Hidden SPEED BUMPS on the Road to “Continuous”

Based on the Sonatype 2015 State of the Software Supply Chain Report

If you build software, you rely on a SOFTWARE SUPPLY CHAIN ...

You've got ...

SUPPLIERS
Open Source Projects

You've got ...

PARTS
Open Source Components & Warehouses

You've got ...

MANUFACTURERS
Software Development Teams

You've got ...

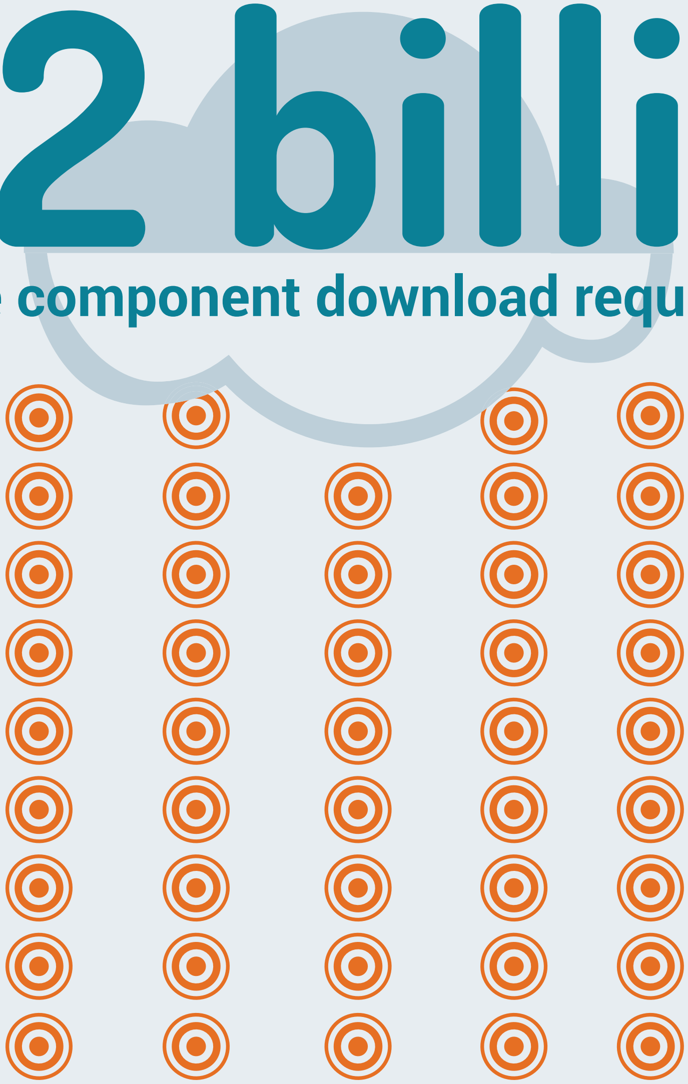
FINISHED GOODS
Software Applications

... in a BIG way!

There are 11 million developers globally.[1] We saw

17.2 billion

open source component download requests in 2014.[2]



And volume introduces hidden ***complexity*** ...

[1] Source: IDC 2014 Worldwide Software Developer and ICT-Skilled Worker Estimates

[2] Source: Analysis of the Central Repository, the world's largest repository of java components.

COMPLEXITY is the enemy of SPEED.

We clutch onto manual approaches that simply don’t work. We have automated so many aspects of software development, EXCEPT the quality, integrity, and traceability of components.

We don’t have effective visibility or controls

We use old & vulnerable component versions

We create technical debt & unplanned work

What if ...

manufacturers built cars the way we build software?

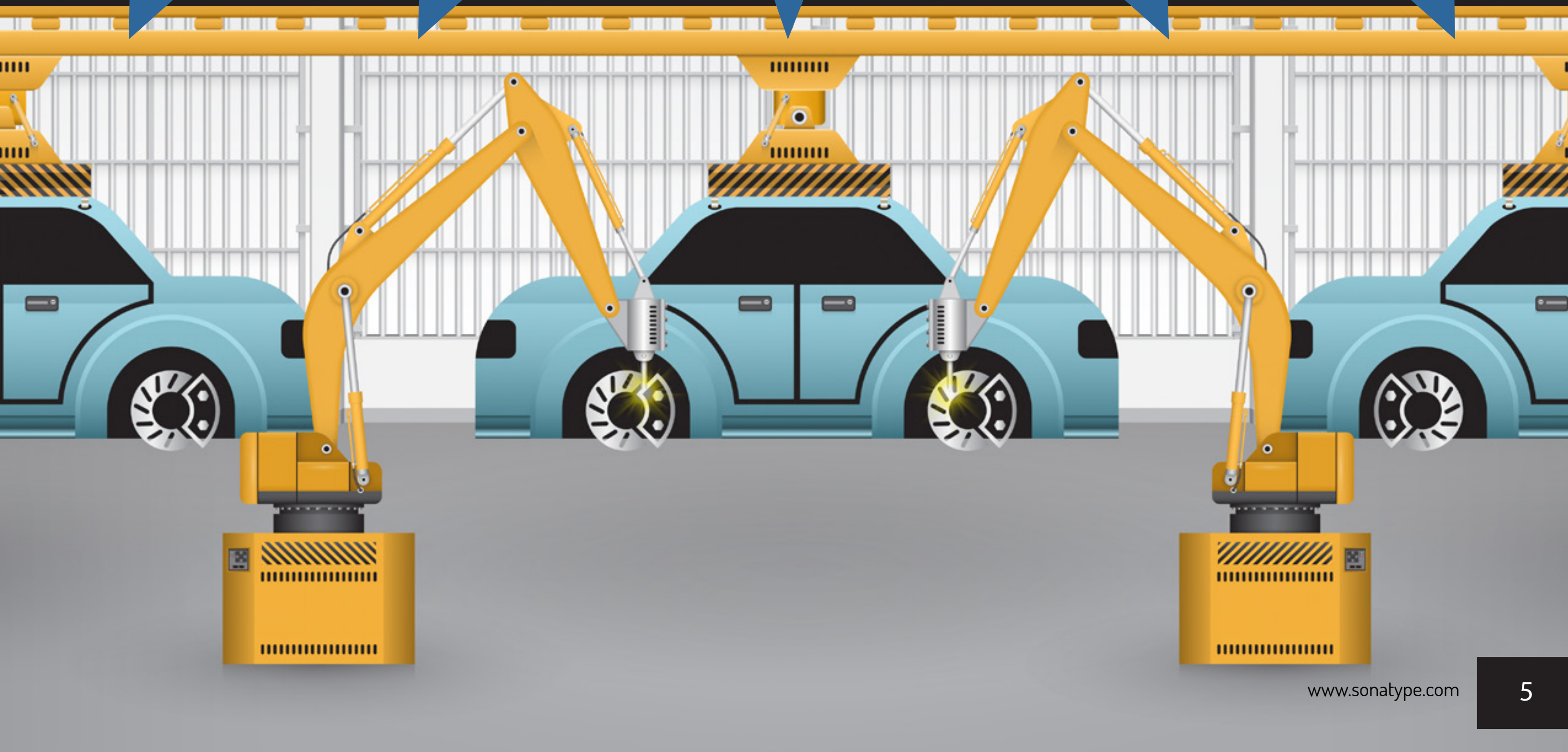
They could choose **any supplier** they want for any given part, regardless of quality.

Any part can be chosen even if it is outdated or known to be unsafe.

There is **no inventory** of the parts that were used, or where.

Since there is no visibility, it is very **slow and costly to recall** a part.

There is **no quality control** or consistency from car to car.



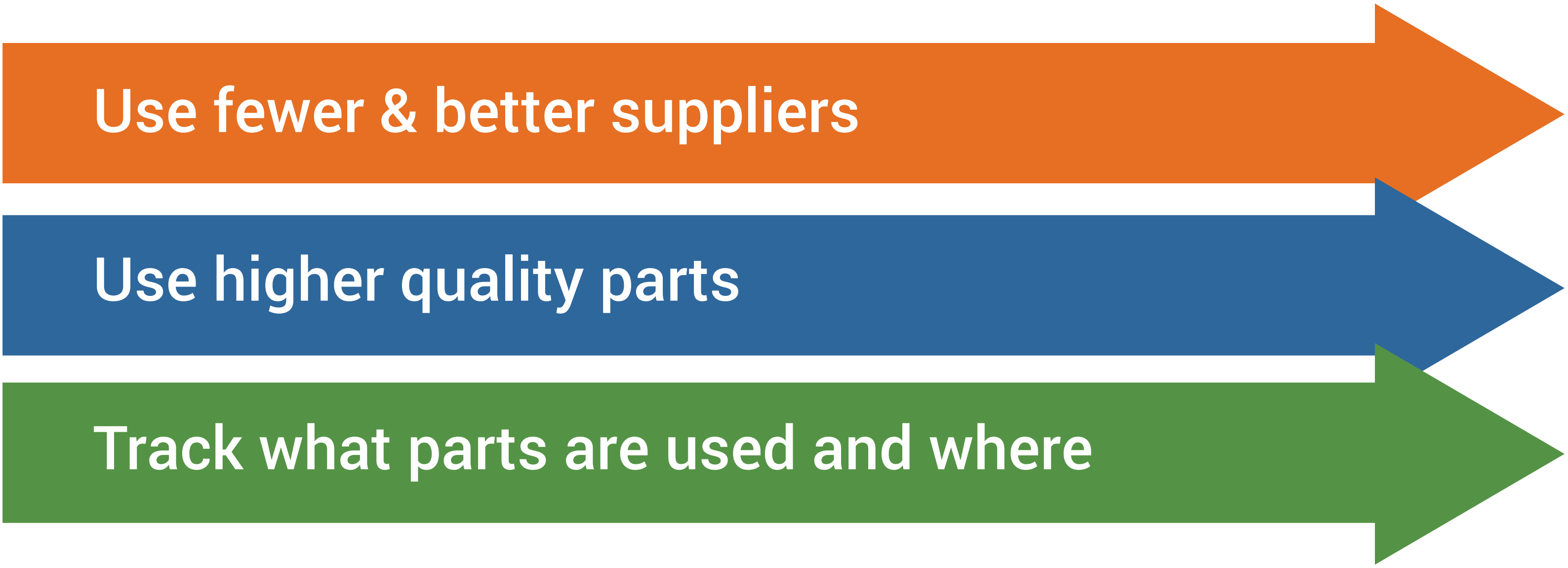
MODERN SOCIETY wouldn’t accept products built this way.

We wouldn’t buy cars built this way. We wouldn’t accept medical devices or food products or anything built this way.

So, what about our **SOFTWARE?**

How did TRADITIONAL SUPPLY CHAINS solve this problem?

Three basic principals CHANGED EVERYTHING:



Use fewer & better suppliers

Use higher quality parts

Track what parts are used and where

Every major transformation in human history started with a new idea that challenged the status quo.

To achieve the next leap in development efficiency, we MUST challenge our assumptions and find new approaches. Our current practices simply are not sustainable.

So, to expose these hidden complexities, we did a deep dive into the Software Supply Chains of

106,000

organizations and we learned ...

THE BEST
deliver better software, even faster.

Increased speed

by automating their software supply chain

Improved quality

by choosing fewer, better open source suppliers

Less unplanned work

by avoiding known vulnerable components

Faster remediation

by continuously monitoring applications for new defects

All together, productivity increases 15 - 40%!

THE REST …
don’t.

ELECTIVE RISK
TECHNICAL DEBT
COSTLY MAINTENANCE
SLOW MEAN-TIME-TO-REPAIR

“Just as in manufacturing, the effective management of our supply chains will create **WINNERS** and **LOSERS**”

Gene Kim, co-author of “The Phoenix Project:
A Novel About IT, DevOps, and Helping Your
Business Win” and upcoming “DevOps Cookbook”

Let’s take a closer look at the eye-opening STATS of **GOOD** and **NOT SO GOOD** software supply chain practices.

How well do we choose ...

Fewer and better suppliers?

There are hundreds of thousands of open source projects (suppliers) — and not all deliver the same quality.

The GOOD ...

- Release new components with updated features 3-4 times a year.
- Patch newly discovered security vulnerabilities in less than 7 days.

The NOT SO GOOD ...

- Have failed to release new versions in 3 or more years.
- Take an average of 390 days to patch known security vulnerabilities in dependent components.

Source: Analysis of the Central Repository, the world's largest repository of java components.

How well do we choose ...

Higher quality parts?

Of the 1 million open source components in the Central Repository, 51,000 have known security vulnerabilities and 340,000 have restrictive licenses.

The GOOD ...

Use automation to deliver up-to-date component intelligence to developers in the tools they use every day, so only the latest, safest and highest quality components are chosen.

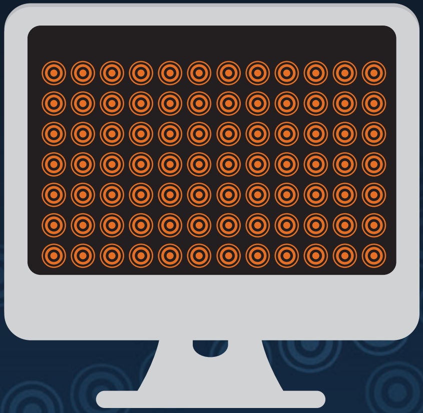
The NOT SO GOOD ...

1 out of every 16 component download requests is for a component with a known vulnerability.

Source: Analysis of the Central Repository, the world's largest repository of java components.

How well do we ...

Track what is used and where?



A typical application has 106 open source components[2] —and they age more like milk than wine. In fact, 50 new critical vulnerabilities[3] are discovered every day.

The GOOD ...

- Maintain a complete Bill of Materials inventory for their applications.
- Instantly know when a component is newly discovered to be vulnerable.

The NOT SO GOOD ...

- 63% have no clear & complete idea of components used in their applications.[1]
- 23% of components in a typical application have known vulnerabilities.[2]
- Average application has 9 restrictive licenses.[2]

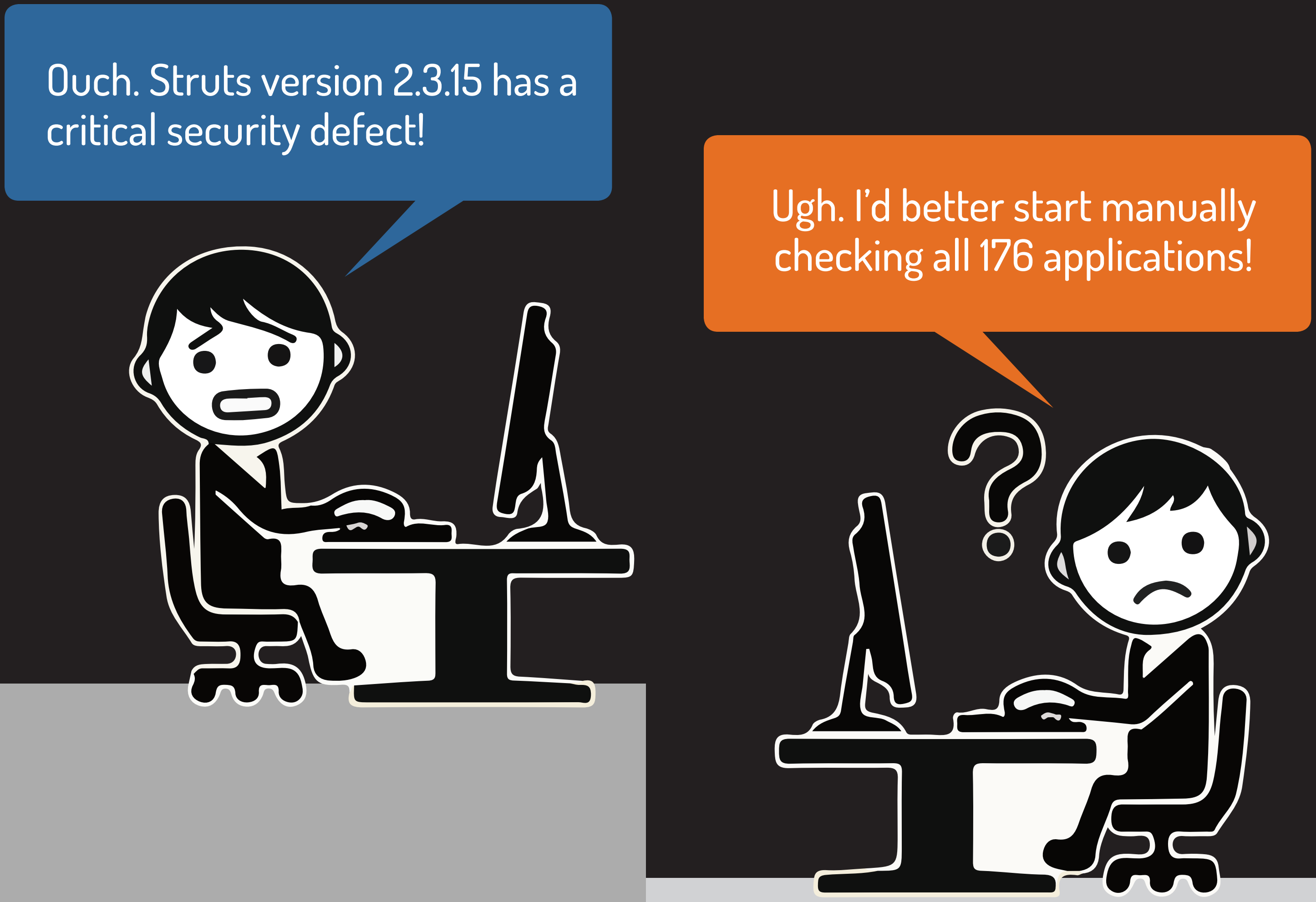
1 Source: Sonatype 2014 Open Source Development and Application Security Survey. Analysis of Sonatype Application Health Checks. 2 Source: Analysis of the Central Repository, the world's largest repository of java components. 3 Source: NIST National Vulnerability Database. https://nvd.nist.gov/

This is not a people problem, this is an automation problem.

And the lack of automation leads to rework, context switching and worse.

> “Software may be eating the world, but rework is choking software”
> John Jeremiah @j_jeremiah

And a visibility problem.


Ouch. Struts version 2.3.15 has a critical security defect!
Ugh. I'd better start manually checking all 176 applications!
?

Give your team the right tools! Automate your software supply chain!

Fewer & better suppliers

+

Higher quality parts

+

Track what is used and where

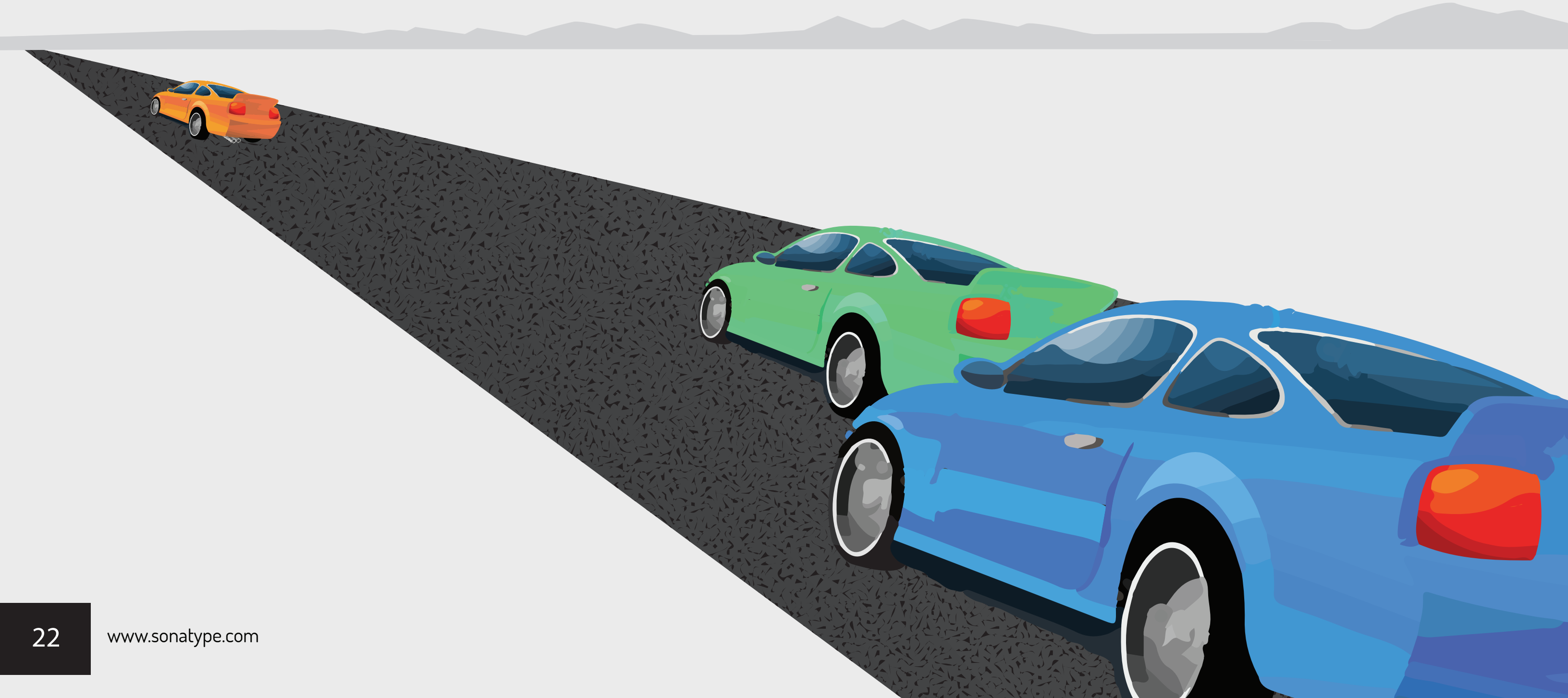
=

QUALITY & CONTINUOUS ACCELERATION

Masterfully outperform your peers.

AUTOMATE.

High performers accelerate away from the pack, and they continue to get better—and better.



Groundbreaking stuff!

Gene Kim

DevOps Authority, Author

“Sonatype uses the metaphor of the ‘software supply chain.’ This metaphor enables some startling revelations on how we should select the components we use and the downstream effects of the decisions we make.”

Nigel Simpson

Director of Enterprise Architecture

“This report draws parallels with traditional manufacturing supply chains, giving us a new way to look at how we build software.”

John Willis

DevOps Days Core Organizer

“In a world that is vastly moving to containers and immutable infrastructure the subject of Software Supply Chains is going to become of increased importance.”

Gareth Rushgrove

Senior Software Engineer, Puppet Labs
and Curator of DevOps Weekly

“This report is required reading for anyone interested in large-scale systems engineering.”

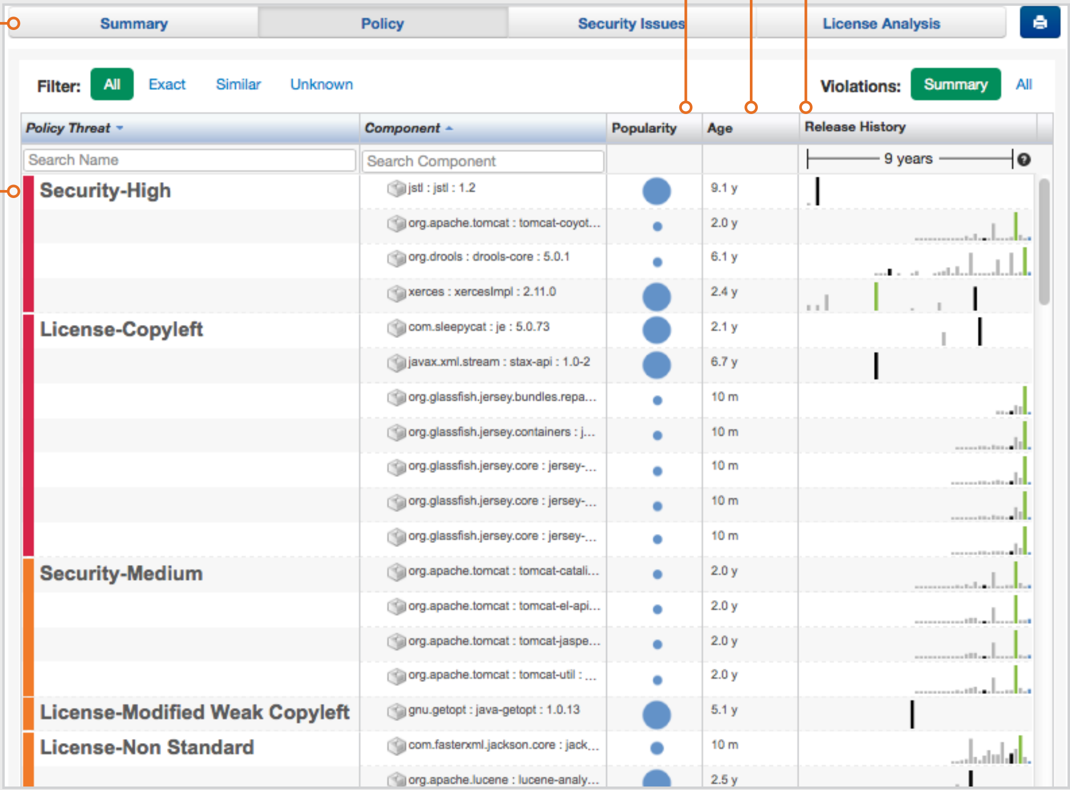
Got Five Minutes?

Create a free software bill of materials!

www.sonatype.com/BoM

Easily toggle between the summary, and detailed policy, security and license data views.

At a glance, see the overall component popularity, age and see if you are using the most recent version.



Policy Threat	Component	Popularity	Age	Release History
Security-High	jstl : jstl : 1.2	●	9.1 y	
	org.apache.tomcat : tomcat-coyot...	●	2.0 y	
	org.drools : drools-core : 5.0.1	●	6.1 y	
	xerces : xercesImpl : 2.11.0	●	2.4 y	
License-Copyleft	com.sleepycat : je : 5.0.73	●	2.1 y	
	javax.xml.stream : stax-api : 1.0-2	●	6.7 y	
	org.glassfish.jersey.bundles.repa...	●	10 m	
	org.glassfish.jersey.containers : j...	●	10 m	
	org.glassfish.jersey.core : jersey-...	●	10 m	
	org.glassfish.jersey.core : jersey-...	●	10 m	
	org.glassfish.jersey.core : jersey-...	●	10 m	
Security-Medium	org.apache.tomcat : tomcat-catali...	●	2.0 y	
	org.apache.tomcat : tomcat-el-api...	●	2.0 y	
	org.apache.tomcat : tomcat-jaspe...	●	2.0 y	
	org.apache.tomcat : tomcat-util : ...	●	2.0 y	
License-Modified Weak Copyleft	gnu.getopt : java-getopt : 1.0.13	●	5.1 y	
License-Non Standard	com.fasterxml.jackson.core : jack...	●	10 m	
	org.apache.lucene : lucene-analy...	●	2.5 y	

See your list of components, including color coded indicators of issue severity, including known security vulnerabilities and license issues.

Sample Software Bill of Materials from Sonatype.

MORE AWESOME INSIGHTS HERE.

We’ve only covered the tip of the supply chain story. Get more ideas, more impact, more vision from the full report at www.sonatype.com/speedbumps

Sonatype

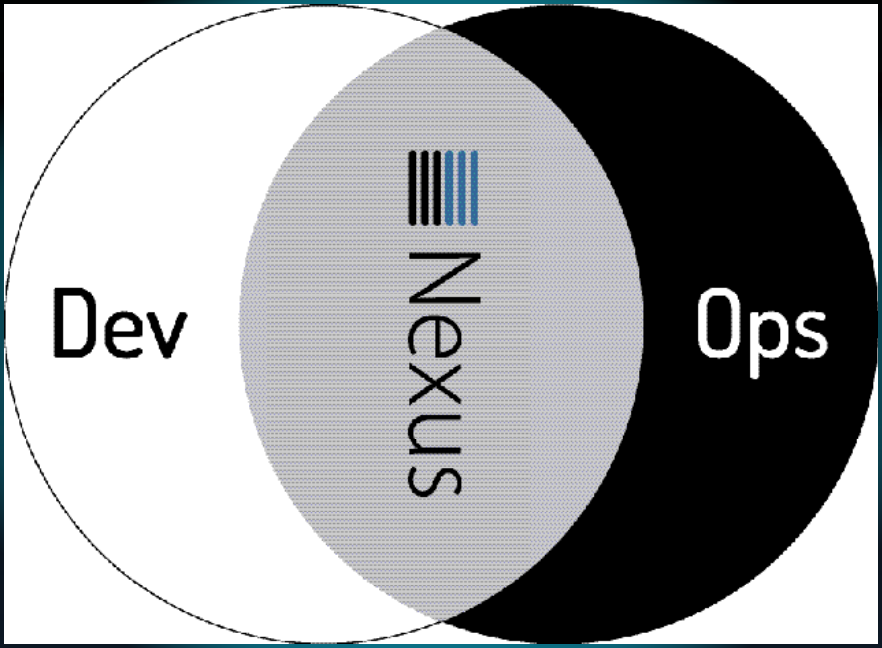
2015 State of the Software Supply Chain Report:

HIDDEN SPEED BUMPS ON THE ROAD TO “CONTINUOUS”

Foreword by Gene Kim, Gareth Rushgrove, John Willis, and Nigel Simpson

RESEARCH REPORT

Share the message!



www.sonatype.com

Sonatype helps organizations build better software, even faster. Like a traditional supply chain, software applications are built by assembling open source and third party components streaming in from a wide variety of public and internal sources. While re-use is far faster than custom code, the flow of components into and through an organization remains complex and inefficient. Sonatype's Nexus platform applies proven supply chain principles to increase speed, efficiency and quality by optimizing the component supply chain. Sonatype has been on the forefront of creating tools to to improve developer efficiency and quality since the inception of the Central Repository and Apache Maven in 2001, and the company continues to serve as the steward of the Central Repository serving 17.2 Billion component download requests in 2014 alone. Sonatype is privately held with investments from New Enterprise Associates (NEA), Accel Partners, Bay Partners, Hummer Winblad Venture Partners and Morgenthaler Ventures. Visit: www.sonatype.com

Sonatype Inc. • 8161 Maple Lawn Drive, Suite 250 • Fulton, MD 20759 • 1.877.866.2836 • www.sonatype.com