OWASP
Open Web Application Security Project

Depending on Vulnerable Libraries

September 21st, 2016

A bit about me…

- Jeremy Long
  - 15 years information security experience
  - 10 years software development experience
  - SAST enthusiast
  - Contributor to the OWASP Java Encoder Project
  - Lead developer/architect for OWASP dependency-check

  - @ctxt / jeremy.long@owasp.org

What are we going to talk about?

- Why should we care?
- Patching programs
- What application teams can do
- Deep dive into dependency-check
- Usage scenarios
- Governance

Why should we care?

- **CVE-2016-5000 -** Apache POI Information Disclosure via External Entity Expansion (XXE)
- **CVE-2016-4216 -** Adobe XMP Toolkit for Java Information Disclosure via External Entity Expansion (XXE)
- **CVE-2016-3081 -** Remote code execution vulnerability in Apache Struts when dynamic method invocation is enabled
- **CVE-2015-8103 -** Remote code execution vulnerability in Jenkins remoting; related to the Apache commons-collections

Black Duck - Open Source Security Analysis

- The State of Open Source Security in Commercial Applications
  - https://info.blackducksoftware.com/rs/872-OLS-526/images/OSSAReportFINAL.pdf
- 95% of applications include open source
- 67% of applications contained open source vulnerabilities
- Average age of open source vulnerability identified: 1,894 days

OWASP Top 10 2013

- Most critical web application risks
- A9 – Using components with known vulnerabilities
  - Prevalence: Widespread
  - Detectability: Difficult
- Difficult for 4 reasons
  - Awareness
  - Visibility
  - Lack of tooling in 2012/2013

Patching Programs

- Generally do not cover application dependencies
  - Lack of awareness of 3rd party or FOSS application dependencies
  - Patching teams cannot push patches
- Patching application dependencies requires
  - Possible code changes
  - Full regression testing

Enter OWASP dependency-check

- Project stated December 2011 (first published in 2012)
- Performs Software Composition Analysis
  - Reports known vulnerabilities
- Easy solution to the OWASP 2013 Top 10 A9 Using components with known vulnerabilities
- Works as:
  - Maven Plugin
  - Gradle Plugin
  - Jenkins Plugin
  - SBT Plugin
  - Ant Task
  - Command Line

Language/Technology Support

- Fully supported: Java & .NET
- Experimental Analyzers:
  - CocoaPods
  - Swift Package Manager
  - Python
  - PHP (composer)
  - Node.js
  - Ruby

OWASP dependency-check

HOW DOES IT WORK?

Vulnerability Data Source

- National Vulnerability Database (NVD)
  - https://nvd.nist.gov
- Contains a listing of Common Vulnerability and Exposures (CVE)
- Each CVE entry contains
  - A description of the vulnerability or exposure
  - A Common Vulnerability Scoring System (CVSS) score
  - A list of the affected platforms identified by their Common Platform Enumeration (CPE)

Library Identification

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

Library Identification Problems

- Development & Security use different identifiers
- Development (GAV coordinates):
  - org.springframework:spring-core:3.2.0.RELEASE
- Security uses Common Platform Enumeration (CPE):
  - cpe:/a:springsource:spring_framework:3.2.0
  - cpe:/a:pivotal:spring_framework:3.2.0
  - cpe:/a:pivotal_software:spring_framework:3.2.0
- No publicly available database exists to map between the two

Evidence Based Identification

- Evidence is extracted from dependencies
  - File name, manifest, POM, package names, etc.
  - Evidence is grouped into Vendor, Product, and Version collections
- Local copy of the NVD CVE is maintained
- Lucene Index of the CPE information is created
- Evidence collected is used to search the index and identify the library by CPE

Evidence Based Identification Issues

- False Positives
  - Evidence extracted may cause incorrect identification
- False Negatives
  - If key elements are not included in the dependency (e.g. jar, dll) the library will not be identified and may result in un-reported risk

Dealing with False Positives

- Invalid dependency identification can be resolved using a suppression file:

```
<suppress>
   <notes><![CDATA[
   This suppresses false positives identified on spring security.
   ]]></notes>
   <gav regex="true">org\.springframework\.security:spring.*</gav>
   <cpe>cpe:/a:mod_security:mod_security</cpe>
   <cpe>cpe:/a:springsource:spring_framework</cpe>
   <cpe>cpe:/a:vmware:springsource_spring_framework</cpe>
</suppress>
```

OWASP dependency-check

USING DEPENDENCY-CHECK

Onboarding an Application

- Basic steps
  - Configure plugin
    - Proxy configuration
  - Run initial scan
  - Create and configure a suppression file (if needed)
  - Plan the upgrade for identified vulnerable components

OWASP dependency-check

DEMO

Use Cases for dependency-check

- Prove the existence of the problem
- Baseline test when conducting POCs with commercial solutions
- OWASP dependency-check is used as the primary tool to identify known vulnerable components

Enterprise Deployments

- Use a centralized database to maintain the local copy of the NVD
  - Single instance of dependency-check used to update
  - Scanning instances do not need to update
- Use an internal Nexus instead of Maven Central
- Run dependency-check within their CI
- Continuous monitoring/reporting using **OWASP dependency-check sonar plugin**, OWASP dependency-track, or ThreadFix

Vulnerable Dependencies as Code Quality

- Fail a build if known vulnerabilities are detected
  - Jenkins, gradle, maven, ant plugins
- Put security into your code quality metrics
  - OWASP dependency-check sonar plugin

Governance

- Known vulnerable dependencies are only one part of the software composition problem
- Organizations should:
  – Control what dependencies are allowed
    - Cleared by architecture, legal, and security reviews
    - Must be easy/quick to engage the governance process

OWASP dependency-check

QUESTIONS?

More Information

- OWASP dependency-check
  - http://jeremylong.github.io/DependencyCheck/
- OWASP dependency-track
  - https://github.com/stevespringett/dependency-track
- OWASP dependency-check-sonar-plugin
  - https://github.com/stevespringett/dependency-check-sonar-plugin

More Information

- Related Projects
  - Ruby Bundler-Audit
  - Retire.js
  - Node Security Project