Managing Vulnerabilities in Open Source Dependencies

Eva Sarafianou

FOSDEM
1 & 2 February, Brussels 2025

Self Intro

- Product Security Engineering Lead @ Mattermost
- Previously, Principal Product Security Engineer at Auth0 & Okta
- Based in Athens, Greece

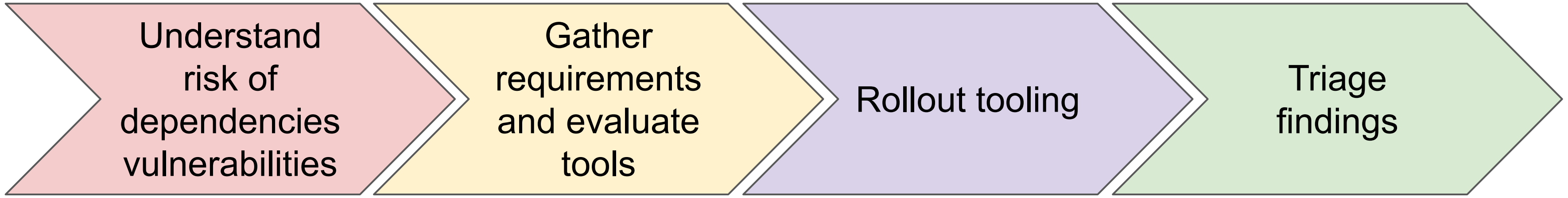
More info: https://evasar.io

Agenda

- Why 3rd Party Dependencies Vulnerabilities Matter
- Evaluating SCA Tools
- SCA Tool Rollout
- Triaging Findings

**SCA: Software Composition Analysis*

Dependency Management Journey


Understand risk of dependencies vulnerabilities
Gather requirements and evaluate tools
Rollout tooling
Triage findings

Why 3rd Party Dependencies Vulnerabilities Matter

- Software composition: In-house code 50% + third-party dependencies 50%
- Increased Attack Surface
- Lack of Control

Dependency Management Goals

Proactivity

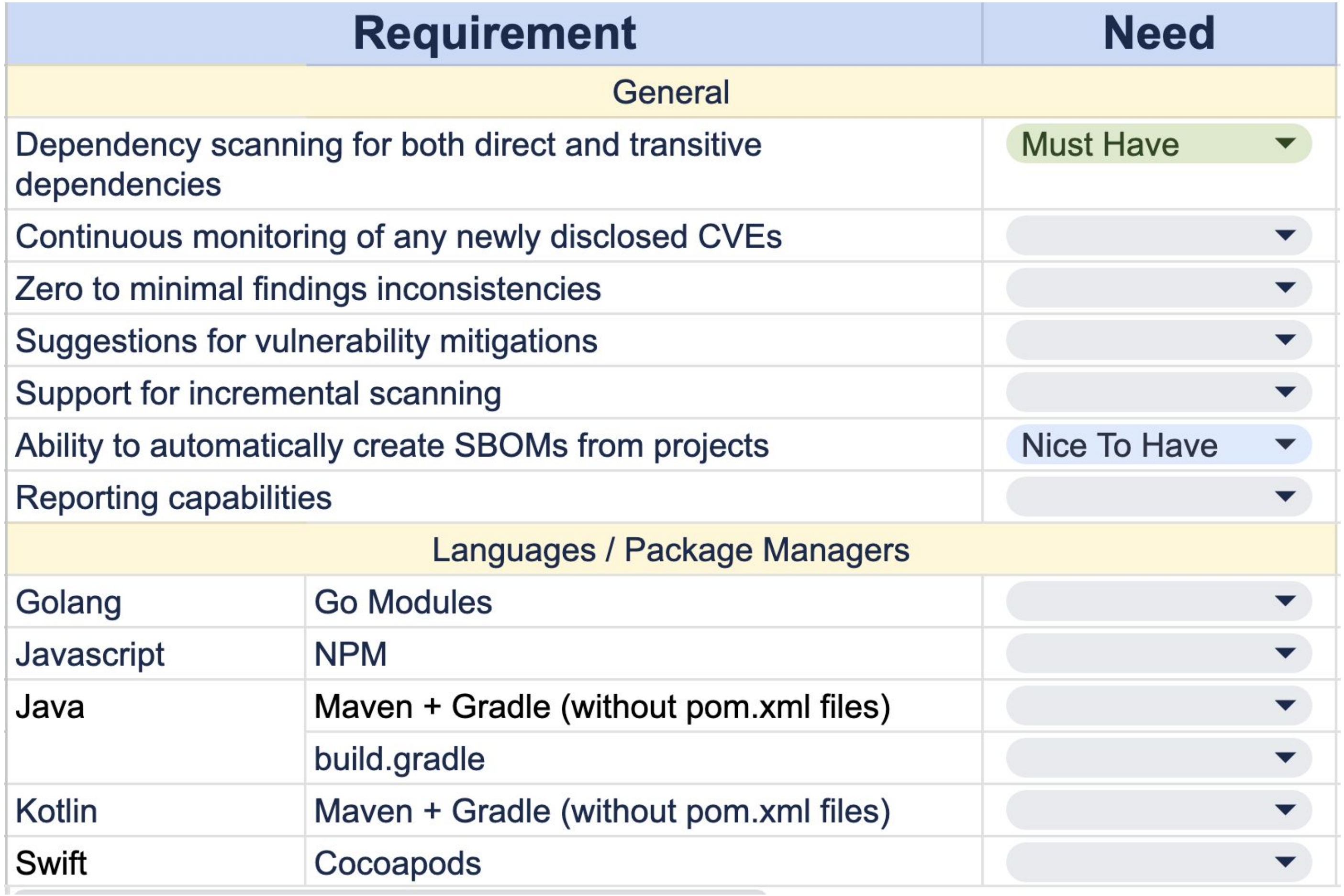
No more vulnerable dependencies added to the project/product

Reactivity

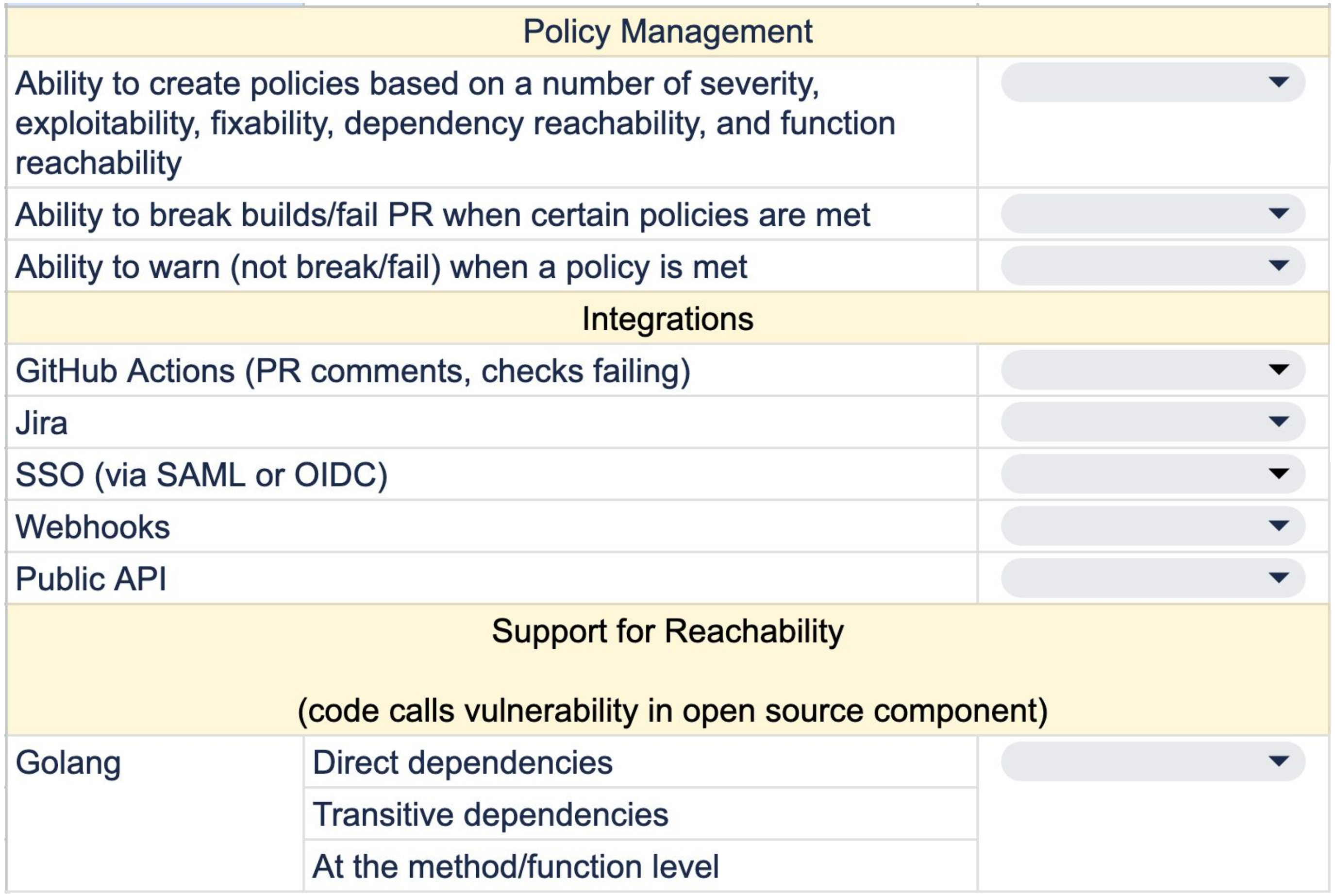
Address vulnerabilities in open source dependencies

Step 1: Evaluating SCA tools

- Define and document your own set of requirements
- For each, define the level of need: **MUST HAVE** **NICE TO HAVE**
- Search for SCA tools to evaluate
  - Compare your requirements with their docs
  - Pick max 3 tools for further evaluation

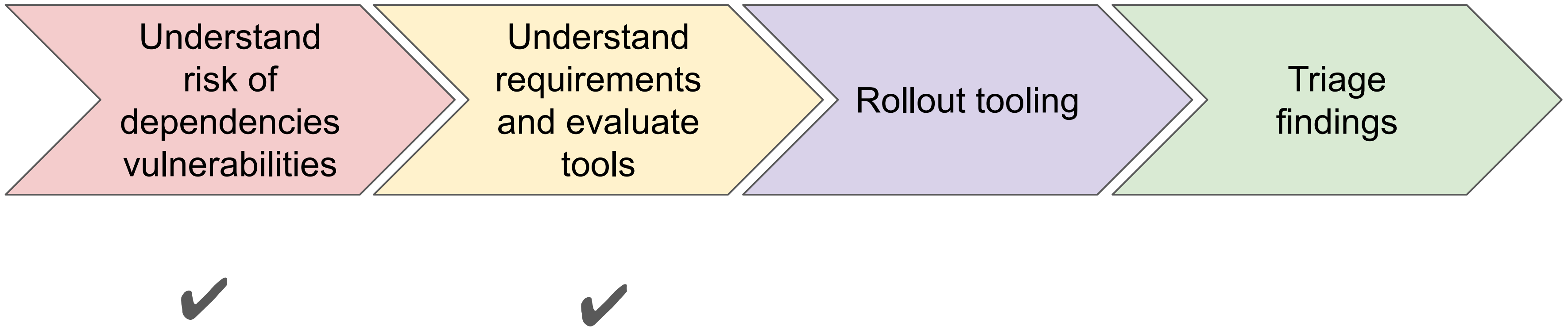


Requirement		Need
General		
Dependency scanning for both direct and transitive dependencies		Must Have
Continuous monitoring of any newly disclosed CVEs		
Zero to minimal findings inconsistencies		
Suggestions for vulnerability mitigations		
Support for incremental scanning		
Ability to automatically create SBOMs from projects		Nice To Have
Reporting capabilities		
Languages / Package Managers		
Golang	Go Modules	
Javascript	NPM	
Java	Maven + Gradle (without pom.xml files)	
	build.gradle	
Kotlin	Maven + Gradle (without pom.xml files)	
Swift	Cocoapods	



Policy Management		
Ability to create policies based on a number of severity, exploitability, fixability, dependency reachability, and function reachability		
Ability to break builds/fail PR when certain policies are met		
Ability to warn (not break/fail) when a policy is met		
Integrations		
GitHub Actions (PR comments, checks failing)		
Jira		
SSO (via SAML or OIDC)		
Webhooks		
Public API		
Support for Reachability (code calls vulnerability in open source component)		
Golang	Direct dependencies	
	Transitive dependencies	
	At the method/function level	

Dependency Management Journey


Understand risk of dependencies vulnerabilities
Understand requirements and evaluate tools
Rollout tooling
Triage findings
✔
✔

Step 2: SCA Tool Rollout

Option 1 - Rollout all at once

- Integrate the SCA tool with all the repos
- Start triaging findings

Step 2: SCA Tool Rollout

Option 2 - Phased Rollout

- Make a list of your most important repos (max 5)
- t=0: repo X is integrated in the SCA tool
- t+15d:
  - repo Y is integrated in the SCA tool
  - Critical/High findings of repo X are mitigated
- t+30d:
  - repo Z is integrated in the SCA tool
  - Critical/High findings of repo Y are mitigated

….

Step 2: SCA Tool Rollout

Factors to consider when deciding options

- Are different teams responsible for the repositories to be added to the SCA tool?
- How many vulnerable dependencies does each repository have?
- How many repositories will be integrated?
- What is the teams’ current capacity and workload?

Dependency Management Goals - SCA Rollout

Proactivity

Once you integrate a repo:

- Make sure that new PRs introducing dependencies are scanned

Reactivity

Risk-based approach:

- Address Critical/High vulnerabilities in the repos that get integrated into the SCA solution.
- Medium/Lows to follow

Step 3: Triaging Findings

Timeboxed investigation

- Is it a test dependency?
- Is the vulnerable dependency or the vulnerable function of the dependency is reachable?

Balance between the time spent for investigation and the time spent to update the dependency

Step 3: Triaging Findings

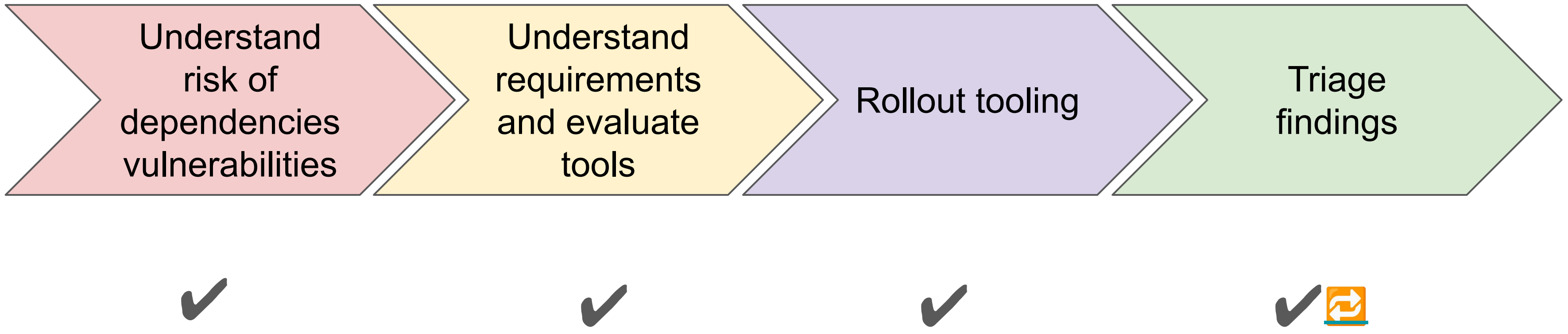
Updating the dependency isn't always an option, if no fix is available

- Notify the maintainer
- Contribute by addressing the issue in the repository
- Explore alternative dependencies with similar functionality

Sustaining Dependency Management Efforts

- Track the resolution of vulnerable dependencies using existing workflows like GitHub issues or Jira for consistency
- Anticipate future vulnerabilities in existing dependencies
- Establish a regular maintenance schedule to proactively update dependencies

Dependency Management Journey


Understand risk of dependencies vulnerabilities
Understand requirements and evaluate tools
Rollout tooling
Triage findings
✔
✔
✔
✔🔁

Key Takeaways

- Choose your SCA tool wisely
- Roll out SCA with a clear strategy for addressing findings and implementing a proactive approach.
- Mitigating vulnerabilities in 3rd party dependencies is an org wide effort
    - Get agreement from Engineering leadership

Thank you!

Eva Sarafianou

FOSDEM
1 & 2 February, Brussels 2025