RESEARCH @ ECLIPSE FOUNDATION

Open Research Webinars

OW2

On-going research to fuel and enhance Eclipse Steady

Serena Elisa Ponta
SAP Security Research

SPARTA

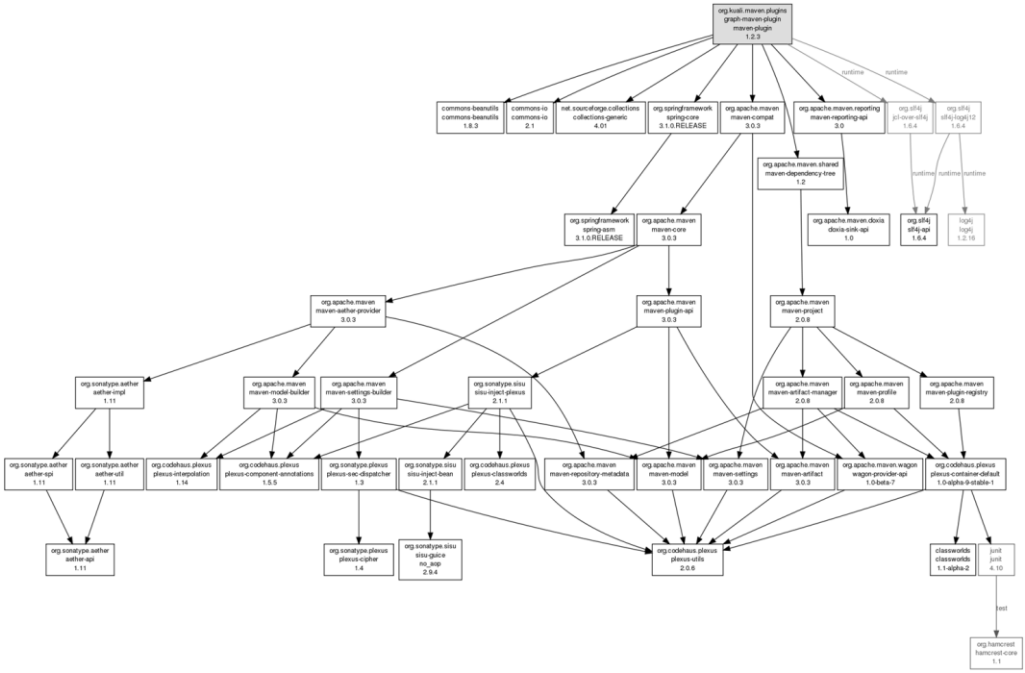
AssureMOSS

Software products embed open-source components

- 80% to 90% of software products on the include OSS components
- 80+% of the codebase of a typical Java application is open-source

Dependency Graph

- Direct/transitive dependencies
- Duplicates and version conflicts
- Automated build systems handle the complexity transparently



Heartbleed, Equifax…

Using components with known vulnerabilities:

- Included in OWASP Top 10 since 2013
- Root cause of major data breaches

Equifax confirmed that their high profile, high impact data breach was due to an exploit of a vulnerability in an open source component, **Apache Struts CVE-2017-5638**. **Apache Struts** is a mainstream web framework, widely used by Fortune 100 companies in education, government, financial services, retail and media.

Behind the Equifax Breach: A Deep Dive Into Apache Struts CVE-2017 ...
https://www.brighttalk.com/.../behind-the-equifax-breach-a-deep-dive-into-apache-struts...

Equifax's Data Breach Costs Hit $1.4 Billion

Massive 2017 Breach Continues to Bite the Credit Reporting Giant's Bottom Line

Mathew J. Schwartz (euroinfosec) · May 13, 2019

Twitter Facebook LinkedIn Credit Eligible Get Permission

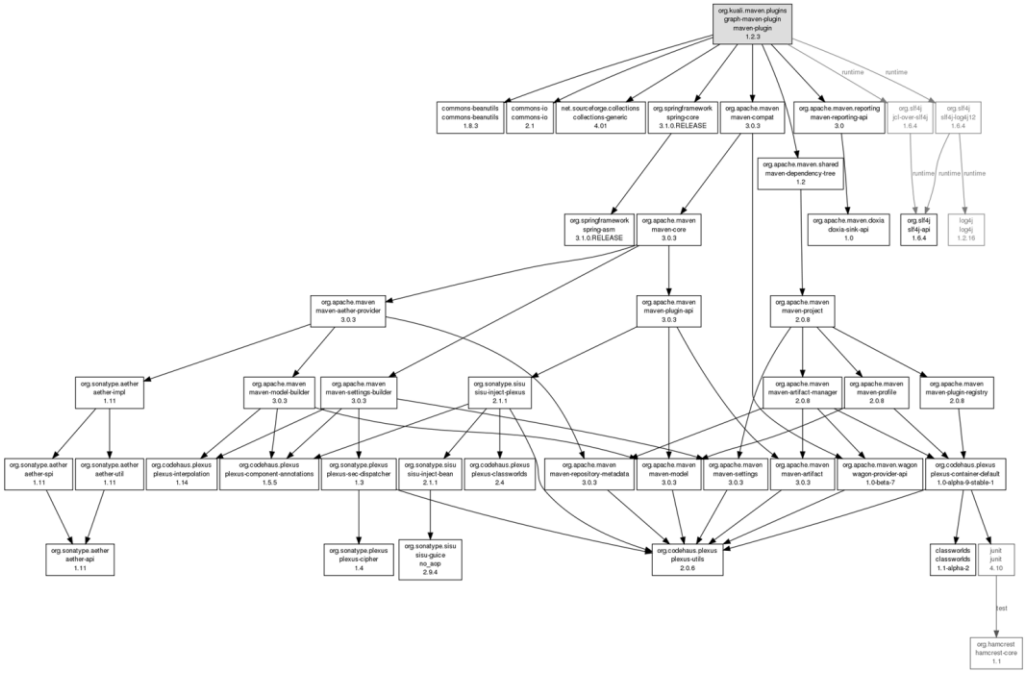
https://www.bankinfosecurity.com/equifaxs-data-breach-costs-hit-14-billion-a-12473

Log4Shell

- Apache Log4j is a widely used logging library in Java
- CVE-2021-44228 allows for remote code execution (RCE)
- Low attack complexity, no privileges required, complete compromise → CVSS 10
- Attack succeeds if strings with JNDI lookups ${jndi:…} are logged by apps depending on vulnerable versions of Log4j (2.0-beta9 to 2.14.1)
- Configuration settings can limit exposure and increase complexity (but not mitigate completely)
- Not only user-facing apps are affected (but any app that receives and logs untrusted input)
- Three other vulnerabilities have been found afterwards (CVE-2021-45046, 45105 and 44832)
- Latest non-vulnerable release is 2.17.1

Known vulnerabilities… Patch Exists!
Simply update?

- Depends on lifecycle phase and deployment model
- May include breaking changes
- Majority of vulnerabilities in transitive dependencies
- Re-bundles can also result in vulnerable apps (3233 artifacts on Maven Central contain the problematic Log4j class JndiLookup)



Open Source Vulnerability Scanners
Two Approaches

Metadata-based

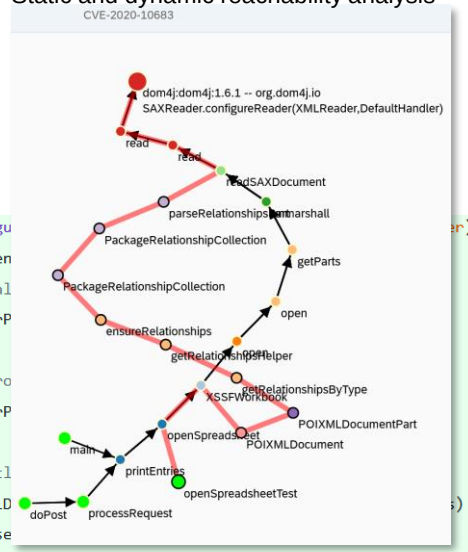
- Primarily rely on package names and versions, package digests, CPEs, etc.
- Example: OWASP Dependency Check (light-weight, maps against CVE/NVD)

Code-centric

- Detect the presence of code (no matter the package)
- Supports impact assessments (static and dynamic analyses), esp. important for later lifecycle phases and non-cloud
- Supports update metrics to avoid regressions
- Example: **Eclipse Steady** (heavy-weight, requires fix-commits)

https://eclipse.github.io/steady/

Static and dynamic reachability analysis



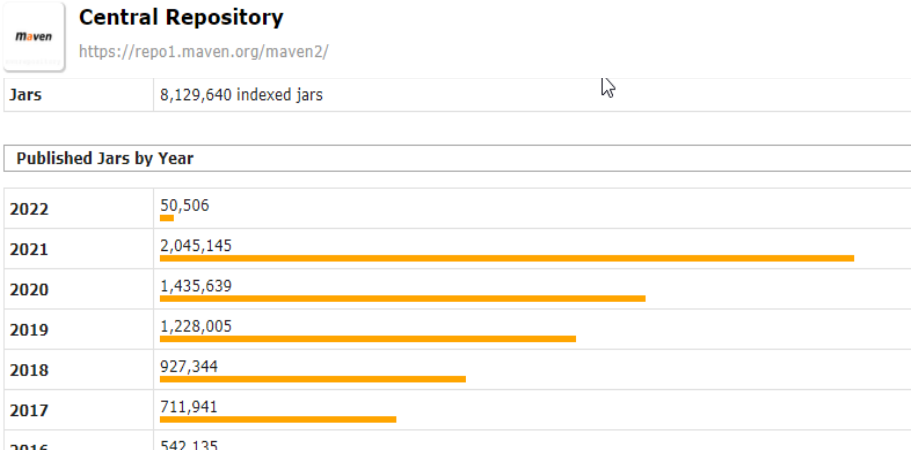
Fix-commit for CVE-2020-10683

Metadata-based
(Some) Limitations

- Short CVE descriptions and varying quality of referenced information
- Error-prone (2.3.5 and 2.3.6 were also affected)
- Coarse-granular reference of entire projects, ignoring reusable components and code (800+ artifact versions contain the resp. classes)
- CPE identifier != package identifier (30 search hits for “mojarra” on Maven Central don’t include `org.glassfish:javax.faces`)

CVE-2018-14371

The `getLocalePrefix` function in `ResourceManager.java` in Eclipse Mojarra before 2.3.5 is affected by Directory Traversal via the `loc` parameter. A remote attacker can download configuration files or Java bytecodes from applications.

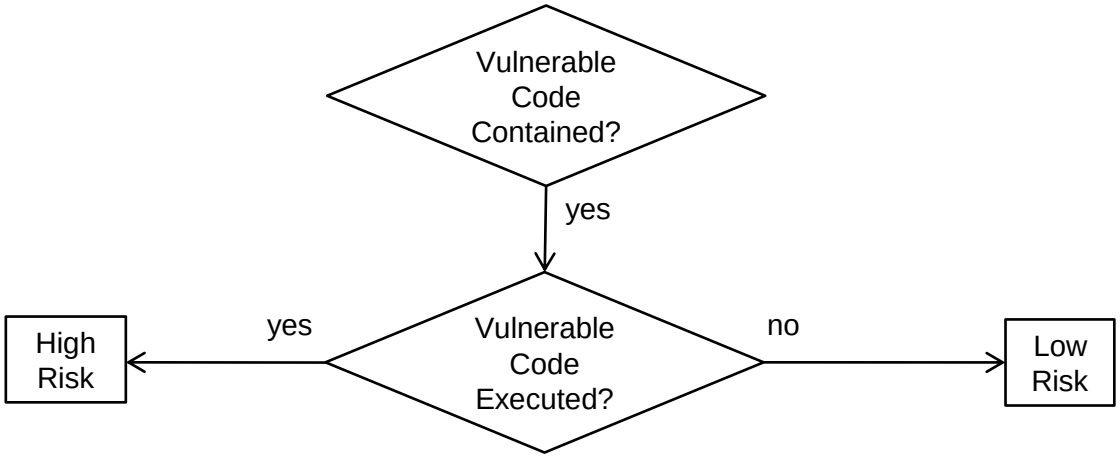


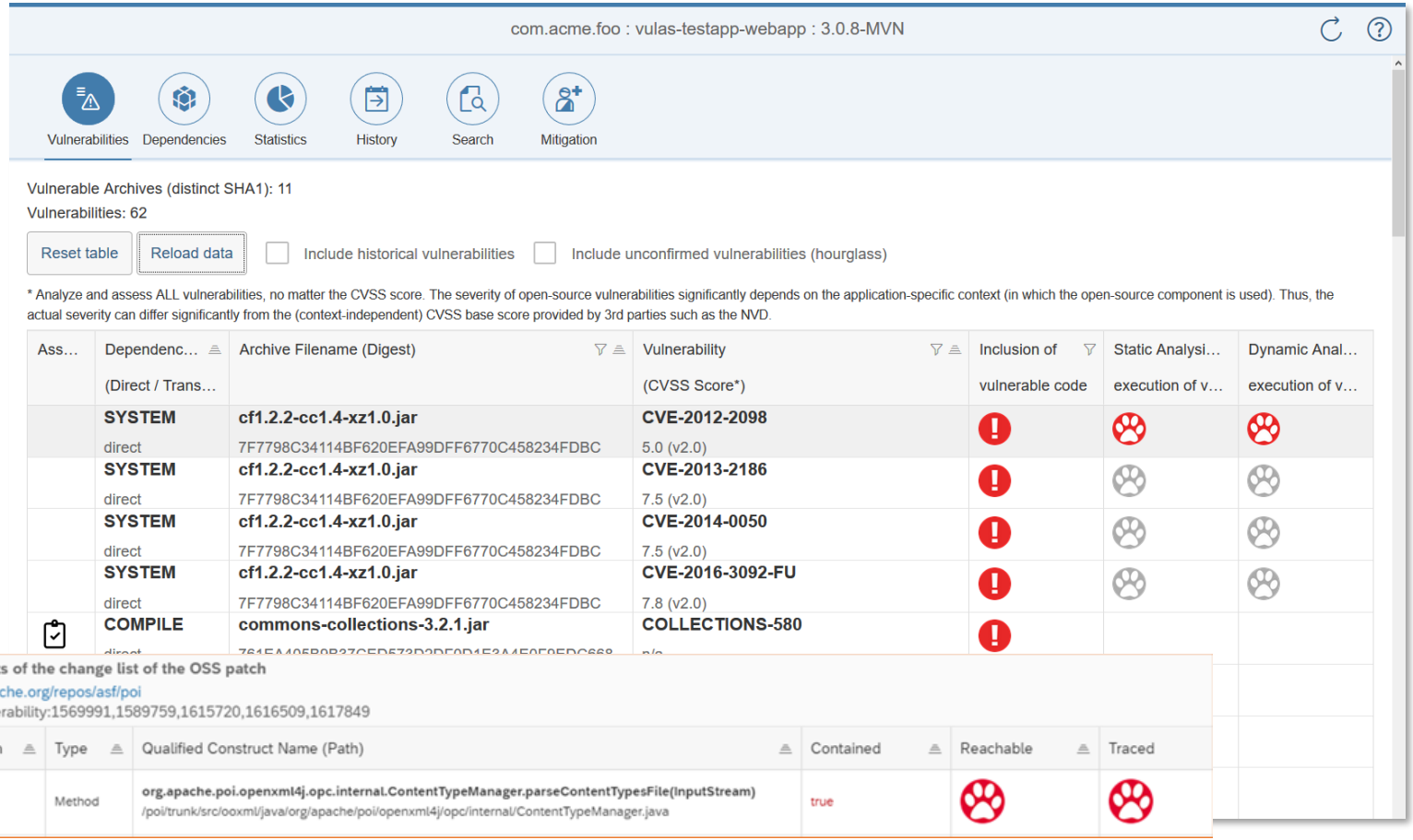
Eclipse Steady

Code-centric detection and application-specific assessment

Validate if vulnerable code is (1) contained and (2) executed by the application

- Applications typically include large pieces of OSS code where only a fraction of it is used
- Combination of static analysis (call graph construction) and dynamic analysis (test/runtime instrumentation)

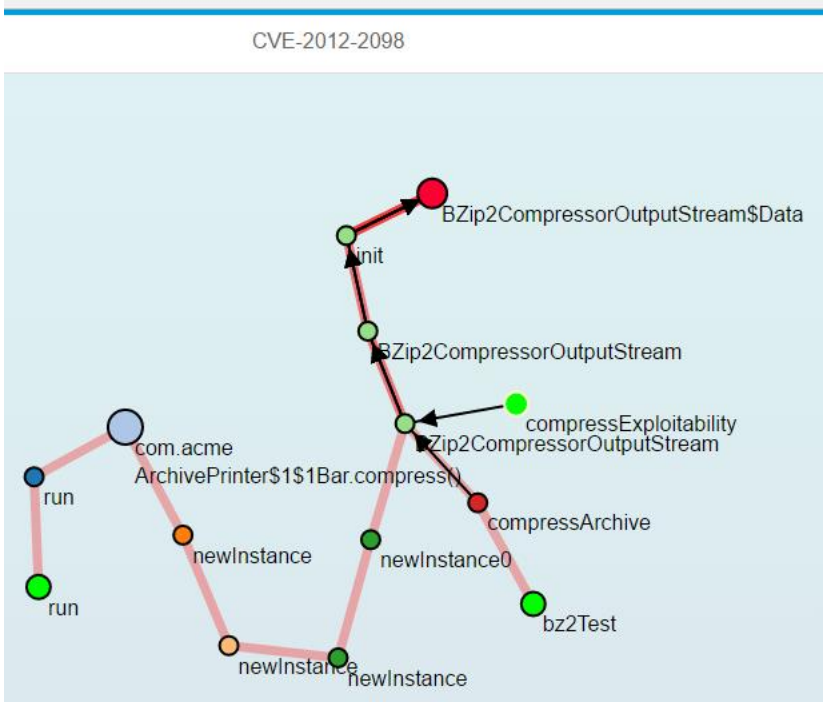




https://eclipse.github.io/steady/

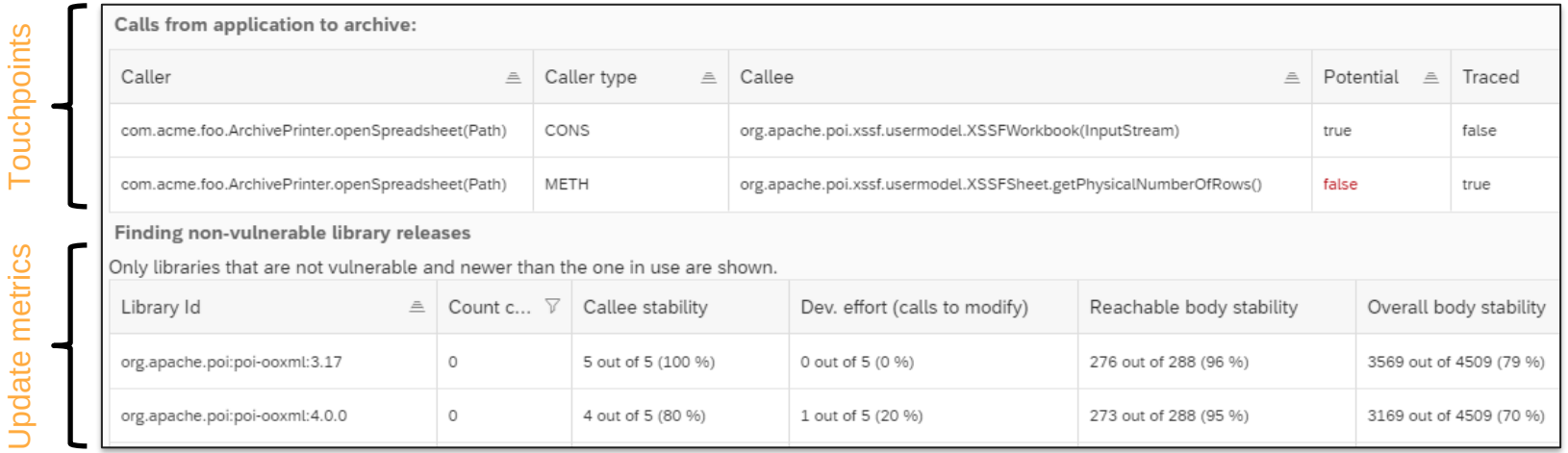
Complementarity of dynamic and static analysis

- Due to missing test case, dynamic analysis does not find path starting from ArchivePrinter.compressExploitability(Path,Path)
- Due to the use of reflection, static analysis does not find path starting from Thread.run()



https://eclipse.github.io/steady/

Mitigation options supported by reachability analyses



Touchpoints

Calls from application to archive:

Caller	Caller type	Callee	Potential	Traced
com.acme.foo.ArchivePrinter.openSpreadsheet(Path)	CONS	org.apache.poi.xssf.usermodel.XSSFWorkbook(InputStream)	true	false
com.acme.foo.ArchivePrinter.openSpreadsheet(Path)	METH	org.apache.poi.xssf.usermodel.XSSFSheet.getPhysicalNumberOfRows()	false	true

Update metrics

Finding non-vulnerable library releases

Only libraries that are not vulnerable and newer than the one in use are shown.

Library Id	Count c...	Callee stability	Dev. effort (calls to modify)	Reachable body stability	Overall body stability
org.apache.poi:poi-ooxml:3.17	0	5 out of 5 (100 %)	0 out of 5 (0 %)	276 out of 288 (96 %)	3569 out of 4509 (79 %)
org.apache.poi:poi-ooxml:4.0.0	0	4 out of 5 (80 %)	1 out of 5 (20 %)	273 out of 288 (95 %)	3169 out of 4509 (70 %)

- Exclude dependency
- Update (non-breaking)
- Fork and down-port security fix
- Implement application-specific safeguards

Vulnerability Data about Open-source Software Should Be Open Too!

Today

- Information about open source vulnerabilities is scattered
- Mining is labor-intense despite advances in AI-based commit classification
- Providers step-in (and compete) with proprietary databases

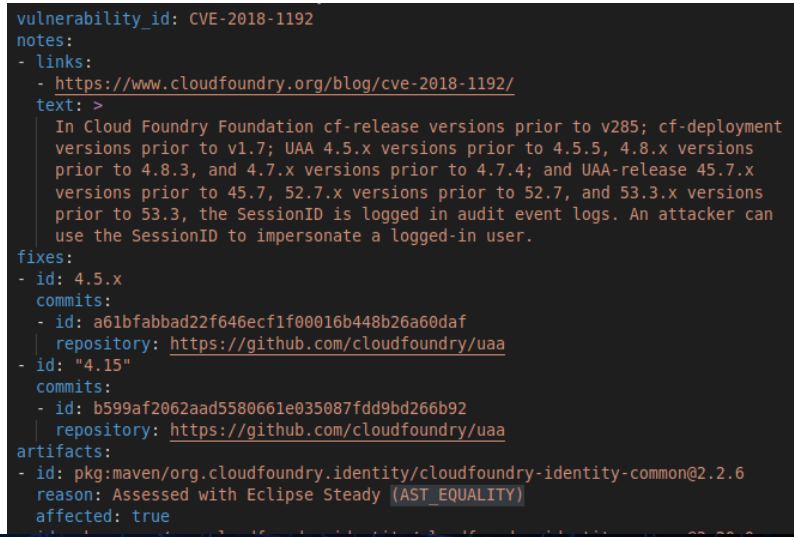
This does not scale, and has the paradoxical consequence that **data about open-source software is not open!**

github.com/SAP/project-kb

Open, **collaborative**, and **trustworthy** knowledge base of vulnerabilities (+fixes) that affect open-source software

Git repositories used to store vulnerability ***statements***

Plain-text data format, machine-readable and human-readable

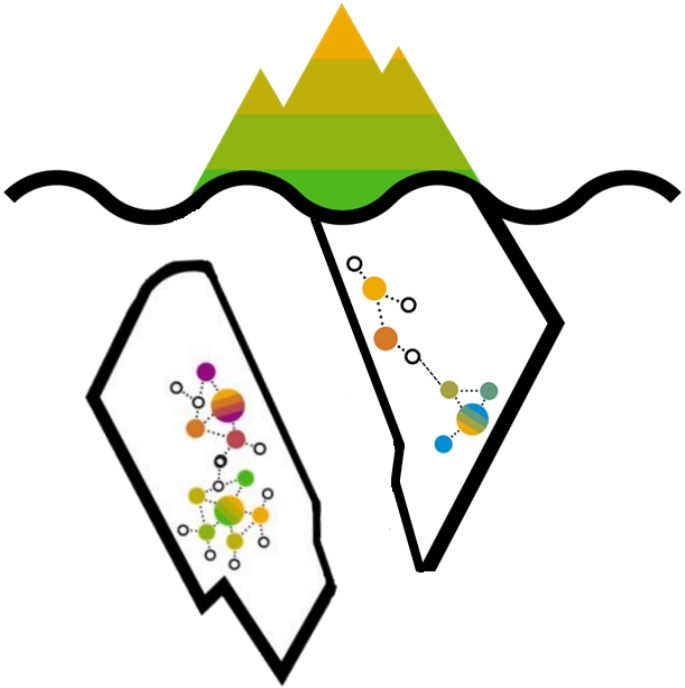


Tool-support

- **Create**, **aggregate** and **validate** statements
- **Find fixes** in open-source code repositories

Reducing the attack surface removing bloated code

- Unused by the application
- Potentially usable by attackers
- Needs maintenance



Case Study

Can existing debloating tools minimize the dependencies of an industrial grade Java application?

- 260 application classes, 62 test classes yielding 446 test cases
- 2725 compile dependency classes

Execution	Classes	Size (KB)	Test success	Vulnerable classes
Vanilla	2725	15033	446	1
DepClean	11	57,26	446	-
Maven Shade	12	57,63	446	-
ProGuard^m	1	4	446	-
ProGuard^c	11	57,26	446	-

Reduced bloated code containing a potential security vulnerability but did not handle a service loader definition

Ponta, S., et al.:The Used, the Bloated, and the Vulnerable: Reducing the Attack Surface of an Industrial Application (2021)

Conclusion

- Need for **precise analysis** techniques for effective vulnerability management
- Code-based approaches reduce FP and FN and support impact assessment
- Code-level information about vulnerabilities and their fixes is key
- Gathering and maintaining this information is best done in a collaborative fashion
- Open formats and tools to enable **publishing**, **sharing** and **aggregating vulnerability data** in an efficient, flexible, trustworthy fashion
- Reducing bloated code may dramatically reduce the attack surface of applications

Eclipse Steady & project "KB"

Dealing with vulnerabilities of open-source software

the open-source way

Links

https://github.com/eclipse/steady

https://eclipse.github.io/steady

https://github.com/SAP/project-kb

https://sap.github.io/project-kb

Acknowledgements

Sparta (EU-funded project)
https://www.sparta.eu/

AssureMOSS (EU-funded project)
https://assuremoss.eu/