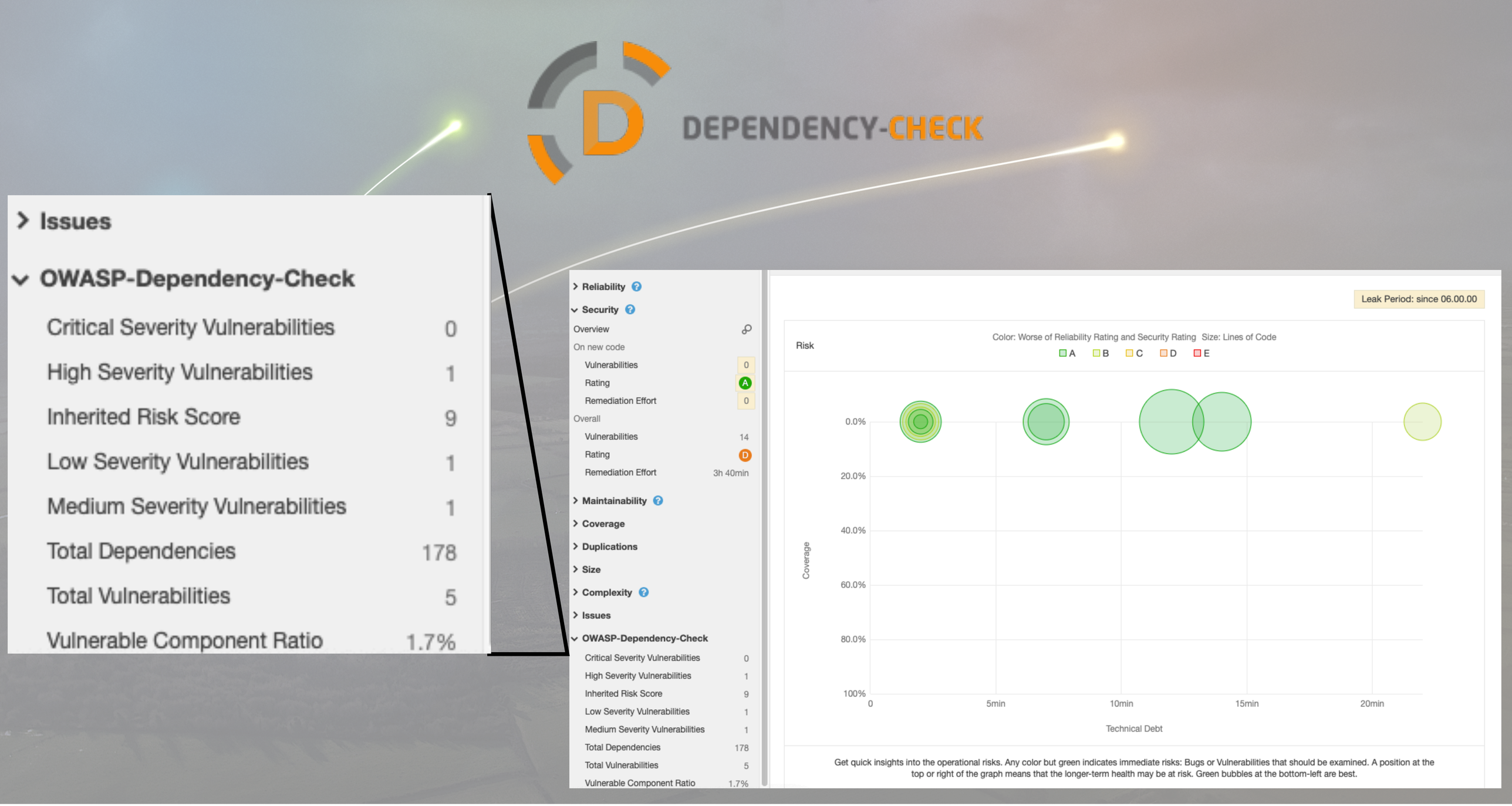

dependencies
example: little maven/springboot demo-project:
github.com/cy4n/broken
6 maven dependencies
71 transitive dependencies

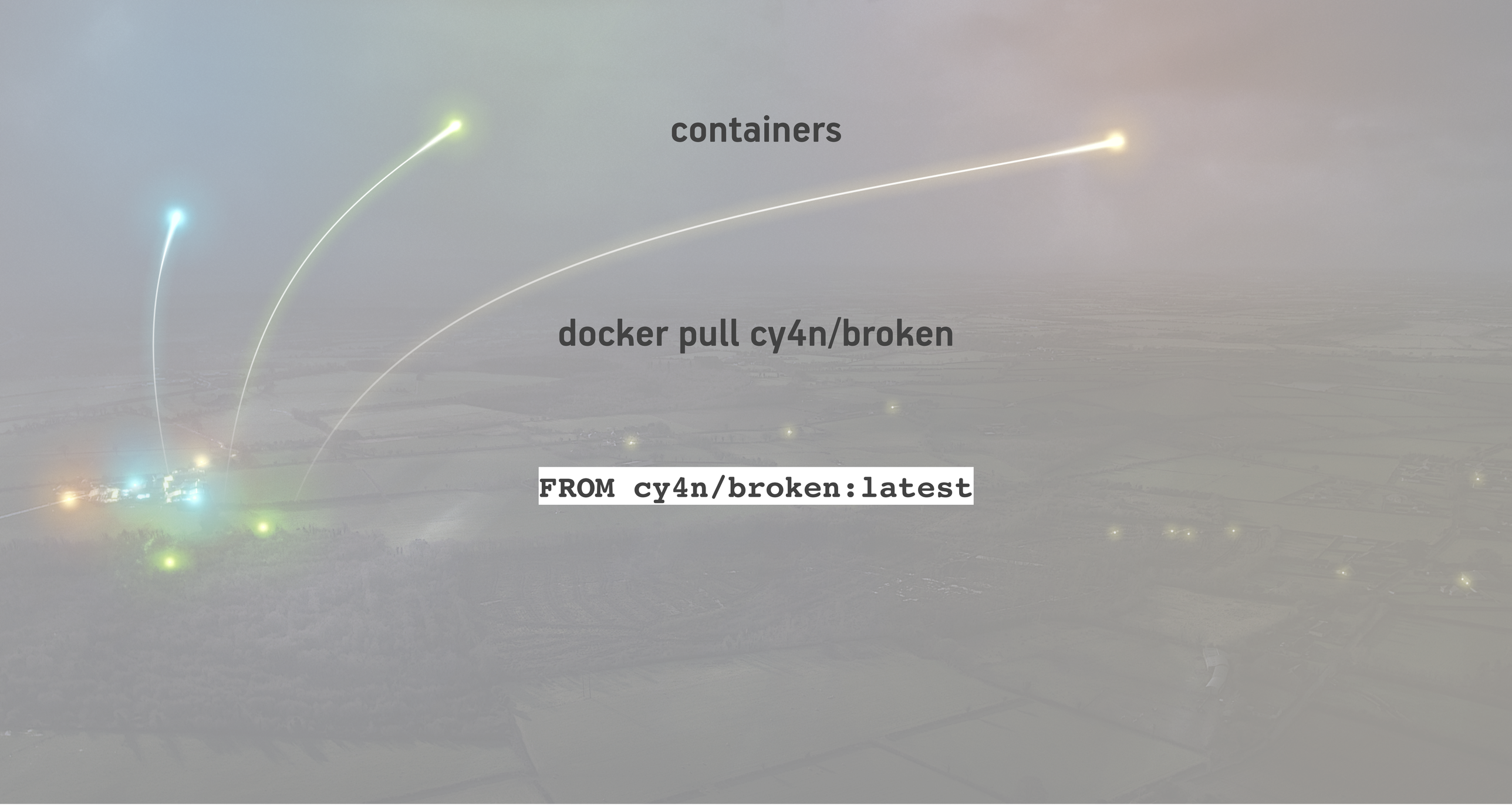
find vulnerable dependencies

DEPENDENCY-CHECK



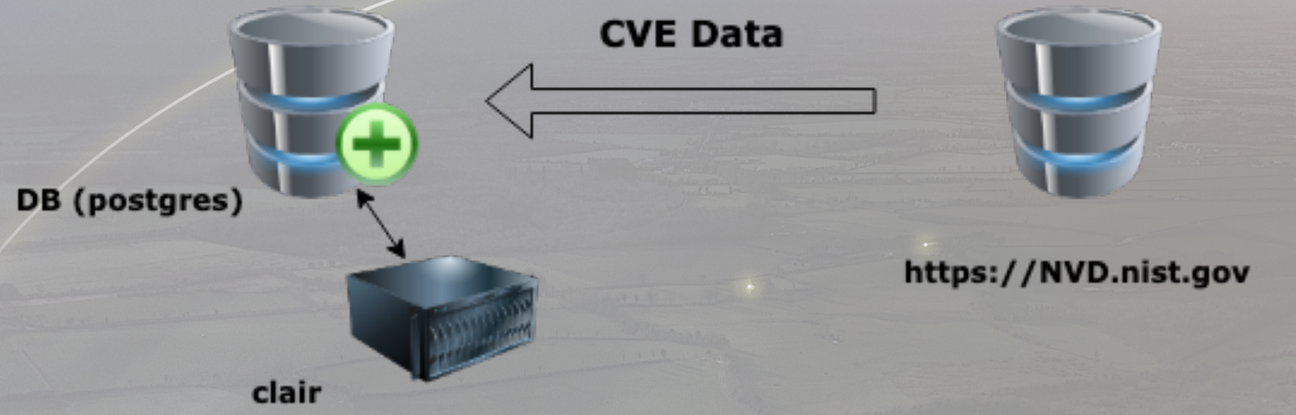

DEPENDENCY-CHECK
CVE Data
local .h2 db
https://NVD.nist.gov
mvn dependency-check:check


DEPENDENCY-CHECK
Issues
OWASP-Dependency-Check
Critical Severity Vulnerabilities 0
High Severity Vulnerabilities 1
Inherited Risk Score 9
Low Severity Vulnerabilities 1
Medium Severity Vulnerabilities 1
Total Dependencies 178
Total Vulnerabilities 5
Vulnerable Component Ratio 1.7%
Reliability
Security
Overview
On new code
Vulnerabilities 0
Rating A
Remediation Effort 0
Overall
Vulnerabilities 14
Rating D
Remediation Effort 3h 40min
Maintainability
Coverage
Duplications
Size
Complexity
Issues
OWASP-Dependency-Check
Critical Severity Vulnerabilities 0
High Severity Vulnerabilities 1
Inherited Risk Score 9
Low Severity Vulnerabilities 1
Medium Severity Vulnerabilities 1
Total Dependencies 178
Total Vulnerabilities 5
Vulnerable Component Ratio 1.7%
Leak Period: since 06.00.00
Risk
Color: Worse of Reliability Rating and Security Rating Size: Lines of Code
A B C D E
Coverage
0.0%
20.0%
40.0%
60.0%
80.0%
100%
0
5min
10min
15min
20min
Technical Debt
Get quick insights into the operational risks. Any color but green indicates immediate risks: Bugs or Vulnerabilities that should be examined. A position at the top or right of the graph means that the longer-term health may be at risk. Green bubbles at the bottom-left are best.


containers
docker pull cy4n/broken
FROM cy4n/broken:latest

clair

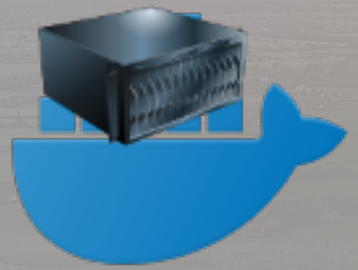
A Container Image Security Analyzer by CoreOS



```
clair-scanner ... <image>
```

https://github.com/arminc/clair-local-scan

arminc/clair-db
(nightly)

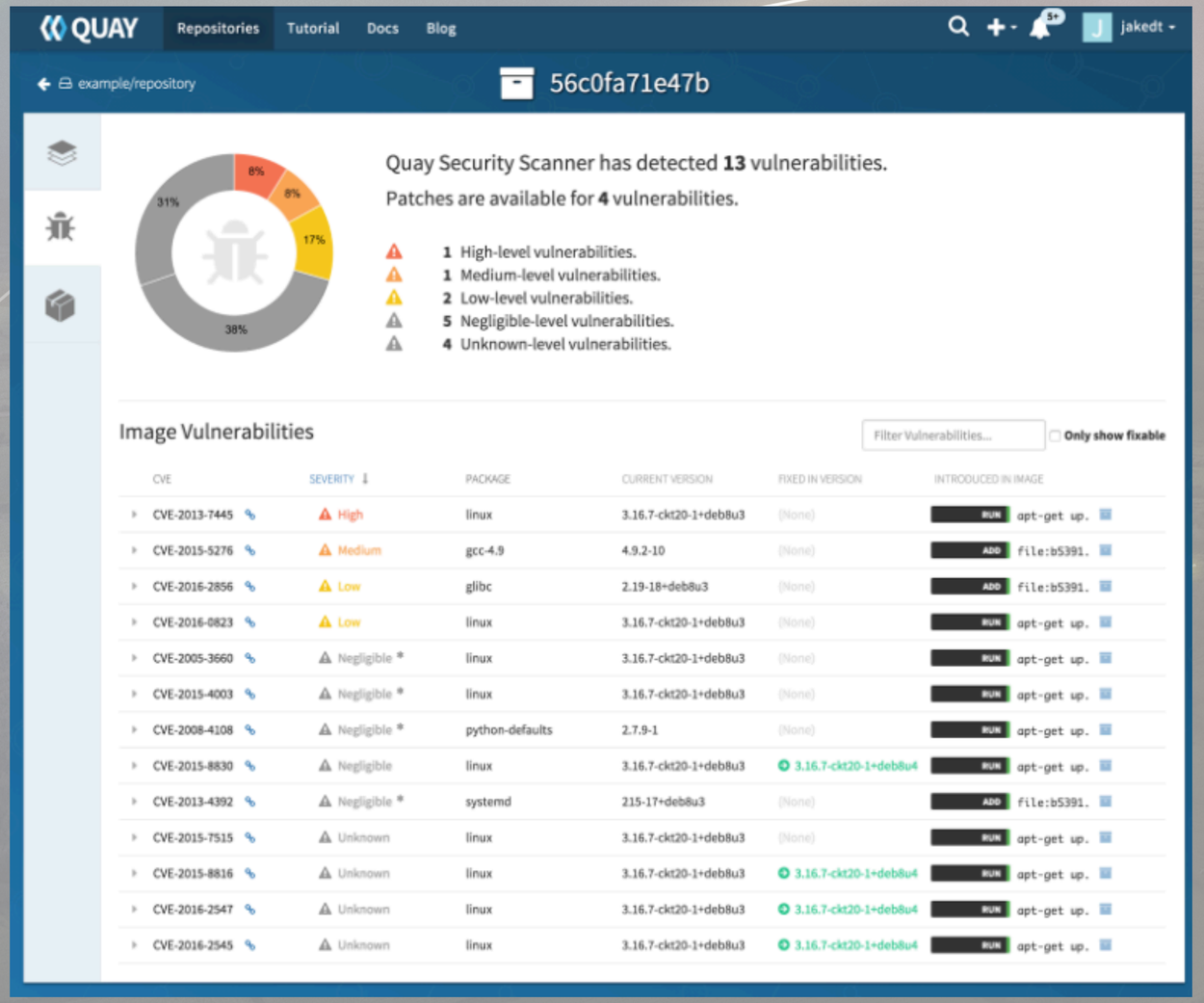


arminc/clair-local-scan

```
docker run -d --name db arminc/clair-db:latest
docker run -p 6060:6060 --link db:postgres -d --name clair arminc/clair-local-scan
```

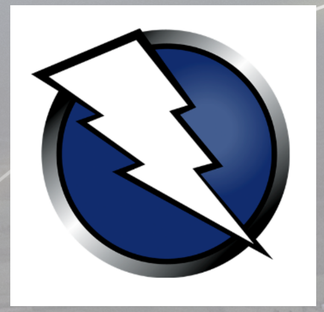
clair

A Container Image Security Analyzer by CoreOS




ools
Help
URL:
http://www.internet.com

API / Webserver

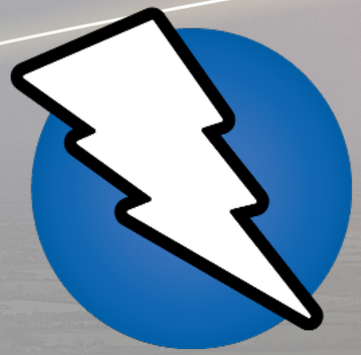


ZAProxy

burp

API / Webserver

OWASP ZAProxy

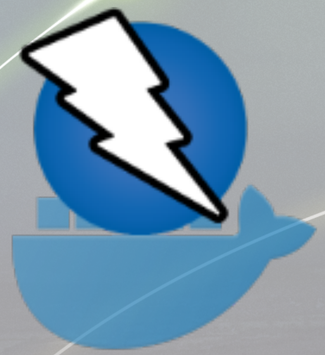


url spider

passive (and active) modes

dynamic scanner

ajax supported



zap2docker

myApp

```
docker run -t owasp/zap2docker-stable zap-baseline.py -t http://<URL>
```

HTTP

get your hands dirty

THM

CCCAMP 2019

Finding insecure third-party librarys in dependencies, containers, APIs (OWASP Top10 - A9)

2019-08-23, 15:15–17:15, Workshop

The OWASP Top Ten project lists the top 10 (web) application security risks. In this Workshop we will take a close look at number 9: "Using Components With Known Vulnerabilities".

we will try to use (open source) tooling to find known vulnerabilities in 3rd party libraries, containers and APIs, then take a look at how we can automate those tools in our ci/cd pipelines you don't need to know about security or vulnerability management to do the workshop, we will cover the basics and you can a lot on the way

---

workshop will feature the following Tools:
- OWASP dependency-check (workshop will focus on java/maven/gradle, but feel free bring your own languages and dependencies so i can learn something too:))
- CoreOS Clair for container scanning
- OWASP Zap for API scanning (technically not A9, but many the others;))

if we have time (or if you're interested after the actual workshop) we can further discuss how we can shape the process of fixing said vulnerabilities in our daily dev/ops/x jobs (or we can just rant about security over some beers)

we speak english and german, so dont be scared if your english is not too good. we will get along :-)

Vortragende

cy

honnel

Sprache: Englisch


how to react?