OPEN INFRA FORUM 24 NOVEMBER 2022

Right and left of security

How to handle security in applications with DefectDojo and open source scanners.

Digitalist

Speaker: Mikke Schirén

Started to code with Commodore 64, copying code from DatorMagazin.
(almost) 12 years @ Digitalst
(former Wunderkraut, former NodeOne).
Worked my way from the frontend to the darker parts.
Senior PHP/Sysadmin/K8s app developer/admin und so weiter.
Tech Lead @ Digitalist Net Services
Responsible for Matomo SaaS

Digitalist

DIgitalist

We work with open tech.

Digitalist

Security - where to look?

From right to left and back again.

Digitalist

Where should we have secuirty?

EVERYWHERE!

How should we keep track of it?

Left - Right, simplified.

Left:
Finding the security issues in the code of the application.

Right:
Protect from security issues hosting the application

DIGITALIST GROUP

But first, let's first talk about some of the problems with DevOPS.

Digitalist

The dream: A developer should be able to deploy to production anytime.

The CI/CD workflow needs a lot of things, let's mention some:

- Building the assets
- Testing of the assets
- Testing the final application
- Security testing of:
  - Code
  - Containers
  - Packages
  - Infrastructure
  - Network
- Stop delivery if important tests or security checks fail.

But, Hey! We are running a node.js application! And the security issues has nothing to do with our delivery!

Let’s trust the developers on this. So, we need an process. And tools for that process. So they can push that button to deliver the code to production.

And also let's think about this - according to a marketing email I got yesterday (*if I trust the developers, I must trust the marketing email, right?*):

85% of commercial applications has known critical vulnerabilities.

So what about security scanning in the left side of the delivery? And how to handle it if all is alright at the day of the delivery, and the day after, it’s not?

MACRO
CONSESUS
?
B Rich
HEDGEYE

So, the developers needs a process, and tools.

Ok, then we also have the old fashioned sysadmins and people in charge of the hosting...
We have containers, servers, VPS, Kubernetes, SaaS, PaaS...

But, hey! Those security issues in X has nothing to do with hosting the application!

So let’s trust the sysadmins.

So… we need processes to handle both left and right, and make sure we know about the security issues. And keep track. Because security itself, as you know, is a process. Suddenly one day we have another Log4j incident, and we need to know we have it and where.

There is a lot of open source tools around security. Some of my favorites:

OWASP Zed Attack Proxy (ZAP)
Nikto
Trivy
Semgrep
Yarn audit
Etc.

Ways to run security scanners:
Terminal
Gitlab runners
Jenkins jobs
Github Actions
Fancy UI:S

So we have runned all the things and have random reports in random formats, with different stakeholders, how could we work with them?

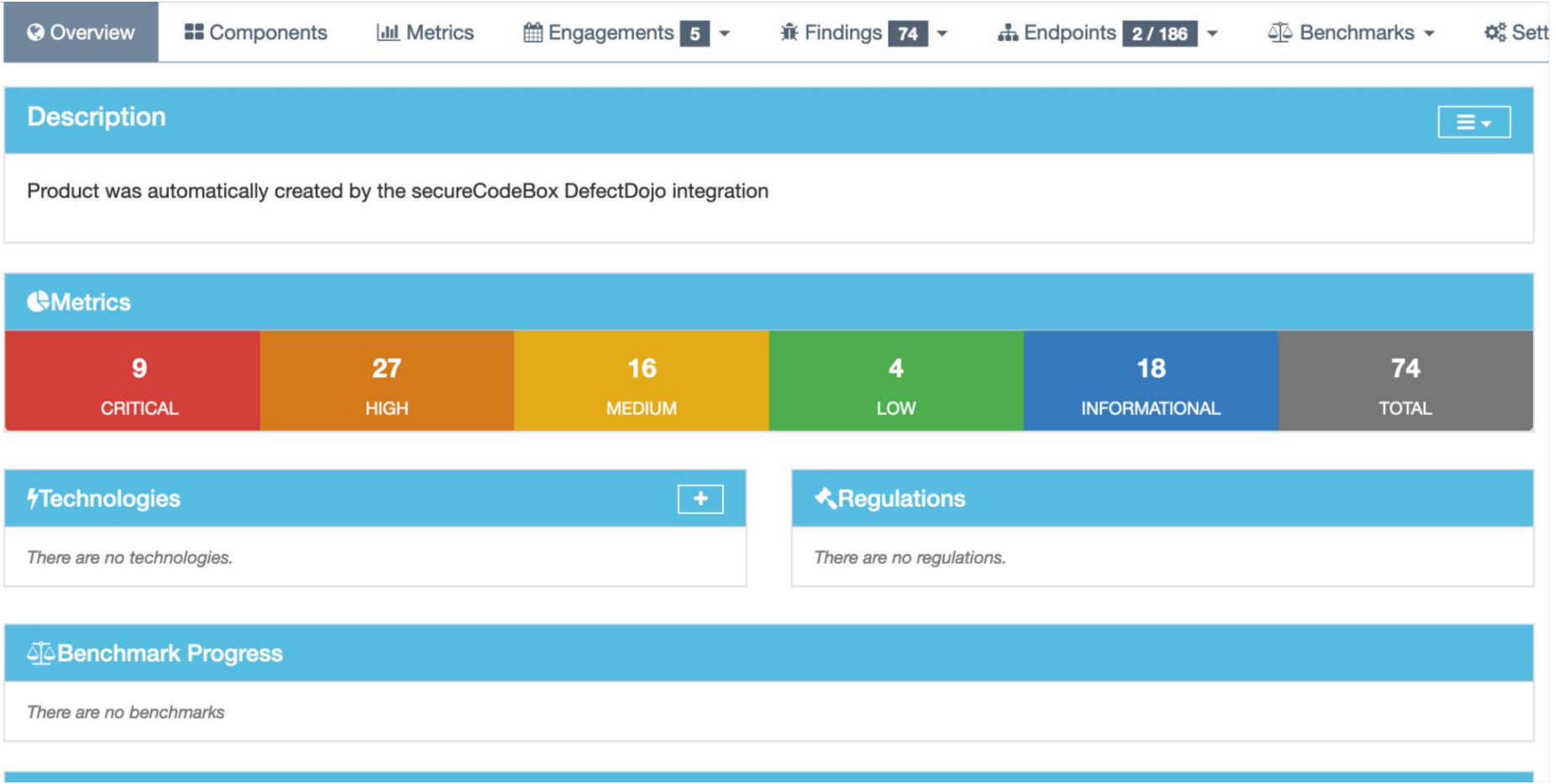

DEFECTDOJO

What is DefectDojo?

DefectDojo is a security tool that automates application security vulnerability management. DefectDojo streamlines the application security testing process by offering features such as importing third party security findings, merging and de-duping, integration with Jira, templating, report generation and security metrics.

What does DefectDojo do?

While traceability and metrics are the ultimate end goal, DefectDojo is a bug tracker at its core. Taking advantage of DefectDojo's Product:Engagement model, enables traceability among multiple projects and test cycles, and allows for fine-grained reporting.


Overview
Components
Metrics
Engagements 5
Findings 74
Endpoints 2 / 186
Benchmarks
Sett
Description
Product was automatically created by the secureCodeBox DefectDojo integration
Metrics
9
CRITICAL
27
HIGH
16
MEDIUM
4
LOW
18
INFORMATIONAL
74
TOTAL
Technologies
There are no technologies.
Regulations
There are no regulations.
Benchmark Progress
There are no benchmarks

Some of the supported scanners in DefectDojo

- **Acunetix Scan** - XML format
- **Acunetix360 Scan** - Acunetix360 JSON format.
- **Anchore Engine Scan** - Anchore-CLI JSON vulnerability report format.
- **Anchore Enterprise Policy Check** - Anchore-CLI JSON policy check report format.
- **Anchore Grype** - A vulnerability scanner for container images and filesystems. JSON report generated with '-o json' format
- **AppSpider Scan** - AppSpider (Rapid7) - Use the VulnerabilitiesSummary.xml file found in the zipped report download.
- **Aqua Scan** -
- **Arachni Scan** - Arachni JSON report format (generated with `arachni_reporter --reporter 'json'`).
- **AuditJS Scan** - AuditJS Scanning tool using SonaType OSSIndex database with JSON output format
- **AWS Prowler Scan** - Export of AWS Prowler in CSV or JSON format.
- **AWS Scout2 Scan** - JS file in scout2-report/inc-awsconfig/aws_config.js.
- **AWS Security Hub Scan** - AWS Security Hub exports in JSON format.
- **Azure Security Center Recommendations Scan** - Import of Microsoft Defender for Cloud (formerly known as Azure Security Center) recommendations in CSV format.
- **Bandit Scan** - JSON report format
- **BlackDuck API** - BlackDuck findings can be directly imported using the Synopsys BlackDuck API. An API Scan Configuration has to be setup in the Product.
- **Blackduck Component Risk** - Upload the zip file containing the security.csv and files.csv.
- **Blackduck Hub Scan** - Upload the zip file containing the security.csv and components.csv for Security and License risks.
- **Brakeman Scan** - Import Brakeman Scanner findings in JSON format.
- **BugCrowd Scan** - BugCrowd CSV report format
- **Bundler-Audit Scan** - 'bundler-audit check' output (in plain text)
- **Burp Enterprise Scan** - Import Burp Enterprise Edition findings in HTML format
- **Burp GraphQL API** - Import Burp Enterprise Edition findings from the GraphQL API
- **Burp REST API** - Import Burp REST API scan data in JSON format (/scan/[task_id] endpoint).
- **Burp Scan** - When the Burp report is generated, the recommended option is Base64 encoding both the request and response fields. These fields will be processed and made available in the 'Finding View' page.
- **CargoAudit Scan** - Import JSON output for cargo audit scan report.
- **Checkmarx OSA** - Checkmarx Open Source Analysis for dependencies (json). Generate with `jq -s . CxOSAVulnerabilities.json CxOSALibraries.json`
- **Checkmarx Scan** - Simple Report. Aggregates vulnerabilities per categories, cwe, name, sinkFilename
- **Checkmarx Scan detailed** - Detailed Report. Import all vulnerabilities from checkmarx without aggregation
- **Checkov Scan** - Import JSON reports of Infrastructure as Code vulnerabilities.
- **Clair Klar Scan** - Import JSON reports of Docker image vulnerabilities from clair klar client.

Useful links

https://www.defectdojo.org/
https://github.com/aquasecurity/trivy-operator
https://www.securecodebox.io/

Demo time!


DIGITALIST GROUP
And yes, we are hiring.
Digitalist

Open positions right now:

Kubernetes/DevOPS
Developers - frontend and backend (React, PHP)
Web strategist

See our web page: digitalist.se/karriar

Thank you for listening!

CONTACT

Mikke Schirén

Tech Lead, Digitalist Net Services

mikke.schiren@digitalist.se

Digitalist