

Supply Chain Risk Management with OWASP Dependency-Check



Matt Kendall

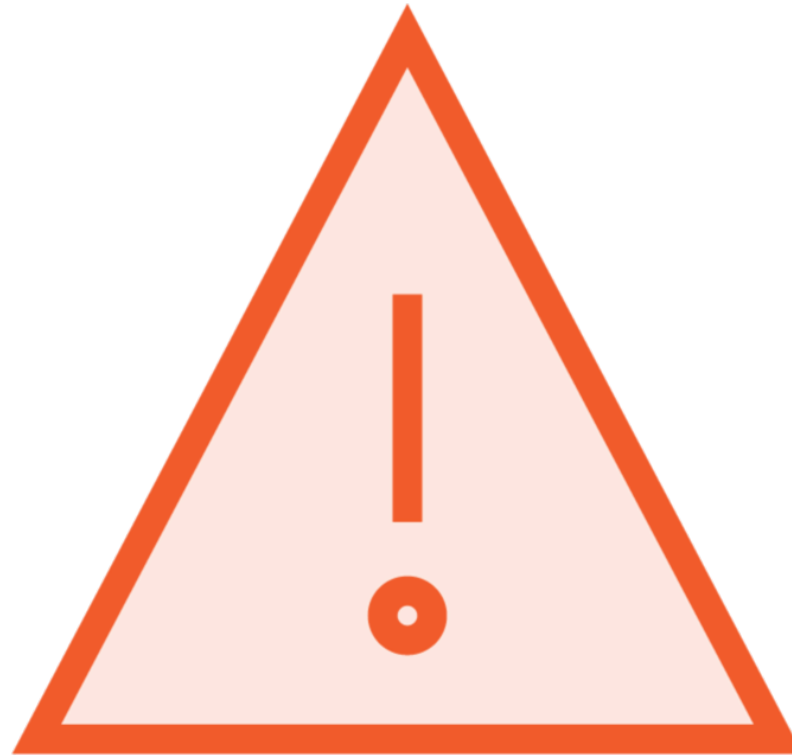
Software Engineer & Security Champion



What's in an Application?



**95% of applications
use OSS components**



**67% of applications
have open-source
vulnerabilities**



**Numerous high profile
incidents (including
Log4Shell)**



An Industry Recognized Risk

OWASP Top 10 2021

**#6 Vulnerable and Outdated
Components**

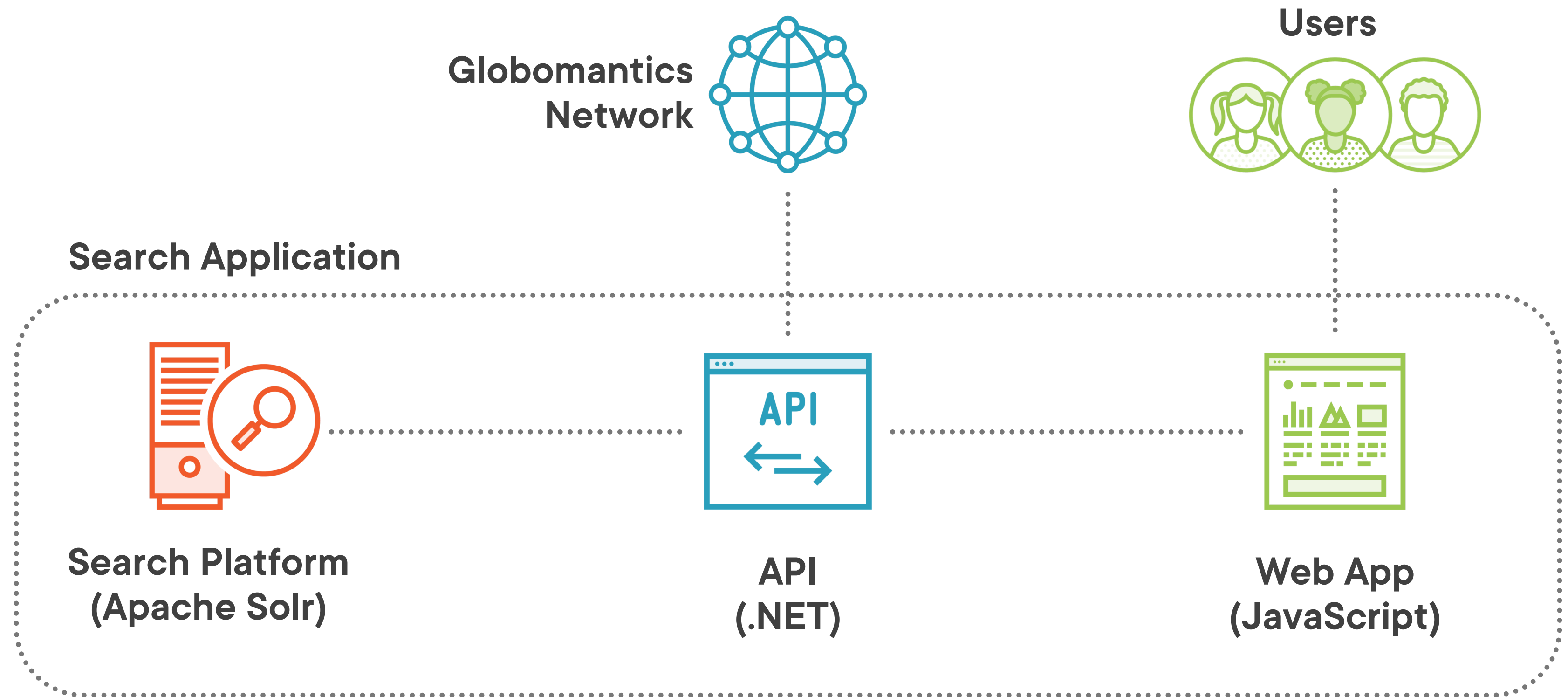
NIST Cyber Security Framework

**ID.SC: Supply Chain Risk
Management**





The Globomantics Search Application



Software Composition Analysis (SCA)



Software Composition Analysis (SCA)



Analyze

Create list of
dependencies and
versions



Compare

Check dependency
list against
vulnerability database



Report

Create human or
machine-readable
report



OWASP Dependency- Check

Command-line tool

Supports multiple technologies

- Java, .NET, JavaScript
- (Experimental) Python, Ruby, Swift, PHP

Checks the National Vulnerability Database

- May also use GitHub Advisory Database, RetireJS, Sonatype OSS Index

Runs a series of specialized analyzers

- All included by default



Demo



Installing OWASP Dependency-Check

Prerequisites

- Java 8 or newer



Demo Clip – Installation



Demo



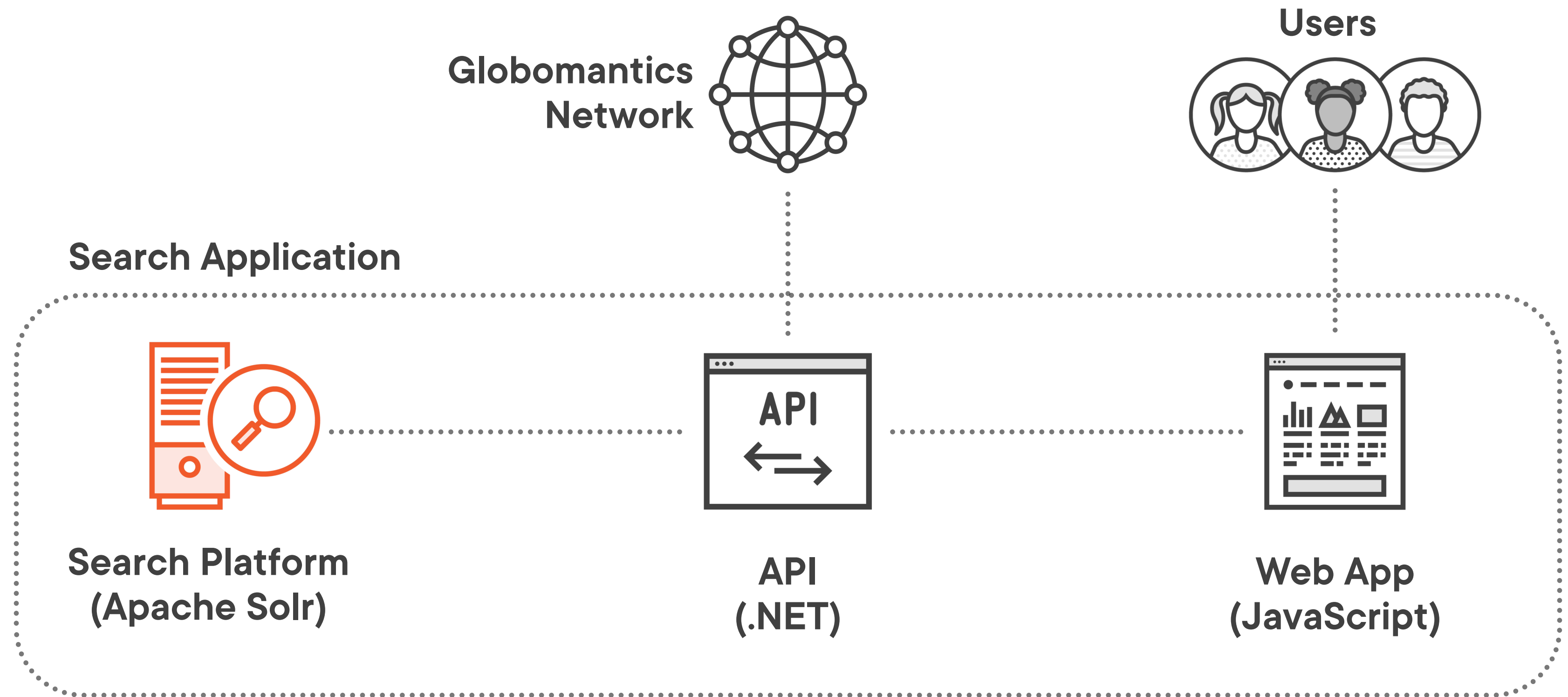
Analyzing a Java Application (Apache Solr)

Understanding output report

- Reading the HTML report
- Configuring for additional report types



The Globomantics Search Application



```
dependency-check
```

```
--scan "C:/solr"
```

◀ **Scan specified path**

Demo Clip – Default scan on a Java project



```
dependency-check
```

```
--scan "C:/solr"
```

```
--format JSON
```

```
--out solr-report.json
```

◀ Scan specified path

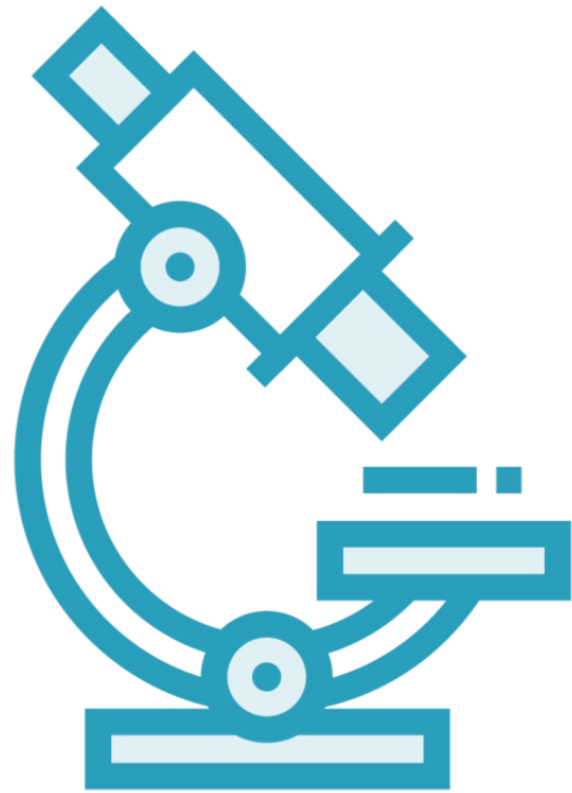
◀ Output report in JSON format

◀ Set output filename to 'solr-report.json'

Demo Clip – Configure report options on a Java project



Dependency Scanning Uses

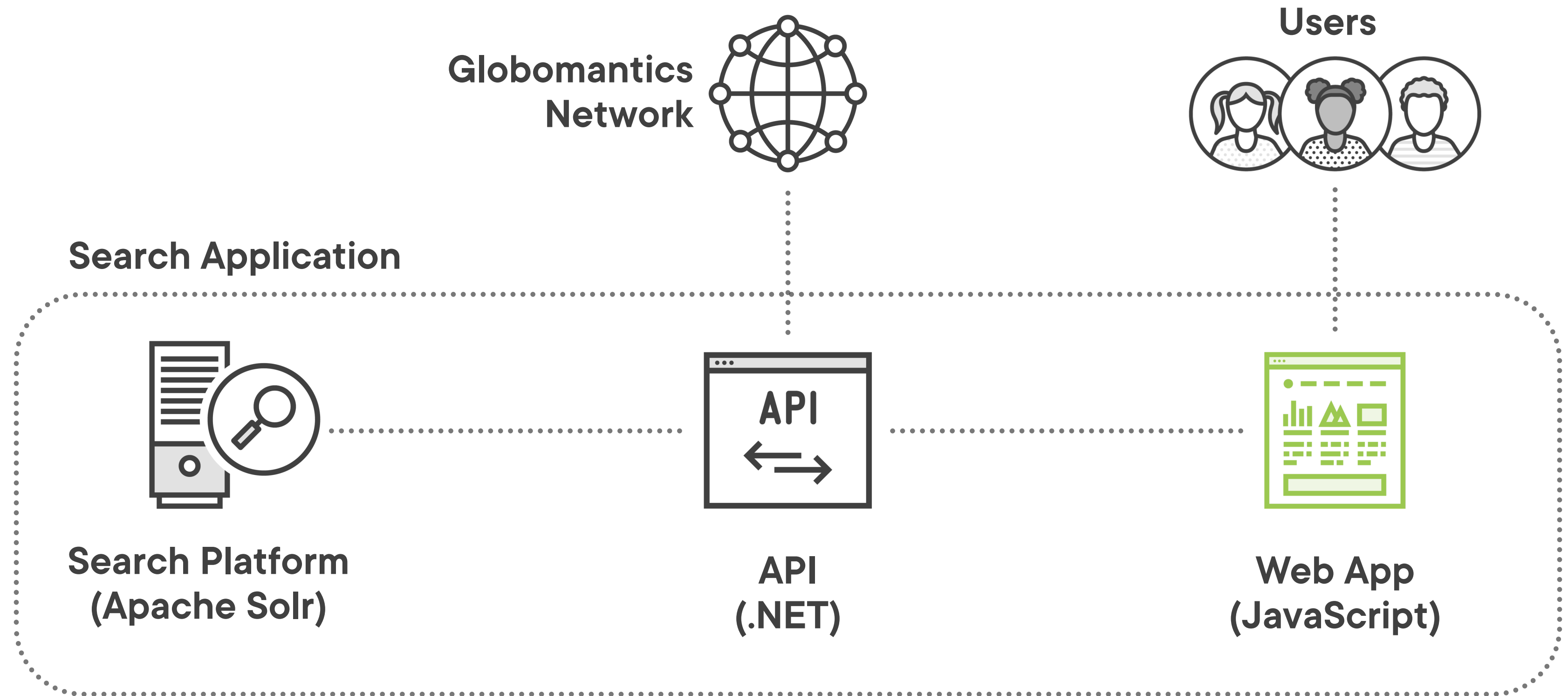


Security Assessment



Continuous Security

The Globomantics Search Application



Demo



Analyzing a JavaScript webapp

Configuring for best results

- Excluding directories
- Disabling unhelpful analyzers
- Excluding JS dev dependencies



Demo Clip – Default scan on a JavaScript project



JavaScript Analyzers

Node Audit

Submits NPM dependency data to NPM Audit API

Uses the GitHub Security Advisory database

Best for NPM projects

NodeJS

Checks for vulnerable packages in NPM / Yarn files

Uses the NVD and Sonatype databases

Good if you can't use the Node Audit analyzer

RetireJS

Scans for vulnerable third-party, client-side .js files

Checks against the RetireJS database

Great for copies of dependencies outside of package manager



dependency-check

--scan .

--disableNodeJS

--exclude "**/node_modules/**"

◀ Scan current directory

◀ Don't use the NodeJS Analyzer

◀ Don't scan the *node_modules* directory

Demo Clip – Configuring analyzers and exclusions on a JavaScript project



Development Dependencies

Third-party dependencies required only at build-time

Used for tasks like

- Transpiling / bundling code
- Running development tests

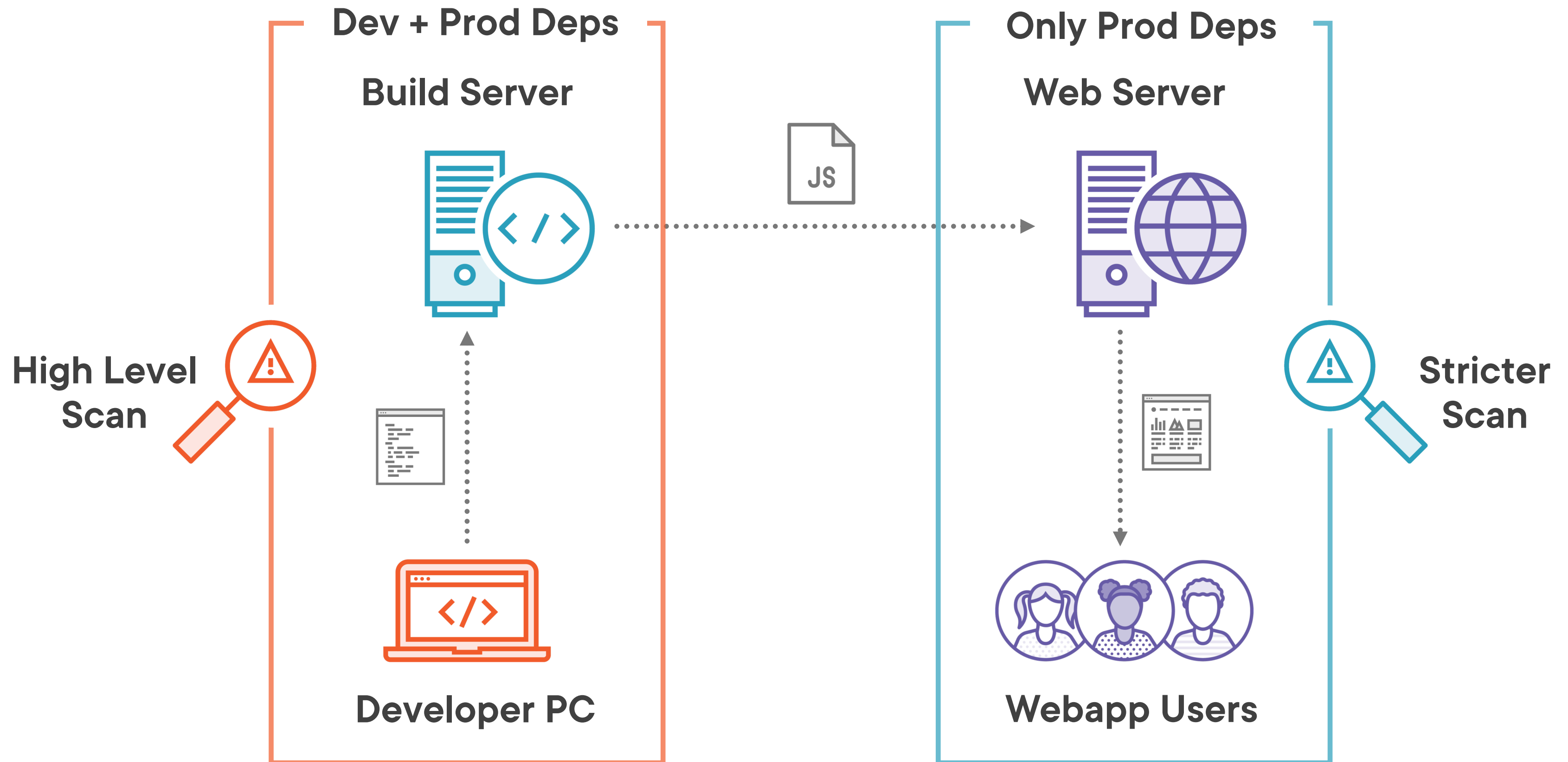
Not included in production code

Vulnerability scanning dev dependencies often ‘drowns out’ production vulnerabilities

- But is it safe to ignore them?



Globomantics Build Process



dependency-check

```
--scan .  
  
--disableNodeJS  
  
--exclude "**/node_modules/**"  
  
--nodeAuditSkipDevDependencies
```

- ◀ Scan current directory
- ◀ Don't use the NodeJS Analyzer
- ◀ Don't scan the *node_modules* directory
- ◀ Have the Node Audit analyzer skip development dependencies

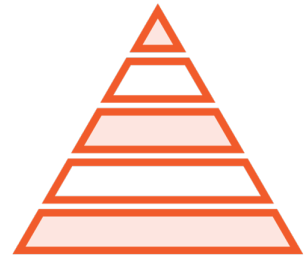
Demo Clip – Skipping dev dependencies on a JavaScript project



Remediating Vulnerable Components



Remediating Vulnerable Components



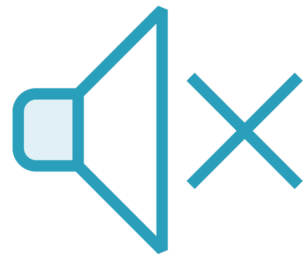
Prioritize by severity



Patch if possible



Investigate whether *your* app is vulnerable



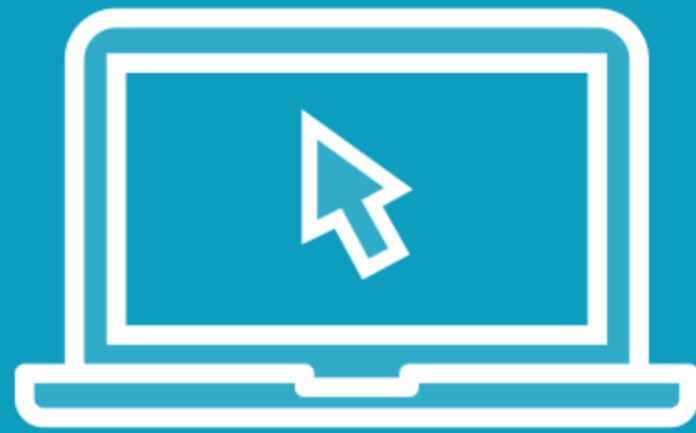
Suppress and document, if appropriate



Make changes to secure your app if required



Demo



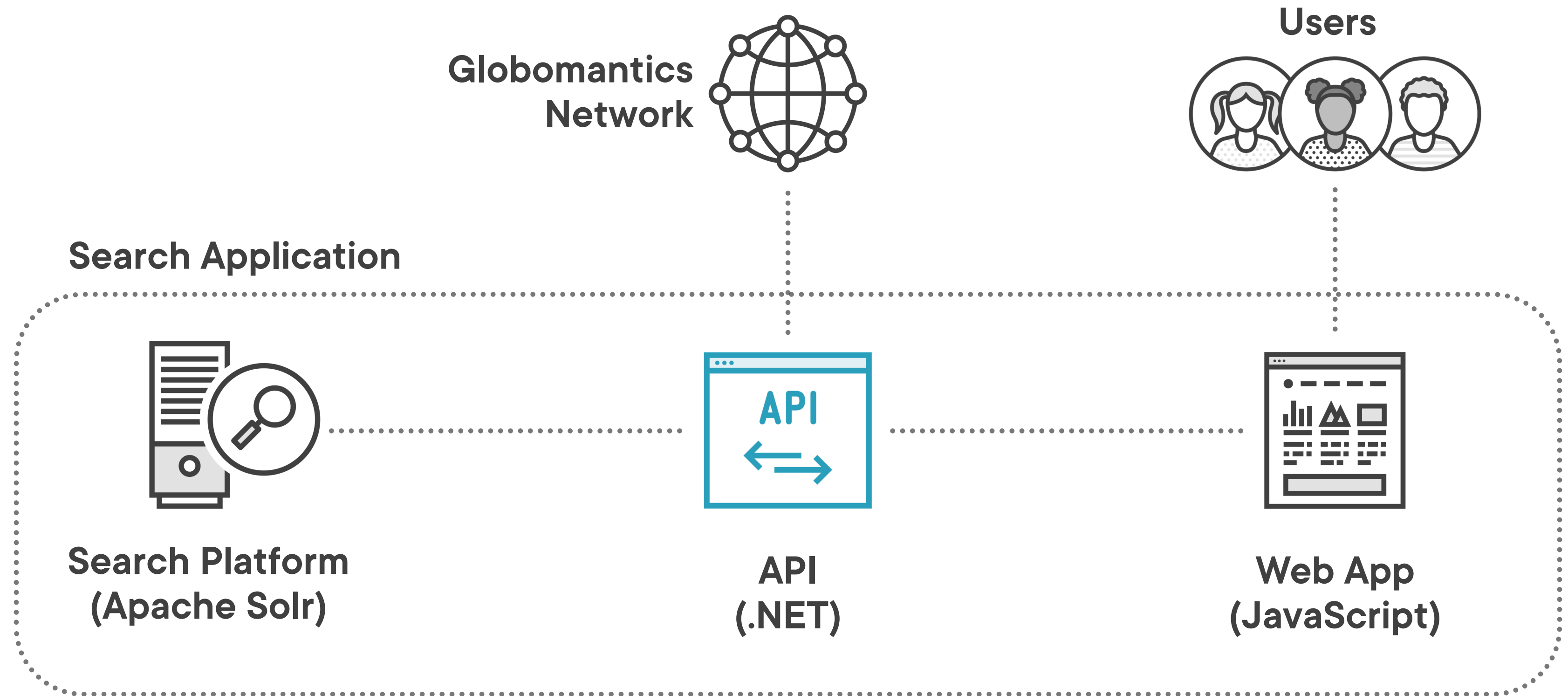
Analyzing a .NET API

Configuring for ongoing monitoring

- Setting a severity failure level
- Suppressing false positives



The Globomantics Search Application



Common Vulnerability Scoring System (CVSS)

| CVSS Score | Severity |
|------------|----------|
| 0.1 – 3.9 | Low |
| 4.0 – 6.9 | Medium |
| 7.0 – 8.9 | High |
| 9.0 – 10.0 | Critical |




```
dependency-check
```

```
--scan .
```

```
--failOnCVSS 7.0
```

◀ **Scan current directory**

◀ **Fail if vulnerability with CVSS 7.0 or greater is found**

Demo Clip – Configuring ‘fail on CVSS’ on a .NET project



dependency-check

--scan .

--failOnCVSS 7.0

--suppression suppression.xml

◀ Scan current directory

◀ Fail if vulnerability with CVSS 7.0 or greater

◀ Ignore vulnerabilities listed in 'suppression.xml'

Demo Clip – Configuring suppression file on a .NET project



Tips for Using Dependency-Check



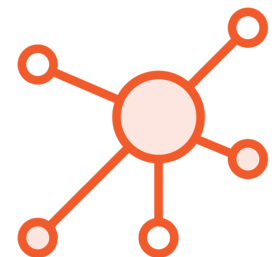
Tips for Using Dependency-Check



Understand how analyzers will interact with your application



Automate for regular scanning



Integrate with other tooling



Set appropriate quality gates for severity



Be pragmatic and start small if needed



Up Next:
Summary & Additional Resources



Summary



Software supply chain

SCA tools and OWASP Dependency-Check

Analyzing applications

- Java, .NET, JavaScript
- Configuring scans and analyzers

Remediating vulnerable components

- Suppressing false positives

Tips & best practices



More Information

Dependency-Check

Dependency-Check OWASP Project

<https://owasp.org/www-project-dependency-check>

CLI Documentation

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

GitHub Issues

<https://github.com/jeremylong/DependencyCheck/issues>

Getting Info on Vulnerabilities

National Vulnerability Database

<https://nvd.nist.gov/vuln/search>

GitHub Advisory Database

<https://github.com/advisories>

Snyk Vulnerability Database

<https://security.snyk.io/>



Thank you

