My container is running stable since a year,

is that good?

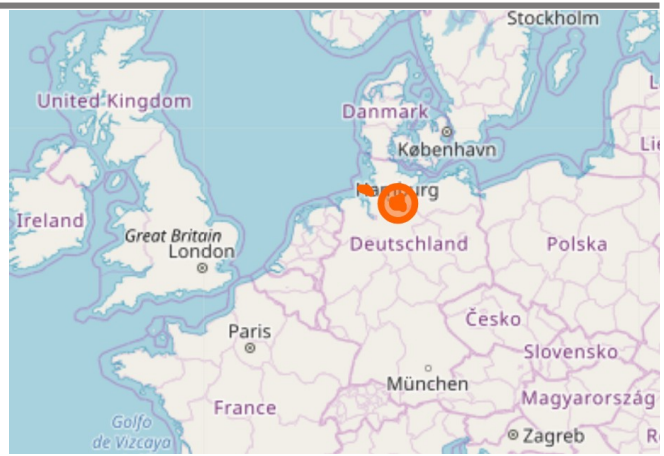
Timo Pagel

✉ devsecops19@pagel.pro

11.05.2019, Kiel

About Me

- DevSecOps Consultant, Hamburg
- Lecturer for *Security in Web Applications* at University of Applied Sciences Kiel/Wedel
- Open Source / Open Knowledge Enthusiast
  - DevSecOps Maturity Model
  - Full University Module Security in Web App.
  - OWASP Software Assurance Maturity Model



Agenda

- Introduction
- Attack Surface Reduction
- Scanning for Known Vulnerabilties
- Fast Patching
- Conclusion

Agenda

- Introduction
- Attack Surface Reduction
- Scanning for Known Vulnerabilities
- Fast Patching
- Conclusion

Patch Management in 2009

My server is running stable since a year

```
$ uptime
12:52:27 up 463 days, [...]
```

My container is running stable since a year

Conclusions?

My container is running stable since a year

Conclusions?

→ Container has not been rebuild since a year

My container is running stable since a year

Conclusions?

→ Container has not been rebuild since a year

→ No patches

→ Host has not been restarted since a year

→ No patches (at least for docker), no kernel updates

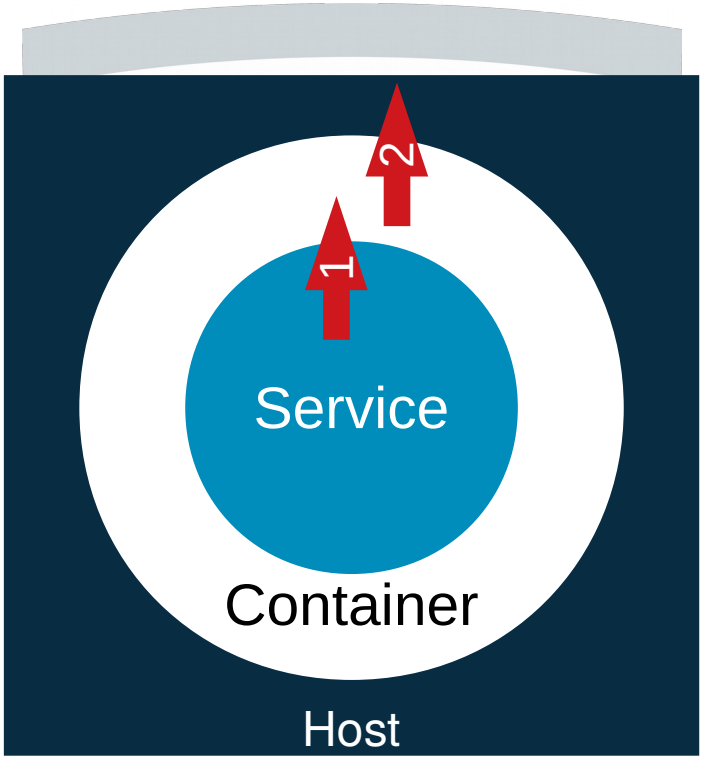
Vulnerabilities

Shellshock, 2014

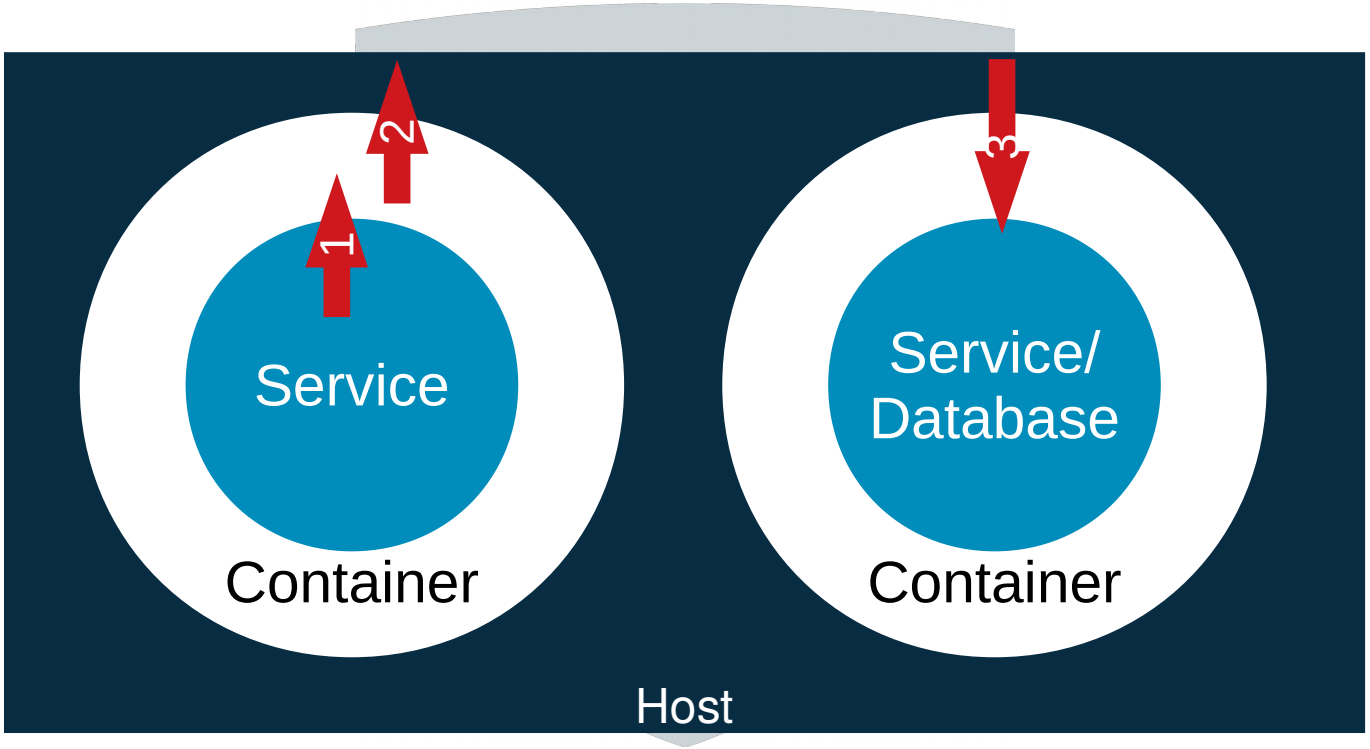
→ privilege escalation

→ run arbitrary commands

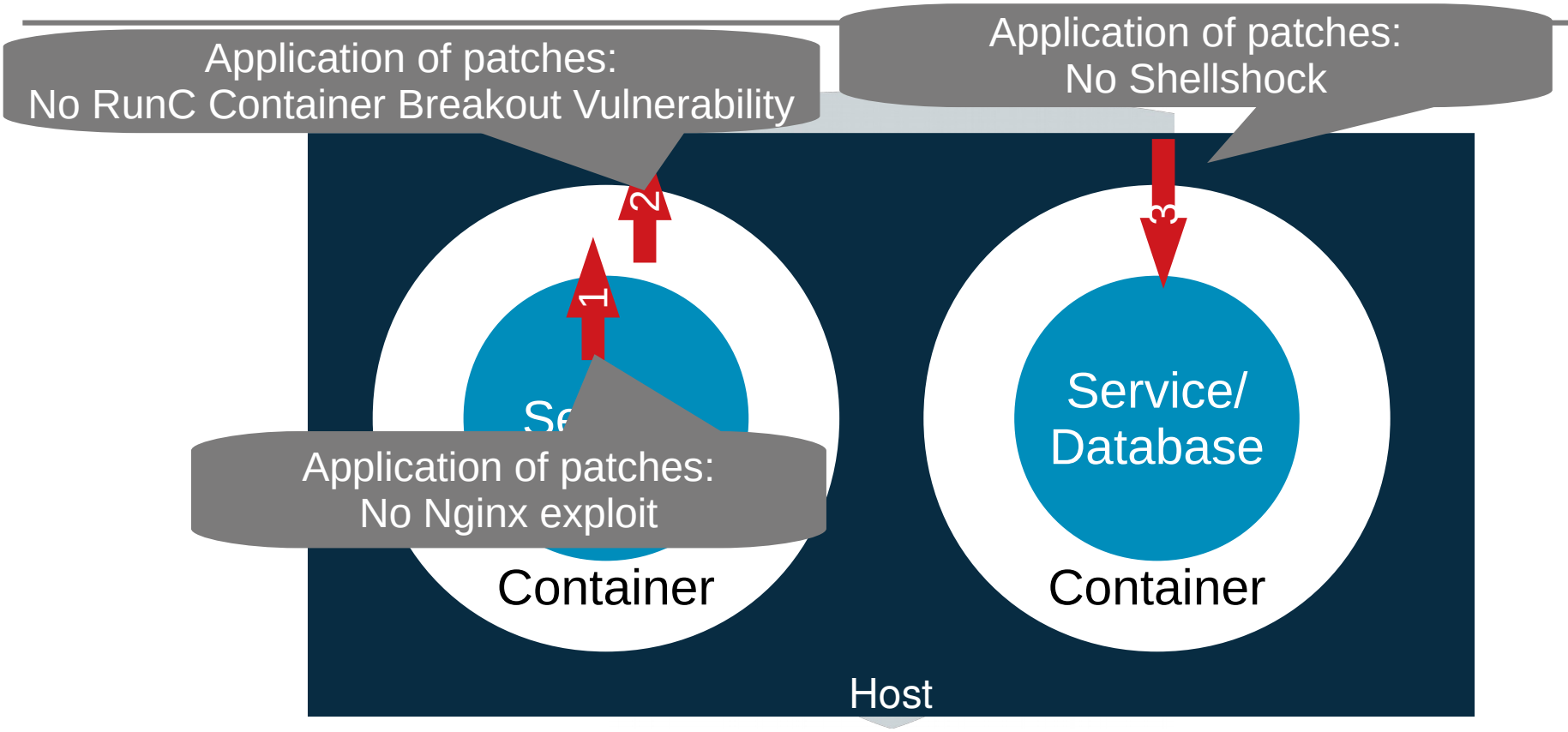
Container Breakout



Container Breakout and more



Container Breakout and Patches



Strategies

- Attack surface reduction
- Scan for vulnerablities
- Fast patching

Agenda

- Introduction
- Attack Surface Reduction
- Scanning for Known Vulnerabilities
- Fast Patching
- Conclusion

Attack Surface Reduction

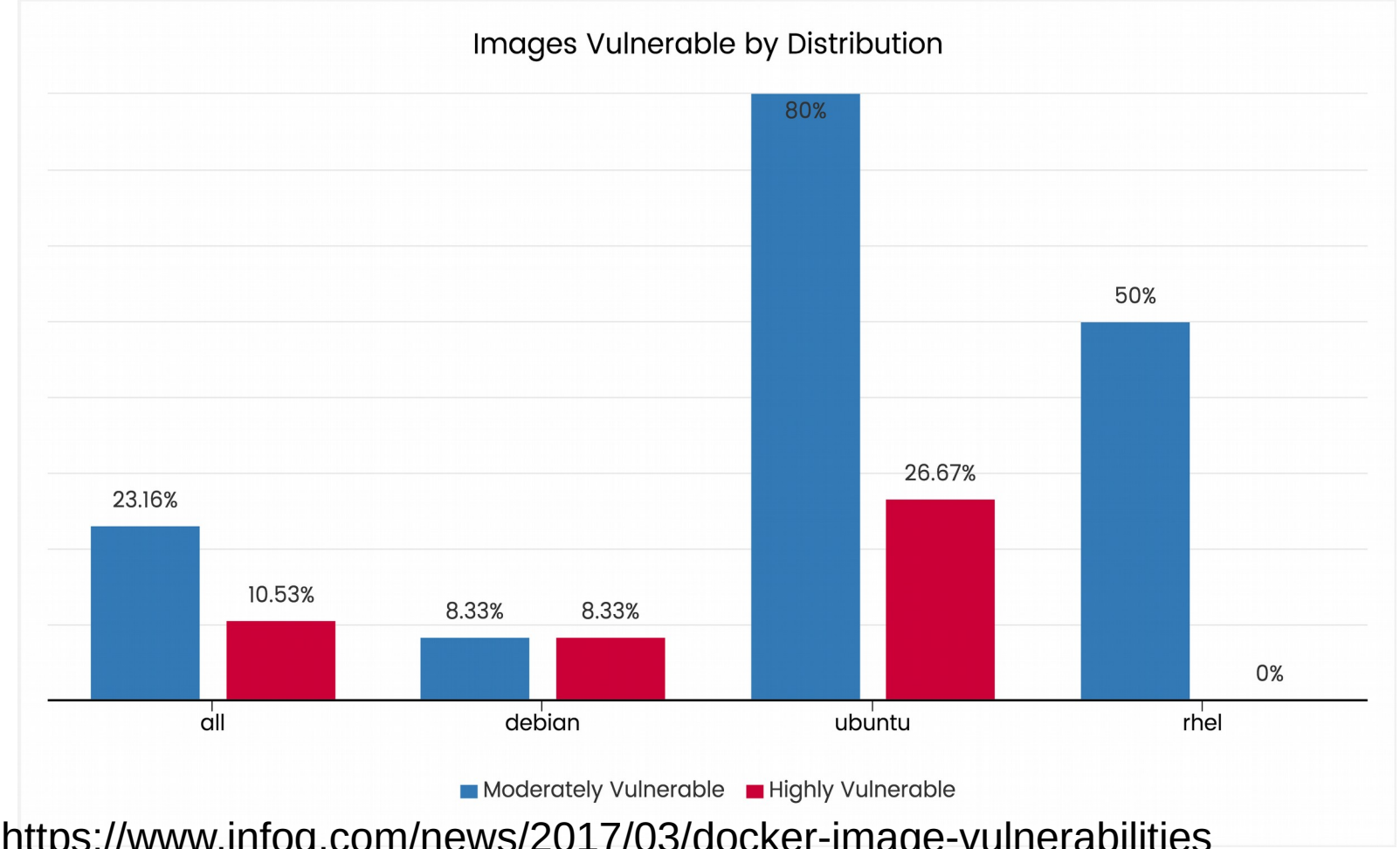
- Careful selection of distribution

Distribution Selection

- Vulnerabilities
- Speed of providing patches
- Size
- Maintainability
- Stability

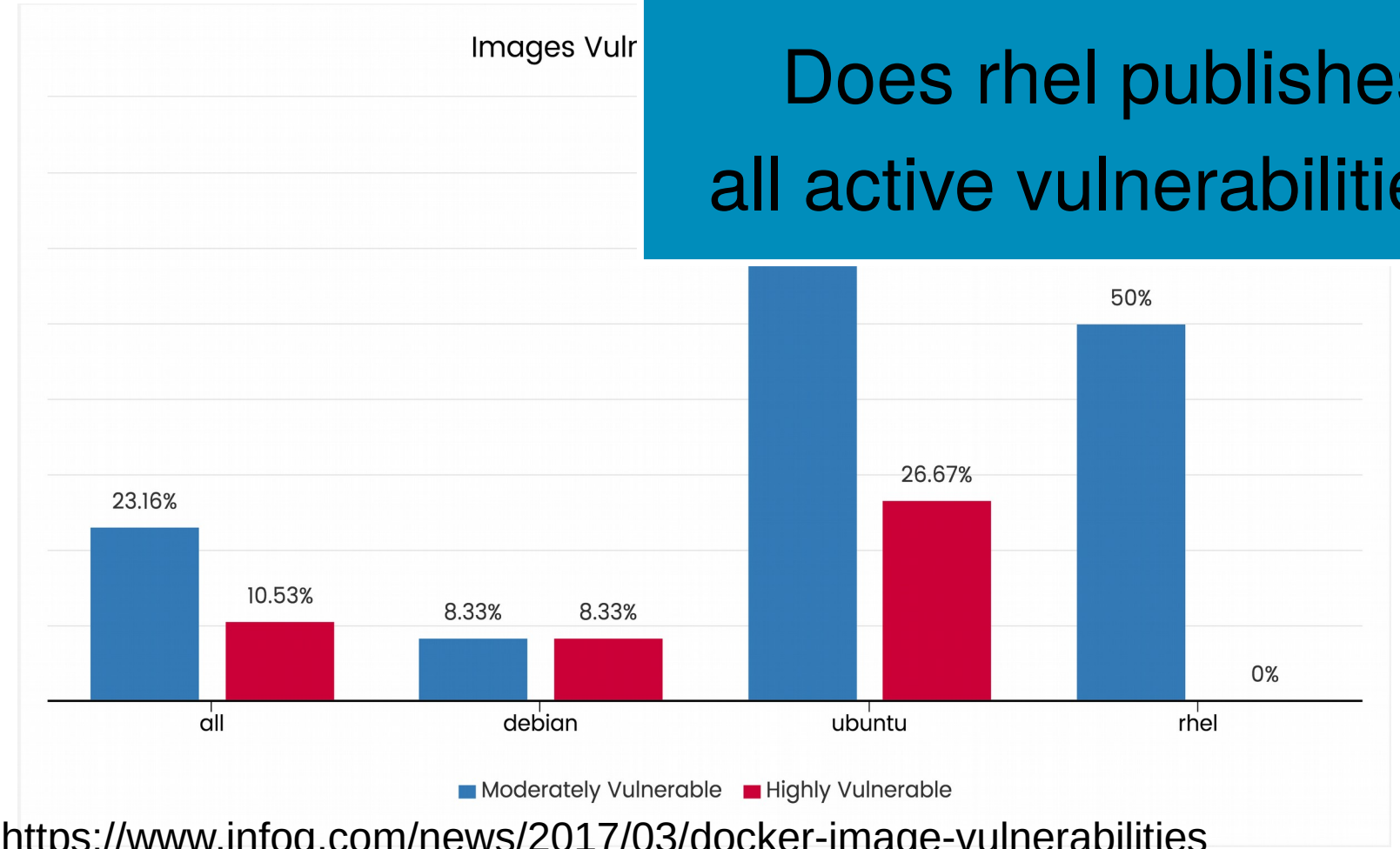
research done with Hendrik Halkow

Image Vulnerable by Distribution



Source: https://www.infoq.com/news/2017/03/docker-image-vulnerabilities

Image Vulnerable by Distribution



Source: https://www.infoq.com/news/2017/03/docker-image-vulnerabilities

OS Selection

- Image size:
  - Alpine: 2.6 MB
  - CentOS: 71.9 MB
  - Debian: 18.3 MB
  - Red Hat Enterprise Linux: 30.8 MB
  - Ubuntu: 31.0 MB

OS Selection

- Image size:
  - Alpine: 2.6 MB
  - CentOS: 71.9 MB
  - Debian: 18.3 MB
  - Red Hat Enterprise Linux: 30.8 MB
  - Ubuntu: 31.0 MB

Does a small image size implies less vulnerabilities?

Maintainability

- Custom *glibc* and *bash* in Alpine

  → Usage of additional *pkg-glibc* in Alpine

  → Not compatible with most Java versions

glibc provides API to the kernel, e.g. *open*, *read*, *write*, *malloc*, …

A manually added Java compatible *glibc* needs to be added

Distroless

Copy only needed files to production images

→ No shell in container

Distroless

- Advantages:
  - Less vulnerable files
- Disadvantages:
  - Requires knowledge
  - For development purpose: original distribution

Hint:

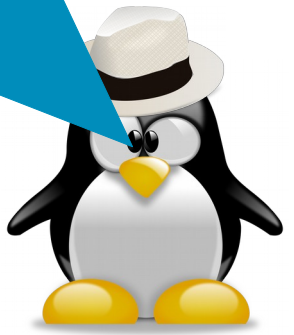
- Scan the original image

  → Results in a lot of false positves

Agenda

- Introduction
- Attack Surface Reduction
- Scanning for Known Vulnerabilities
- Fast Patching
- Conclusion

Whom of you is scanning images for known vulnerabilities?



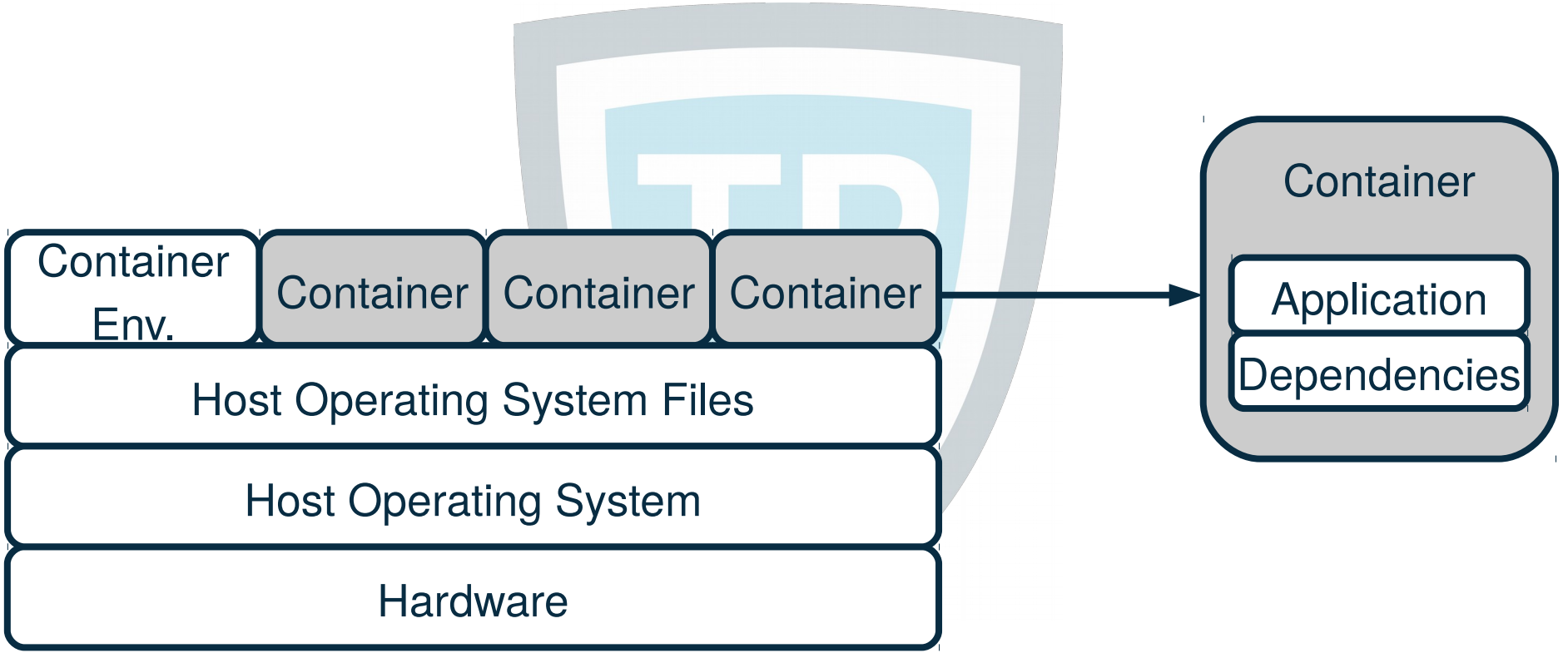
Scanning for Known Vulnerabilties

- What?
- How?
- When?
- Who?

Detection of Components with Known Vulnerabilties

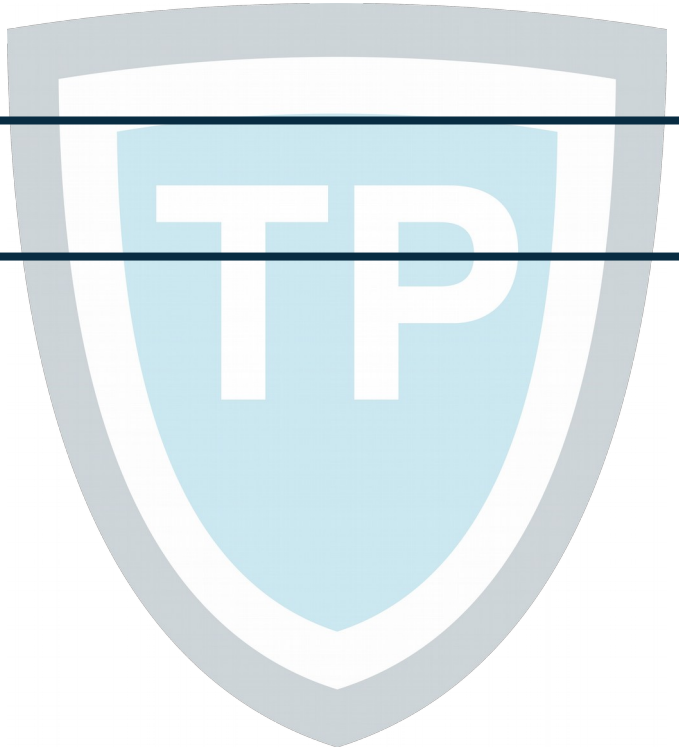
- What?
  - Application
  - Virtualized Operating System (e.g. Container)
  - Operating System
- How?
- When?
- Who?

Docker


Container
Env.
Container
Container
Container
Host Operating System Files
Host Operating System
Hardware
Container
Application
Dependencies

Scanning for Known Vulnerabilties

- What?
- How?
- When?
- Who?



Scan for Vulnerable **Application** Dependencies

- Commercial Tools
  - Blackduck, Artifactory, FOSSA, Whitesource, ...
- Open Source Tools
  - OWASP Dependency Check, NPM, ...

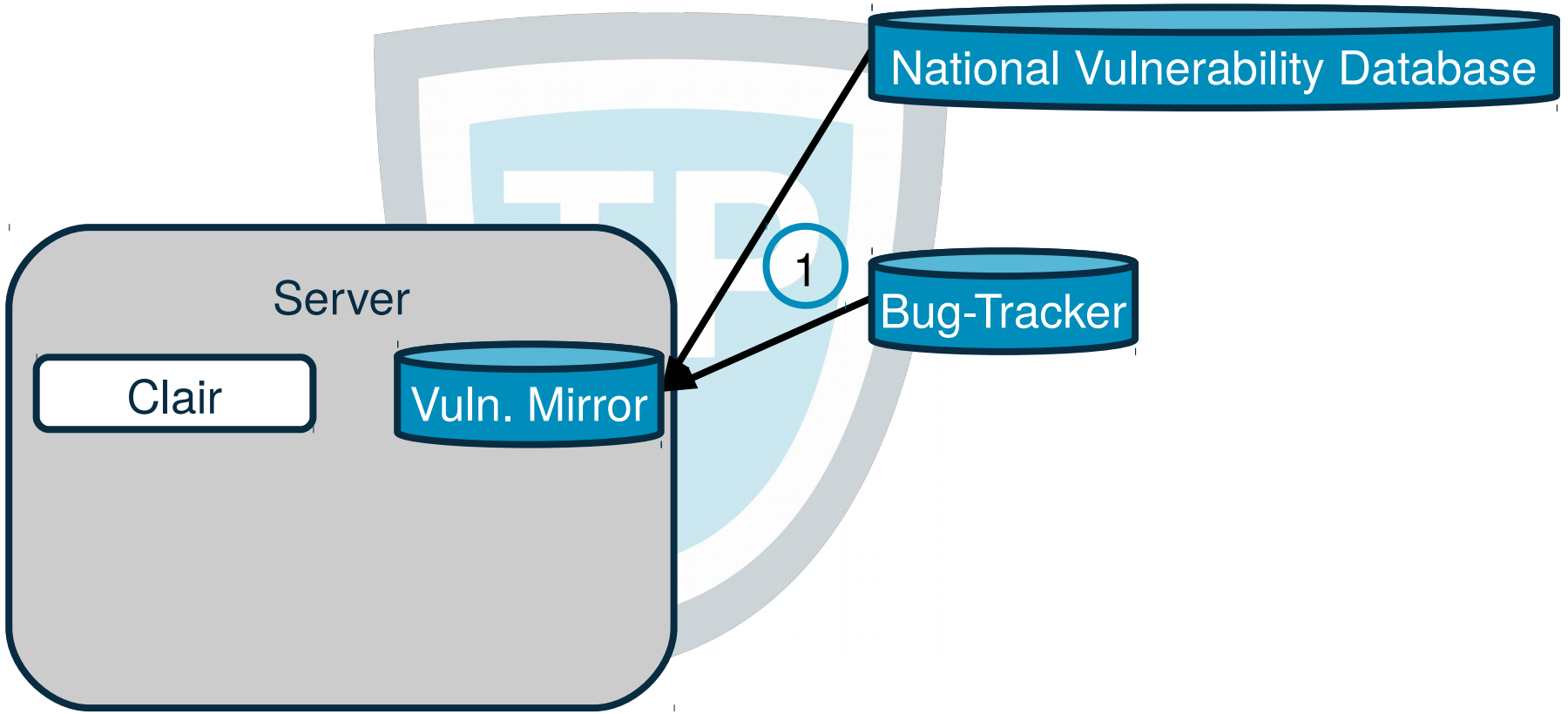
Scan for Vulnerable **Image/Container** Dep.

- Commercial Tools
  - Anchore, Artifactory, Blackduck, hub.docker.com, Tenable.io® Container Security, ...
- Open Source Tools
  - Anchore, CoreOS Clair, OpenSCAP, ...

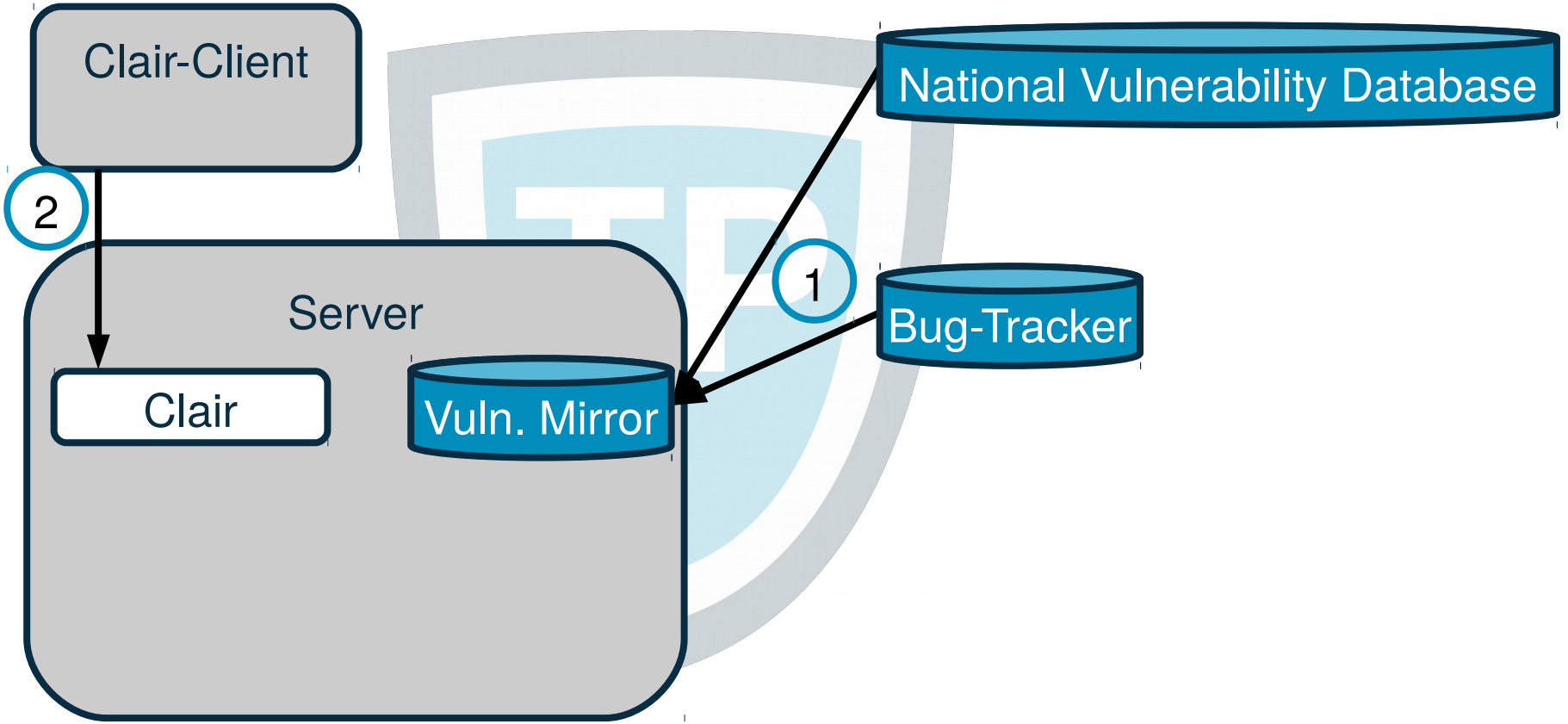
Approaches

- Identification of dependencies incl. version
  - Package managers (App/OS)
  - Fingerprinting (Hash-Sums) of artifacts
  - Pattern recognision
- Vulnerability sources
  - National Vulnerability Database
  - Feeds (e.g. bug tracker)

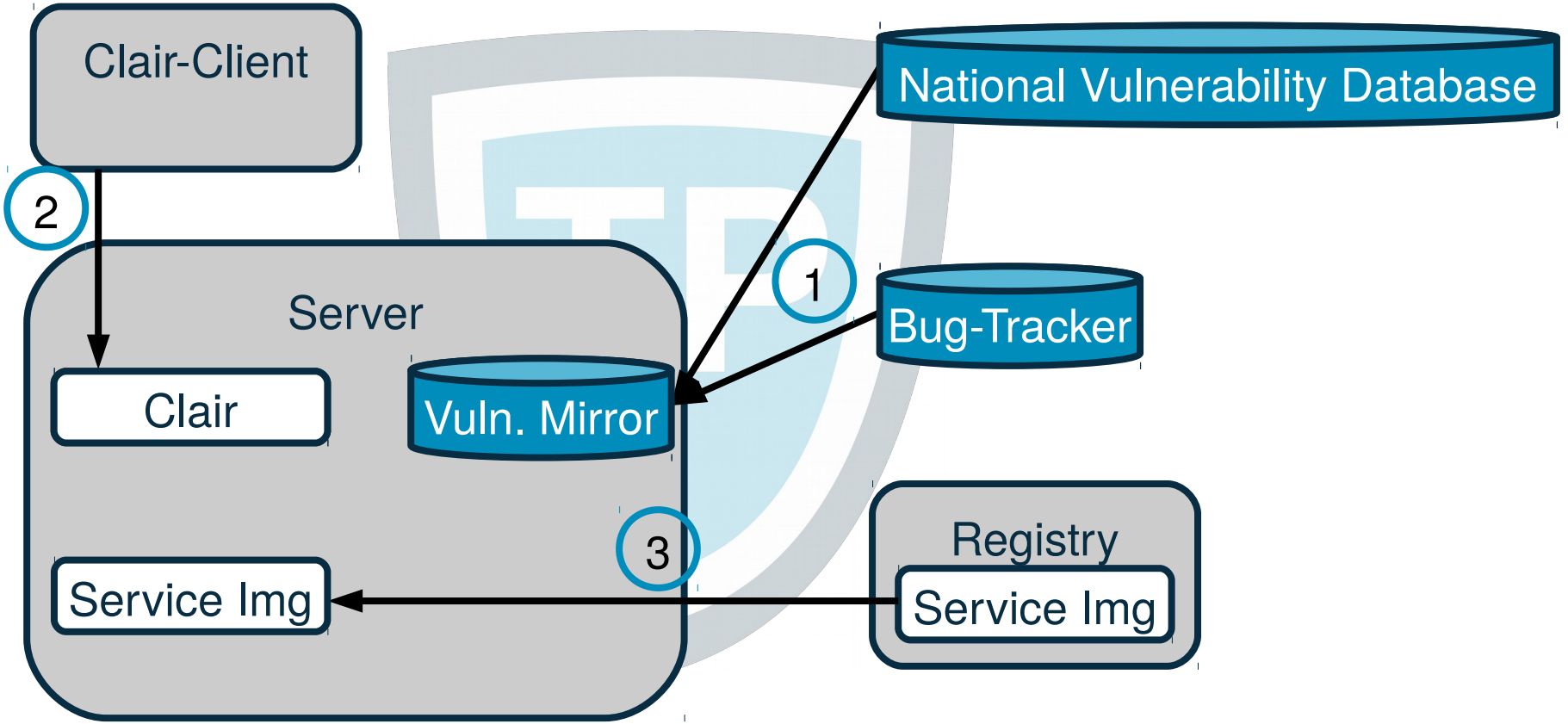
Approach on Example of Clair



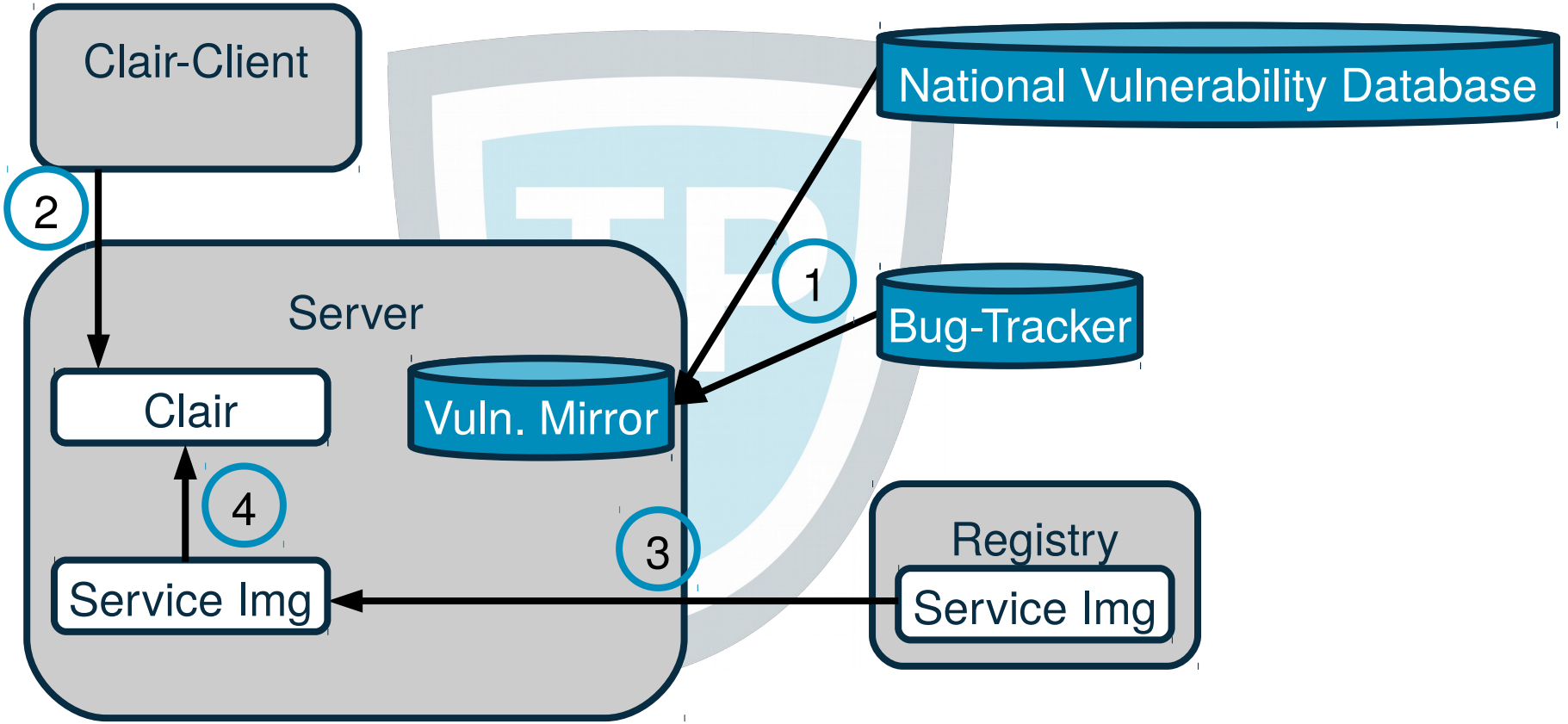
Approach on Example of Clair



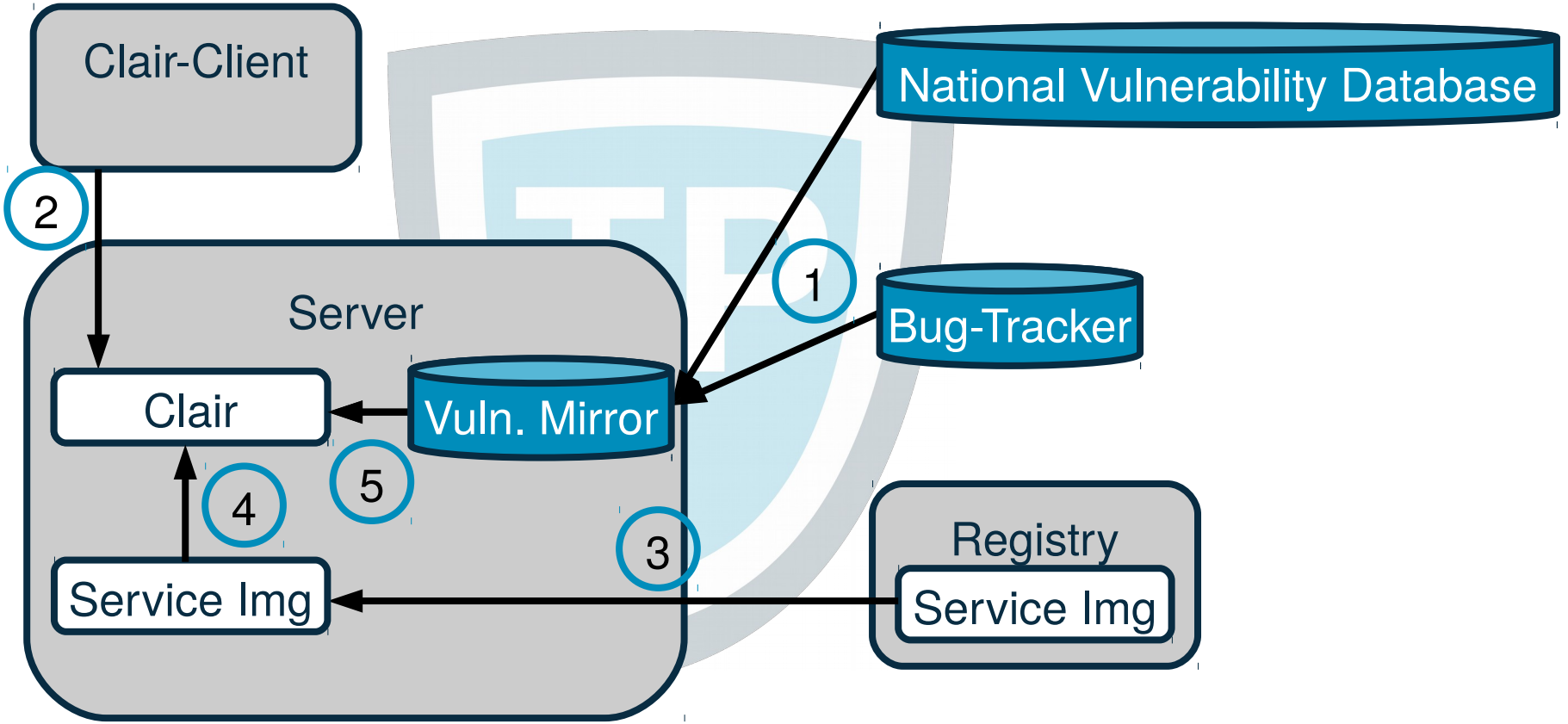
Approach on Example of Clair



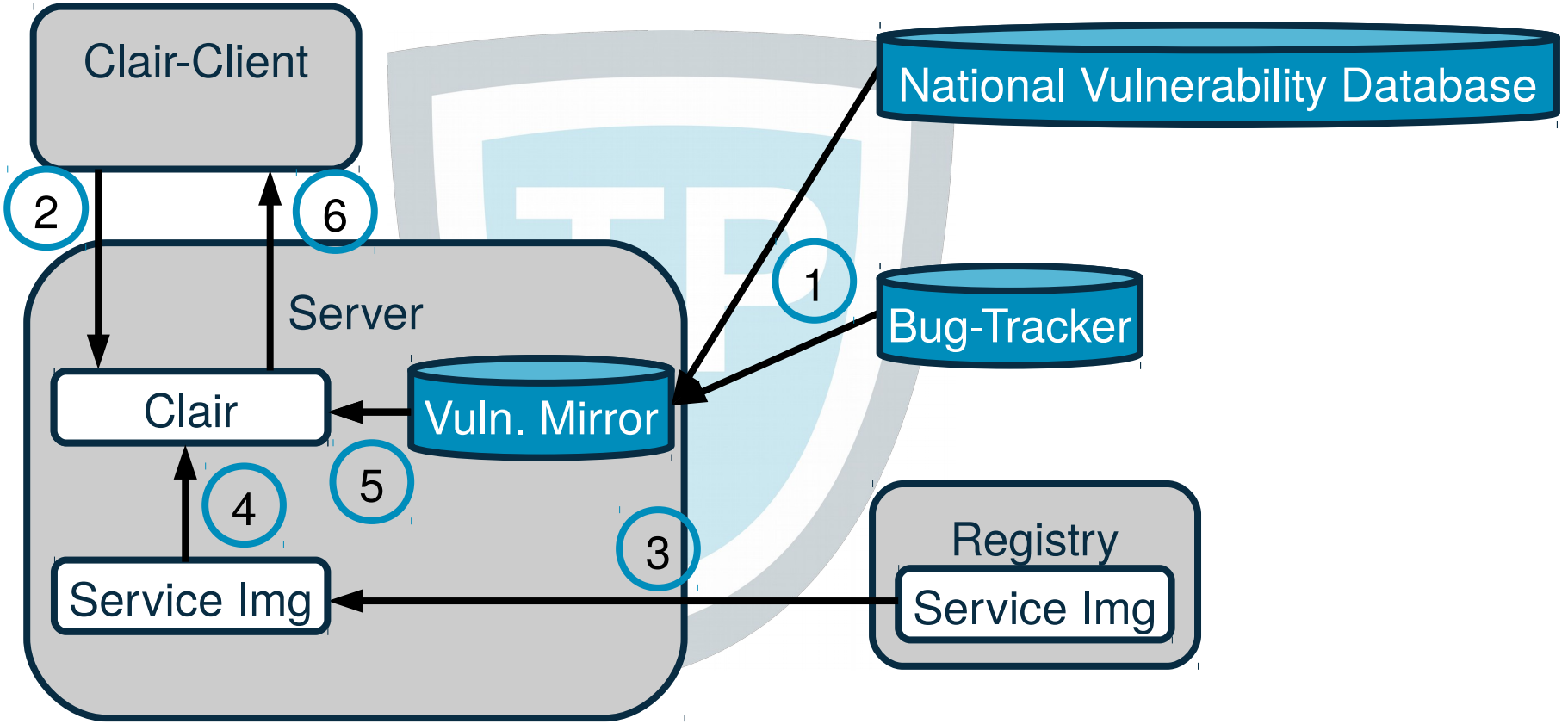
Approach on Example of Clair



Approach on Example of Clair



Approach on Example of Clair



Quality Gates: Thresholds

- Only *n* vulnerabilities
- Only *n* vulnerabilities with criticality *High*
- Dev and ops do not decremt thresholds!

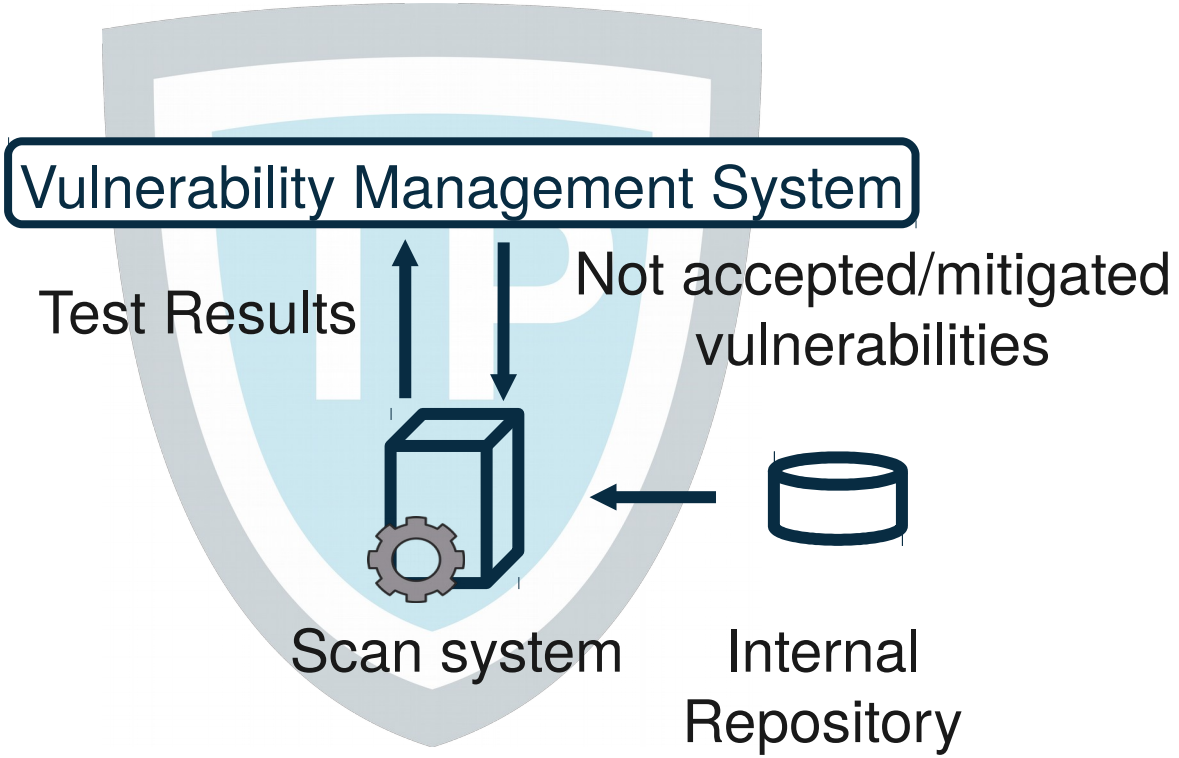
Quality Gates in Regulated Organisations

- Vulnerabilites with ciriticality greater than medium:
  MUST be handeld
- Vulnerabilities with ciriticality low and medium:

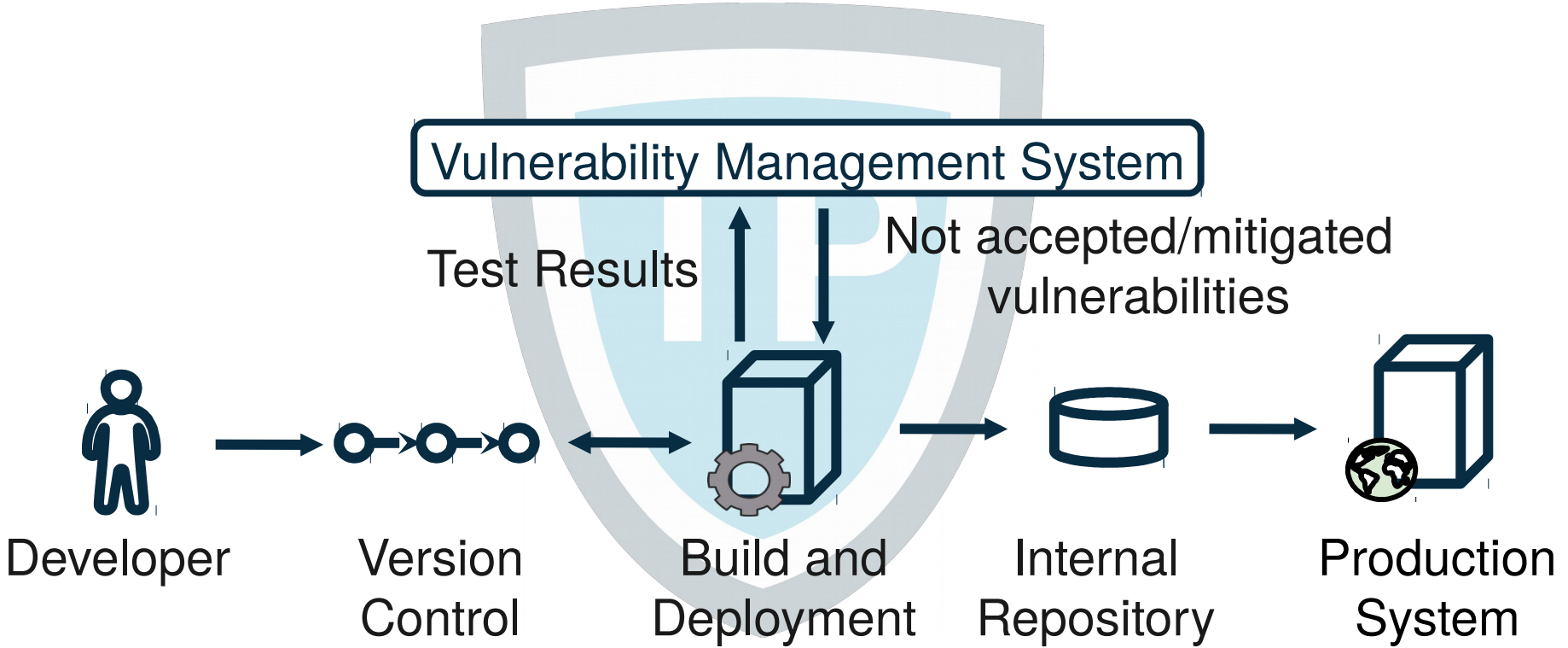
  SHOULD be handeled

  → Not part of automatic quality gate

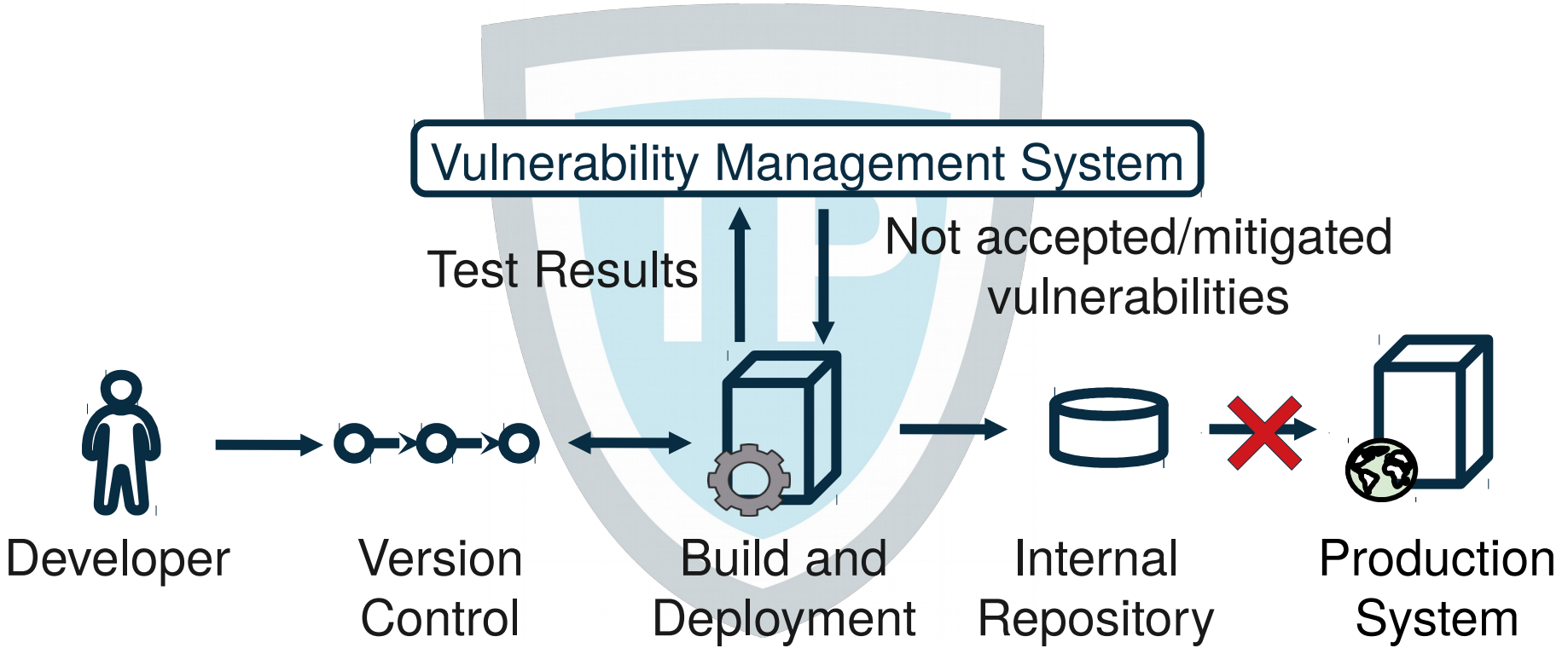
Scan System


Vulnerability Management System
Test Results
Not accepted/mitigated vulnerabilities
Scan system
Internal Repository

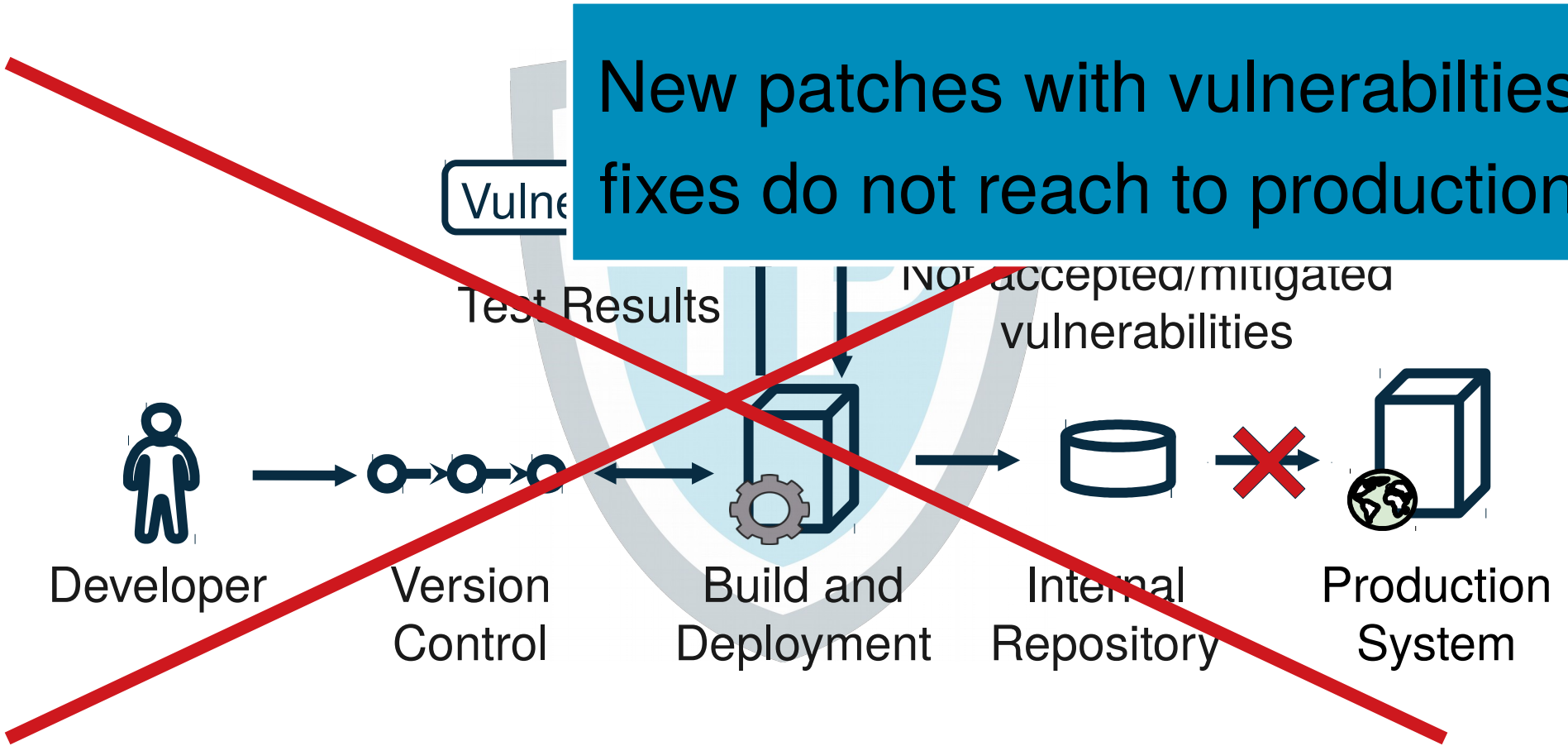
Continuous Integration: Image Scanning



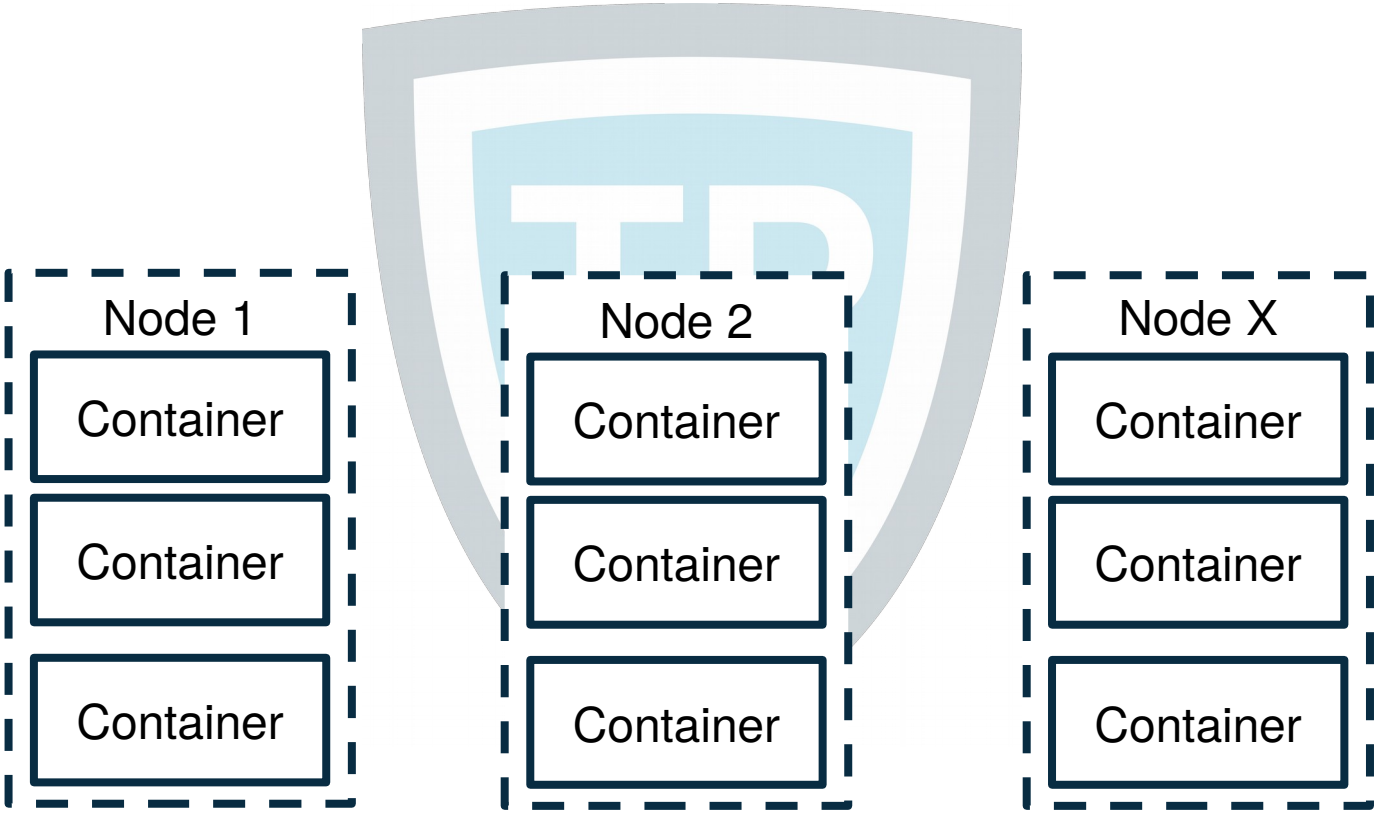
Continuous Integration: Image Scanning



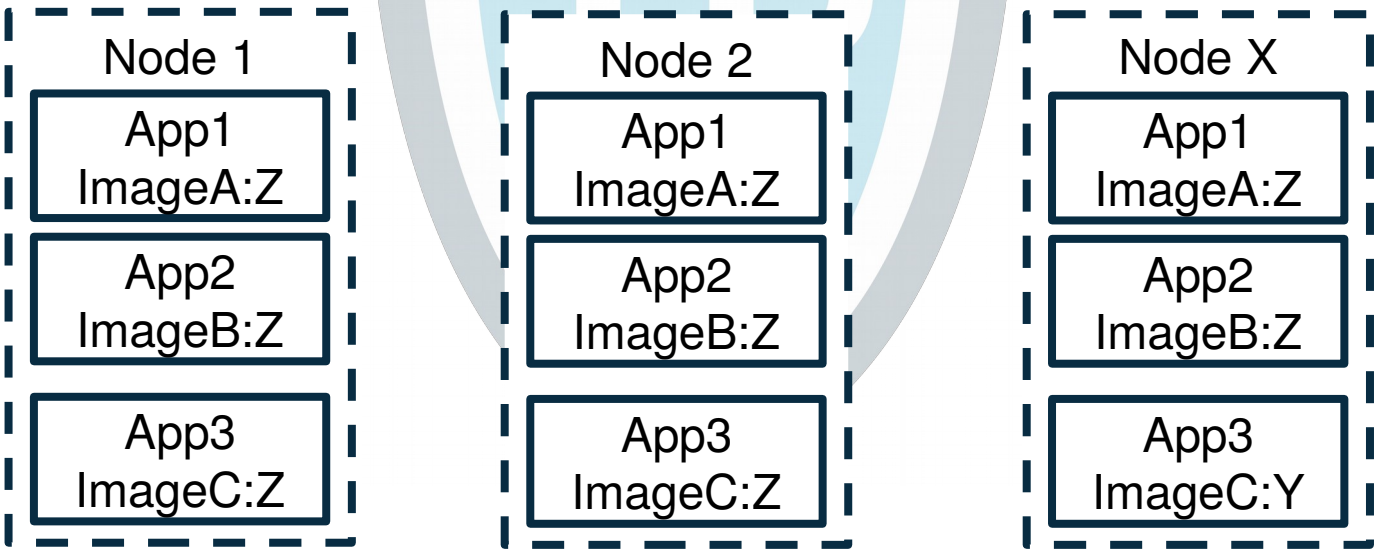
Continuous Integration: Image Scanning



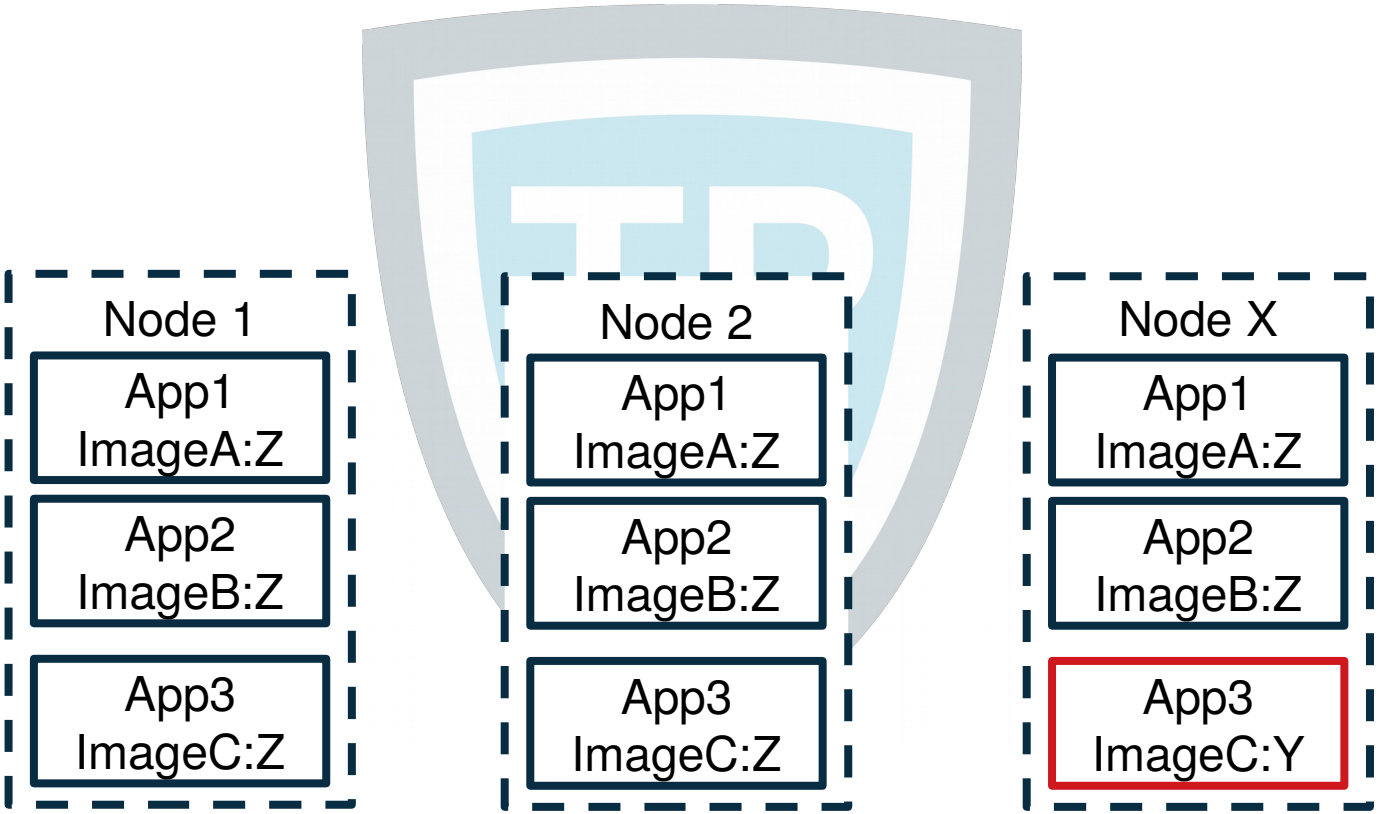
Process to Scan Images



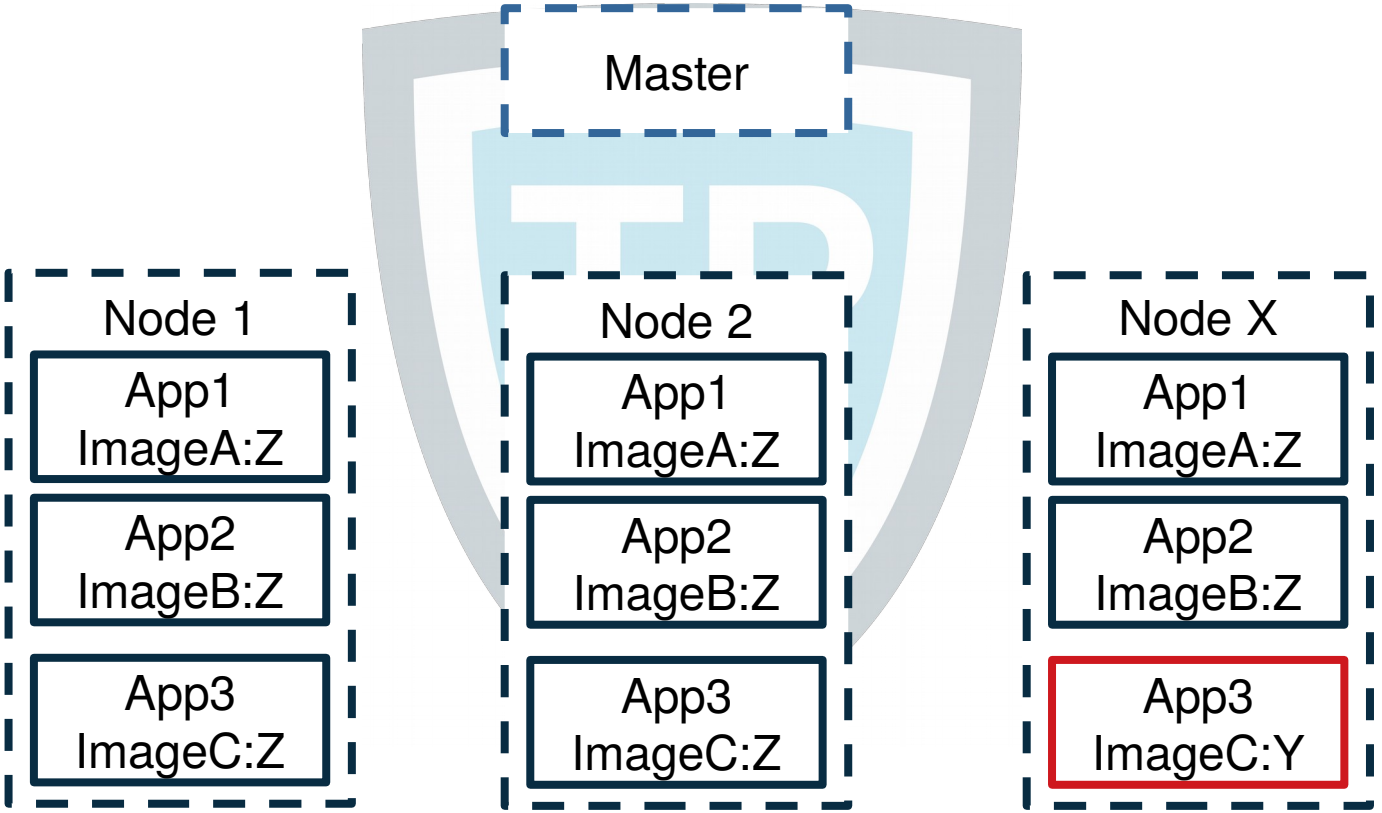
Process to Scan Images



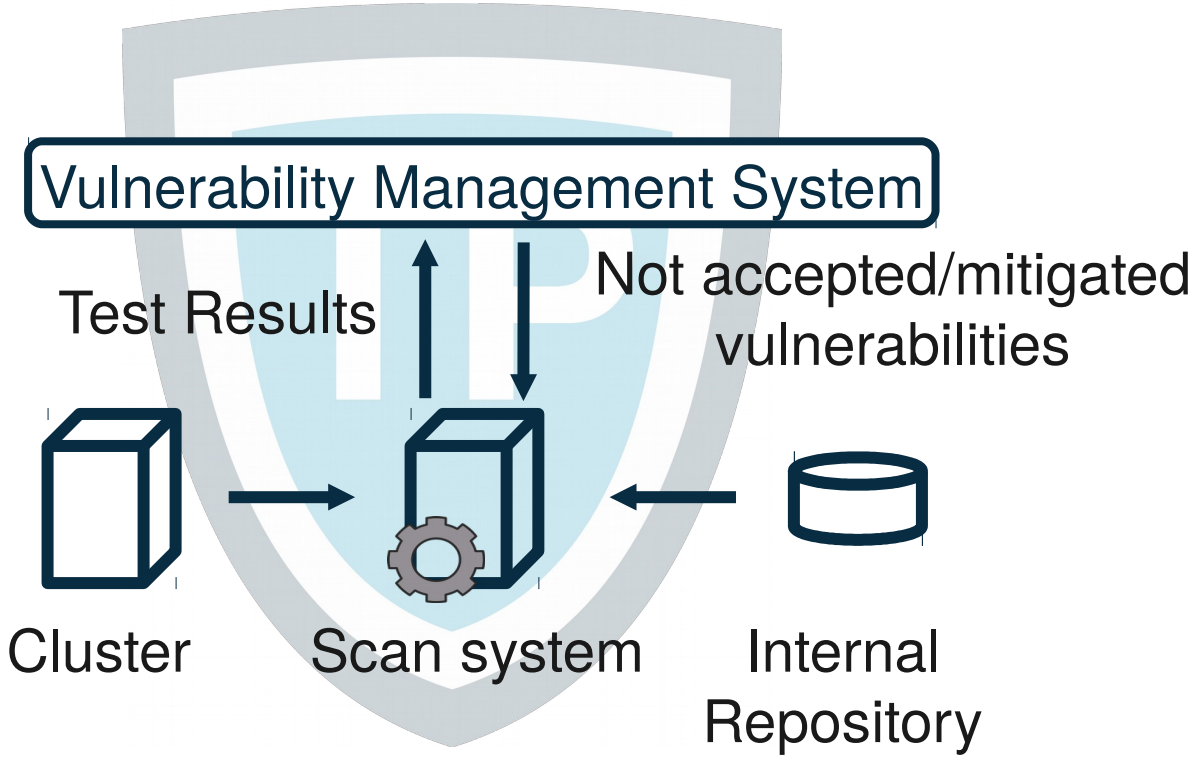
Process to Scan Images



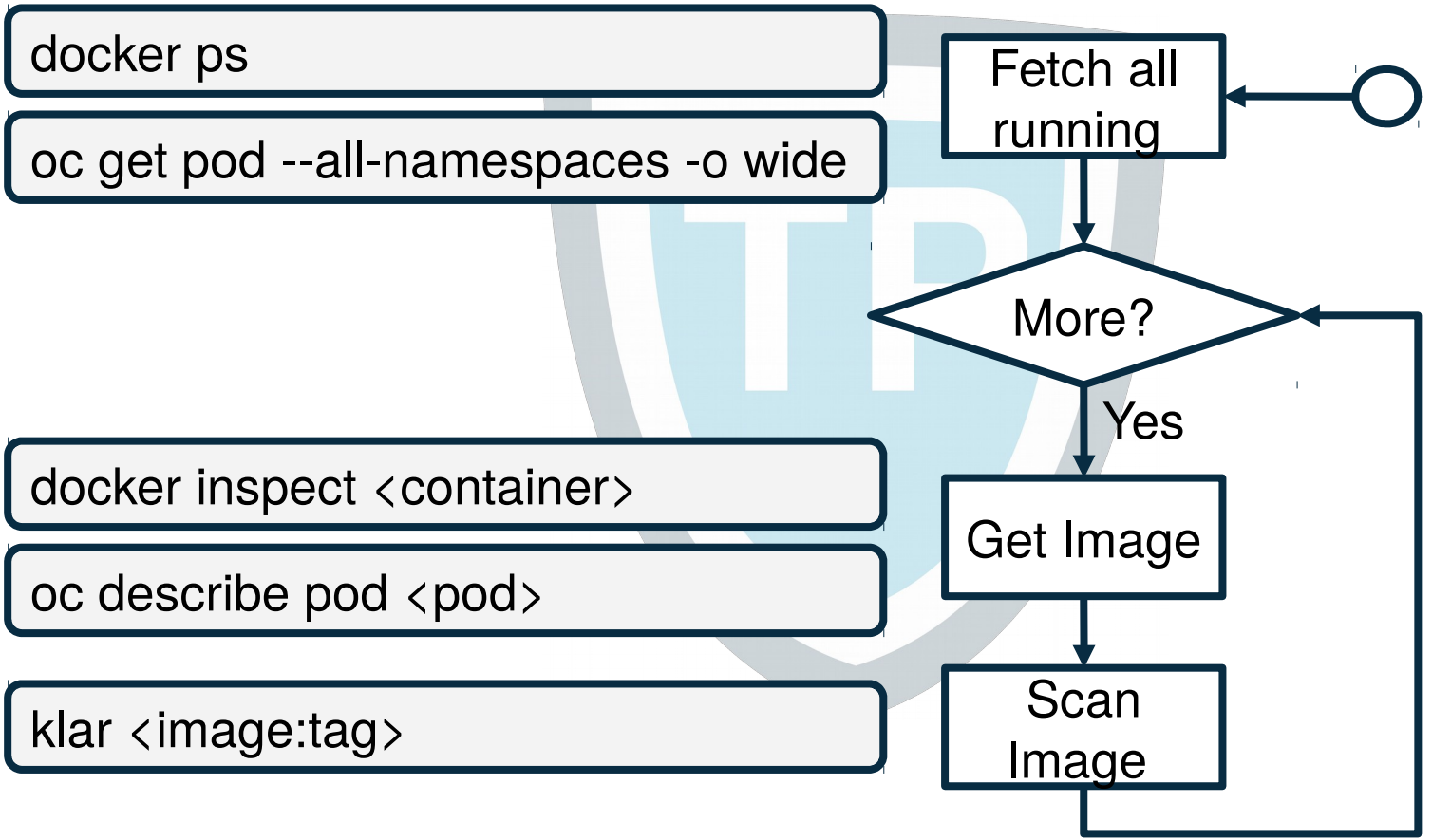
Process to Scan Images



Scan System


Vulnerability Management System
Test Results
Not accepted/mitigated vulnerabilities
Cluster
Scan system
Internal Repository

Process to Scan Images



Praxis Tipp

- Original images (e.g. nginx) often copy compiled version

  → No package manager

  → No dependencies (Open Source)

  → No Vulnerabilities

Praxis Tipp

- Original images (e.g. nginx) often copy compiled version

  → No package manager

  → No dependencies (Open Source)

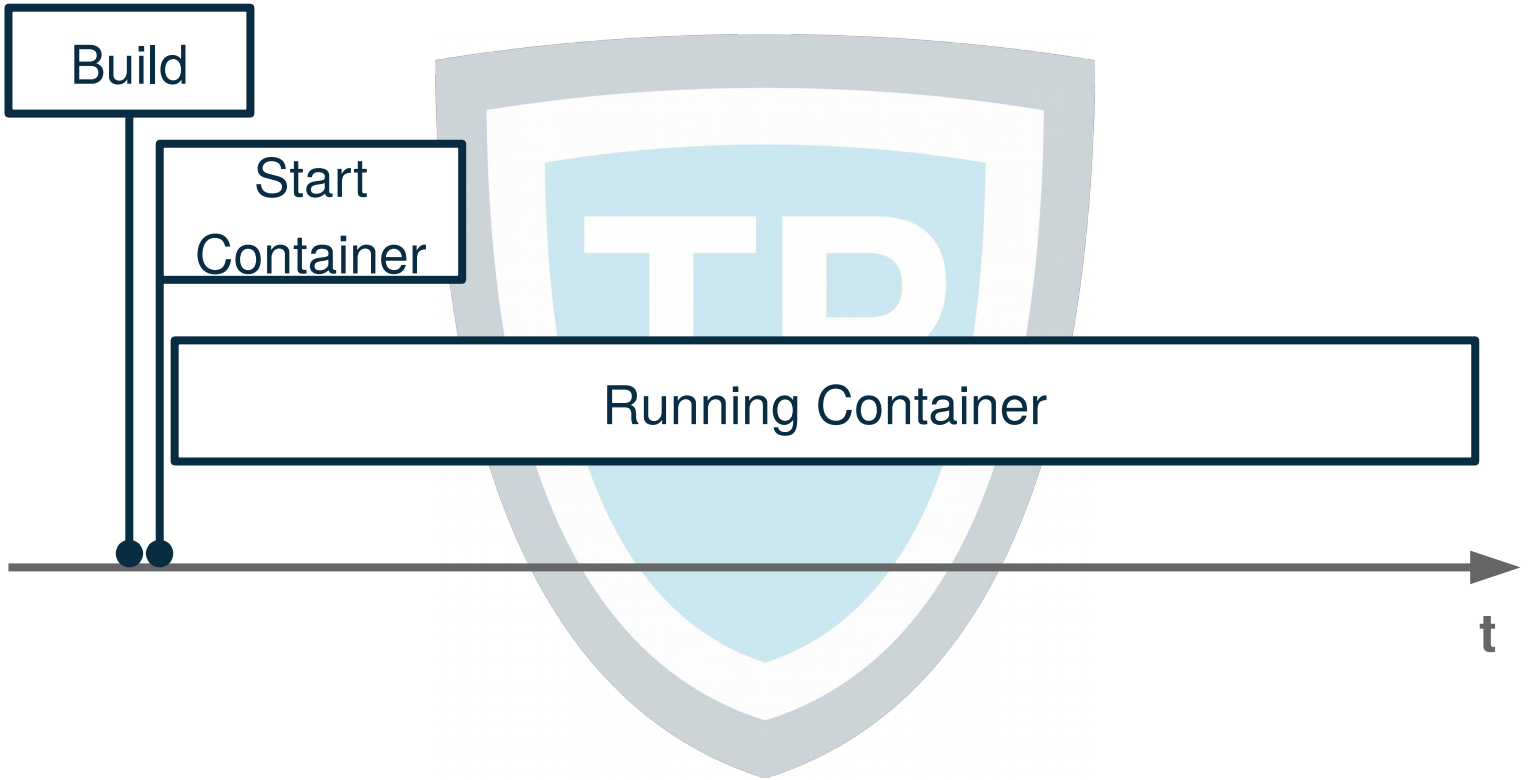
  → No Vulnerabilities

  → Meta package (same version)

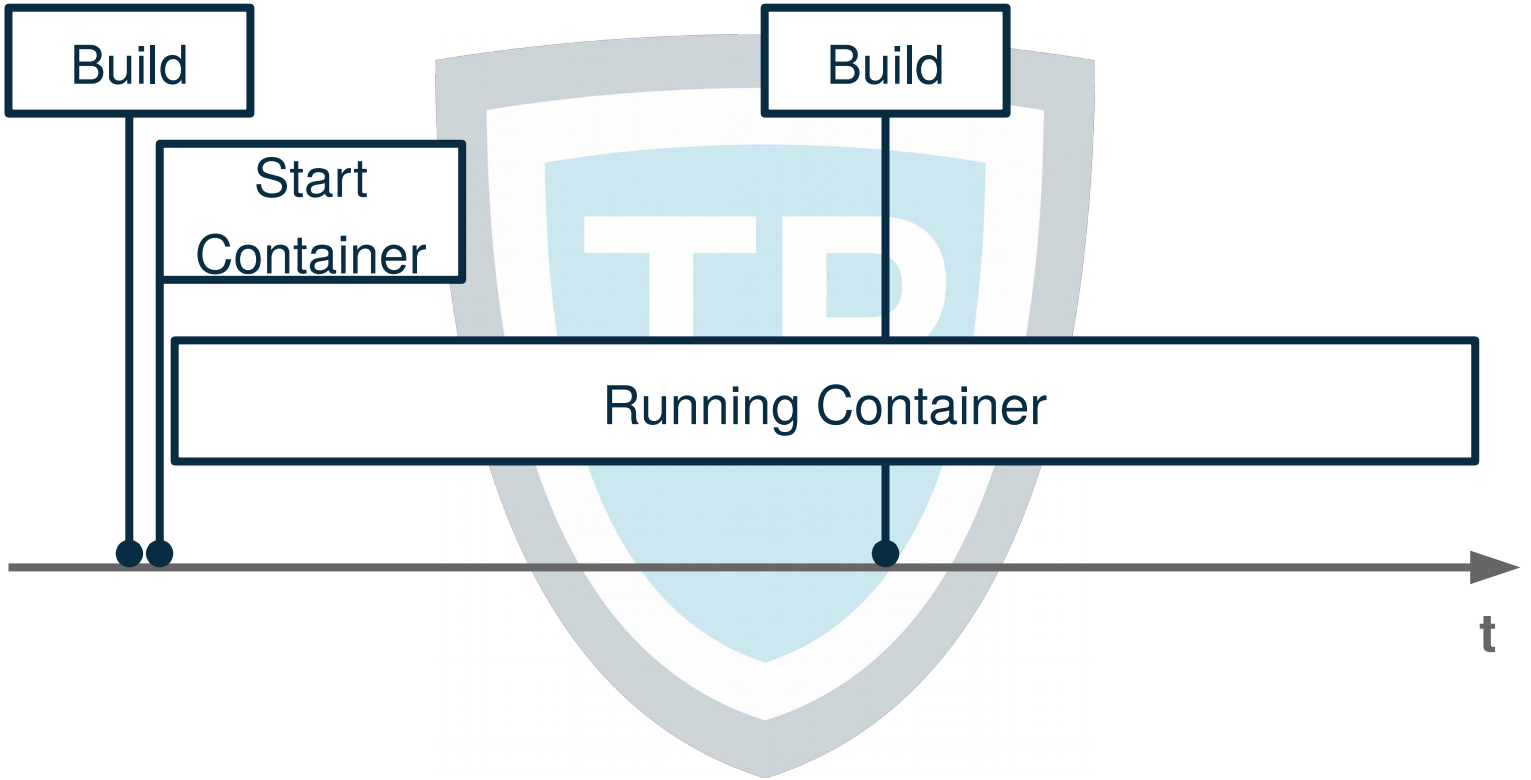
Scanning for Known Vulnerabilties

- What?
- How?
- When?
- Who?

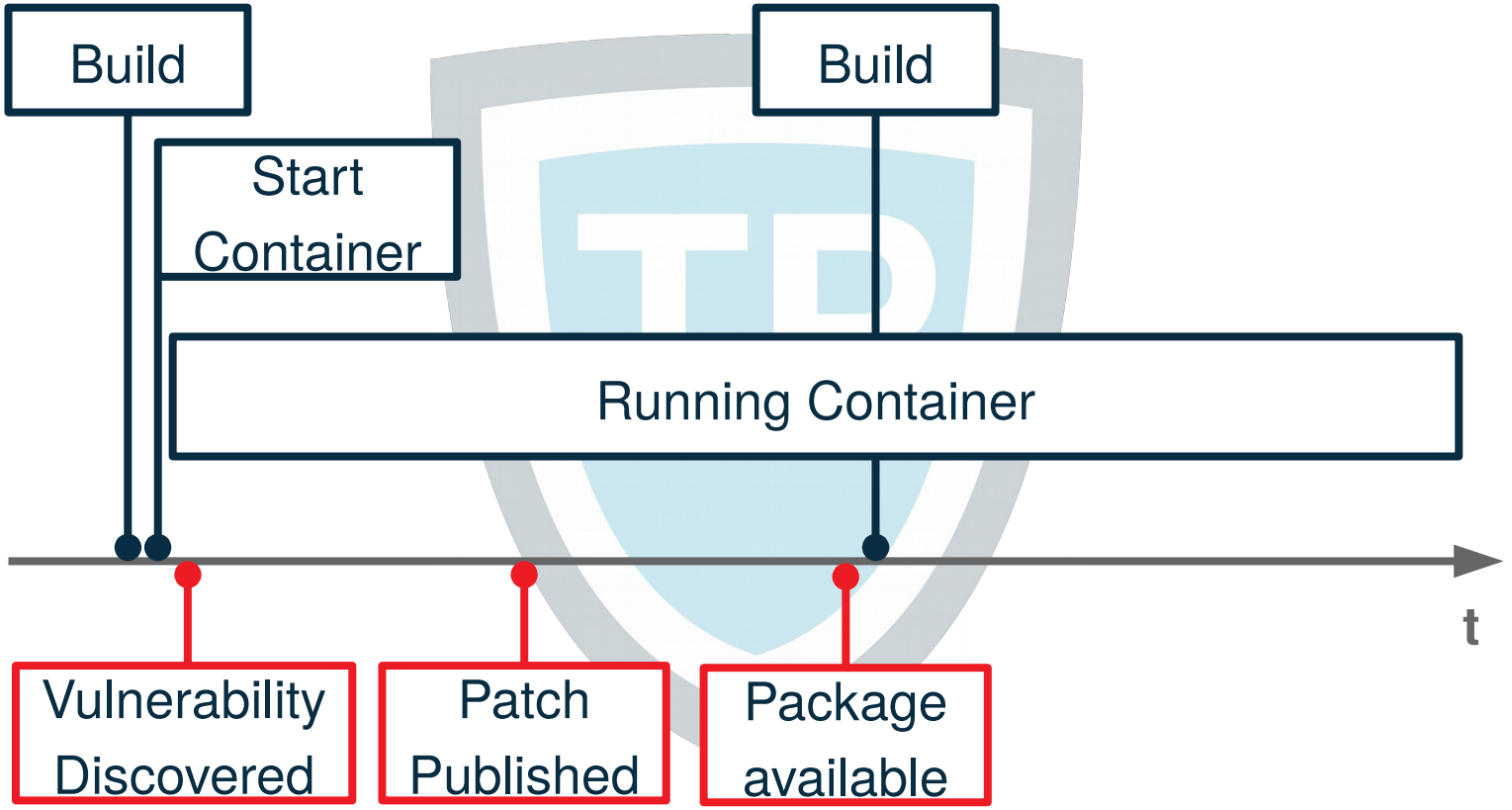
Simplified Vulnerability Lifecycle



Simplified Vulnerability Lifecycle



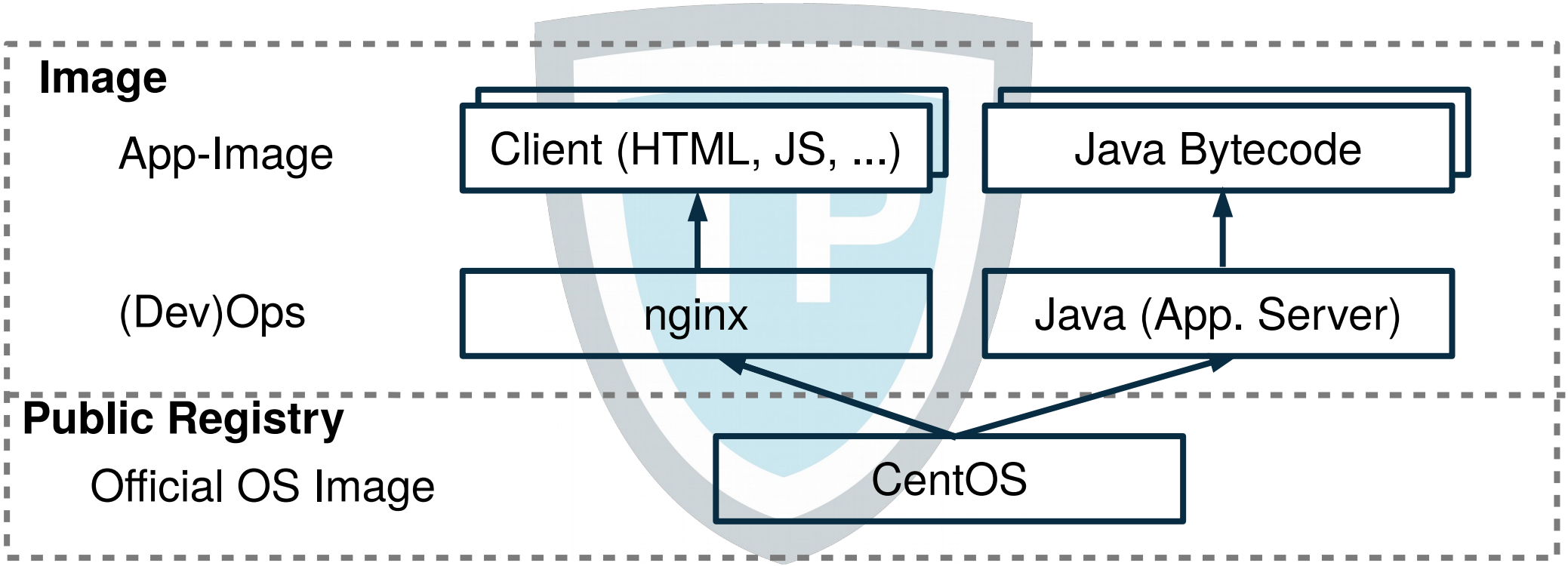
Simplified Vulnerability Lifecycle



Scanning for Known Vulnerabilties

- What?
- How?
- When?
- Who?

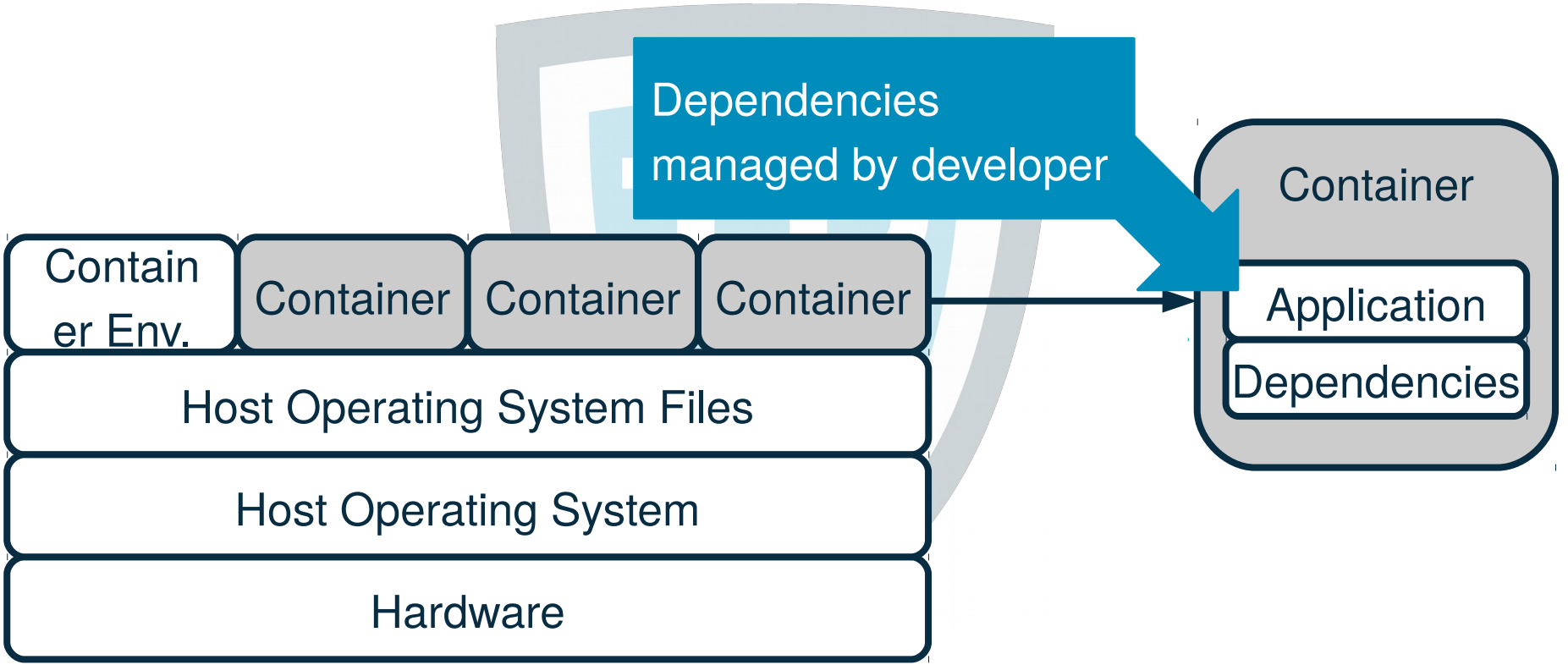
Image Inheritance Tree



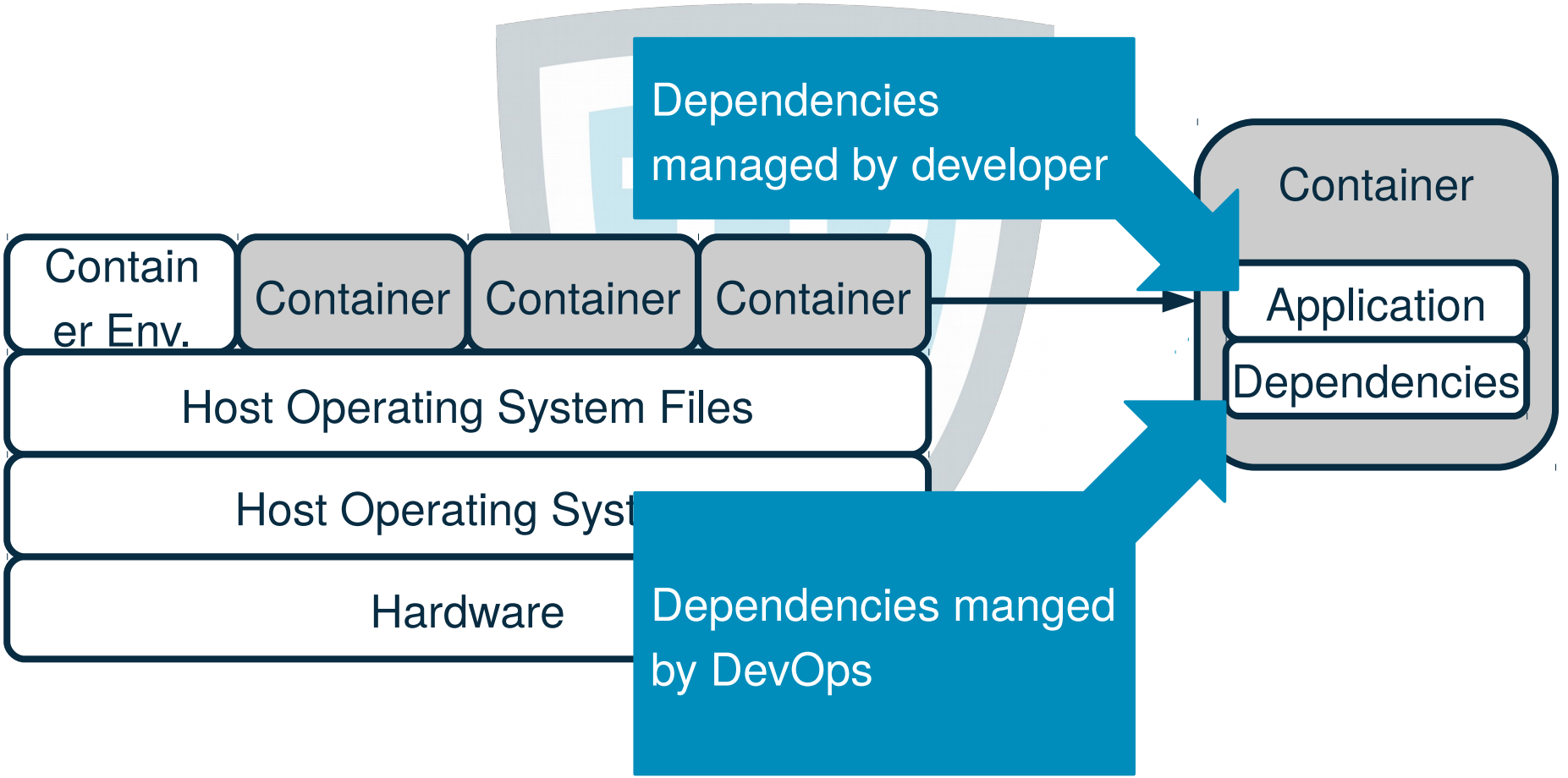
Simplified RACI Matrix

Role	Responsible	Accountable	Consulted	Informed
DevOps-Manager		X		
DevOps-Team	X			
CTO (No Patch)			X	
Customers / Projects				X

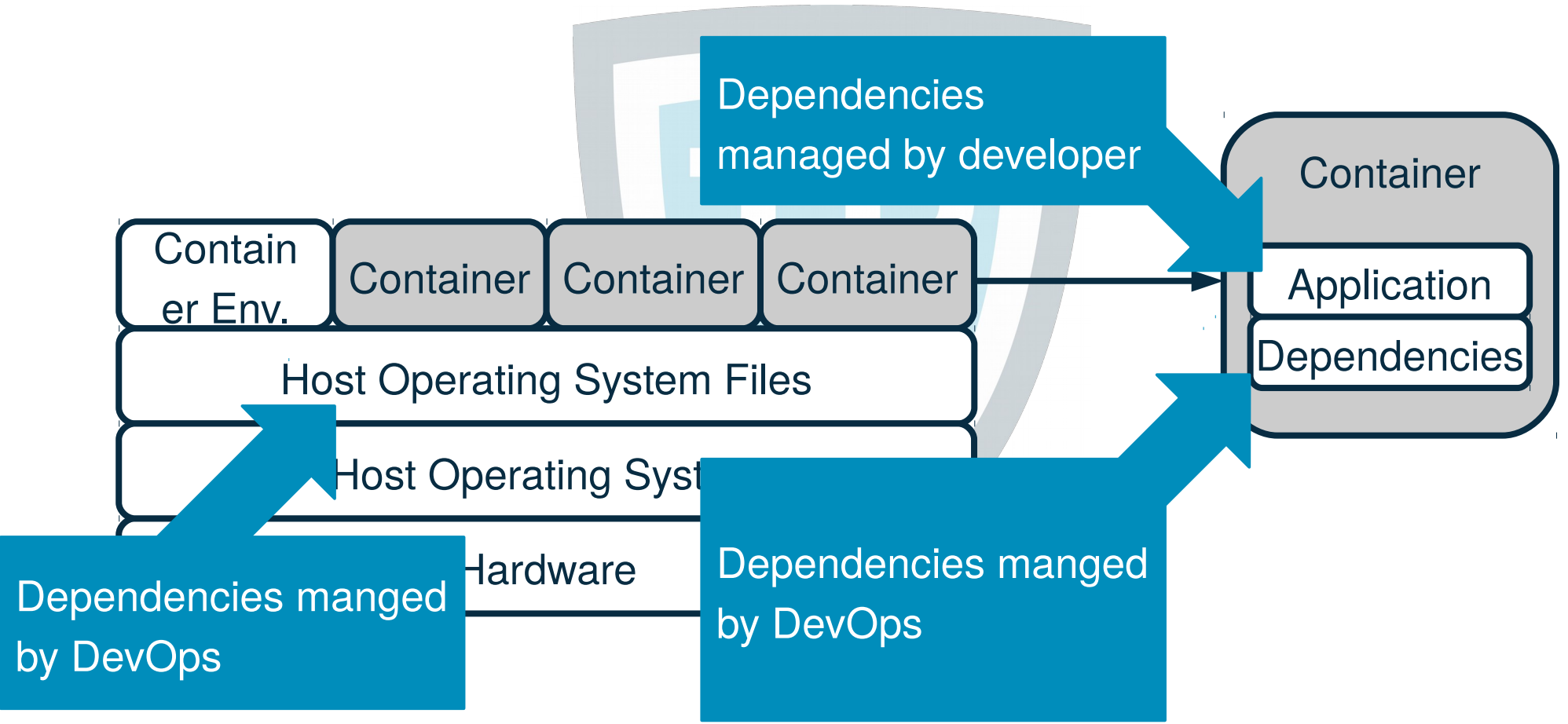
Patch-Responsiblities



Patch-Responsiblities



Patch-Responsiblities



Responding to a Vulnerability

- Transfer
- Avoid
- Mitigate, or
- Accept the risk

Responding to a Vulnerability

- Transfer
- Avoid
- Mitigate, or
- Accept the risk (e.g. temporarily)

Scenario: Critical Vulnerability in *glibc*

What will you do?

- Fix it by yourself (do you have C/C++ developers?)
- Decommission system
- Transfer risk
- Accept (wait for patch in distribution)

Example Denial of Service

CVE-2017-8804

debian

Name	CVE-2017-8804
Description	The xdr_bytes and xdr_string functions in the GNU C Library (aka glibc or libc6) 2.25 mishandle failures of buffer deserialization, which allows remote attackers to cause a denial of service (virtual memory allocation, or memory consumption if an overcommit setting is not used) via a crafted UDP packet to port 111, a related issue to CVE-2017-8779.
Source	CVE (at NVD; CERT, LWN, oss-sec, fulldisc, bugtraq, EDB, Metasploit, Red Hat, Ubuntu, Gentoo, SuSE, Mageia, GitHub code/issues, web search, more)
NVD severity	high (attack range: remote)
Debian Bugs	862086

Example CVE-2015-0235

CVE-2015-0235

Name	CVE-2015-0235
Description	Heap-based buffer overflow in the __nss_hostname_digits_dots function in glibc 2.2, and other 2.x versions before 2.18, allows context-dependent attackers to execute arbitrary code via vectors related to the (1) gethostbyname or (2) gethostbyname2 function, aka "GHOST."
Source	CVE (at NVD; CERT, LWN, oss-sec, fulldisc, bugtraq, EDB, Metasploit, Red Hat, Ubuntu, Gentoo, SUSE bugzilla/CVE, Mageia, GitHub code/issues, web search, more)
References	DLA-139-1, DSA-3142-1
NVD severity	high (attack range: remote)
Debian Bugs	776391

Agenda

- Introduction
- Strategies to Handle Patchmanagement
- Scanning for Known Vulnerabilties
- Fast Patching
- Conclusion

Periodical Patch Management with Containers

Nightly/Weekly:

- Pull external images
- Perform upgrade/update
- Build project images
- Test images (e.g. A/B-Testing)
- Destroy and start containers

Pitfalls

- Image caching during build (no change → old version)
- CentOS: *yum update --security*

Fail Secure?

https://bugs.centos.org/view.php?id=3578

CentOS Bug Tracker

Activities

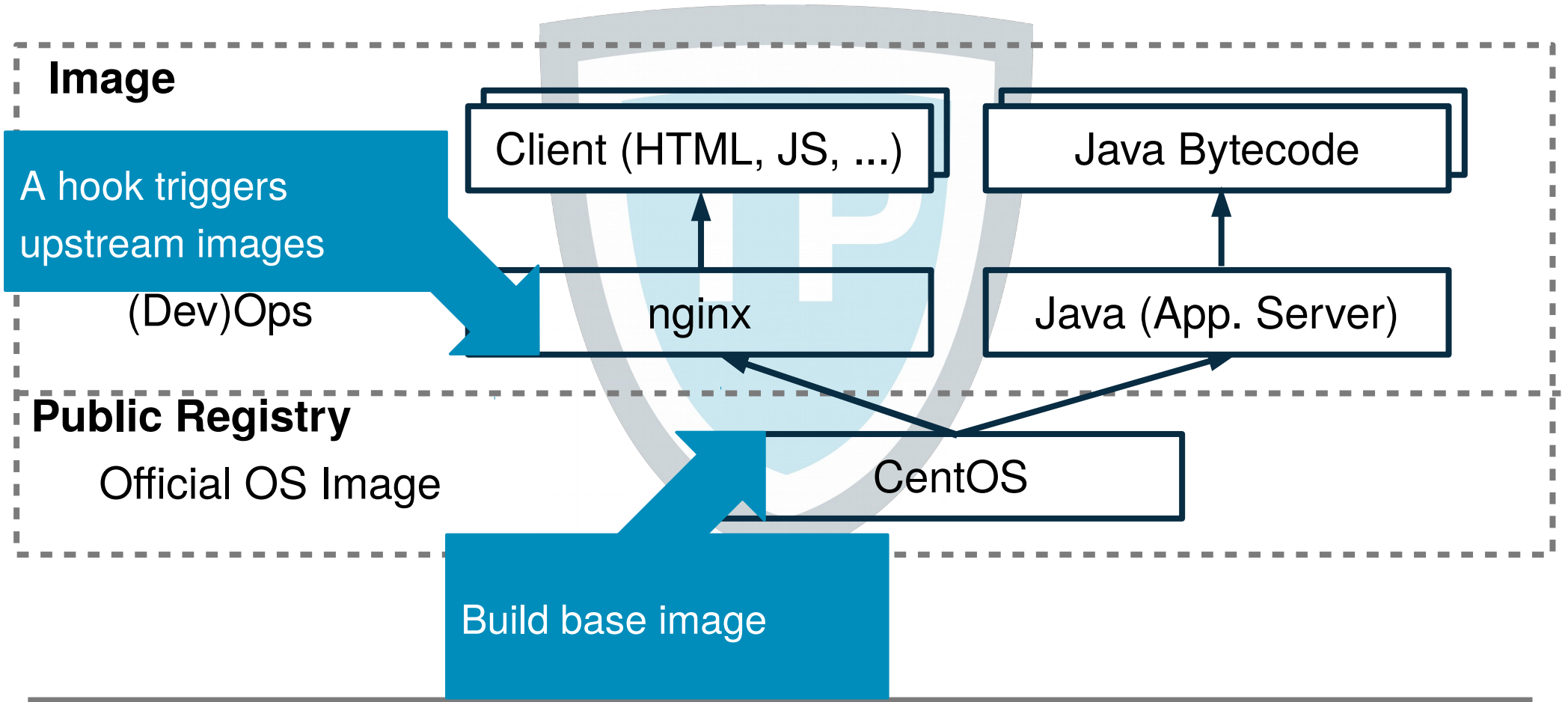
kbsingh@karan.org

2009-04-26 14:36

administrator ~0009256

we dont support yum security in any of the centos repos at the moment

Image Updates



Agenda

- Introduction
- Attack Surface Reduction
- Scanning for Known Vulnerabilities
- Fast Patching
- Conclusion

Pro/Con

Nightly build+deploy

- Good for external images
- High testing requirement
- Continuous Delivery required

Scanning

- Not blind
- High effort in analysing vulnerabilities

→ Time consuming

Conclusion

- Patching is not easy
- Detection of vulnerabilities in running containers
- Quality gates are important and a centralized vulnerability mangement (system)

Questions?

✉ devsecops19@pagel.pro

Trainings	When	Where
Docker Security Workshop	30.08.2019	Hamburg
DevSecOps Workshop	09.09.2019	Hamburg
Sicherheit in Webanwendungen	16.09.2019	Hamburg

Backup