Decoding Software Composition Analysis (SCA): Unveiling Pain Points in SCA

Kaiwe Jiang

18/04/2024

Kaiwen Jiang

- Application Security Engineer at Wise
- Cyber Security Consulta
- Msc Information Securi
  University of London
- Blog: https://appseckiki
- Cat lover & Swiftie

Agenda

1. Introduction of Software Composition Analysis(SCA)
2. Tools Showcase
3. Pain Points Discussion
4. Summary

Introduction of Software Composition Analysis(SCA)

01

Supply Chain Attack: Targeting Industrial Control Systems via Third-Party Software

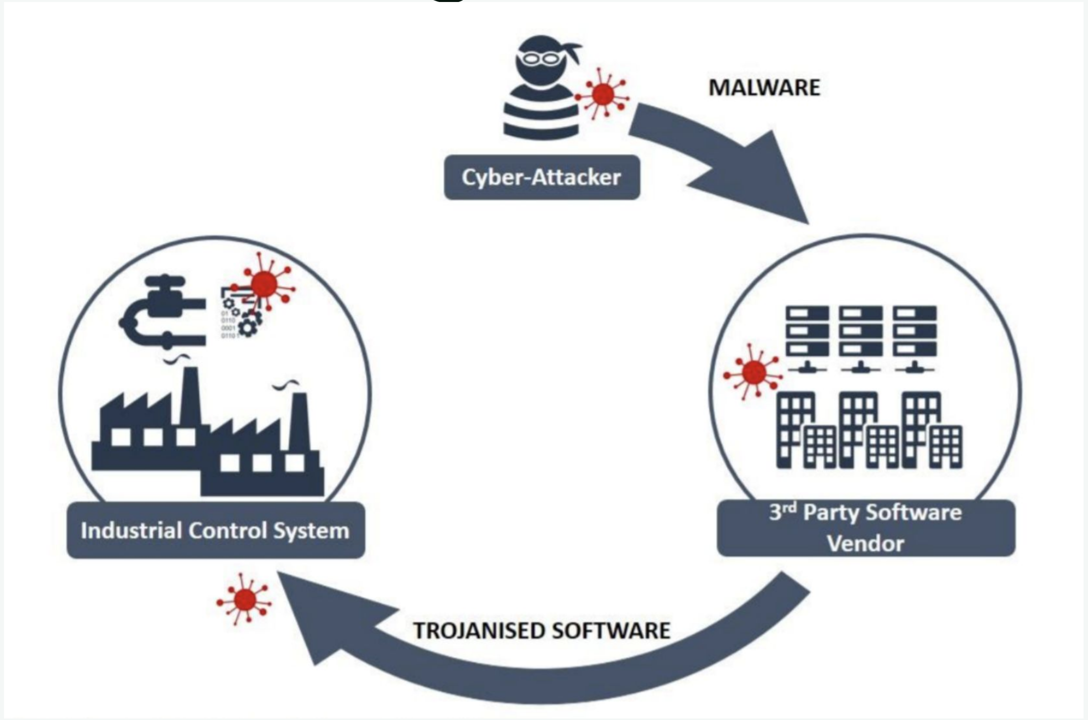


Image credit: https://www.ncsc.gov.uk/collection/supply-chain-security/third-party-software-providers

Supply Chain Attack: Targeting Industrial Control Systems via Third-Party Software

Impact:

- Disruptions: Production delays, equipment failures, and safety hazards can occur due to manipulated control systems.
- Data theft: Sensitive information about industrial processes or system configurations could be stolen for espionage or sabotage purposes.
- Financial losses: The attack can lead to costly repairs, downtime, and potential regulatory fines.

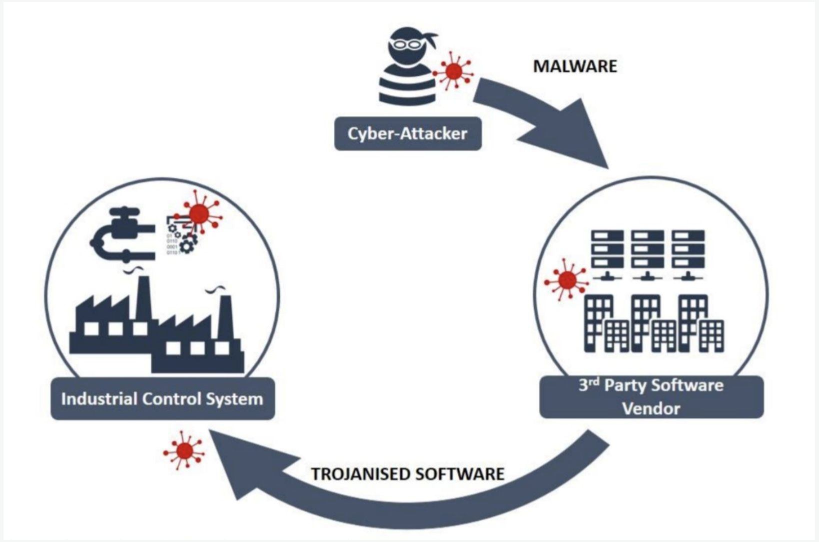
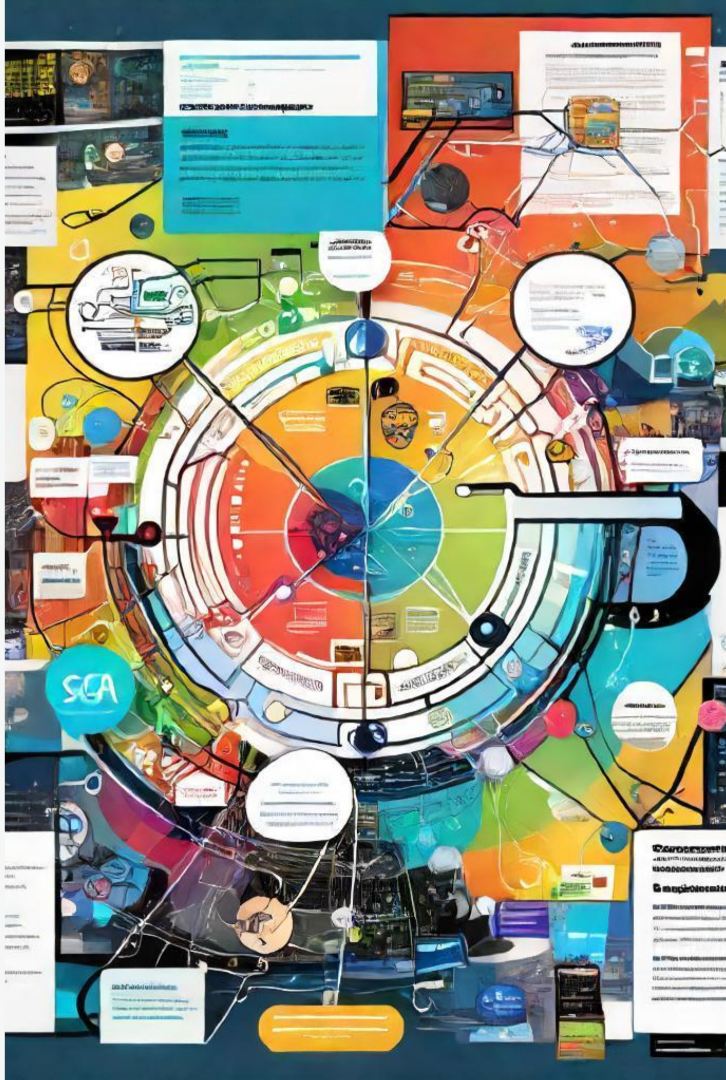


Image credit: https://www.ncsc.gov.uk/collection/supply-chain-security/third-party-software-providers

Software Composition Analysis (SCA)

SCA analyses open-source software (OSS) for two main reasons: security and compliance.

- Security Risks: Most software today relies on pre-written open-source components. SCA helps identify those components and any known security vulnerabilities they may have. This allows developers to fix these weaknesses before they become a target for attackers.
- Compliance: Open-source software comes with various licenses that dictate how you can use it. SCA helps ensure your project follows the terms of these licenses, avoiding potential legal issues in the future.

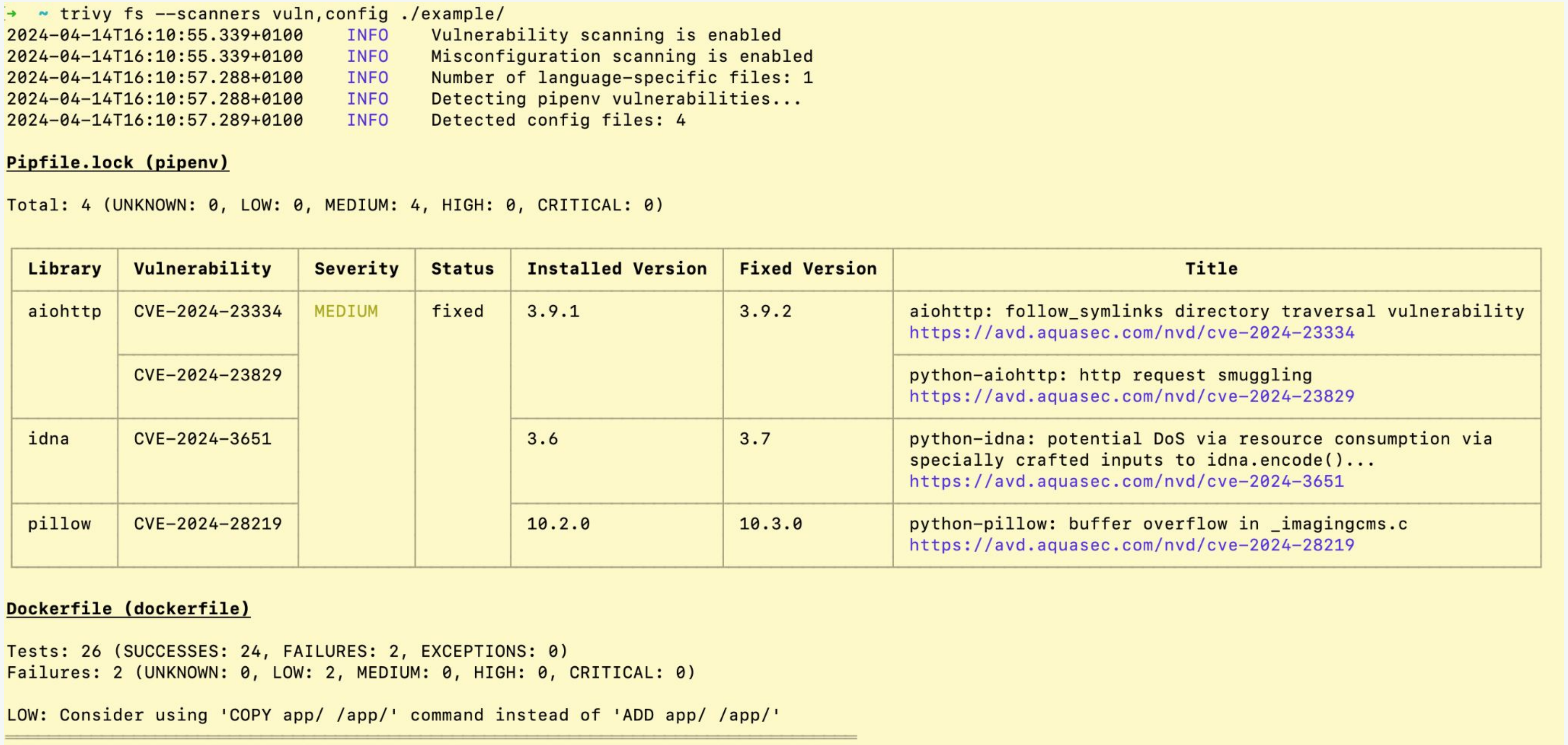


Tools Showcase

02

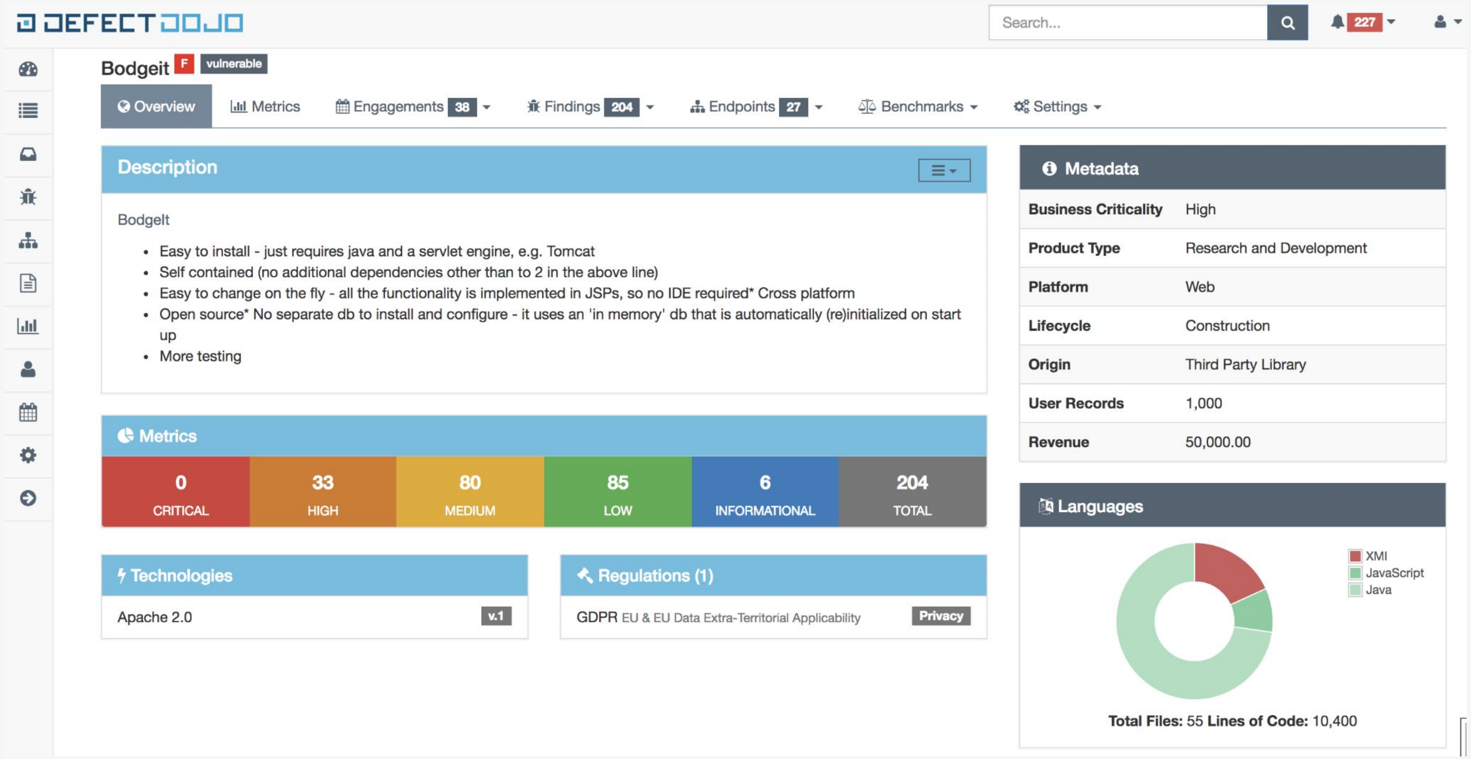
Tool Showcase: Trivy

- Targets (what Trivy can scan):
  - Container Image
  - Filesystem
  - Git Repository (remote)
  - Virtual Machine Image
  - Kubernetes
  - AWS
- Scanners (what Trivy can find there): OS packages and software dependencies in use (SBOM) Known vulnerabilities (CVEs) IaC issues and misconfigurations Sensitive information and secrets Software licenses



```
➜  ~ trivy fs --scanners vuln,config ./example/
2024-04-14T16:10:55.339+0100    INFO    Vulnerability scanning is enabled
2024-04-14T16:10:55.339+0100    INFO    Misconfiguration scanning is enabled
2024-04-14T16:10:57.288+0100    INFO    Number of language-specific files: 1
2024-04-14T16:10:57.288+0100    INFO    Detecting pipenv vulnerabilities...
2024-04-14T16:10:57.289+0100    INFO    Detected config files: 4

Pipfile.lock (pipenv)

Total: 4 (UNKNOWN: 0, LOW: 0, MEDIUM: 4, HIGH: 0, CRITICAL: 0)

┌──────────┬────────────────┬──────────┬────────┬───────────────────┬───────────────┬──────────────────────────────────────────────────────────────┐
│ Library  │ Vulnerability  │ Severity │ Status │ Installed Version │ Fixed Version │                            Title                             │
├──────────┼────────────────┼──────────┼────────┼───────────────────┼───────────────┼──────────────────────────────────────────────────────────────┤
│ aiohttp  │ CVE-2024-23334 │ MEDIUM   │ fixed  │ 3.9.1             │ 3.9.2         │ aiohttp: follow_symlinks directory traversal vulnerability   │
│          │                │          │        │                   │               │ https://avd.aquasec.com/nvd/cve-2024-23334                   │
│          ├────────────────┤          │        │                   │               ├──────────────────────────────────────────────────────────────┤
│          │ CVE-2024-23829 │          │        │                   │               │ python-aiohttp: http request smuggling                       │
│          │                │          │        │                   │               │ https://avd.aquasec.com/nvd/cve-2024-23829                   │
├──────────┼────────────────┤          │        ├───────────────────┼───────────────┼──────────────────────────────────────────────────────────────┤
│ idna     │ CVE-2024-3651  │          │        │ 3.6               │ 3.7           │ python-idna: potential DoS via resource consumption via      │
│          │                │          │        │                   │               │ specially crafted inputs to idna.encode()...                 │
│          │                │          │        │                   │               │ https://avd.aquasec.com/nvd/cve-2024-3651                    │
├──────────┼────────────────┤          │        ├───────────────────┼───────────────┼──────────────────────────────────────────────────────────────┤
│ pillow   │ CVE-2024-28219 │          │        │ 10.2.0            │ 10.3.0        │ python-pillow: buffer overflow in _imagingcms.c              │
│          │                │          │        │                   │               │ https://avd.aquasec.com/nvd/cve-2024-28219                   │
└──────────┴────────────────┴──────────┴────────┴───────────────────┴───────────────┴──────────────────────────────────────────────────────────────┘

Dockerfile (dockerfile)

Tests: 26 (SUCCESSES: 24, FAILURES: 2, EXCEPTIONS: 0)
Failures: 2 (UNKNOWN: 0, LOW: 2, MEDIUM: 0, HIGH: 0, CRITICAL: 0)

LOW: Consider using 'COPY app/ /app/' command instead of 'ADD app/ /app/'
════════════════════════════════════════════════════════════════════════
```

DefectDojo: An OWASP Project for Vulnerability Monitoring


Bodgeit F vulnerable

Overview | Metrics | Engagements 38 | Findings 204 | Endpoints 27 | Benchmarks | Settings

Description

BodgeIt

- Easy to install - just requires java and a servlet engine, e.g. Tomcat
- Self contained (no additional dependencies other than to 2 in the above line)
- Easy to change on the fly - all the functionality is implemented in JSPs, so no IDE required* Cross platform
- Open source* No separate db to install and configure - it uses an 'in memory' db that is automatically (re)initialized on start up
- More testing

Metrics

0	33	80	85	6	204
CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL	TOTAL

Technologies

Apache 2.0 v.1

Regulations (1)

GDPR EU & EU Data Extra-Territorial Applicability Privacy

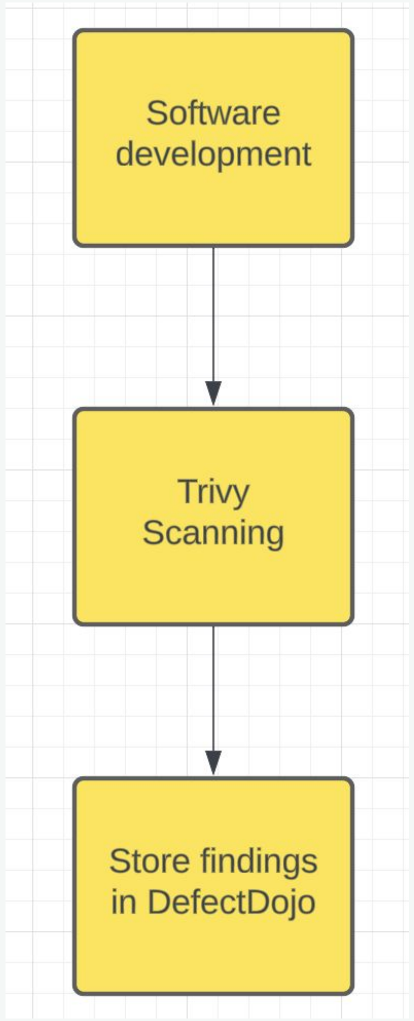
Metadata

Business Criticality	High
Product Type	Research and Development
Platform	Web
Lifecycle	Construction
Origin	Third Party Library
User Records	1,000
Revenue	50,000.00

Languages

🟥 XMI
🟩 JavaScript
🟩 Java

Total Files: 55 **Lines of Code:** 10,400



Visualize the SCA workflow

- Visualize the workflow with a chart: Initiate project -> Development (introduce dependencies) -> SCA (Trivy scan) -> Integrate findings with DefectDojo.
- Benefits: Proactive identification and mitigation of security risks early in the development lifecycle.

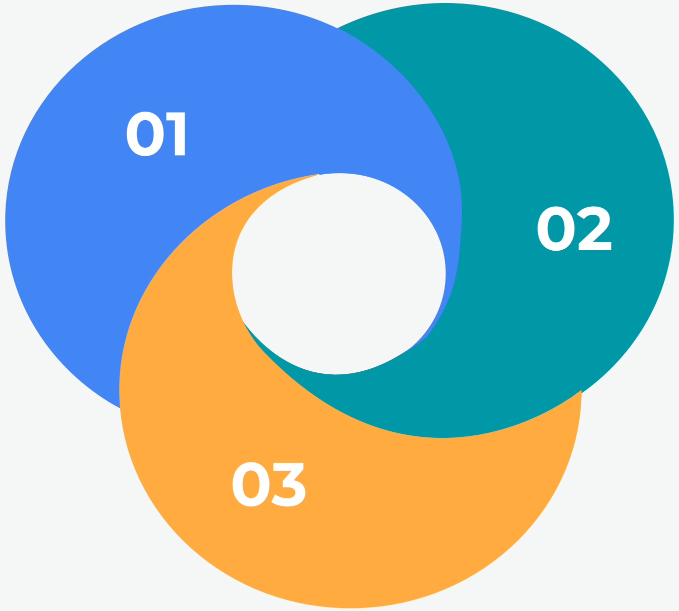
Pain Points Discussion

03

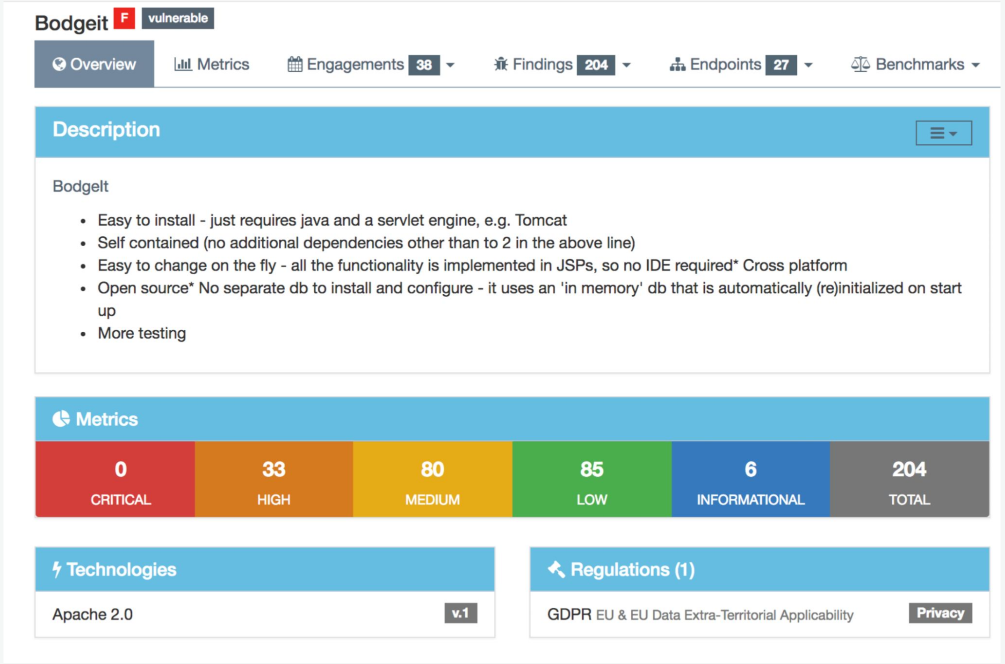
Pain points discussion

SCA findings get overlooked

SLA breach



Streamline dependency upgrading


Bodgeit F vulnerable

Description

BodgeIt

- Easy to install - just requires java and a servlet engine, e.g. Tomcat
- Self contained (no additional dependencies other than to 2 in the above line)
- Easy to change on the fly - all the functionality is implemented in JSPs, so no IDE required* Cross platform
- Open source* No separate db to install and configure - it uses an 'in memory' db that is automatically (re)initialized on start up
- More testing

Metrics

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL	TOTAL
0	33	80	85	6	204

Technologies

Apache 2.0 v.1

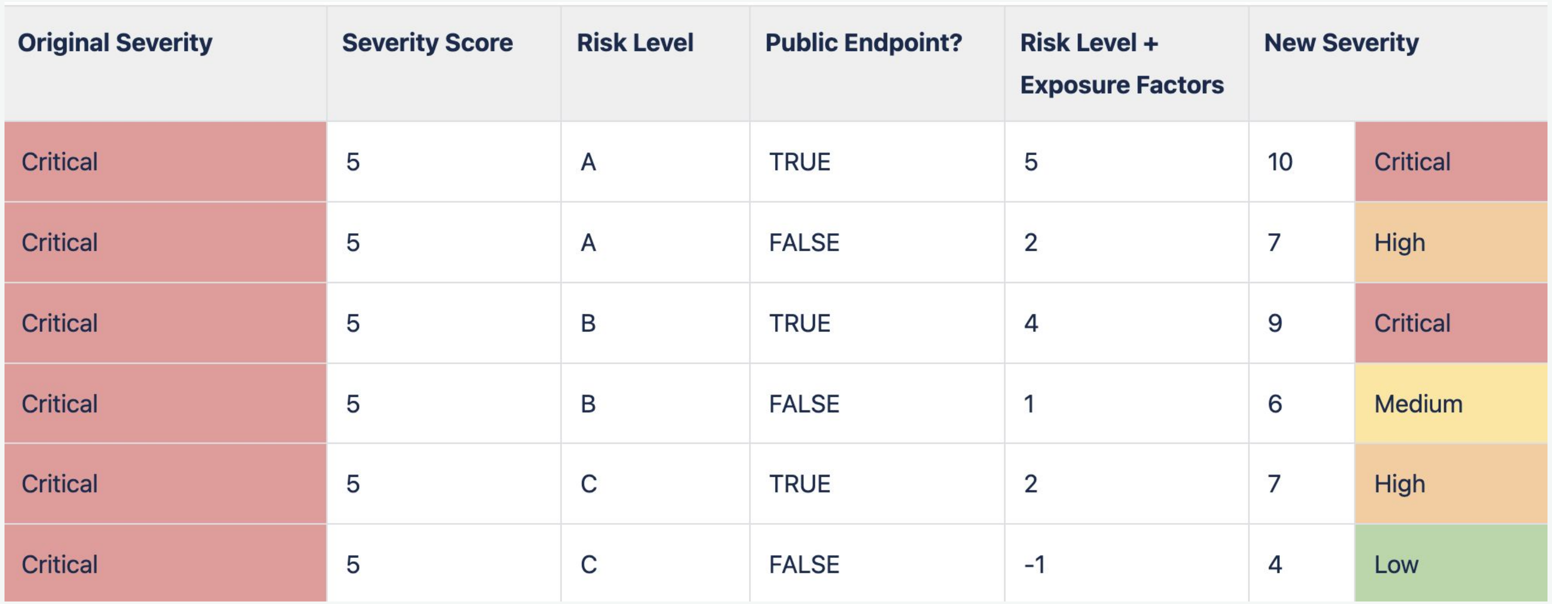
Regulations (1)

GDPR EU & EU Data Extra-Territorial Applicability Privacy

1. SCA findings get overlooked

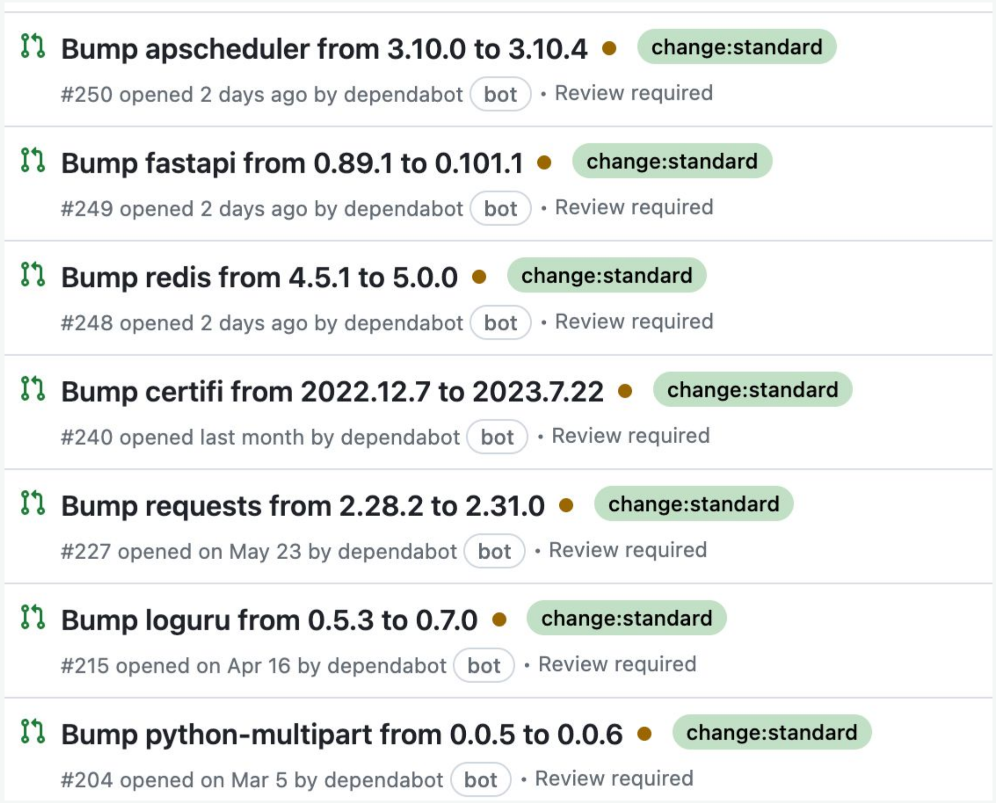
- Service Level Agreement (SLA) implementation
- Vulnerability Reminder Slack bot: The bot sends weekly report with all of the high/critical vulnerabilities that'll reach their SLA soon.
- Training and awareness: offering training to the engineering teams on the importance of SCA findings

2. SLA Breach



Original Severity	Severity Score	Risk Level	Public Endpoint?	Risk Level + Exposure Factors	New Severity	
Critical	5	A	TRUE	5	10	Critical
Critical	5	A	FALSE	2	7	High
Critical	5	B	TRUE	4	9	Critical
Critical	5	B	FALSE	1	6	Medium
Critical	5	C	TRUE	2	7	High
Critical	5	C	FALSE	-1	4	Low

- Too many vulnerabilities to fix in a short time? SLA severity adjustment
    - Public facing vs Internal
    - Service level: High priority, standard, or low
    - Exploitable or not
- Implementing warning mode and blocking PRs



Bump apscheduler from 3.10.0 to 3.10.4 ● change:standard
#250 opened 2 days ago by dependabot bot • Review required

Bump fastapi from 0.89.1 to 0.101.1 ● change:standard
#249 opened 2 days ago by dependabot bot • Review required

Bump redis from 4.5.1 to 5.0.0 ● change:standard
#248 opened 2 days ago by dependabot bot • Review required

Bump certifi from 2022.12.7 to 2023.7.22 ● change:standard
#240 opened last month by dependabot bot • Review required

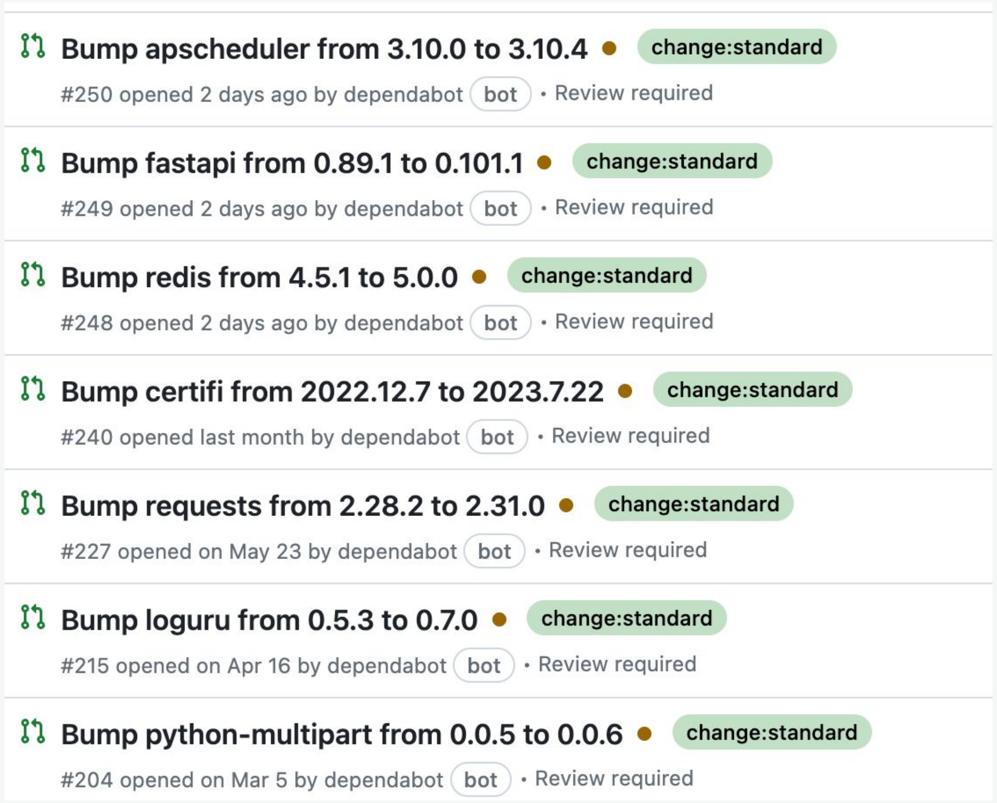
Bump requests from 2.28.2 to 2.31.0 ● change:standard
#227 opened on May 23 by dependabot bot • Review required

Bump loguru from 0.5.3 to 0.7.0 ● change:standard
#215 opened on Apr 16 by dependabot bot • Review required

Bump python-multipart from 0.0.5 to 0.0.6 ● change:standard
#204 opened on Mar 5 by dependabot bot • Review required

3. Streamline dependency upgrading

- Streamlining the process for addressing vulnerabilities, for example with automated remediation tools such as Dependabot.
    - Dependency PR overload and don’t get to merge
    - Group dependency PRs by upgrade level



Bump apscheduler from 3.10.0 to 3.10.4 ● change:standard
#250 opened 2 days ago by dependabot bot • Review required

Bump fastapi from 0.89.1 to 0.101.1 ● change:standard
#249 opened 2 days ago by dependabot bot • Review required

Bump redis from 4.5.1 to 5.0.0 ● change:standard
#248 opened 2 days ago by dependabot bot • Review required

Bump certifi from 2022.12.7 to 2023.7.22 ● change:standard
#240 opened last month by dependabot bot • Review required

Bump requests from 2.28.2 to 2.31.0 ● change:standard
#227 opened on May 23 by dependabot bot • Review required

Bump loguru from 0.5.3 to 0.7.0 ● change:standard
#215 opened on Apr 16 by dependabot bot • Review required

Bump python-multipart from 0.0.5 to 0.0.6 ● change:standard
#204 opened on Mar 5 by dependabot bot • Review required

3. Streamline dependency upgrading

- Improving testing to improve confidence level of dependency upgrading
  - Implementing unit testing/ integration testing/end to end testing in CI
  - Improving higher test coverage
- utilisation of automated deployment to staging environment

3. Streamline dependency upgrading

- Should we use dependency bots to auto merge pull request (PR)?

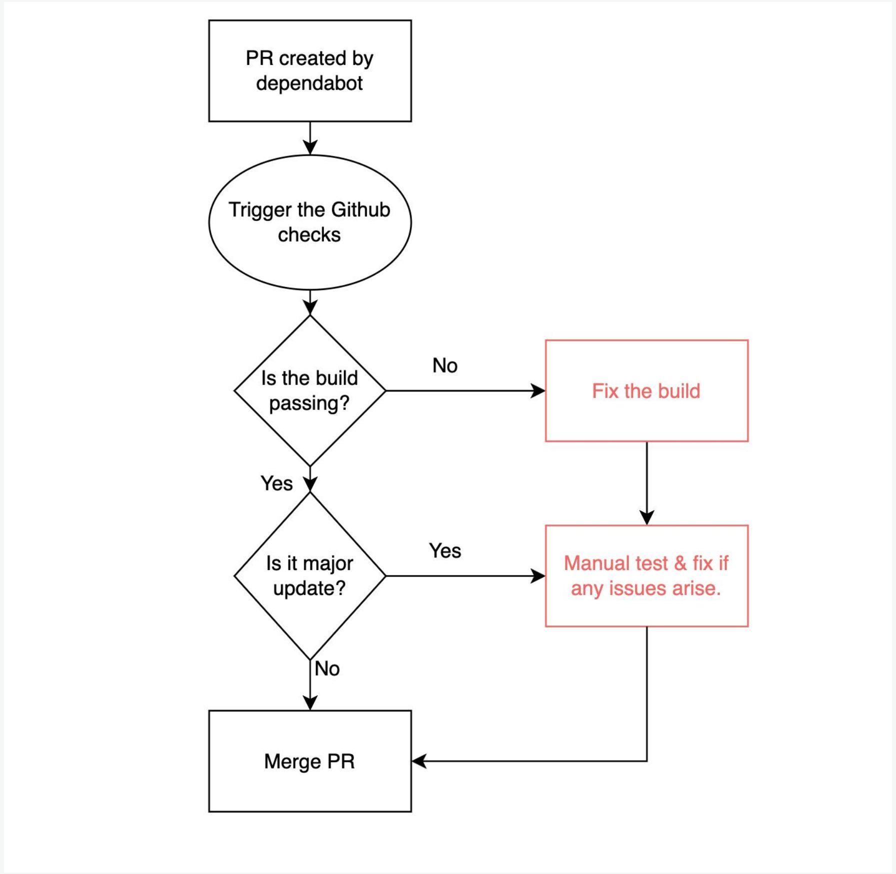
Pros of Auto-Merging with Dependency Bots

- Efficiency: Saves developers time by automating the merging of routine dependency updates, freeing them to focus on more complex tasks.
- Consistency: Ensures all projects consistently receive the latest updates, reducing the risk of outdated dependencies with known vulnerabilities.
- Reduced Errors: Eliminates the possibility of human error during manual merging.

Cons of Auto-Merging with Dependency Bots

- Security Risks: Auto-merging introduces the risk of introducing new vulnerabilities or compatibility issues with the latest updates. Malicious actors could potentially exploit this by publishing compromised versions of dependencies.

- Lack of Review: Bypasses the opportunity for human review of the changes, which might catch potential issues beyond simple version updates. Important code changes within the dependency update could be missed.



Finding the right balance

- Testing strategy: a robust testing with high coverage to make teams more comfortable with auto-merging for minor, well-tested upgrades.
- Security Posture: Organisations with stricter security requirements might be more cautious about auto-merging.
- Dependency Criticality: Auto-merging might be less risky for non critical dependencies compared to core components.

Summary

04

Summary

- SCA is essential for secure coding: Identifies vulnerabilities in open-source components and dependencies.
- Tools like Trivy simplify vulnerability scanning and provide actionable insights.
- Vulnerability management platforms like DefectDojo centralise data and streamline remediation.
- Clear communication and processes are crucial to ensure timely action on SCA findings. SLAs and automated workflows can improve efficiency but require careful planning for security.
- Balance is key: Prioritise security without hindering development speed excessively.

Call to Action

- Integrate SCA tools and vulnerability management into your development process.
- Foster a culture of security awareness and collaboration between developers and security teams.
- Continuously improve your secure coding practices for a more secure development lifecycle.

Thank you!

Questions?

kaiwen.jiang@wise.com

Kaiwen Jiang