

Automated Security Testing of deployed infrastructure

Presenting our work and knowledge from SCS

@ ALASCA Summit 2024

Dominik Pataky, Kurt Garloff


Overview

1. Intro to the topic of pentesting
2. Background and context of the project
 - Infrastructure layer
3. Implementation in SCS
 - Container layer
4. Review and methodology transfer

Pentesting?

- = Penetration testing
- Method of testing for security weaknesses and vulnerabilities in IT
- Experts run offensive tests in specified scope
 - Beware of legal issues in DE (Hackerparagraph §202c StGB) and other countries
- Also part of “red teaming” (attack team)
- Complements the theoretical and design work by looking for issues with the implementation

Pentesting? (2)

- Should be done regularly
 - Own interest to keep everything secure
- Some people are required to test
 - Compliance reasons
 - Mainly by contracting a third party
 - Highly skilled and expensive engineers required
-  Can partially be automated!

Pentesting in SCS

- Scope and lawful compliance can be worked on by many orgs
- Automated pentesting according to SCS Security Standards is equivalent to Compliance Test Suite for Platform Standards
- Continuous integration and CI testing has enabled a new level a quality assurance and development velocity
- Do the same for Security: Secure SDLC, shift left (automated tests in stage of development)

Context for SCS VP09c

- VP09c name of the SCS tender for “penetration testing”. Two steps:
 1. Run pentests against SCS infra and find problems to be fixed
 2. Automate these tests as much as possible
- Automation allows re-running and replicability
 - Important, if tests shall run at multiple CSP sites

Preliminary work: pentesting IaaS

- Pentesting experts installed IaaS layer testbed instances
- Used environment for real pentest
 - Results substitutional for SCS instances at CSP sites
 - Findings were reported to upstream and fixed
- Final report available to SCS for further reference

SCS Best Practices in Security (1)

- SCS also empowers open discussions around security topics
 - Also driven by VP09c
- Leads to SCS Best Practices, standards, guides
- Ecosystem exchange with patches and docs

SCS Best Practices in Security (2)

- Reporting potential vulnerabilities
 - Use Security Advisories in Security Tab in github SCS issues repo -
 - > allows for restricting audience prior to publication
 - Prevent confusion by false positives
 - Avoid helping black hat hackers from using insight to hack our CSPs
 - Separate from our security contact mailing list reporting process
 - Separate from public advisories (SCS Blog)

Implementation in SCS

- The testbed pentest provided the ground work for implementation of security tooling
- Pentesters gained experience, insight on where to look
 - SCS = stack of open source components
 - Each component has its specific attack surface

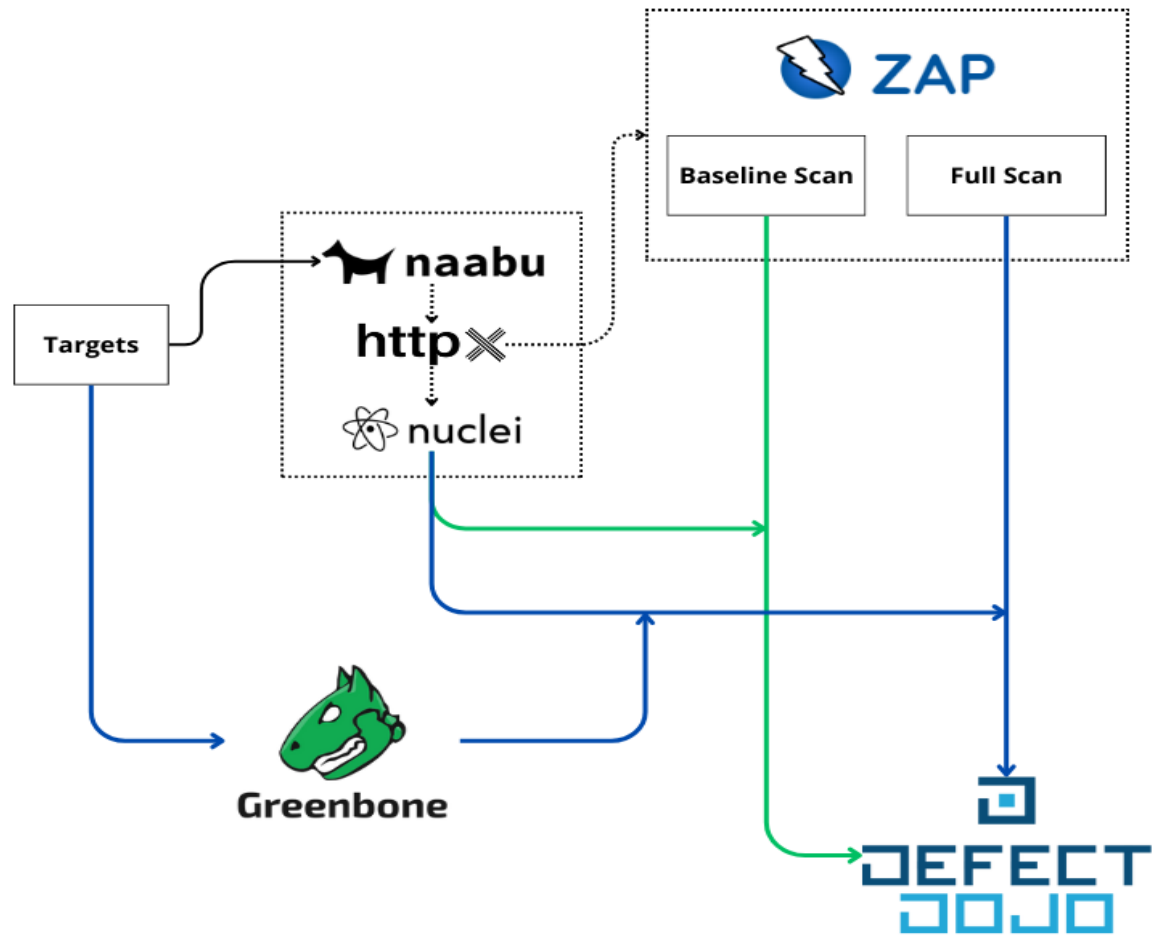
Implementation in SCS: IaaS

- For IaaS, deployed infra is scanned regularly
- Build server pipeline as reference implementation
- Docs @ <https://docs.scs.community/docs/category/pentesting-iaas>

(<https://docs.scs.community/docs/category/pentesting-iaas>)

Implementation in SCS: IaaS tools

- Pipeline: Zuul
- Basic scanning: Naabu, httpx, Nuclei
- Vulnerability scan: ZAP, OpenVAS
- Report management: DefectDojo



- 🕒
- ☰
- 📄
- 👤
- 🏠
- 📅
- 👤
- 📄
- 📊
- 👤
- 📅
- 🏠
- ⚙️
- 🚫

7
Active Engagements

[View Engagement Details](#)

518
Last Seven Days

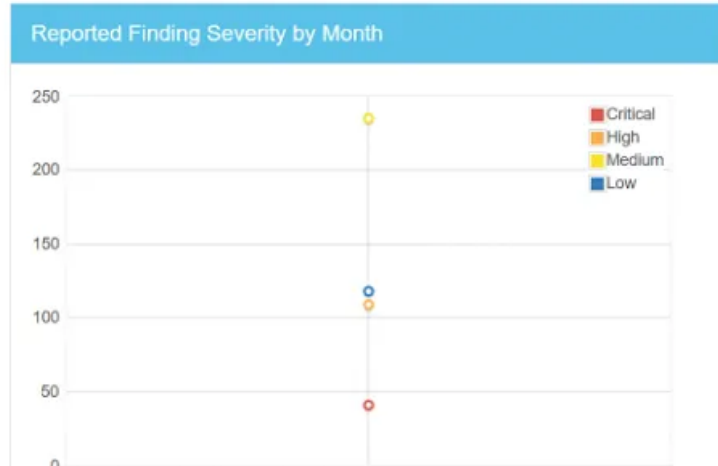
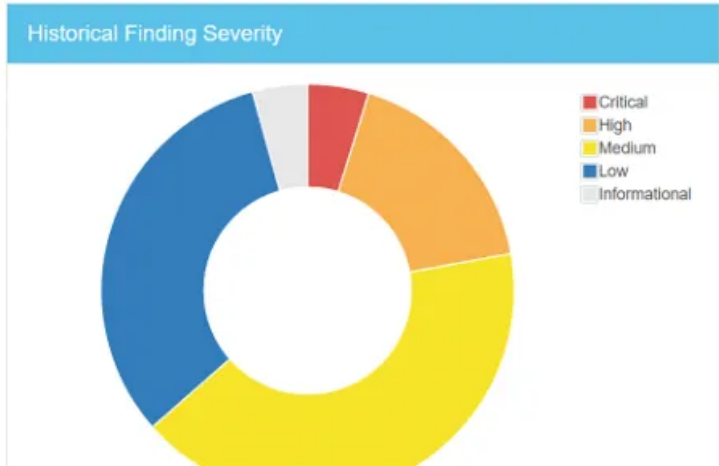
[View Finding Details](#)

0
Closed In Last Seven Days

[View Finding Details](#)

0
Risk Accepted In Last Seven Days

[View Finding Details](#)



Implementation in SCS: KaaS

- Container layer, Kubernetes
- Two modes:
 - ad-hoc from Zuul (unauthenticated, “black box testing”)
 - continuously as Operator (authenticated test from inside the cluster)
- Docs @ <https://docs.scs.community/docs/category/pentesting-kaas>

(<https://docs.scs.community/docs/category/pentesting-kaas>)

Implementation in SCS: KaaS tools

- Focus on Trivy as scanning tool
- Widely accepted security utility in Kubernetes environments
 - k8s-native and tailored to common weaknesses in Kubernetes
 - Surprise: No native export to DD or S3
- Export to DefectDojo ◦ Self-built export cronjob

Review of methodology

- Chosen tools fit SCS components
- BUT: reference implementation! Tools are interchangeable
- More important than tool choice is a good methodology
- Creating the pipeline of tools proved very helpful
 - especially in context of automation (infra-as-code, IaC)

Outlook and adaptability

- Methodology can be adapted to ALASCA projects
 - Trivy for all things k8s (creates a set of reports)
- YAOOK ◦ IaaS scanner instances spawned automatically
- YAKE?

Conclusion and questions

- Summary
- Questions from audience?