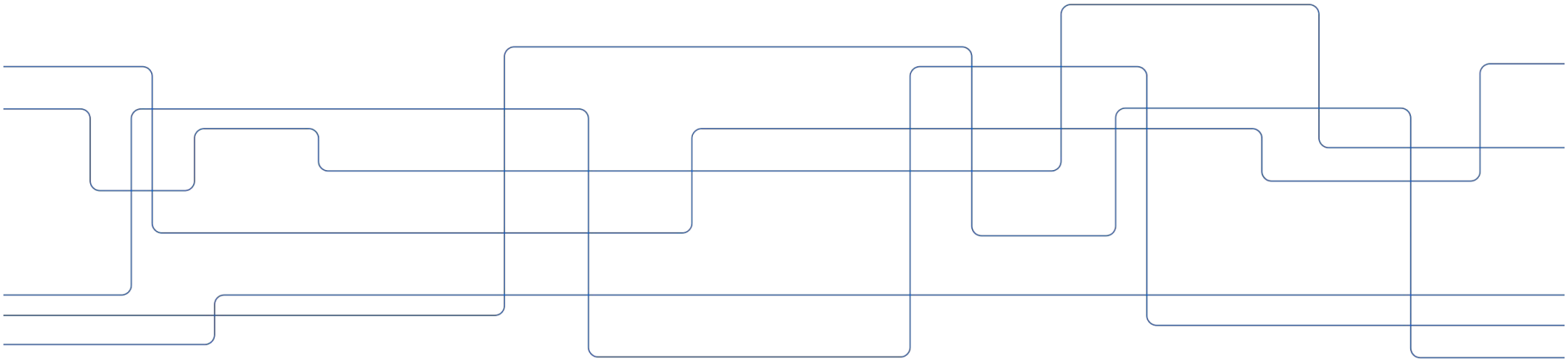
KTH ROYAL INSTITUTE
OF TECHNOLOGY

Challenges of Producing Software Bill Of Materials for Java

Aman Sharma & Martin Wittlinger



Contents

1. What is software supply chain?
    a. Supply chain attacks
    b. Examples
2. What is an SBOM?
    a. Use cases
    b. Content of an SBOM
3. Analysis of SBOMs
4. Qualitative Analysis
5. Quantitative Analysis
    a. Ground truth
    b. Metrics computation
    c. Results
6. Takeaways
7. Future Work

What is a Software Supply Chain?

"The sequence of steps resulting in the creation of an artifact."

SLSA

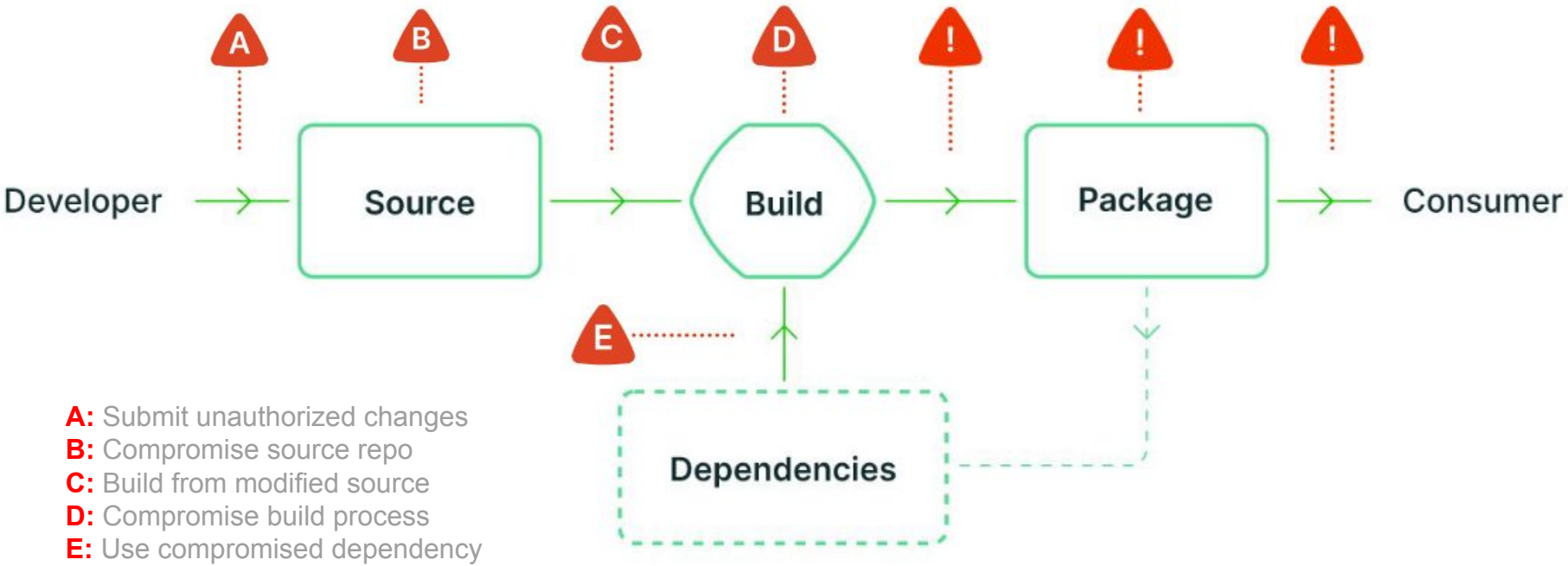
"The software supply chain is made up of everything and everyone that touches your code in the software development lifecycle (SDLC), from application development to the CI/CD pipeline and deployment."

Redhat

"A software supply chain is composed of the components, libraries, tools, and processes used to develop, build, and publish a software artifact."

Wikipedia

Software Supply Chain Attack

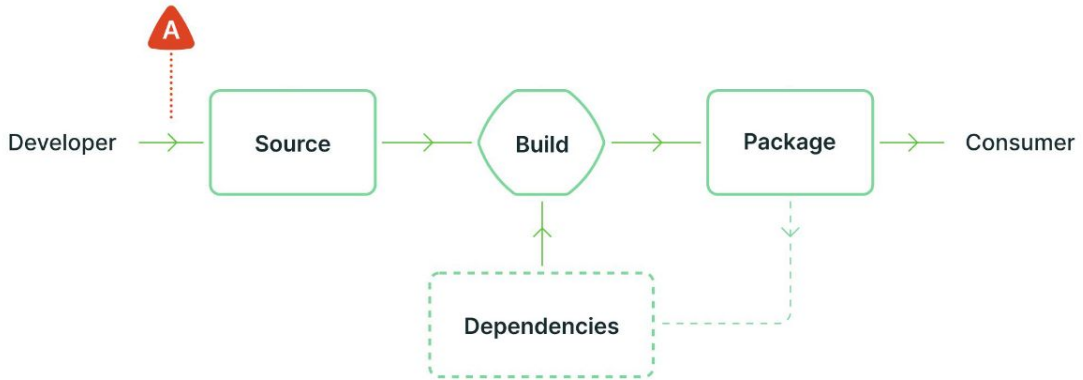


A: Submit unauthorized changes
B: Compromise source repo
C: Build from modified source
D: Compromise build process
E: Use compromised dependency

Source: https://slsa.dev/

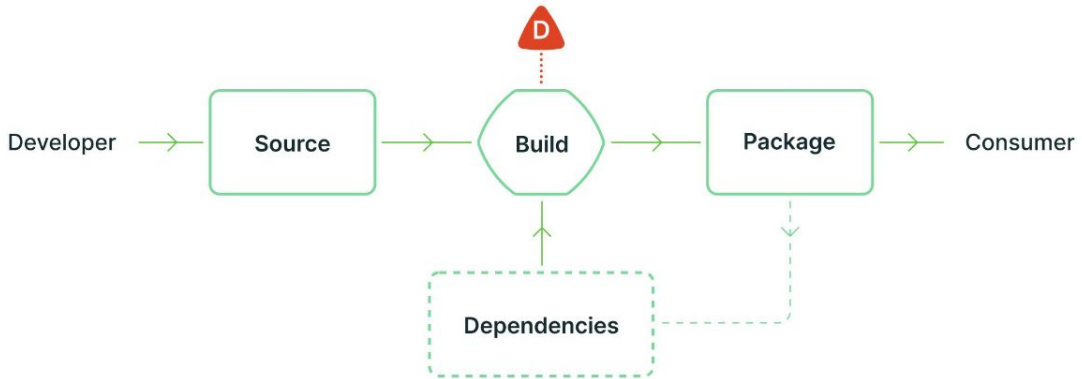
Misconfigured Dev and QA tool compromised (2021)

- Write access to VSCode main repository without permissions
- Attacks your local code editor
- Link to attack - https://github.com/cncf/tag-security/blob/main/supply-chain-security/compromises/2021/vscode.md



GCP Golang Buildpacks Old Compiler Injection (2022)

- Old version of go compiler pulled
- Old compiler versions have known vulnerabilities
- Could have detected it with *SBOM*
- Link to attack - https://github.com/cncf/tag-security/blob/3c63c2b4fd7763479222766b89cc5ff81eba9291/supply-chain-security/compromises/2022/golang-buildpacks-compiler.md



What is an SBOM?

“An SBOM is a formal, machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships.”

- NTIA

- Machine-readable
- List of components and dependencies
- Information about components
- Hierarchical relationships

Dependencies

```
<dependency>

<groupId>fr.inria.gforge.spoon</groupId>
    <artifactId>spoon-core</artifactId>
    <version>10.3.0</version>
</dependency>
```

Direct Dependencies

```
org.apache.maven.shared
maven-invoker
3.2.0
```

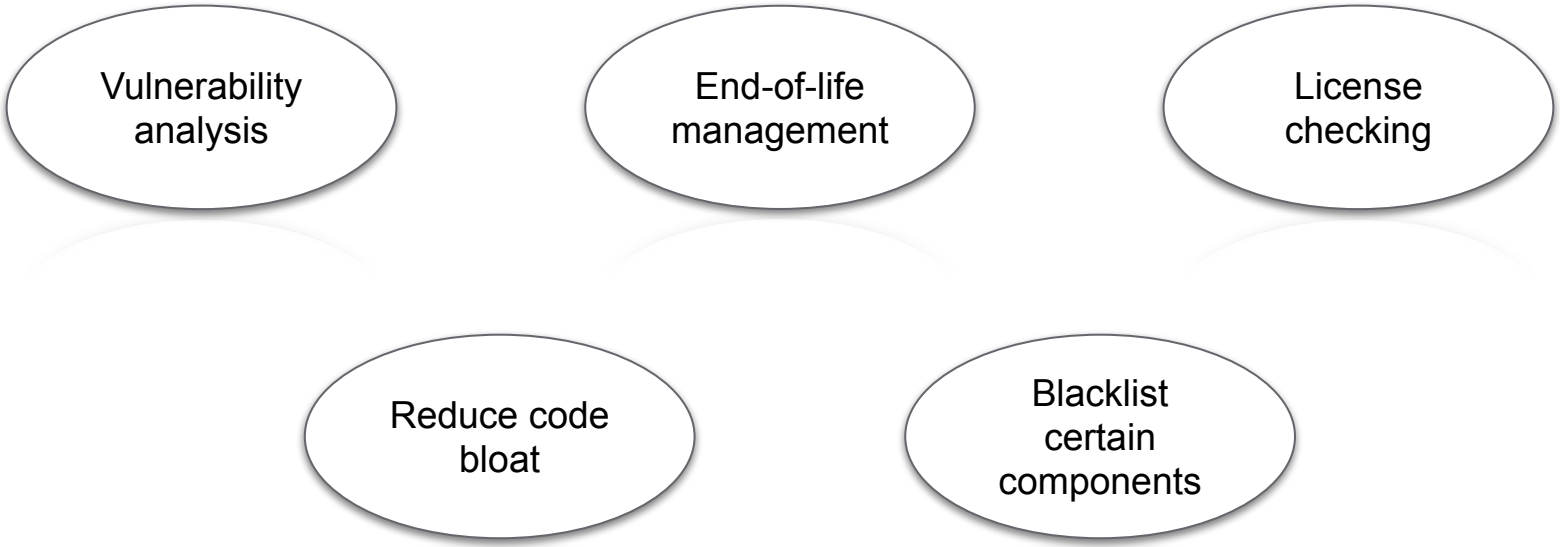
```
org.apache.maven.shared
maven-invoker
3.2.0
```

...

...

Transitive Dependencies

Use cases


Vulnerability analysis
End-of-life management
License checking
Reduce code bloat
Blacklist certain components

Content of an SBOM

1. Metadata
2. Project
3. Dependencies
4. Relationship between dependencies and projects

https://cyclonedx.org/
https://spdx.dev/

Content of an SBOM: Metadata

```
{ "bomFormat" : "CycloneDX",
 "specVersion" : "1.4",
 "metadata" : {
   "timestamp" : "2023-02-20T16:14:42Z",
   "tools" : [
     { "name" : "CycloneDX Maven plugin",
       "version" : "2.7.5" }
   ],
```

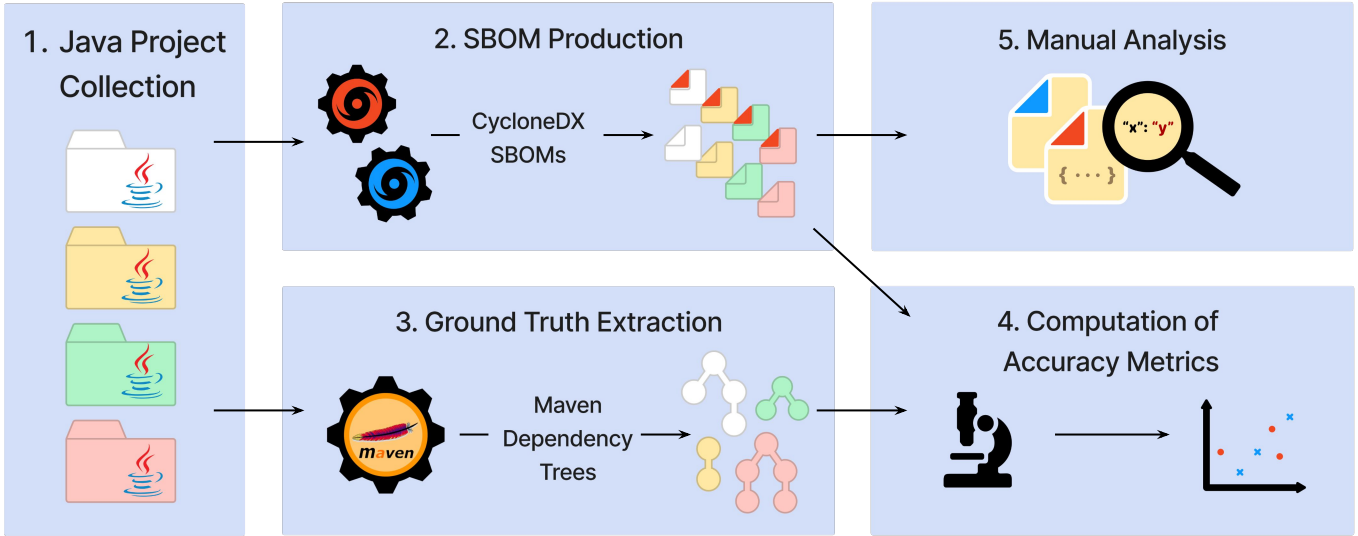
Content of an SBOM: Project

```
"component" : {
     "group" : "org.asynchttpclient",
     "name" : "async-http-client-project",
     "version" : "2.12.3",
     "hashes" : [ { "alg" : "SHA-512",
         "content" : "e5435852...7b3e6173" }, ... ],
     "licenses" : [...],
     "externalReferences" : [ {
       "url" : "http://github.com/AsyncHttpClient/async-http-client"   }
     ],
     "bom-ref" :
"pkg:maven/org.asynchttpclient/async-http-client-project@2.12.3?type=pom"
   }
```

Content of an SBOM: Libraries & Relationships

```
"components" : [
   { "group" : "com.sun.activation",
     "name" : "jakarta.activation",
     "version" : "1.2.2",
     "bom-ref" :
"pkg:maven/com.sun.activation/jakarta.activation@1.2.2?type=jar"
   } ...
 ],
"dependencies" : [ {
     "ref" :
"pkg:maven/org.asynchttpclient/async-http-client-project@2.12.3?type=pom"  ,
     "dependsOn" : [
       "pkg:maven/com.sun.activation/jakarta.activation@1.2.2?type=jar"
       ....
     ]
   } ... ] }
```

Analysis of SBOMs



Qualitative Analysis

SBOM Producer	Checksums	Hierarchy	Deterministic	Scope
Build-Info-Go	3	✓	✗	✗
cdxgen	8	✓	✓	✓
cyclonedx-maven-plugin	8	✓	✓	✓
depscan	8	✓	✓	✓
jbom	2	✗	✗	✓
OpenRewrite	0	✓	✓	✓

Qualitative Analysis

SBOM Producer	Checksums	Hierarchy	Deterministic	Scope
Build-Info-Go	3	✓	✗	✗
cdxgen				✓
cyclonedx-maven-plugin				✓
depscan				✓
jbom				✓
OpenRewrite	0	✓	✓	✓

Takeaway:

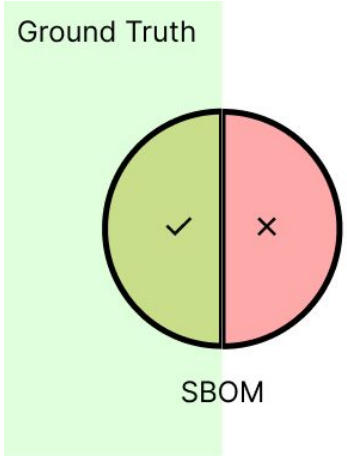
Different SBOM producers have distinct properties

Ground Truth: Maven Dependency Tree

- Integral part of the Maven build system
- Proven by use; first release in 2007
- It uses the same maven resolver as the build
- Returns group ID, artifact ID, and version of each dependency
  - Example: '`fr.inria.gforge.spoon:spoon-core:10.3.0`'
  - `fr.inria.gforge.spoon` is the group ID
  - `spoon-core` is the artifact ID
  - `10.3.0` is the version

Metrics computation

We compute precision and recall based on group ID, artifact ID, and version

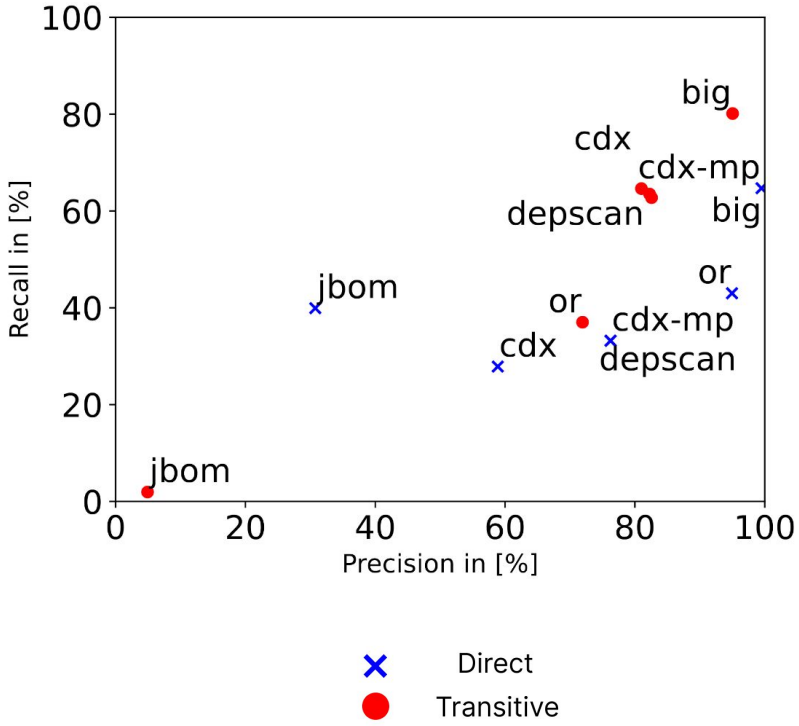


$$\text{Precision} = \frac{\text{green}}{\text{green} + \text{red}}$$

$$\text{Recall} = \frac{\text{green}}{\text{green} + \text{green}}$$

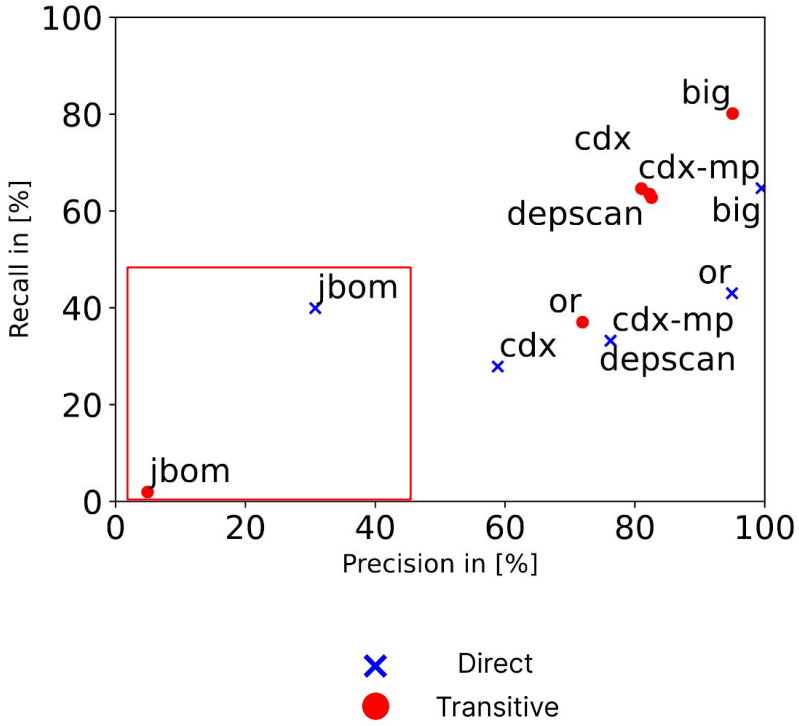
Quantitative Analysis

- Compare 6 producers against *the ground truth*
- The average of 26 runs on each datapoint
- Blue crosses are direct dependencies
- Red circles are transitive dependencies



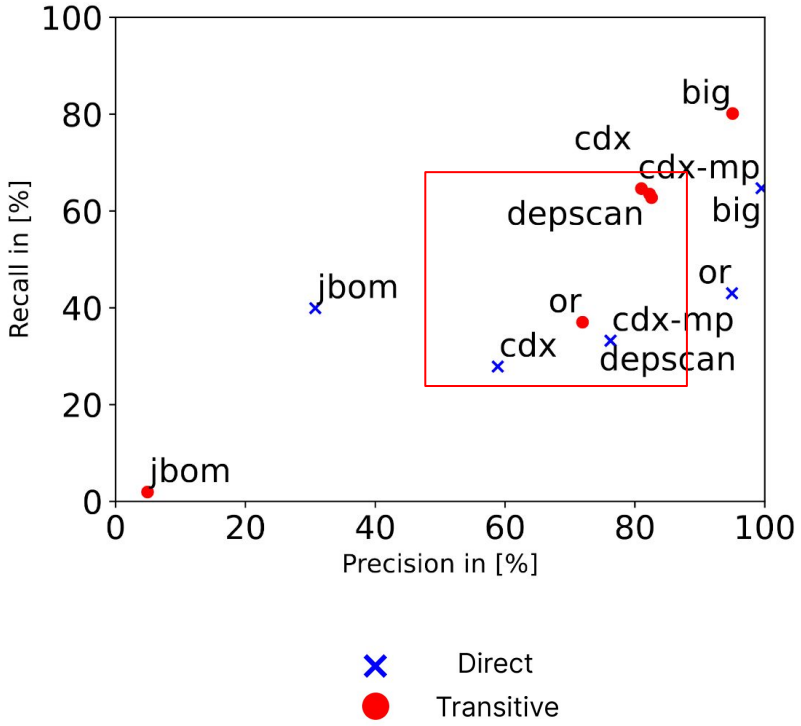
Results: jbom

- Very low precision and recall on direct dependencies
- Transitive dependencies are detected even less because it does not capture the hierarchy information



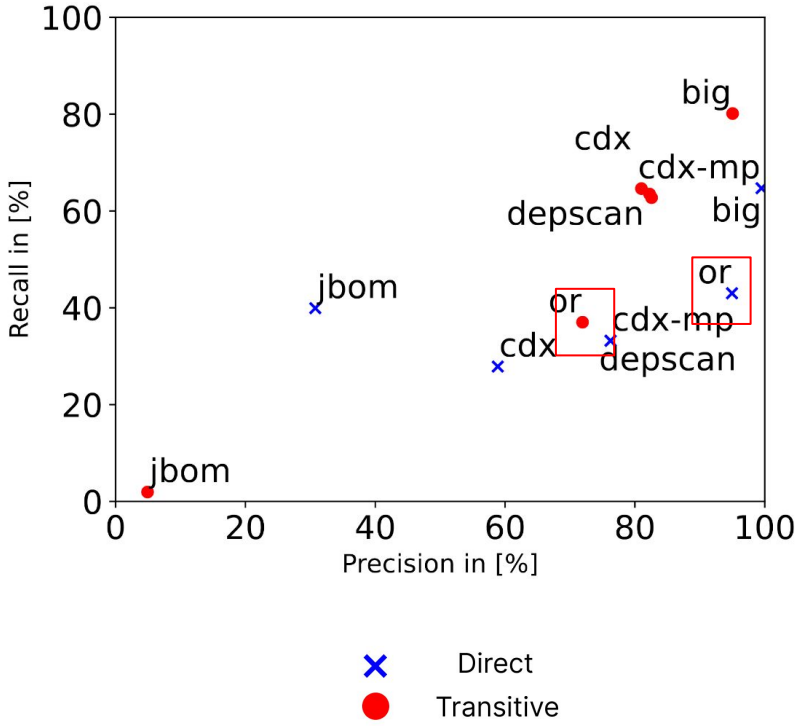
Results: cdxgen

- cdxgen , cyclonedx-maven-plugin, and depscan have similar results
- They share the same backend as cyclonedx-maven-plugin
- Problems with complex maven builds
- Test dependencies missing



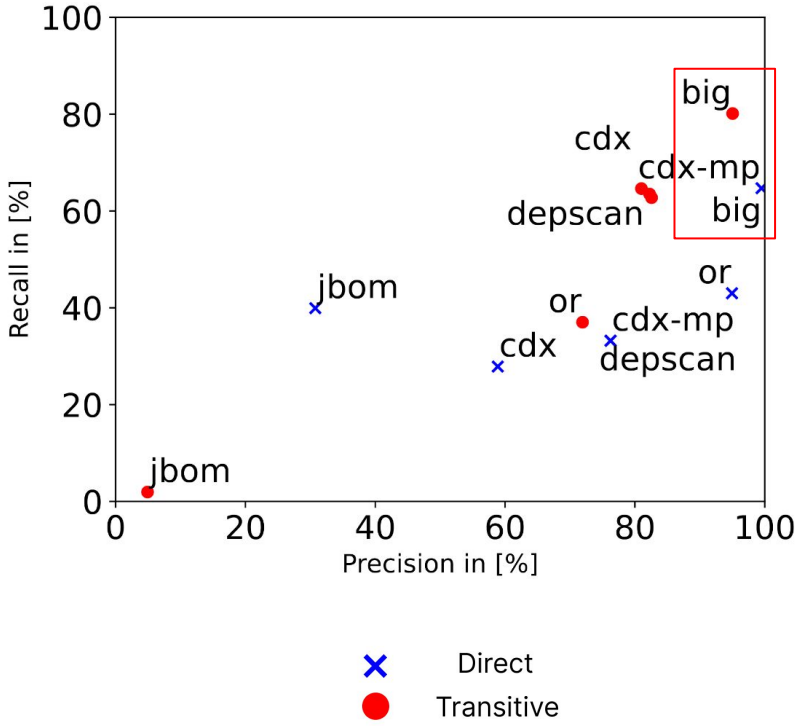
Results: openrewrite

- Highly precise on direct dependencies
- Misses out on test dependencies which also affects transitive



Results: build-info-go

- Best producer overall
- 99.5% precise on direct dependencies
- Still misses 35% of the dependencies



Takeaways: Java Developer

- Build-Info-Go is the best SBOM producer
- Different SBOM producers provide distinct feature set
- There is no silver bullet
  - Quality of different producers varies on different projects
  - Quality of the SBOM depends upon the maven build complexity

Takeaways: SBOM Consumers

- Input SBOM varies with SBOM producer
- Standard leaves room for interpretation
- Quality of producers will increase with consumption
- Higher adoption will improve the standard

Takeaways: Researchers

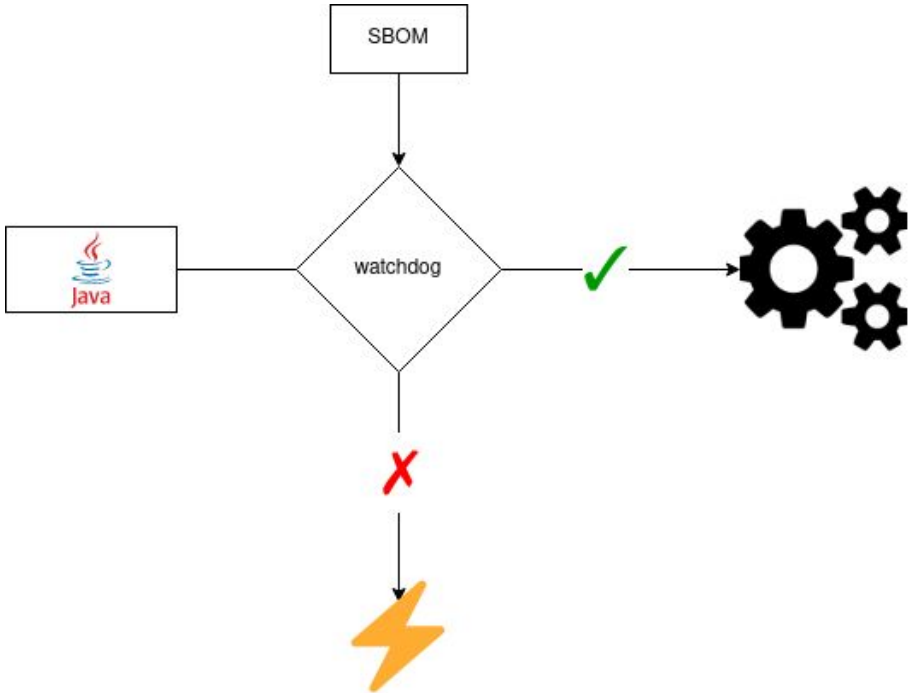
Production step of SBOM is an open question.

- When should we produce an SBOM?
- Shall we produce multiple SBOMs at different stages?
- At which stages in the supply chain?

Related Work

- Export SBOM for GitHub repository
- GraalVM produces SBOM during build
- Microsoft SBOM tool
- Snyk
- ….

Future Work: Runtime as the production step



Thank you!

Aman Sharma: amansha@kth.se

Martin Wittlinger: marwit@kth.se

Paper link: https://arxiv.org/abs/2303.11102