
MUNICH
SECURE CONTAINERS
Do component reduction strategies
fix your container security nightmares?
secureIO

PRESENTERS

Michael Wager

- Developer, Hacker, Consultant
- Topics: DevSecOps, Automated Security Assurance, Vulnerability Management
- Interested in music, traveling, cooking and all stuff cyber security related

Michael Helwig

- Strategic Consulting: Security Programs / SSDLC / DevSecOps
- Interested in all things security (Security Testing, Threat Modeling, Cloud, Reverse Engineering, …)
- Company founder

AGENDA

1. Intro: Container Security Challenge
2. Component reduction methods ("distroless" concept)
3. Demo (Node.js)
4. Research & Comparison
5. Conclusion

WHY ARE CONTAINERS A SECURITY CHALLENGE?

Lack of processes in early adoption

- Lack of transparency into vulnerabilites in early adoption phases
  (no container scanning, no awareness, no CI/CD integration)
- No trusted repositories / base image selection
- Containers are everywhere (Cloud Services, vendor delivieries, …)

Responsibility Shift (Shift-Left)

- Containers managed by dev teams; servers and OS traditionally managed by ops team.
- "It's not our code"

Complex attack surfaces

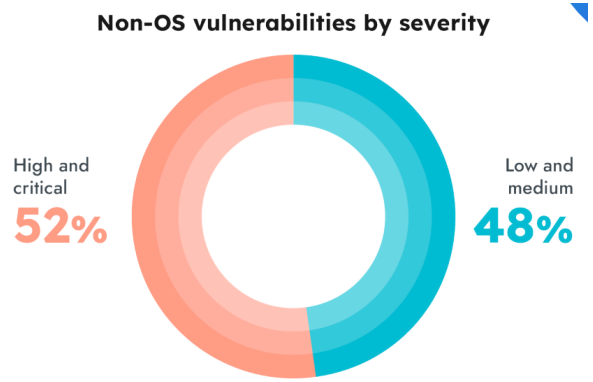
- Application
- OS layer / container images
- Configuration
- Network
- Hypervisor

Security degrades over time

- Security is not constant, new vulnerabilities and attack vectors appear. The more you have to maintain, the more effort you need.

CONTAINER SECURITY AND VULNERABILITY TRENDS

- High number of images with high or critical vulnerabilities
- Only a small number (2%) are exploitable but a large number is patchable
- Most of the vulnerable libraries are not actually used or needed by the application



Source: https://sysdig.com/blog/2023-cloud-native-security-usage-report/

WHY ARE CONTAINERS A SECURITY CHALLENGE?

"the likelihood of a greater number of vulnerabilities increases with the complexity of the software architectural design and code."

Minimize your attack surface

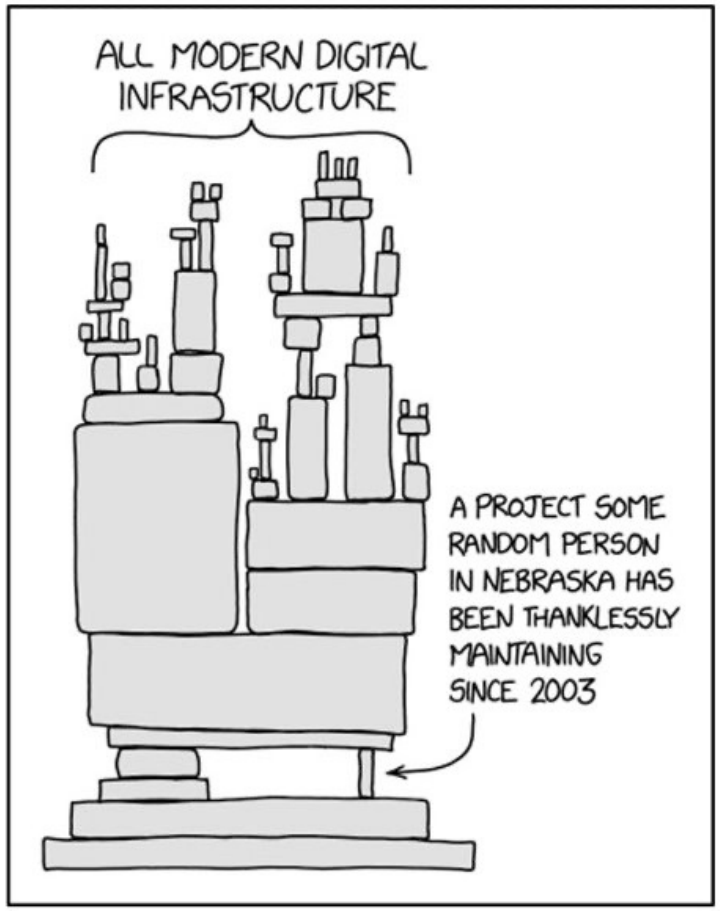
https://github.com/OWASP/DevGuide/blob/master/02-Design/01-Principles%20of%20Security%20Engineering.md

„IT‘S SECURE BECAUSE IT‘S RUNNING IN A CONTAINER“

„IT‘S SECURE BECAUSE IT‘S RUNNING IN A CONTAINER“

https://youtu.be/RMqjQ_i9eP0?si=AnCp9gWmIuLhia0s&t=227

COMPARISON WITH OPEN SOURCE COMPONENTS



https://daniel.haxx.se/blog/wp-content/uploads/2021/04/xkcd-2347-curl-adjusted-by-tjost.png

- Teams are responsible for the functionality and security of OSS dependencies - so they are responsible for the security of the selected base images

CONTAINER IMAGE SCANNERS

- Goal: identify known vulnerabilities (CVEs) in container images
- Easy to integrate into CI/CD pipelines

Some tools: trivy, Anchore grype, docker scout, twistcli

COMPONENT REDUCTION TOOLS

Google "distroless"

- Open source project by google (since 2007)
- Provides prod ready images for several runtimes (java, node.js, go)
- Very small in size (e.g. static-debian11: ~2MB)

Ubuntu "chisel"

- Open source project by Canonical (since 2023)
- Provides some prod ready images, others need to be built yourself („chiseled“)
- Ubuntu long-term supported (LTS) releases (0 critical 0 high findings, 24h)

RedHat UBI "micro"

- Based on RedHat's "Universal base images"
- RedHat enterprise linux (RHEL) well maintained
- Same security response team, the same security hardening

Chainguard images

- Security vendor founded in 2021
- Provides prod ready images for a lot of popular runtimes (free & support)
- Hardened images with 0-known vulnerabilities

DEMO

Sourcecode available: github.com/mwager/nodejs_exploit

RESEARCH

- Research in collaboration with University of Applied Sciences Augsburg
- 3 Research Questions:
  - RQ1: Does the reduction of components significantly reduce the amount of vulnerabilities within a container image?
  - RQ2: Are typical vulnerabilities found through container security scanners actually exploitable and therefore a risk to the application?
  - RQ3: What are implications on development, deployment and maintenance when introducing component reduction methods?

Hochschule Augsburg University of Applied Sciences

RESEARCH - RESULTS

Image	total	critical	high	medium	low
alpine_latest:latest	0	0	0	0	0
amazonlinux_2:latest	15	1	7	7	0
chainguard-jre_latest:latest	0	0	0	0	0
chainguard-node_latest:latest	0	0	0	0	0
chainguard-php_latest:latest	0	0	0	0	0
chainguard-python_latest:latest	0	0	0	0	0
chainguard-wolfi-base:latest	0	0	0	0	0
chiselled-base_22.04:latest	0	0	0	0	0
distroless-base-debian12:latest	3	0	0	0	3
distroless-java-base-debian12:latest	2	0	0	0	2
distroless-java11-debian11:latest	9	0	0	0	9
distroless-java17-debian12:latest	2	0	0	0	2
distroless-nodejs18-debian12:latest	4	0	0	0	4
distroless-nodejs20-debian12:latest	4	0	0	0	4
distroless-python3-debian12:latest	9	0	0	1	8
distroless-static-debian12:latest	0	0	0	0	0
ibmjava_jre:latest	12	0	1	3	8
node_14-slim:latest	37	2	6	4	25
node_16-slim:latest	31	1	3	2	25
node_18-slim:latest	8	0	0	2	6
node_18.14.1-alpine:latest	13	1	5	7	0
node_20-alpine:latest	0	0	0	0	0
node_20-slim:latest	8	0	0	2	6
php_fpm-buster:latest	63	2	7	7	47
php_latest:latest	14	0	0	2	12
piotrkardasz-php-distroless_8.1-debug	178	9	29	90	50
redhat-ubi8-ubi-micro:latest	0	0	0	0	0
redhat-ubi8-ubi-minimal:latest	32	0	3	14	15
redhat-ubi9-openjdk-11-runtime:latest	71	0	5	45	21
redhat-ubi9-openjdk-17-runtime:latest	70	0	5	44	21
ubuntu-jre_17-22.04_edge:latest	0	0	0	0	0
ubuntu-jre_8-22.04_edge:latest	0	0	0	0	0

Table 2: All scanned images and their vulnerability distribution

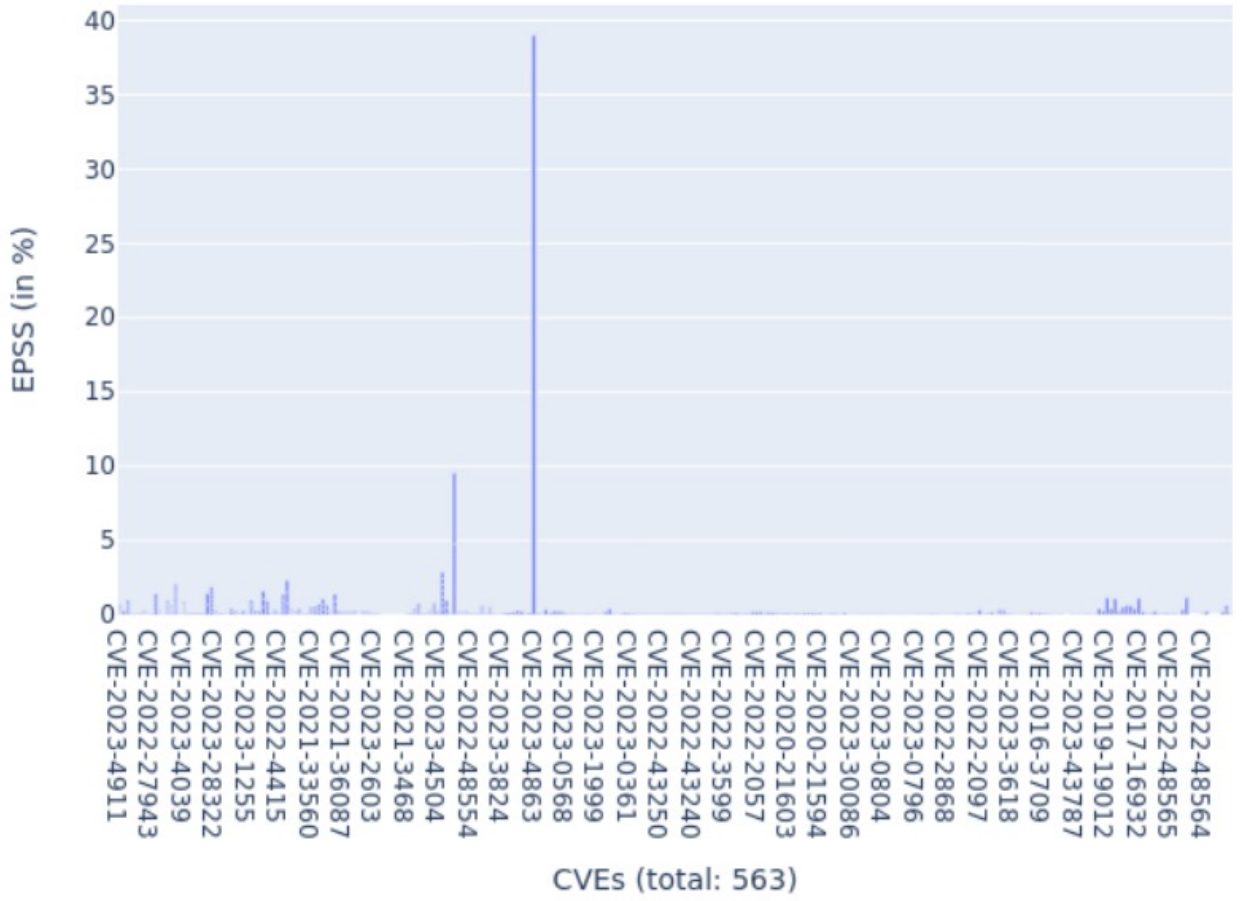
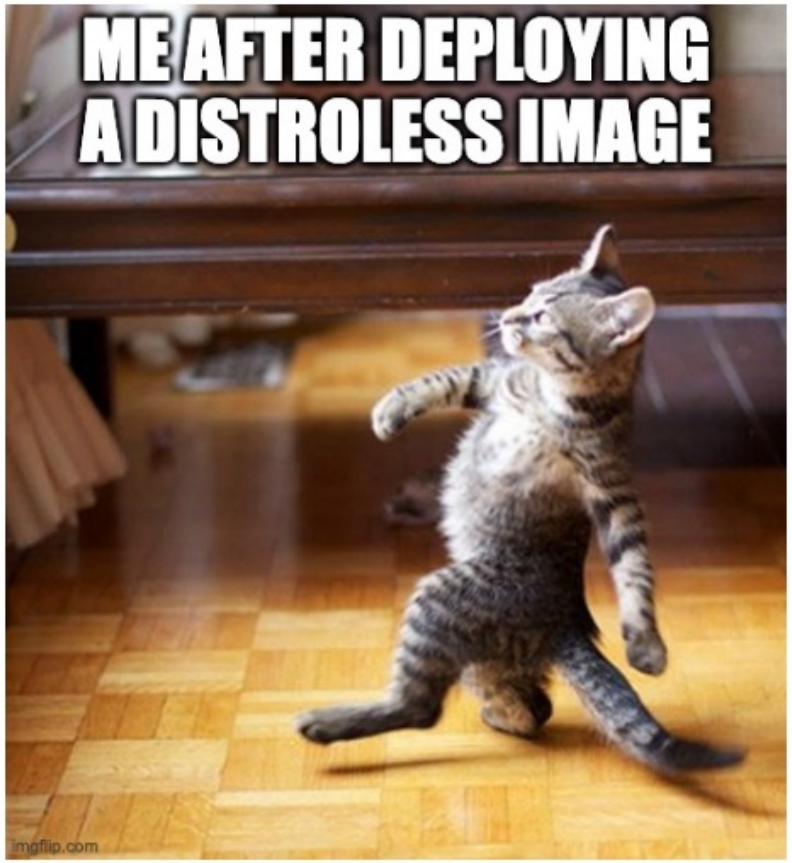


Figure 13: EPSS exploitation probability over found vulnerabilities

ADVANTAGES

- Minimal images containing only runtime environment and the application (no shells, no package managers, etc)
- Reduced attack surface
- Less findings of security scanners
- Removes entire classes of attacks
- Faster transfer times, less storage size, resource efficiency => less costs
- Faster build times



DISADVANTAGES & CHALLENGES

- **Complexity**
  Requires deep understanding of all underlying systems, from user i/o to kernel namespace, docker internals etc

- **Compatibility Issues**
  Some applications may rely on specific features or libraries that are missing in distroless containers

- **Debugging / No shell access**
  If your application needs to execute system commands, Distroless won't work. (Chainguard/Alpine does!)

- **No support for certain runtimes**
  Google Distroless does not support PHP out of the box, but chainguard does.

https://i.imgflip.com/5dq5my.jpg

CONCLUSION

- Teams are responsible for the selection and security assurance of their base images (same as with their source code and open source dependencies)
- Distroless methods make your apps more secure (scientifically proved)
- **Recommendations**
  - Use Google distroless or chainguard images (or Alpine if possible)
  - Scan your images (fail your build!)
  - Do not build your images as root!
  - Create awareness / establish community

Contact

Twitter:
@michael_wager
@c0dmtr1x

www.secure-io.de