



OWASP Dependency-Check

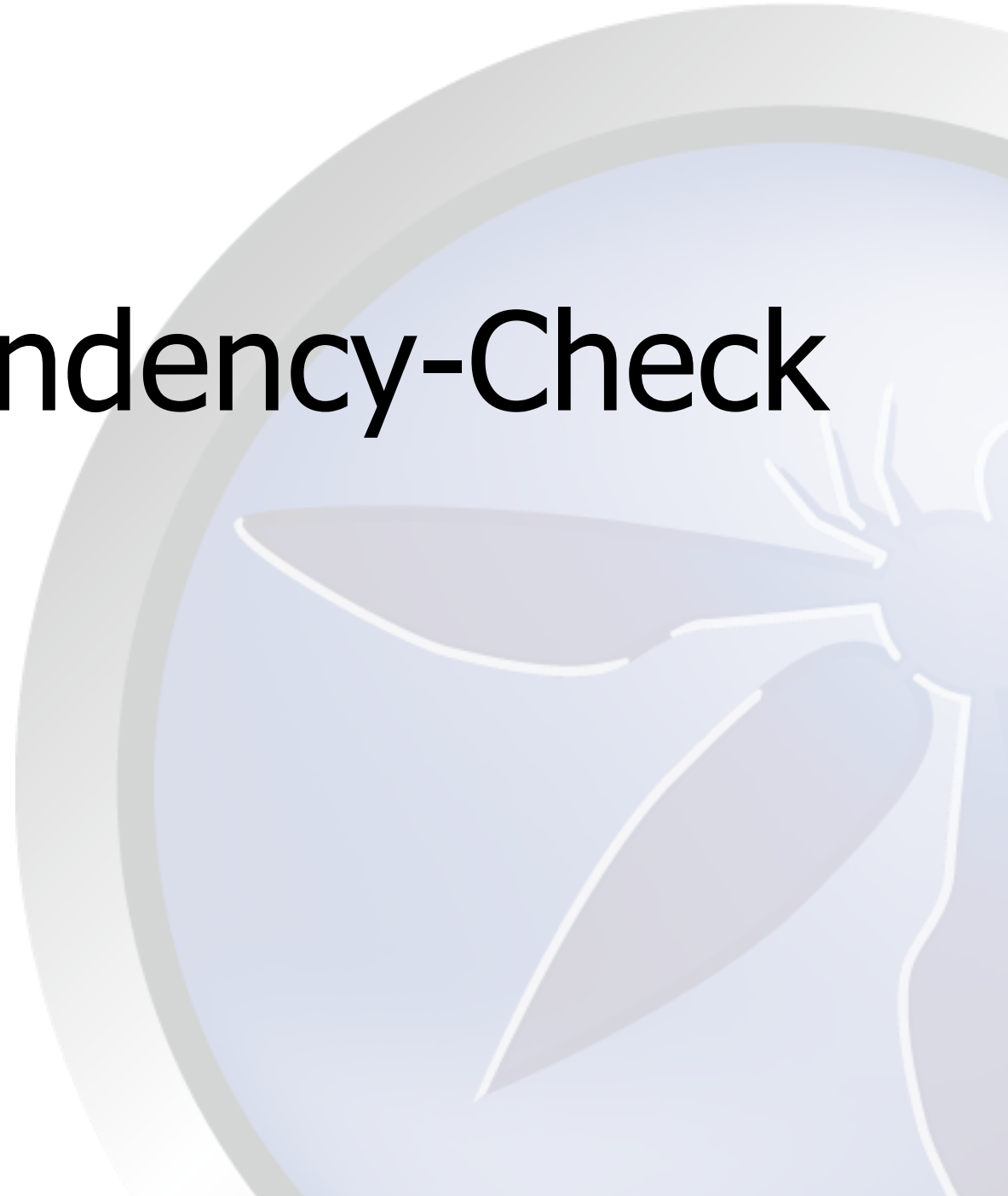
Jeremy Long

jeremy.long@owasp.org

Twitter: [@ctxt](#)

Steve Springett

steve.springett@owasp.org




Jeremy Long



- 10 years information security experience
- 10 years software development experience
- Senior Information Security Engineer at a large institution
- Northern Virginia OWASP Chapter board member
- Lead developer/architect for OWASP Depend

Steve Springett



- 19 years software development experience
- 4 years information security experience
- Principal application security engineer at  a business
- Provide direction, best practices & education
- Contributor to OWASP Dependency-Check

Vulnerabilities in 3rd Party Libraries

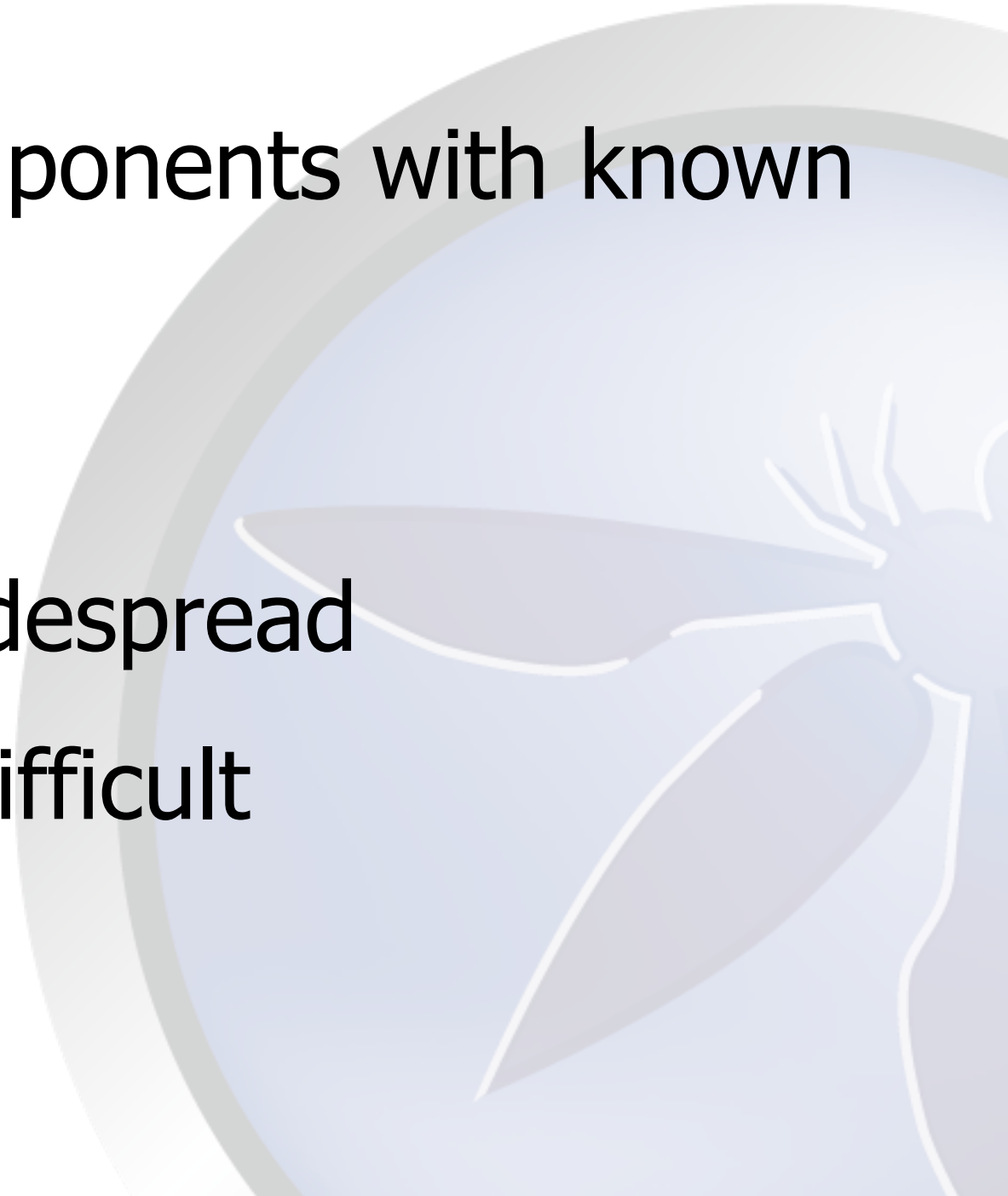
- 88% of code in today's applications come from libraries and frameworks
- 113 million downloads analyzed for the 31 most popular Java frameworks/libs
- 26% had known vulnerabilities
- Most vulnerabilities are undiscovered

OWASP Top Ten 2013

- A9 – Using components with known vulnerabilities

Prevalence: Widespread

Detectability: Difficult

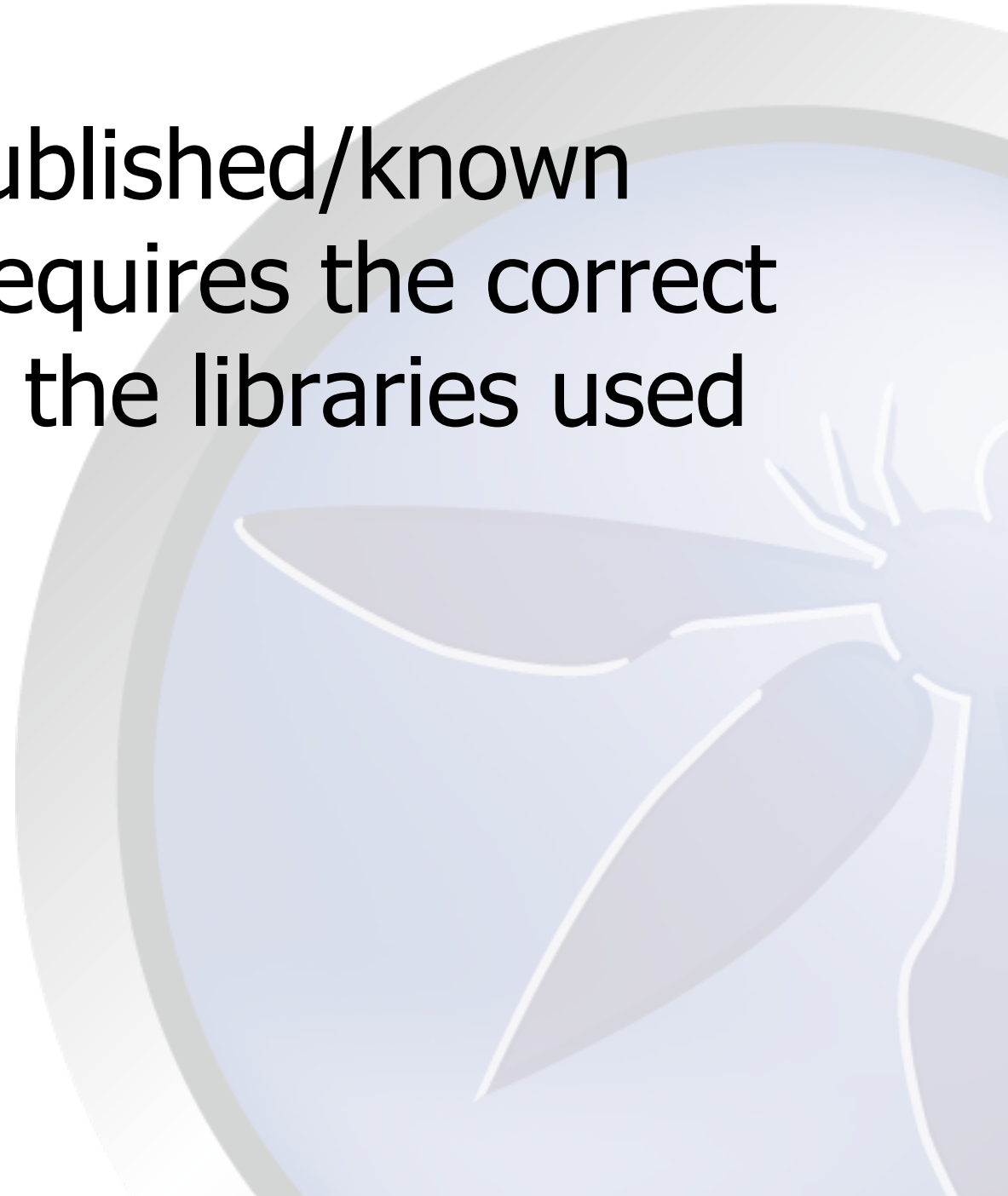


Dependency-Check

- Simple answer to the A9 problem
 - Identifies libraries and reports on known/published vulnerabilities
 - Currently limited to Java libraries
- Project Team:
 - Jeremy Long – lead developer/architect
 - Steve Springett - contributor

Library Identification

- Reporting on published/known vulnerabilities requires the correct identification of the libraries used



Problems w/ Library Identification

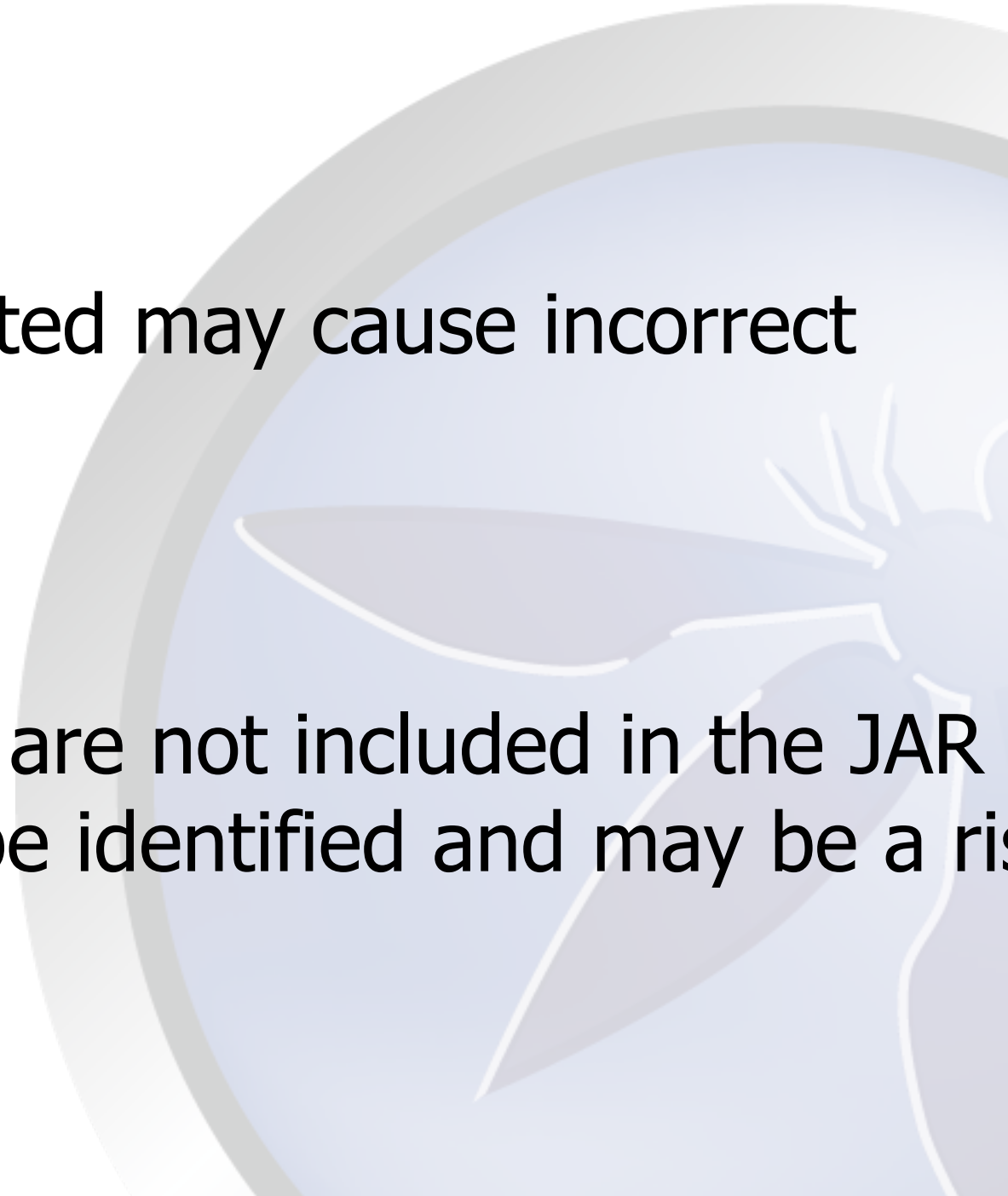
- No standard labeling mechanism for identification
- CPE identifiers are used in NVD CVE:
 - `cpe:/a:springsource:spring_framework:3.0.0`
 - `cpe:/a:vmware:springsource_spring_framework:3.0.0`
 - `cpe:/a:apache:struts:1.2.7`
 - `cpe:/a:apache:struts:2.1.2`
- File hashes could be used to aid in identification
 - Hash database must be maintained
 - Hashes may change if library is built from source

Library Identification:

Evidence Based Identification

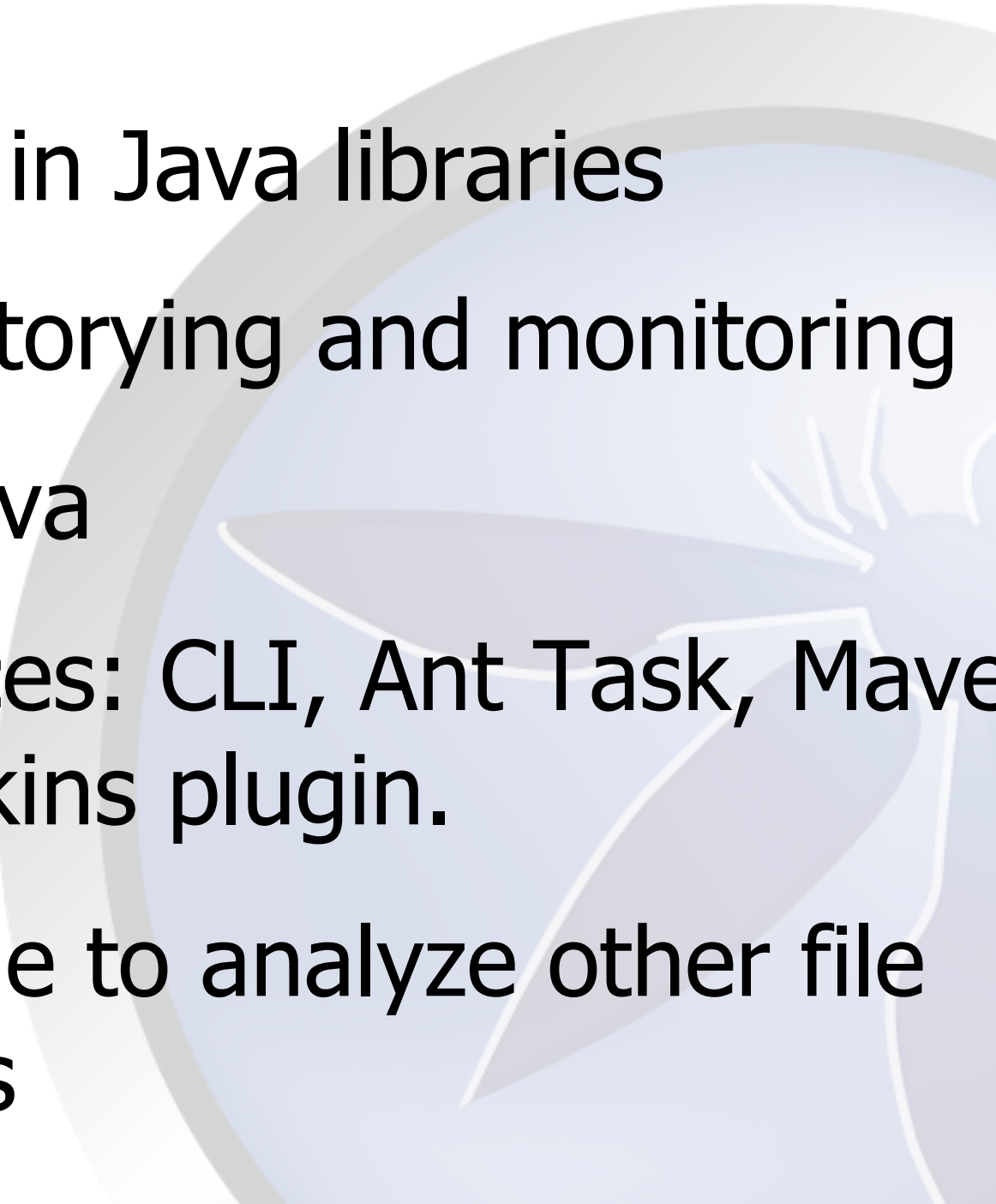
- Local copy of the NVD CVE is maintained
- Evidence collected is used to search the local database to identify the library and vulnerability
- Data extracted from libraries
 - File name, manifest, POM, package names, etc.
- Mapping of library to CPE/CVE not needed
 - Future enhancements may include a file hash analyzer – this is not currently available

Evidence Based Identification: Problems

- False Positives
 - Evidence extracted may cause incorrect identification
 - False Negatives
 - If key elements are not included in the JAR library will not be identified and may be a risk
- 

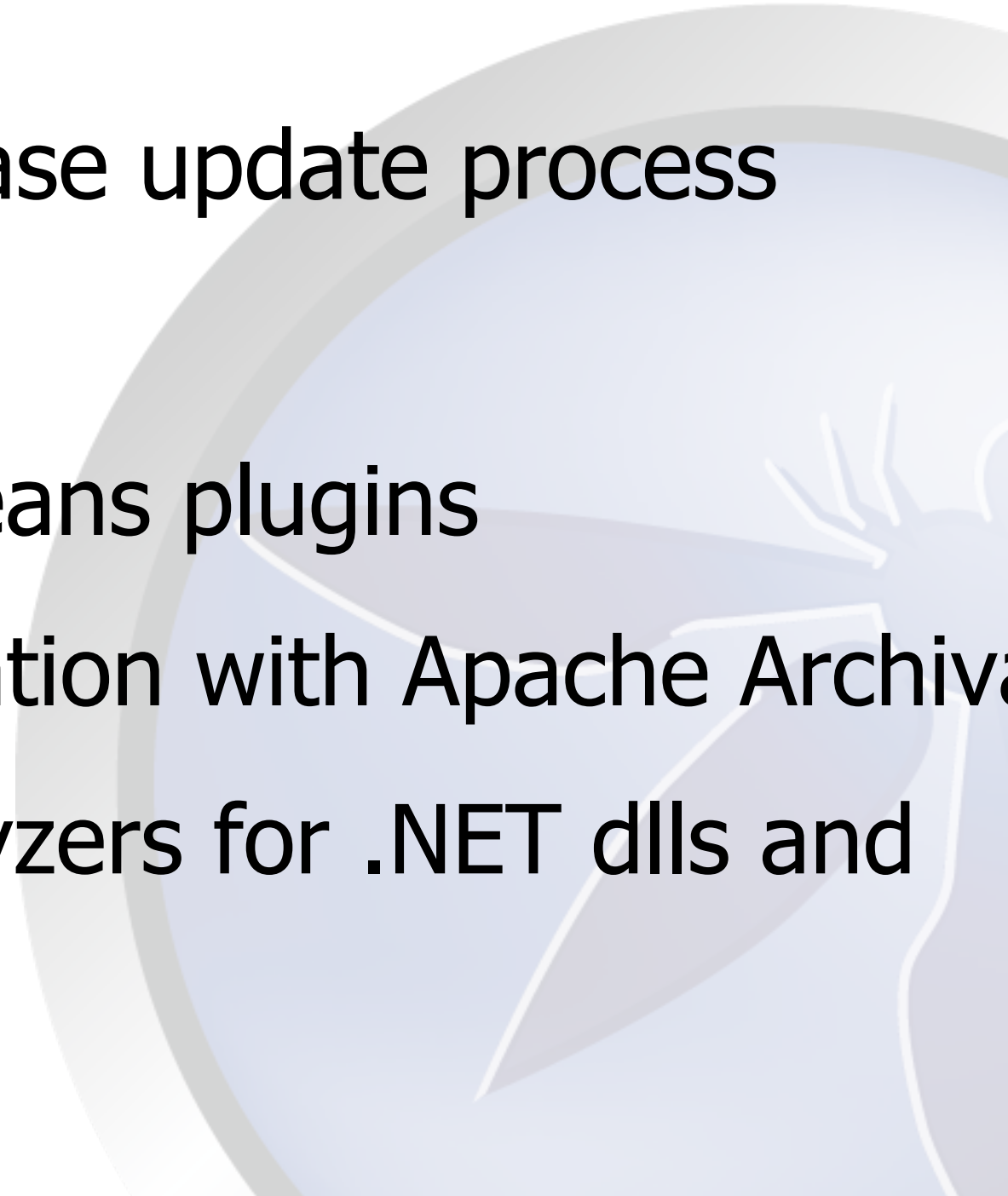
Dependency-Check:

Current State

- Identifies CVE's in Java libraries
 - Useful for inventorying and monitoring
 - Developed in Java
 - Current Interfaces: CLI, Ant Task, Maven Plugin, and Jenkins plugin.
 - Easily extendable to analyze other file types/languages
- 

Dependency-Check:

Roadmap

- Improve database update process
 - Sonar Plugin
 - Eclipse & Netbeans plugins
 - Possible integration with Apache Archiva
 - Additional analyzers for .NET dlls and JavaScript
- 

Dependency-Check

- License - GNU GPL v3 license
- Important Links:

OWASP Project Page:

https://www.owasp.org/index.php/OWASP_Dependency

SCM:

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

Mailing List:

Subscribe: dependency-check+subscribe@googlegroups.com

Post: dependency-check@googlegroups.com

DEMO

