Getting your Ducks in a Row

Contents

1 Getting your Ducks in a Row

An Introduction to Managing Components in your Software Supply Chain

Manfred Moser - #simpligility - www.simpligility.com

Sonatype - www.sonatype.com

2 Data

- 2011-2014 Open Source Software Development Survey
  - >11k responses
- 2015 State of the Software Supply Chain Report
  - >110k organizations,
- Central Repository Usage Statistics
  - Largest binary repository
  - >17 Billon Downloads in 2014
- Sonatype Nexus Usage Statistics
  - >50k installations

3 Development Today

- Uses components as building blocks
- 80-90% of typical application
- average app from application health check has approx 100 components
- Mostly open source

4 So What is a Component?

- All the stuff you use to create your applications
- The plumbing you don't want to write yourself
- Logging, IoC, persistence layer, ORM, . . .
- And that you get easily by declaring dependencies with Maven, Gradle, nuget, npm. . .
- Comprises 80-90% of your application

Warning
And you are responsible for **all** of them.

---

5 So What is a Component?

- But also all the parts you create and use
  - Including the application deployment packages
- JARs, WARs, EARs..
  - but also rpm and npm packages or tar.gz files
- what is a component
- percentage of app

6 Applications Use Components

lots of them

7 Software Supply Chain

Like manufacturing

8 Software Supply Chain Management

Components are integrated part of Software Development Life Cycle (SDLC)
It is an endless circle of activities:

- Research what components to use
- Implement usage of components in development
- Check component usage in QA and release process
- Monitor applications and components used in production
- Go back to the start and change versions or entire components

Warning
It only ends when any production deployments are decommissioned and **not** when development stops.

9 Supply Chain

- Better and fewer suppliers
- Higher quality parts
- Improved visibility and traceability

10 Step: Control Component Source

and improve for performance and stability

use a repository manager

in your own datacenter

95% of downloads from Central Repository are done via build tools and such - inefficient

only 5% via repo manager

11 Component Management

Supply change management for components

- Security and authentication
- Repositories and repository targets
- Component information

→ Set the stage for first repository and component management

12 Sonatype Nexus as Central Hub

→ Nexus is a key component of your enterprise development infrastructure

13 Component Management with Nexus OSS

- Used by 66% of repository manager users
- 18 % of respondents to component survey use NO repository manager
- Internal caching and storing of components → enables collaboration
- Reduced dependency on external repositories
- One component storage location for backup, audit, control…
- Highly performant
- Reduced bandwidth usage and costs
- Efficient search
- Repository Health Check
- Some meta data
- Formats include Maven, NuGet, NPM, site, Yum and JRuby/Gems

→ Way better than manual management or ignoring the need

Important
Yet easy to implement…

14 Step: Know Your Components and Suppliers

tbd

E.g. Central Repository:

- 985k OSS components
- 100k suppliers (OSS projects)
- 11 Million OSS users
- annual downloads - 17B in 2014 central (0.5B in 2007 - huge growth), 15B npmjs.org, 5B rubygems (since inception), nuget gallery (300million)
- approx 1000 new components added to central daily
- How many open source projects used on average per large organization?

Comparison - e.g. Toyota manages about 800 suppliers, F

15 Step: Know Component Security Characteristics

tbd

Average org 250k different component downloads per year 15k components with known vulnerabilities downloaded some of them probably are used in production apps

46 million vulnerable components downloaded in 2014 from Central

16% must prove they are not using known security vulnerabilities

New vulnerabilities found regularly, new releases all the time - yet 63 % do NOT monitor for changes in vulnerability data

How would you know? A PITA to find out

approx 50K components in Central have known security vulnerabilities

OWASP dependency check

Show in Nexus Pro

16 Step: Understand License Implications

63 % have incomplete view of license risk from components

Only 32 % examine all open source components for license compliance

58 % say they are no concerned about license risk

approx 280k components in Central have restrictive licenses tbd

17 Step: Know Their Dependencies

tbd

dependency:tree

```
M2e view
```

```
crack open the war and look

sha checksum search

can you produce a BOM?
```

18 Step: Upgrade Often

tbd

19 Step: Avoid Duplication

- multiple logging frameworks

20 Step: Reduce Their Numbers

- KISS
- less complexity
- less learning effort
- Application Health Check analysis of 1500+ applications
- 106 components
- 24 with known vulnerabilities
- 9 with restrictive licenses

21 Step: Reduce Different Versions Used

- across enterprise
- e.g. Google mandates versions of libraries use, only one or two in most cases

22 Step: Reduce Vendors

tbd

23 Step: Select Projects and Vendor

- large vs small open source project
- active vs inactive in terms of commits
- commercial company backing or not
- Communicate with vendors!

24 Step: Upgrade Often

- just like devops - release often
- reduces complexity of updates
- open source projects work on master - latest == best, nearly always, avoid issues easily
- especially regarding security issues, backports are rare!
- sometimes you will get burned with regressions
- Typical component 3-4 releases per year

Average org downloads from Central - top 100 components, 27 are outdated and newer versions exists

25 Step: Communicate with Supplier

- most often open source project
- report bugs, feature requests
- help with documentation
- be present and support (mailing lists, forums, issue tracker, IRC, stackoverflow)

26 Next Steps

- Define policies
  - No components older than 5 years
  - No components with know security vulnerabilities of score > 8
  - No GPL licensed components
- Only 56 % have policies
- Of these only 68% follow policy
- Often manual, slow
- But 78% say they have never banned a component
- Things dont add up..

27 Next Steps

- Add tools to automate the process
- Configure tools with policies

28 Aim of Tools

1. Empower developers with the right information at the right time
2. Design frictionless, automated approach for continuous DevOps processes
3. Create, manage and monitor component bill of materials for each application

29 Bill of Material

- Tracking productions applications BOM
  - 40% including dependencies
  - 23% NOT including dependencies

30 Example Tools

lots of them out there for different stacks, examples to follow

31 Maven

- Dependency Plugin
- Dependency Management
- BOM POM file
- M2e - effective POM view, dependency view

32 OWASP Dependency Check

tbd

33 Sonatype Nexus Pro

tbd

34 Nexus Pro+

In a nutshell:

- Configurable component policies - **very** powerful
- Managed on the Sonatype CLM server
- Tied into Nexus staging

Tip
Demo time!

---

35 Sonatype Nexus Lifecycle

Expands Nexus Pro+

- Manual analysis via web interface upload
- Eclipse IDE integration
- Continuous Integration Server Jenkins/Hudson/Bamboo support
- SonarQube support
- Command line scanning

36 Resources

- 2014 Open Source Software Development Survey Results
- 2015 State of the Software Supply Chain Report
- Sonatype slides
- The Nexus Community
- www.sonatype.com
- Repository Management with Nexus
- Application Health Check
- modulecounts.com

37 The End

- Questions
- Remarks
- Discussion
- Slides - OSCON site or email manfred@sonatype.com now

38 Disclaimers

Image sources:

- Ducks in a Row from wikimedia