Software Composition Analysis

By Richard Stephanus
Agenda

1. Introduction
   Overview and Motivation

2. How to solve this problem?

3. How to get better?
Introduction
A real life example

Target: analyzed application was a middleware for REST services.

Some technical aspects:
• Key technologies: Java
• ~ 28,000 lines of code
• ~ 55 3rd party libraries
• ~ 38 APIs (REST). 82 functions
• Partly cloud based (MS Azure)
• ~ 120 Mio. Requests (clicks) per year

Project key facts:
• Project duration: 8 months
• Agile approach with 16 Sprints (every 2 week)
• Up to 26 team members (18 developers)
Programmers wrote
\~ 28,000 Line of code (loc)
Sounds much?

Anticipate a 3\textsuperscript{rd} party component consists only of 10,000 lines of code (log4j = 180K loc*)

Whole application consists of
28K + 55 \times 10,000 \rightarrow 578K loc

* https://www.openhub.net/p?query=log4j
Introduction

What is this code doing?
Introduction

Vulnerabilities in LOCs

Quality of source code changes depending on:

- Programming Language
  Example: compare Assembler with JavaScript?

- Type of Application
  Example: Mobile App and command line tool?

- Experience of developers
  ➔ Very different numbers
  ➔ 6 defects / 100.000 LOC *

<table>
<thead>
<tr>
<th>Package</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>2,862,087</td>
</tr>
<tr>
<td>PHP</td>
<td>3,882,984</td>
</tr>
<tr>
<td>Apache Tomcat</td>
<td>1,136,822</td>
</tr>
<tr>
<td>Linux</td>
<td>25,646,844</td>
</tr>
<tr>
<td>Mozilla Firefox</td>
<td>14,045,424</td>
</tr>
<tr>
<td>Google Chrome</td>
<td>15,441,702</td>
</tr>
<tr>
<td>Log4j</td>
<td>180,173</td>
</tr>
<tr>
<td>Spring</td>
<td>1,239,948</td>
</tr>
<tr>
<td>Hibernate ORM</td>
<td>720,095</td>
</tr>
</tbody>
</table>

➔ Very different numbers
➔ 6 defects / 100.000 LOC *

* The Economics of Software Quality By Capers Jones, Olivier Bonsignour
To say it clearly!

This is no problem of open source!
Introduction
Which libraries am I using?

Who is building OSS libraries?

Icons from https://www.flaticon.com/authors/freepik
Why should you trust your libs?

- Our open source team approved them
- They’re compiled!
- We control our software?
- Open source? Many eyes?
- We pentest?
- We patch?
- Static analysis?

OWASP Top 10

- 2010 → Security Misconfiguration → Position 6
- 2013 → Split of Security Misconfiguration
  → Position 5 (Security Misconfiguration)
  → Position 9 (Using Components with Known Vulnerabilities)
- 2017 → Using Components with Known Vulnerabilities → Position 9

- Examples
  → Apache CXF (CVE-2012-3451) → Bypass authentication
  → Commons collections, ver. 3.2.1 (CVE-2015-6420) → Execute arbitrary code

How do you get aware of problems with libraries?
How to solve this problem?
How to solve this problem?

Security Testing!

- OWASP Dependency Check
  Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities. Currently Java and .NET are supported; additional experimental support has been added for Ruby, Node.js, Python, and limited support for C/C++ build systems (autoconf and cmake).

- Sonatype Nexus Health Check
  A fully integrated health check for all components within a repository

- Artifactory
  Integration with external service (Blackduck) possible.

- ...

Commercial solutions
How to solve this problem?

OWASP Dependency Check

Commandline tool to check libraries for security issues

**Configuration**
- Locate used libraries
- Create configuration

**Perform the scan**
- Update local database from MITRE
- Identify distinct library and version → CPE
- Check CPE for known vulnerabilities (CVE)

**Create a report**
- Per default a XML-file is created
- XML-file can be integrated in various solutions

... and many more!

By The Jenkins project: [https://jenkins-ci.org/](https://jenkins-ci.org/)
By SonarSource: SonarQube, CC BY 3.0, [https://commons.wikimedia.org/w/index.php?curid=27920948](https://commons.wikimedia.org/w/index.php?curid=27920948)
How to solve this problem?

OWASP Dependency Check

**Dependency Check Result**

**Warnings Trend**

<table>
<thead>
<tr>
<th>Category</th>
<th>Total</th>
<th>High Priority</th>
<th>Normal Priority</th>
<th>Low Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Details**

<table>
<thead>
<tr>
<th>Dependency</th>
<th>CVE</th>
<th>Confidence</th>
<th>Evidence Credit</th>
<th>Inherited Risk</th>
<th>Vulnerable Component Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>716</td>
<td>5.7%</td>
</tr>
</tbody>
</table>

**Inherited Risk**

- Dependencies: 968
- Vulnerabilities: 260
- Vulnerable Dependencies: 46

**Vulnerable Component Ratio**

- High Severity: 1
- Medium Severity: 229
- Low Severity: 36
How to solve this problem?

Sonatype Nexus

Technical aspects:
- Only available in Nexus PRO (commercial)
- Additional information: license analysis
- Nice reporting
How to get better?

Choose wisely!
How to get better?
Which dependency is more secure?

Stay critical!
How to solve this problem?

Monitoring

Apache Commons Lang

Apache Commons Lang, a package of Java utility classes for the classes that are in java.lang's hierarchy, or are considered to be so standard as to justify existence in java.lang.
How to get better?

**Action plan**

**Immediate: Inventory**
- Scan for libraries
- Create tracking sheet

**Short term: analyze**
- Purge unnecessary libraries
- Code review to check necessity
- Check signatures

**Tactical: Control**
- Centralize library control
- If possible: consider sanboxing

**Monitor**
- Manage your libraries
- Get security intelligence

Thank you!

Happy to discuss with you!
Automated checking for vulnerable components: https://dependabot.com/ (free, integrated in GitHub)

Automated merging after unit testing: https://github.com/marketplace/mergify (commercial solution, simple to script)
Excursion Container (Docker)

1. Gen. – Physical Server
2. Gen. – Virtual Servers
3. Gen. – Containerized Apps

App
OS
Hardware

App
App
App
OS
OS
OS
Hardware

App
OS
Hardware

App Sec.
Server Hardening
Supply Chain
Container Image

Contains custom code and foreign components
Excursion License

Which license suits best for my commercial project?

- **Permissive**
  - Examples:
    - MIT
    - BSD
    - Apache v2

- **Weak Copyleft**
  - Examples:
    - LGPL 2/3

- **(Strong) Copyleft**
  - Examples:
    - GPL 2/3
    - AGPLv3

Web hint: [https://choosealicense.com/](https://choosealicense.com/)
OWASP Dependency Track

- Project by Steve Springett
- Allows centralized tracking of dependency vulnerabilities
- Utilizes SBoM file (CycloneDX)

https://dependencytrack.org/
https://cyclonedx.org/