# SOFTWARE COMPOSITION ANALYSIS

**Maya Kaczorowski**

***Product Manager***

*Software Supply Chain Security*

GitHub

Brought to you by: OWASP®

## Maya Kaczorowski

Product Manager, Software Supply Chain Security

GitHub

@MayaKaczorowski

## Agenda

## What's Software Composition Analysis (SCA)?

- Know your environment
- Manage your dependencies
- Monitor any changes

## Application security tools: categories

**SAST:** Static Application Security Testing

**DAST:** Dynamic Application Security Testing

**IAST:** Interactive Application Security Testing

**RASP:** Runtime Application Self Protection

## Application security tools: categories

**SAST:** Static Application Security Testing

**DAST:** Dynamic Application Security Testing

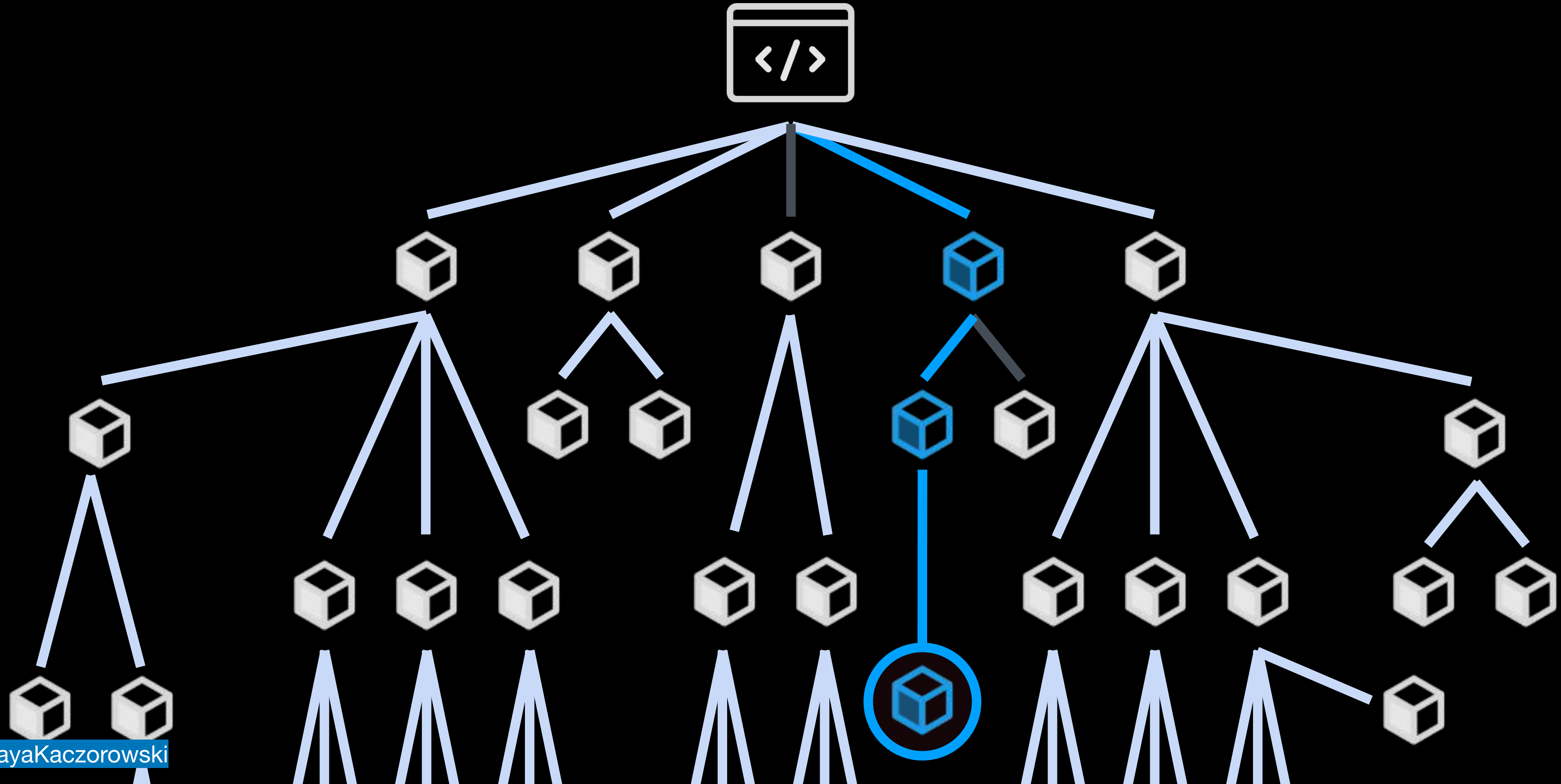
**IAST:** Interactive Application Security Testing

**RASP:** Runtime Application Self Protection

**SCA: Software Composition Analysis**

@MayaKaczorowski

## 1 What’s Software Composition Analysis (SCA)?

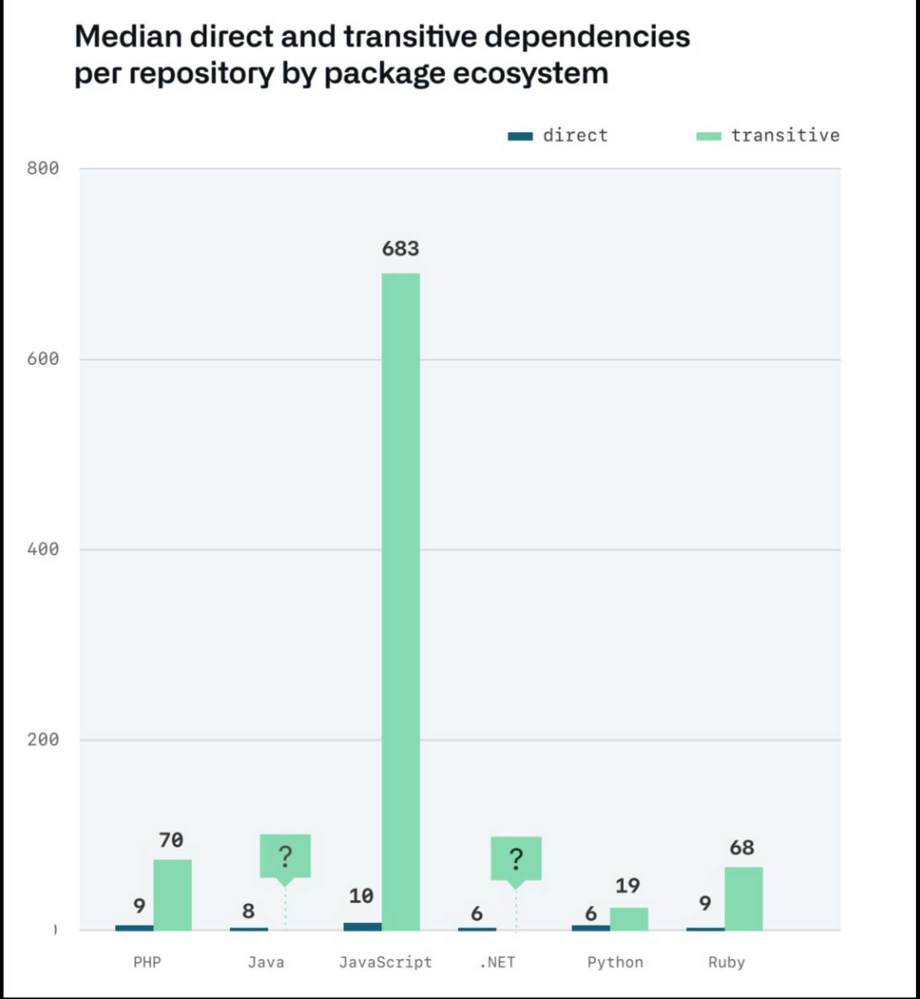


A dependency is another binary that your software needs in order to run

@MayaKaczorowski

## A **dependency** is another binary that your software needs in order to run

- Author
- Date of contribution
- Date of publication
- Security reviews
- Known vulnerabilities
- Supported versions
- License information
- etc.



# Software dependencies are pervasive

@MayaKaczorowski

## Software composition analysis

Analysts coined the term “software composition analysis”

Software composition analysis is identifying the dependencies and components used in a piece of software that is shipping, and their vulnerabilities, licenses, and other metadata.

Can be combined with Application Security Testing (SAST, DAST, IAST and SCA), or considered as its own category

## 2 Software Composition Analysis capabilities

Developers

Know what’s in your environment

Manage your dependencies

Monitor your supply chain

## Software composition analysis: capabilities

**Know**
**what's in your environment**

- **Discover your dependencies**, including transitive and checked in dependencies
- **Understand your risks**, such as vulnerabilities and licensing restrictions

## Software composition analysis: capabilities

**Know**
**what's in your environment**

- **Discover your dependencies**, including transitive and checked in dependencies
- **Understand your risks**, such as vulnerabilities and licensing restrictions

**Manage**
**your dependencies**

- **Determine if you are impacted** by a new security issue
- **Update** for the latest functionality and security patches
- **Review changes** to understand and approve new dependencies you're introducing
- **Remove unnecessary dependencies**, to reduce surface of attack

# 1.4x

faster to apply a patch
when an automatic pull request is generated

# Software composition analysis: capabilities

**Know**
**what's in your environment**

- **Discover your dependencies**, including transitive and checked in dependencies
- **Understand your risks**, such as vulnerabilities and licensing restrictions

**Manage**
**your dependencies**

- **Determine if you are impacted** by a new security issue
- **Update** for the latest functionality and security patches
- **Review changes** to understand and approve new dependencies you're introducing
- **Remove unnecessary dependencies**, to reduce surface of attack

# Software composition analysis: capabilities

## Know
**what's in your environment**

- **Discover your dependencies**, including transitive and checked in dependencies
- **Understand your risks**, such as vulnerabilities and licensing restrictions

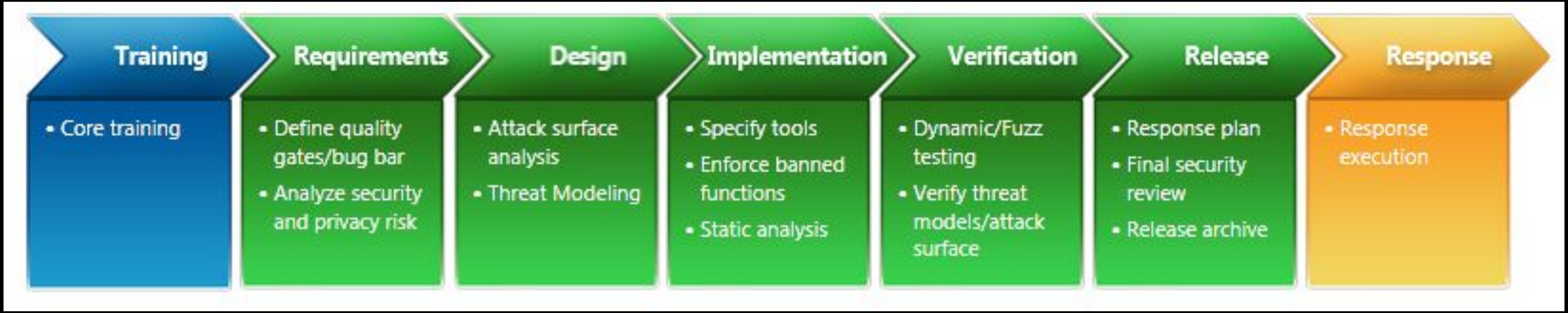
## Manage
**your dependencies**

- **Determine if you are impacted** by a new security issue
- **Update** for the latest functionality and security patches
- **Review changes** to understand and approve new dependencies you're introducing
- **Remove unnecessary dependencies**, to reduce surface of attack

## Monitor
**your supply chain**

- **Audit** your current environment for potential risks
- **Enforce policies** to prevent new issues from being introduced

# Software composition analysis: methods



*Microsoft's Secure Development Lifecycle (SDL)*

$\underbrace{\hspace{10em}}$

Apply SCA once you define your dependencies

@MayaKaczorowski

## Software composition analysis: methods

You can identify your dependencies…

… in code

… at build time

… in artifacts

@MayaKaczorowski

## Software composition analysis: shift left

Catch issues earlier:

- In IDE
- In PR
- At build time
- In artifacts
- … anytime before it’s exploited!

## Software composition analysis: enforcement

- Decide what you want to enforce
  - Vulnerabilities: severity, exploitability, etc.
  - Licenses
- Decide where you want to enforce
- Warn then enforce
- Prevent then remediate
- Focus on risk

@MayaKaczorowski

## Software composition analysis: OWASP dependency-check

- Lots of integrations! CLI, Ant, Maven, Jenkins, SBT, Homebrew
- Vulnerability information comes from NVD
  - Runs locally but needs to access the internet to get NVD data
- Reports on CPEs and CVEs
- Supports Java and .NET

https://jeremylong.github.io/DependencyCheck/

## Software composition analysis: Other OSS tools

- tl;drLegal: https://tldrlegal.com/
- npm audit: https://docs.npmjs.com/cli/v6/commands/npm-audit
- PyPI Safety: https://pypi.org/project/safety/
- Freemium tools
  - GitHub dependency graph, dependency review, and Dependabot alerts and updates
  - WhiteSource Bolt
  - Sonatype OSS Index
  - fossa-cli
  - FlexNet Code Aware
  - nexB/scancode-toolkit

## Considerations for choosing SCA tools

**Dependency detection**

- How are dependencies detected?
- Are transitive dependencies included?
- Are checked-in dependencies included?
- Which languages or ecosystems are supported?

**Vulnerability & license data**

- Where is vulnerability data sourced?
- What information on vulnerabilities is used to help prioritize remediation, e.g., severity?
- Where is license data sourced?

**Remediation actions**

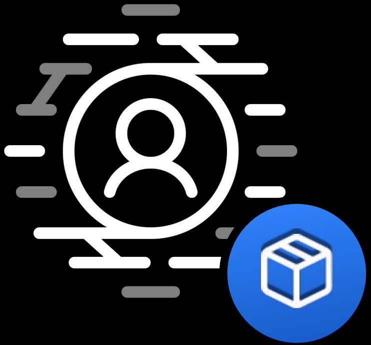
- What actions can you take from the tool, e.g., generate an alert, generate a fix, block an issue?

## Considerations for implementing SCA tools

- Responsibility for dependencies
- Current development pipeline
- Desired actions or enforcement
- Tracking and reporting security issues
- Languages and ecosystems

@MayaKaczorowski

## GitHub's SCA tools are free for open source



Developers

### Know your environment

**Dependency graph**

Understand your project's dependencies

**Dependency review**

Identify new dependencies and vulnerabilities in a PR

### Manage your dependencies

**Dependabot alerts**

Get notified of a vulnerability in a dependency

**Dependabot security updates**

Review a PR to update to the minimum fixed version

**Dependabot version updates**

Review a PR to update to the latest stable dependency version

## Demo

## Summary

- **Software Composition Analysis (SCA)** is about managing the security of what's in your supply chain
- Three main capabilities:
  - Know what's in your environment
  - Manage your dependencies, and
  - Monitor your supply chain
- Dependencies can be detected in code, at build time, or in artifacts
  - Consider shifting left
- When choosing an SCA tool, consider
  - How dependencies are detected
  - Where vulnerability and license information comes from
  - What remediation actions and enforcement options you have
  - How it'll work in your environment!