OWASP Dependency-Check

Jeremy Long
jeremy.long@owasp.org
twitter: @ctxt

Jeremy Long

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

Steve Springett

- 19 years software development experience
- 4 years information security experience
- Principal application security engineer at axway
- 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

Jeff Williams & Arshan Dabirsiaghi
The Unfortunate Reality of Insecure Libraries
Aspect Security (March 2012)

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 & .NET libraries
- Project Team:
  - Jeremy Long – lead developer/architect
  - Will Stranathan - contributor
  - 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 identifying
- 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
  - Components bundled via one jar, maven-shade-plugin, etc.

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 vulnerabilities
- 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 the library will not be identified and may be a risk

False Positives

- Suppression Filters – added in 1.0.7
  - Provides a simple way to remove false positives

```
<?xml version="1.0" encoding="UTF-8"?>
<suppressions
xmlns="https://www.owasp.org/index.php/OWASP_Dependency_Check_Suppr
ession">
   <suppress>
      <notes><![CDATA[
      file name: spring-core-3.0.0.RELEASE.jar
      ]]></notes>
      <sha1>4F268922155FF53FB7B28AECA24FB28D5A439D95</sha1>
      <cpe>cpe:/a:vmware:springsource_spring_framework:3.0.0</cpe>
   </suppress>
</suppressions>
```

Dependency-Check:
Current State

- Identifies CVE's in Java and .NET 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

- Sonar Plugin
- IDE plugins (such as Netbeans)
- Possible integration with Apache Archiva
- Additional analyzers for JavaScript (jquery, Node libraries, etc.)

Dependency-Check

- License – Apache 2.0
- Important Links:

OWASP Project Page:
https://www.owasp.org/index.php/OWASP_Dependency_Check

SCM:
https://github.com/jeremylong/DependencyCheck

Mailing List:
Subscribe: dependency-check+subscribe@googlegroups.com
Post: dependency-check@googlegroups.com