2016 STATE OF THE SOFTWARE SUPPLY CHAIN

Sonatype's 2nd annual report on managing
open source components to accelerate innovation.

TABLE OF CONTENTS

INTRODUCTION 4

SUMMARY 5

PART 1
A MASSIVE SUPPLY OF OPEN SOURCE COMPONENTS

The Software Supply Chain Index 8

PART 2
SUPPLIER NETWORKS ARE GROWING RAPIDLY

Suppliers: The Open Source Projects 11

Warehouses: Central Component Repositories 14

Industry Spotlight: J. Paul Reed 17

Local Warehouses: Repository Managers 18

PART 3
NOT ALL PARTS ARE CREATED EQUAL

Manufacturers: Software Development Team 23

Finished Goods: Software Applications 25

Industry Spotlight: The Quality of Encryption 28

PART 4
SOFTWARE SUPPLY CHAIN MANAGEMENT PRACTICES ARE GAINING TRACTION

Industry Spotlight: Global Insurer Boosts Innovation Budget 30

Embracing Principles of Automation in the Public and Private Sectors 31

Industry Spotlight: Intuit 34

Calculating the Cost of Rework 35

CONCLUSION 38

SOURCES 39

INTRODUCTION

WELCOME

Introduction

As caretakers of the Central Repository, Sonatype receives more than 31 billion requests per year for open source components. We literally feed millions of developers the software parts they require to manufacture and continuously deliver modern applications.

From this unique vantage point, we've amassed a great deal of data and we've developed deep intelligence with respect to the staggering volume and variety of open source components flowing through software supply chains into development environments.

We've studied the patterns and practices exhibited by high-performance organizations. We've also documented how these innovators are utilizing the principles of software supply chain automation to manage the massive flow and variety of open source components and consistently deliver higher quality applications for less.

The **State of the Software Supply Chain Report** blends public and proprietary data with expert research and analysis to reveal the following:

1. Developers are gorging on an ever expanding supply of open source components.

2. Vast networks of open source component suppliers are growing rapidly.

3. Massive variety and volume of software components vary widely in terms of quality.

4. Top performing enterprises, federal regulators and industry associations have embraced the principles of software supply chain automation to improve the safety, quality, and security of software.

Read the report and leverage the insights to understand how your organization's practices compare to others. Then, let us know what you think @sonatype.

Wayne Jackson
CEO, Sonatype

Summary

Warehouses

6.1%
Component downloads are vulnerable

Manufacturers

5.6%
Components in repository managers are vulnerable

Finished Goods

6.8%
Components in applications are vulnerable

Good news: enterprises can improve net innovation via improved software supply chain management practices.

1. Developers are feasting on a massive supply of open source components.

The use of open source and third party components is exploding. In 2015, **31 billion download requests** were recorded from the Central Repository. The trend is accelerating; the previous year, over 17 billion download requests were registered.

2. Supplier networks are growing rapidly.

Components are delivered to organizations via software supply chains that operate with many parallels to traditional manufacturing supply chains. A vast network of component suppliers creates **1,000 new projects and delivers 10,000 new versions per day.**

3. Not all component parts are equal.

While parts are the fuel of software supply chains, they have two big weaknesses: **(1) parts are not created equal, and (2) parts age and grow stale quickly.** Last year, the average enterprise downloaded 229,000 open source components. If properly sourced and managed, open source components are a tremendous source of energy for accelerating innovation. If not, they lead directly to security vulnerabilities, licensing risks, enormous rework, and waste. Our analysis of these downloads revealed that 6.1% (1-in-16) had a known security defect.

Furthermore, data from 25,000 applications demonstrated that **6.8% of components in use had a known security defect.** However, because a single component may contain multiple vulnerabilities, it's important to understand that an average application consisting of 106 components — of which 6.8% are known bad — could contain numerous unique vulnerabilities. [0]

4. Software supply chain management practices are gaining traction.

To counter the effects of massive component consumption, top performing development organizations embrace supply chain management best practices, including: (1) procure components from fewer and better suppliers, (2) procure only the best parts from those suppliers, and (3) continuously track and trace the precise location of every component. Furthermore, Federal regulators and industry associations like **FDA, FTC, UL, and FS-ISAC are taking action** to build awareness and establish guidelines for sound software supply chain management practices.

SSC BY THE NUMBERS

31B

COMPONENT DOWNLOAD REQUESTS ANNUALLY

page 9

229,898

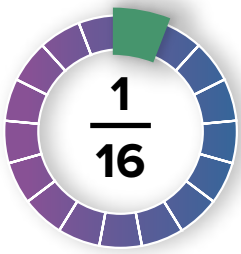
AVERAGE COMPONENT DOWNLOADS PER COMPANY

page 24

5,275

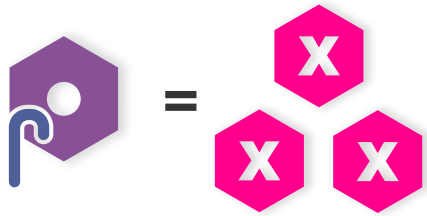
AVERAGE UNIQUE COMPONENT VERSIONS (PARTS) DOWNLOADED ANNUALLY

page 23



DOWNLOADS INCLUDED AT LEAST ONE KNOWN SECURITY VULNERABILITY

page 19



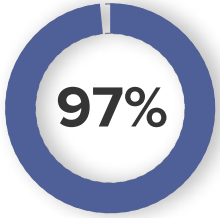
OLDER COMPONENTS IN APPS HAVE **3X HIGHER RATE OF VULNERABILITIES**

page 26

$7.42M

ESTIMATED COST TO REMEDIATE 10% DEFECTS ACROSS 2000 APPLICATIONS

page 35



COMPONENT DOWNLOADS CANNOT BE EASILY TRACED OR AUDITED

page 20

5.8M

VULNERABLE BOUNCY CASTLE LIBRARIES DOWNLOADED LAST YEAR

page 28

125K

REPOSITORY MANAGERS ARE USED TO IMPROVE BUILD SPEEDS AND RELIABILITY

page 19

69

NEW VULNERABLE COMPONENTS DOWNLOADED TO REPOSITORY MANAGERS PER QUARTER

page 21

150M

NPM COMPONENTS DOWNLOADED DAILY

page 14

PART 1

A **MASSIVE SUPPLY** OF **OPEN SOURCE COMPONENTS**

The Software Supply Chain Index

Developers are gorging on an ever expanding supply of open source components. As consumption volumes continue to skyrocket, three supply chain automation principles are improving net innovation.

Component use is exploding

In 2015, Sonatype measured over 31 billion download requests from the Central Repository [1] – the largest and most active public repository of open source components for the Java development community. Accessed by over 10 million developers worldwide [2], the data from this repository reflects a 64x increase in volume of open source download requests since 2007.

Components minimize the need to code from scratch

Today, modern applications typically consist of 80% - 90% component parts. [3] And, while it is impossible to measure the precise productivity gains associated with componentized software development; it is easy to imagine how billions of hours have already been saved due to the simple fact that developers no longer have to write code from scratch.

Through this lens, it is easy to understand how the incredible power of open software innovation has been fueled by an enormous volume and variety of reusable parts. It's also easy to see how these parts are being assembled together through a supply chain-like process that closely resembles how physical goods, like automobiles, are manufactured.

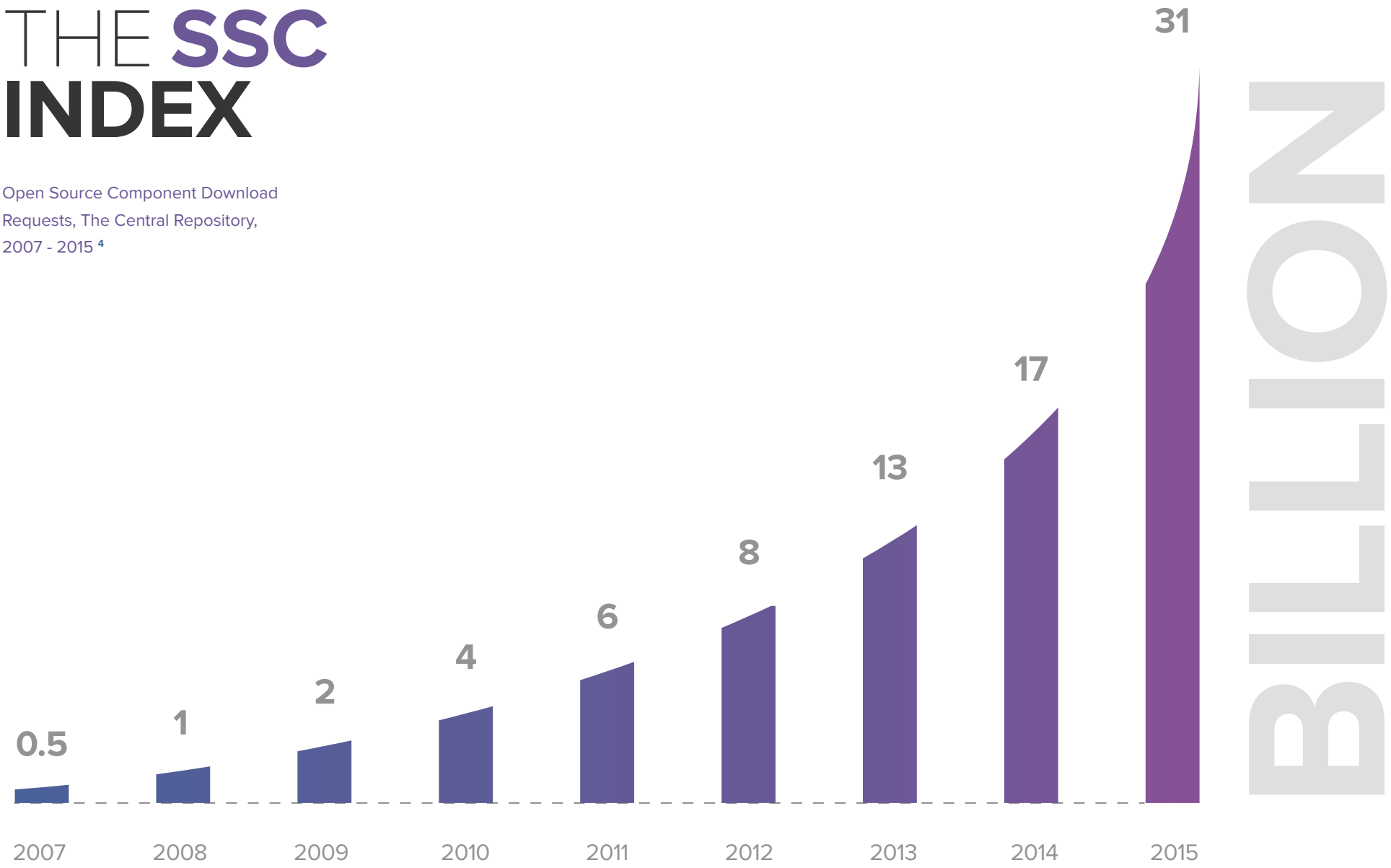
Three principles to improve net innovation

In the wake of massive component consumption, top performing development organizations are embracing supply chain management best practices and automation tools to help **(1) procure components from fewer and better suppliers, (2) procure only the best parts from those suppliers, and (3) continuously track and trace the precise location of every component.** For these organizations, automation is not about just going faster and increasing the number of deploys per day. They focus on driving value through net innovation as they balance speed with quality, security, maintainability, and repeatability.

Enterprises trying to keep pace with consumption through manual evaluations and governance of component quality, risks, and security can not keep pace with the volume and they inevitably fall behind and become prone to increased costs associated with rework, bug fixes, and context switching.

THE SSC INDEX

Open Source Component Download Requests, The Central Repository, 2007 - 2015 [4]



PART 2

SUPPLIER NETWORKS ARE GROWING RAPIDLY

SUPPLIERS

The Open Source Projects

Millions of developers feed new components into software supply chains. While pioneers of open source initially focused on Java, there is a rising tide of components across development languages.

Building on immutable infrastructure

Adron Hall blogged in September 2015, "With the advent of continuous delivery and integration, application deployment via immutable infrastructure has become a best practice. This generally involves being able to fully script and recreate a production (or staging or UAT or other) environment in minutes. This gives the ability to rollback and push deployments to production in a matter of seconds. By proxy of building on immutable infrastructure like this it dramatically increases the ability for a team to test deployments without fear." [5]

The importance of immutable parts — Reflections on npm-gate

In March 2016, a developer named Azur Koculu unpublished all of his packaged software components from the global npm registry. [6] Among these was a component containing 17 lines of code that were used by thousands of development teams across the globe. Within minutes of unpublishing, builds at companies large and small, including the likes of Facebook and Spotify, were failing because the packages were no longer available from their supplier. The npm-gate issue was quickly resolved but it highlights the importance of a healthy supply chain consisting of trustworthy suppliers, and immutable parts.

While some repositories like the Central Repository and now the npm registry are managed as immutable sources of parts, other repositories like the NuGet Gallery are mutable. Modified components can be published on top of previous components at the same coordinate location. In these cases, the absence of immutability presents a challenge for organizations attempting to identify which version of a component they really have residing within their software supply chain.

Suppliers fuel innovation

Suppliers of open source software components like Azur live at the heart of software supply chains. Without them, the entire supply chain is empty. Azur Koculu represents one of 3.7 million developers delivering open source and third party components used across software supply chains. Sites like Openhub.net count over 672,000 open source projects across multiple environments. [7]

Components are updated 14x per year

An analysis of 380,000 open source projects reveals that components are updated with new releases an average of 14 times per year. This average is skewed by over 300 projects that release new versions more that 100 times a year. Data reveals that 50% of projects release new versions between 3 - 10 times per year, while 17% of projects release new versions once or twice per year. Just like any supplier offering a product, version updates introduce new features, boost performance, repair defects, fix security flaws, introduce new dependencies, etc.

Some suppliers are better than others

Not all parts from these suppliers are created equal. Some are better than others. And it is not automatically obvious which parts are good and which parts are bad. While new, more powerful component versions are constantly made available from active suppliers, older versions are never retired.

Data from last year's report revealed that 59% of open source projects with known security vulnerabilities in their dependencies were never repaired; while newer versions of those components were released, the security defects in them were never fixed. [8] By contrast, the best suppliers were remediating flaws within a week of vulnerability disclosures, while the average repair time for the remaining 41% clocked in at 390 days. More details of this analysis are available in Dan Geer and Josh Corman's USENIX article "Almost Too Big to Fail". [9]

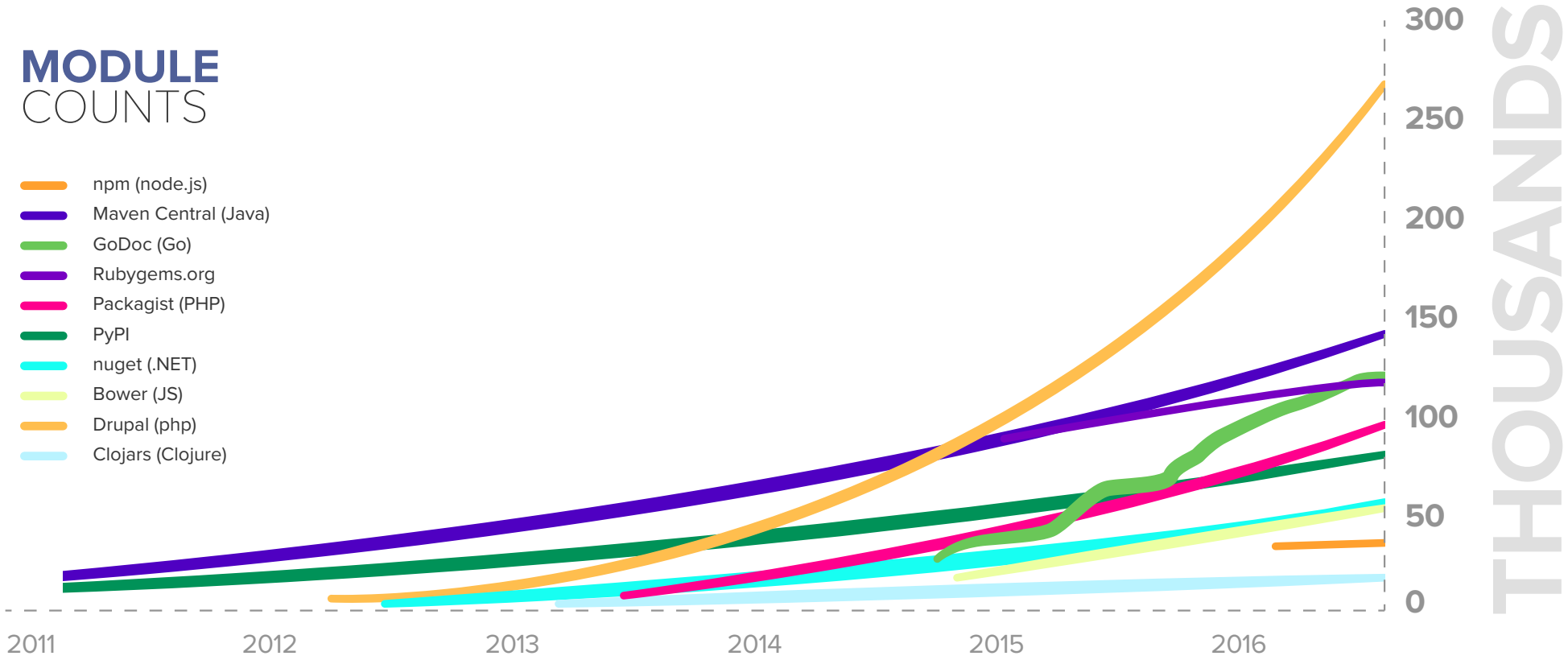
Componentized software development is here to stay. Going forward, however, the global development community will actively seek information to determine which suppliers are offering the highest quality components.

3.7M DEVELOPERS ↔ AT THE **HEART OF THE SSC**

The rising tide of components is multilingual

Modulecounts.com tracks component counts available across the software supply chain for various development languages. The growth leads to an ever-greater supply of components available to supply chains. Component formats like npm and Go have experienced significant growth in the number of parts being created by developers, while Java, RubyGems, PHP, and PyPI also show steady climbs over time.

Deeper analysis of the data available from modulecounts.com reveals the magnitude of the open source component wave. Approximately 1,000 suppliers -- open source projects -- join the network each day across all component formats. When accounting for all of the suppliers across these development ecosystems, we estimated about 10,000 new component versions are introduced daily.



Source: Modulecounts.com

WAREHOUSES

Central Component Repositories

As downloads reach into the billions per year, software quality is threatened by defect rates and a lack of supply chain management rigor.

The rapid growth of central repositories

In order to facilitate distribution of, and access to, open source and third party components, central warehouses have been set up to better serve the development community. Serving as immutable sources of components required by development and operations teams, these warehouses hold millions of components.

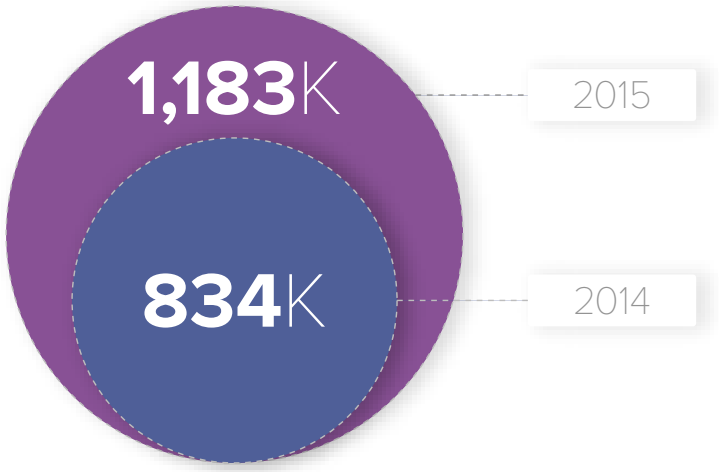
The components have all been contributed by open source projects and third-party developers to be made freely available to the estimated 18 million developers worldwide. [10] In 2015, billions upon billions of downloads were registered. RubyGems.org is now approaching 8.4 billion component downloads over its history [11], and Bower component downloads are regularly peaking over 80,000 per day. [12] Well-established central warehouses like those for Maven and npm components also grew at very healthy rates.

Container registries are part of the software supply chain

In 2015, we also witnessed explosive growth from relative newcomers like Docker Hub. For Docker, containerized applications grew from 14,500 in 2014 to over 150,000 by mid-2015 -- a 934% increase. [13] At the same time, Docker Hub surpassed half a billion downloads. While open source binaries and containerized applications are different beasts, the supply chains they utilize behave with similar participants, flow, work centers, and controls for those in the development and IT operations community.

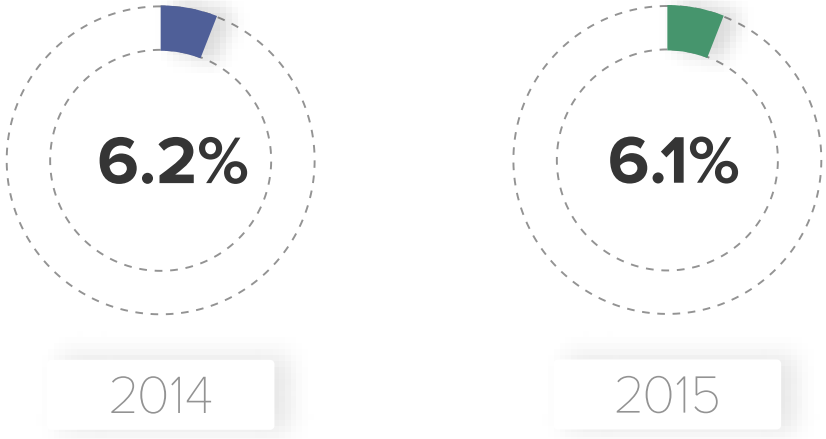
Sample of Open Source Repositories and Registries	2014 Volume of Download Requests	2015 Volume of Download Requests	% YOY Growth
Central.sonatype.org	17 billion	31 billion	82%
npmjs.org	15 billion	46 billion	306%
NuGetGallery.com	280 million	756 million	270%
Docker Hub [14]	2.750 million	500 million	18,082%

NUMBER OF UNIQUE COMPONENTS IN THE CENTRAL REPOSITORY



By the end of calendar 2015, the supply of components available from the repository grew by 42% over calendar 2014. In May 2016, the component inventory had reached 1.36 million -- averaging almost 1,400 component additions each day. [15]

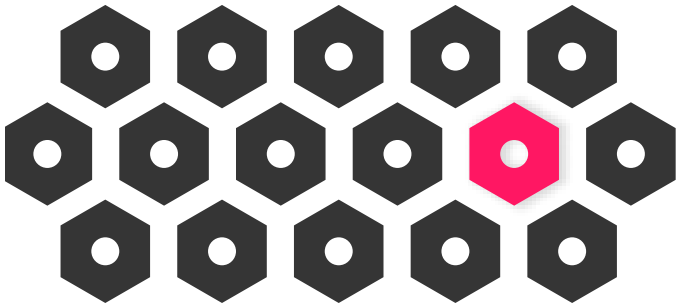
SSC DEFECT DOWNLOAD RATIO



Each year, development organizations are alerted to numerous discoveries of security vulnerabilities in open source components including the likes of handlebars.js, commons collection, and Spring Social Core. In the Central Repository alone, over 71,000 components had known security vulnerabilities associated with them. [16] As with all mature warehouses for components, even though component parts may have a known vulnerability, they cannot be removed from the warehouse -- and for good reason. Warehouses are immutable sources of components for development teams. In order to replace a defective part, developers must first rebuild the original application using all of the original components including the known vulnerable one. Once reconstructed, they can then replace the vulnerable component with a known safer version and release a new version of the application.

[...] in 2015 just over 1-in-16 downloads were components known to have security vulnerabilities.

Across the billions of downloads from the Central Repository in 2015 just over 1-in-16 downloads (6.1%) were components known to have security vulnerabilities. [17] In 2014, the percentage of vulnerable downloads was only slightly higher at 6.2%. [18]



To effectively manage a software supply chain, organizations must improve their visibility, reduce the variety, and govern the quality of components coming into their organizations. Enterprises must also seek to control how and where those components enter the development process, evolving away from free-for-all consumption practices.

As software supply chain practices mature, suppliers and manufacturers will need to share more information. Right now the communication channel between suppliers and manufacturers is limited. This lack of communication is a real problem when you consider the fact that suppliers release updates to components an average of 14x per year. The result is that software manufacturers are often unaware that new and improved versions of components exist.

INDUSTRY SPOTLIGHT

The value of optimizing software chains for development and release engineering teams:

"One of the interesting things we're finding with DevOps is that we can frame the work we do in the context of a pipeline. By identifying and optimizing some of the business value within that pipeline, businesses are receptive. Developers are receptive. Different parts of the business are receptive in ways I've almost never seen in my career, and it's great to be a part of that.

From a Rugged DevOps or security perspective, I think if we could move that work into the pipeline, not only do we make it visible in terms of the costs and trade-offs, but then also we could possibly do more. It's part of that whole. When you shift work further upstream, you can address it earlier and actually have a chance at fixing the problem.

One of the things I wouldn't think keeps release engineers up at night as much as it keeps security engineers up at night are the questions: where is our software coming from, and what issues may it have in it?

That's not something developers, for whatever reason, traditionally seem to think about. And that's not to denigrate them. A lot of times they're under deadlines just like release engineering team. They go to the Internet and grab whatever version of a library suits their need. In fact, the one version I usually see being used is the upgraded version because there's some API that they need.

There's a concern there -- when you think about it -- of where that's coming from. I recently told this story about an engineer who was missing a DLL from a build. The engineer just Googled for the DLL, downloaded it, and then threw it on all the build machines. That was pretty scary.

If you have one vulnerable library in your product, that is a security problem. If you've got multiple versions of the same library and multiple versions of those are vulnerable, that's a release engineering problem. That's one of the best ways upfront that release engineers can contribute to Rugged DevOps and contribute to the security space in terms of helping to detangle that problem." [19]

J. Paul Reed
Managing Partner
Release Automation Approaches

LOCAL WAREHOUSES

Repository Managers

Use of repository managers improves build reliability and performance while also supporting better governance practices. Increased rigor around inventory management can lead to a substantial decrease in technical and security debt.

Improving the flow of components

In any business, dealing with moving parts from one location to another is vital. Streamlining distribution involves the planning and efficient use of supply chain resources and often involves working with intermediaries. While supply chains represent the organizations, people, technologies, activities, and resources involved in moving goods from suppliers to customers, logistics help us command the efficient and effective flow of those goods.

Two common paths for procurement: direct and indirect

Enterprise development organizations generally ingest open source components one of two ways. The more popular (but less hygienic) method is to ingest components directly from cloud based repositories such as the Central Repository, Docker Hub, the NuGet Gallery and others described in the previous chapter. The less popular (but more hygienic) method is to ingest components from local repositories located inside the organizational firewall.

Our analysis of download patterns in 2015 revealed that 96.7% of all components were sourced by development teams using a direct connection from their tools to the Central Repository [20] -- a slight uptick from 2014 where direct connections represented 95.2% of the consumption [21]. The dominant path is also the longest and least efficient path for sourcing components. Additionally, when less mature central warehouses like Bintray, Docker Hub and the NuGet Gallery experience outages, or in the case of npmjs.org where components were unpublished during “npm-gate”, development in organizations relying on components stored in those warehouses grinds to a halt.

The less popular (but more hygenic) method for sourcing components is to use local warehouses located inside the software supply chain -- commonly referred to as repository managers or container registries. Nexus, Artifactory, Archiva, and private Docker registries are common examples of these local warehouses. Locating components closer to where they are being used accelerates build and deploy times. These warehouses also cache the components locally; once components are downloaded from central warehouses, they can be accessed an infinite number of times from the repository

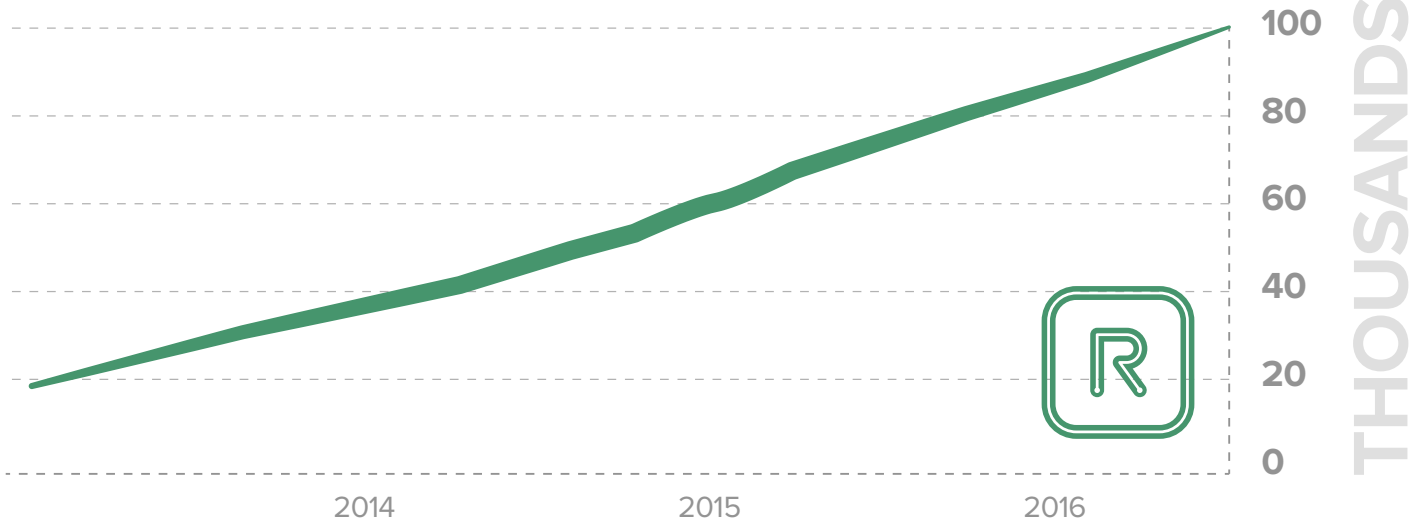
manager. Local repositories enable teams to store open source, third-party, and proprietary components in a private, secure location -- a capability not offered by most cloud-based public warehouses.

Use of local repository managers grows steadily

The use of local repositories continues to gain traction among DevOps and continuous delivery architects looking to optimize their software supply chains. It is estimated that over 125,000 repository managers are in use today. [22]

Performance benchmarks reveal that sourcing components locally can accelerate builds 10 - 20x. For larger size components like a containerized application, the performance benefits can be even greater. Yet downloads from caching repository managers only represented 3.3% of downloads from the Central Repository in 2015 [23].

NEXUS
REPOSITORY
GROWTH [24]



Repository managers improve visibility and control

Organizations seeking greater visibility to the components flowing through their software supply chains benefit from the use of local repositories and registries. Some organizations have mandated that downloads from cloud-based, public repositories must be directed through local repository managers. The mandates effectively limit the number of doors components can enter through, giving organizations better visibility and control as to what parts are being used. These organizations are effectively introducing a standard procurement path, where free-for-all sourcing from any location is banned.

When consumption follows a free-for-all pattern, 500 developers might equate to 500 doors through which components can be sourced. With components entering the organization through hundreds and sometimes thousands of locations, vetting of component quality, auditing of inventories, and traceability of components is impossible to achieve.

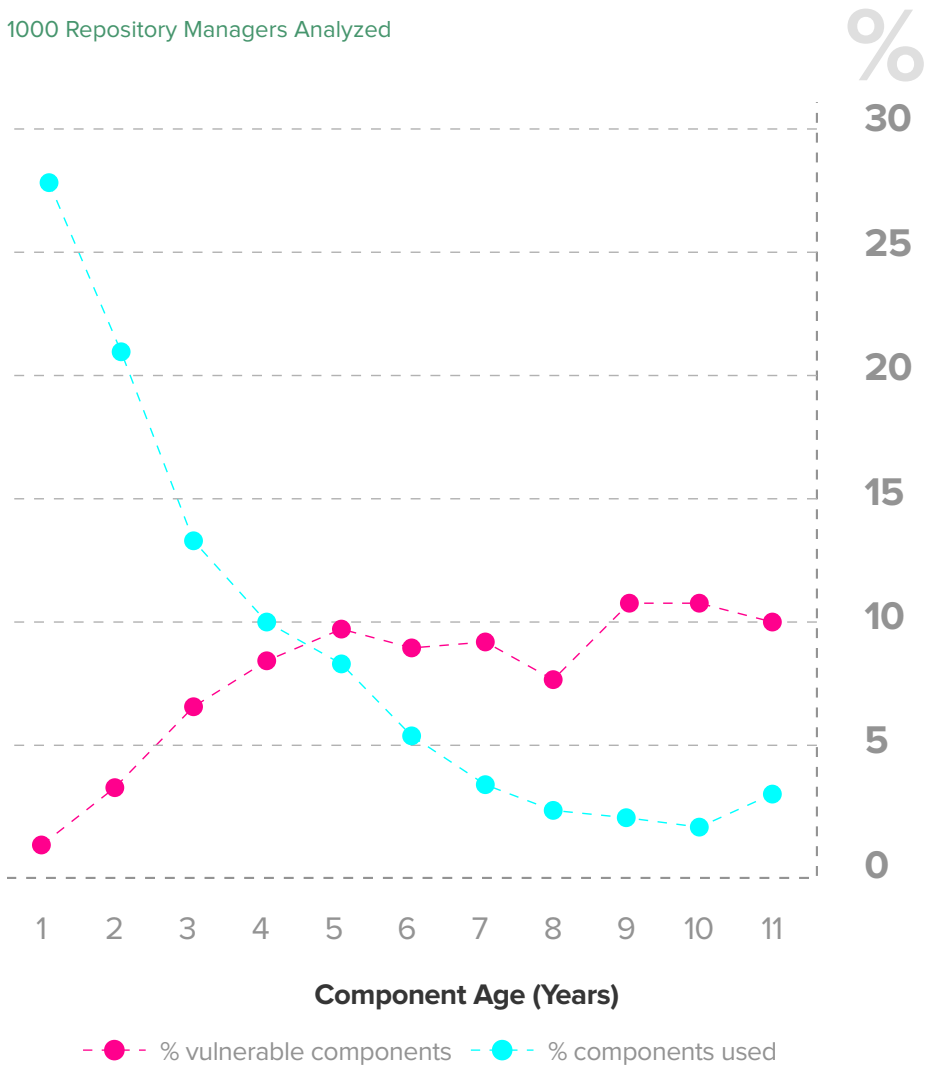
Newer components in repositories have fewer defects

Just as traditional operations managers monitor the quality of physical parts they bring into and store in their warehouses, the same practice should hold true for software components and repository managers. An in-depth analysis of 1,000 repository managers by Sonatype revealed interesting usage patterns across component inventories.

Components less than two years old account for over 47% of the parts accessed by development teams to build their applications. The average associated defect rates for those components were under 5%. By comparison, component versions older than two years accounted for 80% of the risk [25]. Capital One, Google, and other leading edge organizations are applying policies across their development practices to favor newer, higher quality components in order to reduce unplanned, unscheduled work while also reducing technical debt.

COMPONENT USE VS VULNERABILITIES

1000 Repository Managers Analyzed



Older parts are often unsupported

Sonatype's analysis of the inventories across 1,000 repositories also revealed a high percentage of older projects that have not released newer versions in years. For example, over 10% of components greater than five years old housed in repository managers had never released a newer version [26].

Use of outdated, unsupported, and dead projects can be problematic for DevOps organizations. If a severe defect or security vulnerability were discovered in any of the unsupported or dead projects, organizations would need to seek out a new supplier or build a new component with the same functionality from scratch.

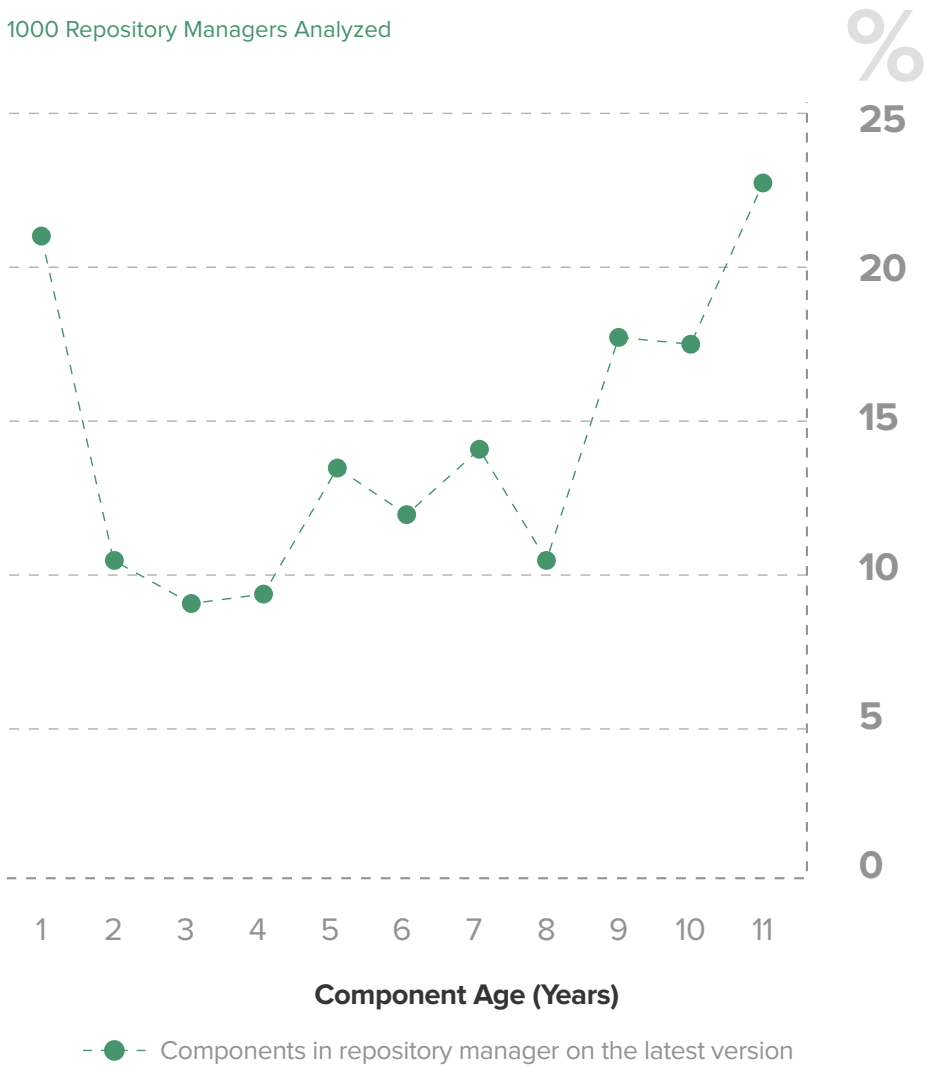
The analysis of repository inventories tells us outdated, unsupported, and vulnerable components are present in significant volumes. Because the inventory analysis represents a single point in time, we thought it important to shed light on the flow of known vulnerable components into repository managers.

Steady sourcing of components with known defects

In 2015, Sonatype evaluated an even broader set of repository managers. The evaluation targeted downloads performed by 7,000 repository managers, spanning a three month period. Each repository manager contained 500 or more components. During that time, the average number of new vulnerabilities that flowed into the repositories was 69 -- an average of 23 per month or just over one per workday. Seventy percent (70%) of the components with known vulnerabilities had a Common Vulnerability Scoring System (CVSS) level of 5 or greater [27].

Virtually no repository managers were immune from vulnerable downloads. Analysis revealed that 98.3% consumed at least 1 vulnerable component and 97.7% consumed at least one higher risk (CVSS > 5) component [28].

COMPONENT AGE DISPERSION INSIDE REPOSITORY MANAGERS



PART 3

NOT ALL PARTS ARE CREATED EQUAL

MANUFACTURERS

Software Development Teams

Data shows enterprises consumed an average of 229,000 software components annually, of which 17,000 had a known security vulnerability.

Component volume and variety at the heart of development

Very few CIOs, software development executives, enterprise architects, and especially personnel residing outside of the IT organization realize the extent of their organization's reliance upon open source components. While legal, security, audit, open source review boards, and other functional organizations have attempted to detail and track consumption behaviors, they often fall far short of gaining full visibility.

In 2015, the global community of software developers downloaded billions of components from the Central Repository, npmjs.org, RubyGems.org, Docker Hub and other public repositories. When performing detailed assessments of Central Repository downloads from over 3,000 companies, we saw the average company consume 229,898 components. [29] In 2014, component downloads were comparable at 240,000 per company. [30] It is important to note that these downloads only account for Java components; when multiple component formats (e.g., RubyGems, npm, PyPI, NuGet) are being used, the overall download volumes per enterprise will be significantly higher.

Further study of these downloads reveals an average of 5,275 unique components downloaded during the year. This points to an extreme inefficiency in sourcing behaviors by developers and their tool chains. In fully optimized software supply chains, a unique component part only needs to be downloaded once in order to be reused an unlimited number of times. Yet in 2015, the average company downloaded each unique component 111 times. If companies selected only the best components from each supplier, they would have only requested 2,071 components during the year, a reduction of greater than 50%. [31]

	Orders (downloads)	Suppliers (artifacts)	Parts (versions)
Averages (2015)	229,898	2,071	5,275

Sourcing practices pull in components of varying quality

The use of multiple versions of the same components adds complexity to managing environments while also growing technical debt. Use of multiple component versions also contributes to greater context switching and compatibility issues for developers.

Assessment of development organizations pulling in well over 200,000 components annually showed that 1-in-12 downloads had known security flaws. [32]

For many organizations, use of repository managers within the supply chain improves sourcing efficiencies while also reducing security debt. Only a small portion (5.08%) of the overall downloads take the efficient path. Within this path, only 1-in-20 (5.00%) component downloads included a known security defect. [33]

Quality is not a people problem

Most development teams strive for ever-increasing speed and throughput. Yet the software assembly process remains rife with inefficiencies, largely due to a lack of enforceable policies and precise data that can help developers make better, safer decisions. Use of poor quality components is not so much a people problem as it is a data precision and automation problem. Automating the availability of information about components is key to quickly making decisions. Providing precise data about the quality of components in an automated way, as early in the development lifecycle as possible, offers the best outcome.

	Orders	Quality Control		Quality Control with repository managers	
	Average downloads	# with known vulnerabilities	% with known vulnerabilities	Downloads to repository managers	Vulnerable downloads to repository managers
Detailed assessment of 3,000 companies	229,898	17,206	7.48%	11,695	5.00%

FINISHED GOODS

Software Applications

Analysis of 25,000 applications reveals 6.8% of components used included known defects. Organizations standardizing on components between 2 - 3 years of age can decrease defect rates substantially.

Last year's State of the Software Supply Chain report disclosed that an average of 106 components comprise 80 - 90% of a total application, yet few organizations have visibility into what components are used where. [34]

Known defective components lead to quality and security issues within applications. While developers save tremendous amounts of time by sourcing software components from outside their organizations, they often don't have time to check those component versions against known vulnerability databases or internal policies.

An analysis of 25,000 scans reveals that 6.8% of components being used in applications contained at least one known security vulnerability. [35] This finding demonstrates that defective components are making their way across the entire software supply chain -- from initial sourcing to use in finished goods.

Warehouses

6.1%
Component downloads are vulnerable

Manufacturers

5.6%
Components in repository managers are vulnerable

Finished Goods

6.8%
Components in applications are vulnerable

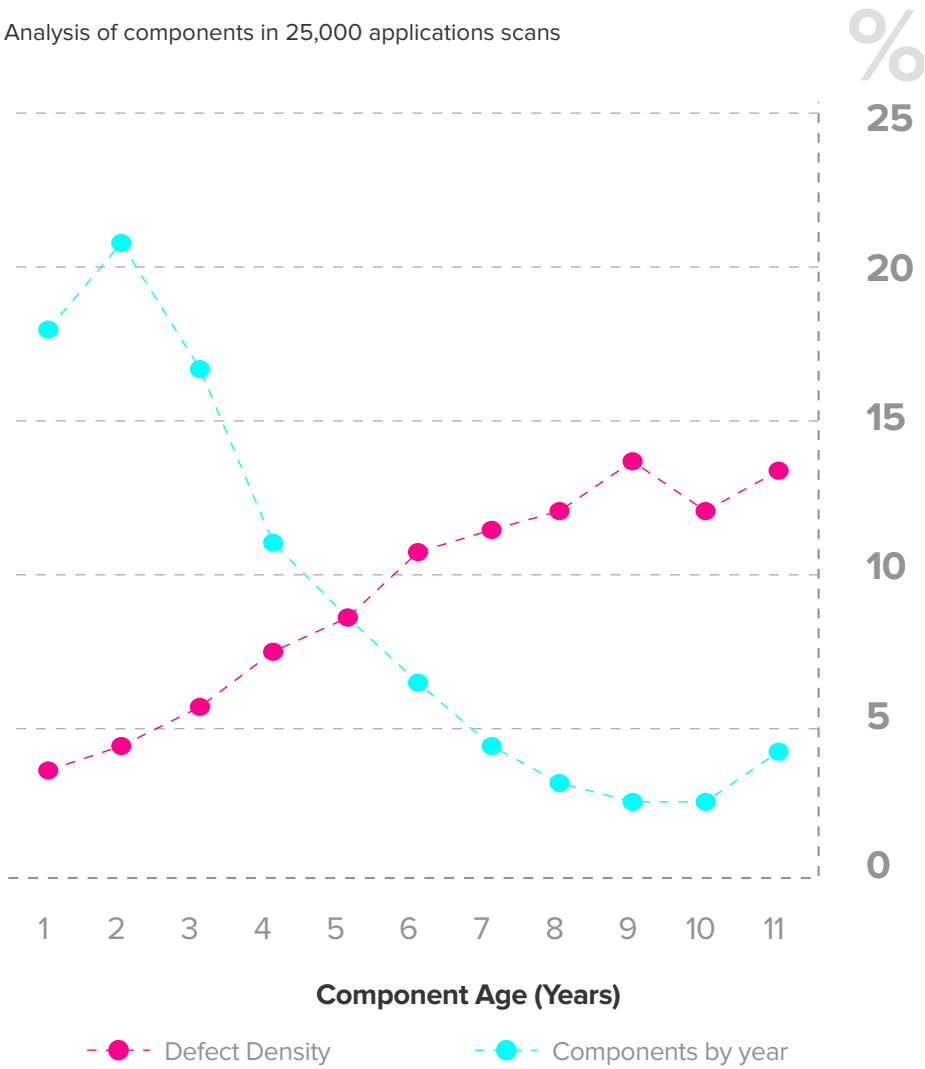
Newer components make better software

Analysis of the scanned applications revealed that the latest versions of components had the lowest percentage of known defects. Components under three years in age represented 38% of parts used in the average application with security defect rates under 5%. By comparison, components between five and seven years old had 2x the known security defect rate. [36] The 2016 Verizon Data Breach and Investigations Report confirms that the vast majority of successful exploits last year were from CVE's (Common Vulnerabilities and Exposures) published 1998 - 2013. Combining the Verizon data with Sonatype's analysis further demonstrates the economic value of using newer, higher quality components.

In summary, components greater than two years old represent 62% of all components scanned and account for 77% of the risk. Better component selection not only improves the quality of the finished application, it also reduces the number of break-fixes and unplanned work to remediate the defects. [37]

NEWER COMPONENTS MAKE BETTER SOFTWARE

Analysis of components in 25,000 applications scans



Older components die off

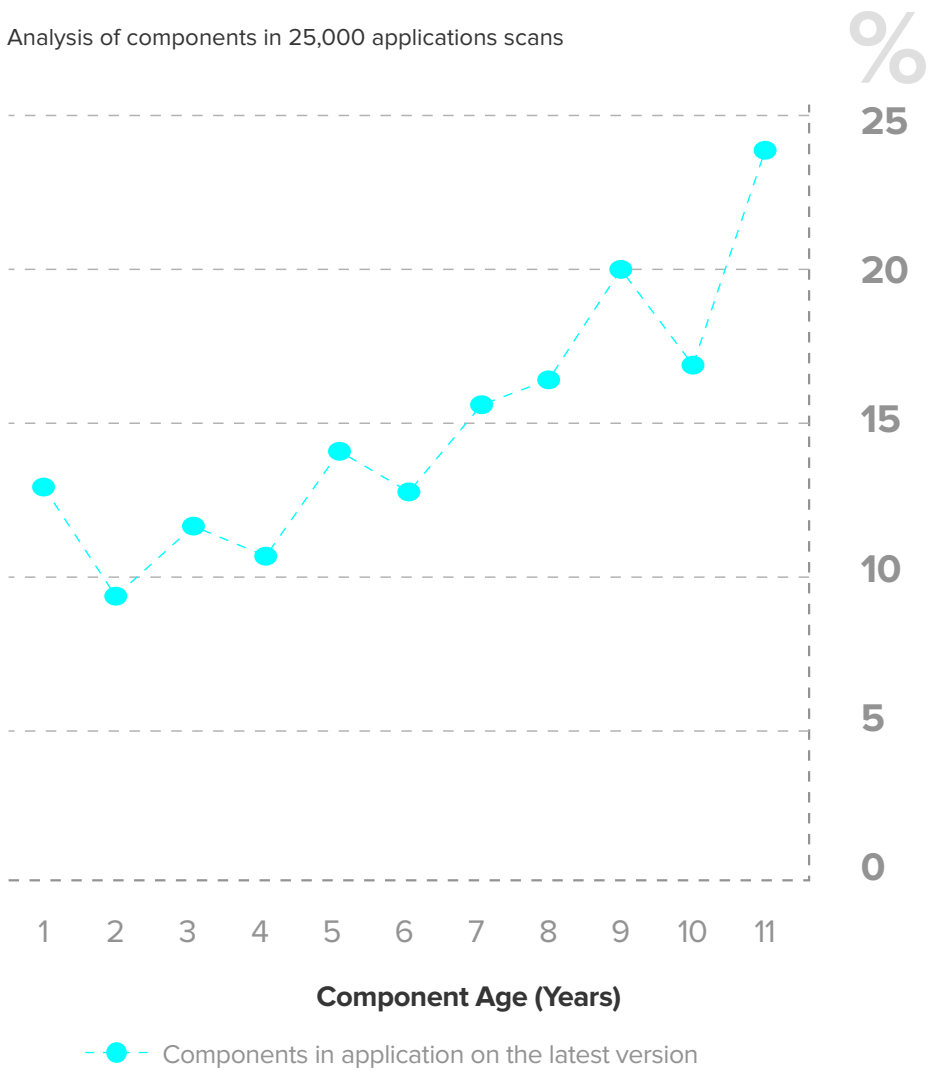
As noted earlier in the report, new versions of components are released an average of 14x per year. The new versions deliver greater functionality, improved performance, and fewer known defects. Just as in traditional manufacturing, using the newest versions of any part typically results in a higher quality finished product.

Versions that were seven years or older made up approximately 18% of the component footprint of the 25,000 scans. For the older components, analysis showed that as many as 23% were on the latest version -- meaning, the open source projects for those components were inactive, dead, or perhaps incredibly stable. [38] Discovery of components with known security vulnerabilities or other defects used in applications is not something anyone desires. Unfortunately, when these defects are discovered in older components, chances of remediating the issue by upgrading to a newer component version are greatly diminished. If a new version does not exist, only a few options exist: (1) keep the vulnerable component in the application, (2) switch to a newer like component from another open source project, (3) make a software change to add a mitigating control, or (4) code the functionality required from scratch in order to replace the defect. None of these options comes without a significant cost.

As discussed in Cisco's 2015 Midyear Security Report, "With open-source software in place in many enterprises, security professionals need to gain a deeper understanding of where and how open-source is used in their organizations, and whether their open-source packages or libraries are up to date. This means that, moving forward, software supply chain management becomes even more critical." [39]

OLDER COMPONENTS DIE OFF

Analysis of components in 25,000 applications scans



INDUSTRY SPOTLIGHT

Encryption: the importance of security, privacy

For developers wanting to add cryptographic libraries to their applications, a number of open source components are available to them. Of course, anyone seeking to add encryption to an application has an important requirement for the privacy and security it provides.

One of the more popular components for encryption is known as The Legion of Bouncy Castle Cryptographic Library. Download records from the Central Repository reveal that 17.4 million Bouncy Castle components across all versions were dowloaded last year. Of these, 5.8 million (33%) were known vulnerable versions of Bouncy Castle.

It's a sobering fact, but it's true. Last year alone, organizations downloaded vulnerable versions of the Bouncy Castle cryptographic library 5.8 million times. The defective components downloads occurred across 93,253 unique IP addresses from 13,824 organizations in 197 countries. [40]

5.8M

PART 4

SSC **MANAGEMENT PRACTICES** ARE **GAINING TRACTION**

INDUSTRY SPOTLIGHT

Global Insurer Boosts Innovation Budget by 30%

W. Edwards Deming taught the world that it was possible to sustain competitive advantage by following three basic principles: (1) use fewer and better suppliers, (2) use higher quality parts, and (3) track what is used and where.

One organization – a global insurer with over $50 billion in revenue – applied these three principles to their software development lifecycle. They had two simple goals:

1. **Reduce defects from 10 per 10,000 lines of code**
2. **Reduce rework and improve productivity by 15%**

The insurer's first step was to minimize complexity in their software supply chain by moving toward a single supplier (OSS projects) for any one type of component. By identifying the best suppliers of software components, they reduced defects per 10,000 lines of code by 60% (from 10 to 4).

Next, equipped with continuous component intelligence the insurer's development team used only the best component versions. Armed with real-time intelligence about their component quality, the organization reduced defects per 10,000 lines of code from 4 to 1.

The third step was to track and trace every component across their supply chain by creating a software bill of materials (BOM). With a BOM in place, discovery of component defects in the future brings their mean time to identify issues to near-zero while ensuring corrective measures can be taken immediately.

Use fewer & better component suppliers

Use only the highest quality component parts

Continuously track when & where components are used

By applying the three principles, the global insurer improved developer productivity by 30%. Reducing the complexity of operations, improving the quality of components, and maintaining clear visibility to components enabled the insurer to shift hundreds of millions of dollars from their maintenance budget to their innovation budget.

Embracing Principles of Automation

Federal regulators, industry associations and top performing enterprises have embraced the principles of software supply chain automation to improve the quality, safety, and security of software.

Software bill of materials

Based on the volume and velocity of open source and third-party software components being consumed, it is impossible to check everything manually. It is simply too expensive and too slow—especially when considering the sub-components or dependencies which are less obvious. For many organizations, tracking a complete inventory of the component parts used in an application requires producing a "software bill of materials" (BOM).

Mayo Clinic and Exxon

According to TheHill.com, Mayo Clinic and Exxon are two examples of organizations that have procurement policies that force vendors to accept liability for software flaws that cause a breach. And Mayo Clinic forces vendors to go through extensive testing and to provide a bill of materials to insure none of the software has known vulnerabilities. [41]

Financial Services - Information Sharing and Analysis Center (FS-ISAC)

Those two are not the only organizations enforcing policies and a software bill of materials. The FS-ISAC released new guidelines in 2016 that advise organizations to use a software "bill of materials that clearly identifies the open source code libraries that are part of a commercially developed software package offered to financial service firms." [42] FS-ISAC recommends a BOM to help prioritize remediation of known security vulnerabilities, create awareness of potential intellectual property issues, and improve vendor evaluation criteria.

American Bankers Association (ABA)

The Financial Services Sector Coordinating Council (FSSCC) in coordination with the ABA produced the 2016 Cyber Insurance Buyer's Guide. For banks attempting to identify and mitigate cyber risks, the guide points to new ways organizations can transfer those risks through the purchase of insurance products. For companies purchasing software, the guide describes procurement policies that address the analysis of "all third-party open source components, and shall, at a minimum, identify all known

vulnerabilities." [43] It also recommends a "Bill of Materials that clearly identifies all known third-party software components contained in the supplier product", including "any changes resulting from product updates, patches, etc." [44]

For business executives considering cyber insurance purchases, the report advises, "Companies that implement these procurement policies should find themselves more insurable in the market, both in terms of the dollar amount of the insurance and scope of coverage." [45]

U.S. Federal Trade Commission (FTC)

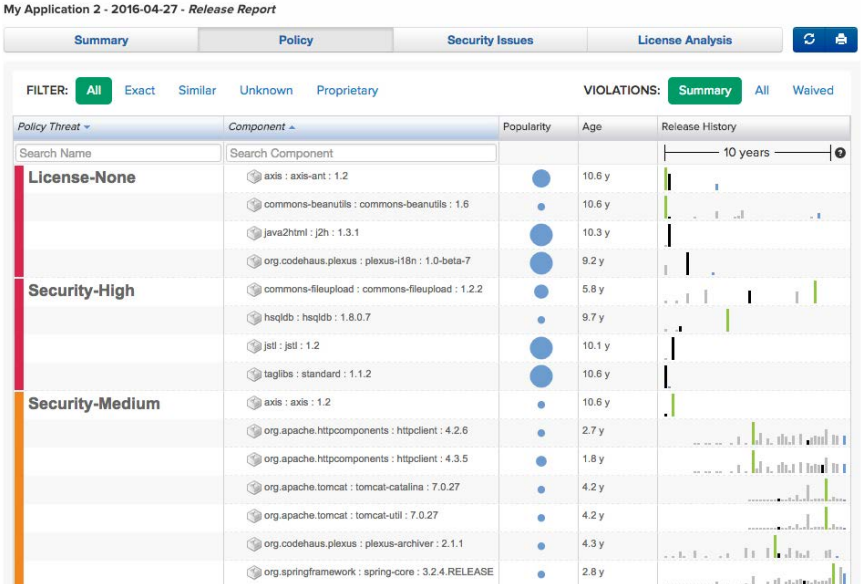
Where Mayo Clinic and Exxon have introduced procurement policies to eliminate software vulnerabilities from entering their software supply chains, we have also seen government organizations like the FTC le complaints against companies using known vulnerable software. [46] In February 2016, ASUSTeK Computer, Inc. settled charges with the FTC related to its use of insecure software in home routers and cloud services that put consumers at risk.[47]

U.S. Food and Drug Administration (FDA) and Philips Medical

In January 2016, the conversation on software bill of materials also extended to the US Food and Drug Administration's (FDA) discussions titled "Collaborative Approaches to Medical Device Cybersecurity". The FDA was studying how to best work with medical device manufacturers to address known vulnerabilities in their software and the need for a software BOM.

In the FDA workshops, Michael McNeil, Global Product Security & Services Officer at Philips Healthcare was asked if it was Philips' intention to share the BOM with provider organizations that are using its products. Mr. McNeil offered, "I would definitely say that from a Philips perspective, our bill of

A SAMPLE SOFTWARE BILL OF MATERIALS



materials and what we use to leverage for our particular threats in the vulnerabilities within those solutions would be shared." [48]

Underwriters Laboratories (UL)

In April 2016, Underwriters Laboratories, launched the UL Cybersecurity Assurance Program (UL CAP) that helps software vendors identify security risks and suggest methods for mitigating those risks in their products. This program developed a series of standards, under UL 2900, to protect against

vulnerabilities and software weaknesses. The standards include identifying components and mitigating known security vulnerabilities across a wide range of industry functions, including: industrial control systems, medical devices, automotive, HVAC, lighting, building automation, and consumer electronics. [49]

An anticipated effect of having software supply chain security standards introduced by UL is growing attention from insurance underwriters. With a baseline for software security testing in place, insurers can begin to establish premiums that best reflect an organization's supply chain hygiene.

U.S. Department of Defense (DOD)

The Department of Defense outlines a process similar to UL and ABA in its February 2016 publication "How to Put Software Assurance into Contracts". The document suggests language that may be tailored for use in Request for Proposal (RFP) packages and contracts with third-parties "to provide assurance regarding developed software and its ability to meet the mission needs." The publication further infuses credibility in the procurement approach by citing current law: "Common industry practice and Section 933 of the National Defense Authorization Act for Fiscal Year 2013 [Public Law 113-239], define 'software assurance' to mean the level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software, throughout the lifecycle". [50]

Energy Sector Control Systems Working Group

The Energy Sector Control Systems Working Group's (ESCSWG) report, "Cybersecurity Procurement Language for Energy Delivery Systems", offers guidance to help protect against cybersecurity threats that pose serious or ongoing challenges for North America's energy infrastructure. Procurement guidance offered in the report includes, "The Supplier shall verify and provide documentation that procured products (including third-party hardware, software, firmware, and services) have appropriate updates and patches installed prior to delivery to the Acquirer". The report further recommends that a "Supplier shall provide appropriate software and firmware updates to remediate newly discovered vulnerabilities" and that "updates to remediate critical vulnerabilities shall be provided within a shorter period than other updates, within [a negotiated time period (e.g., 7, 14, or 21 days)]." [51]

18F Group

18F is an office inside the U.S. General Services Administration that helps other federal agencies build, buy, and share efficient and easy-to-use digital services. The 18F group recommends extensive use of open source components anytime a savings of just 20 lines of custom code can be achieved. At the same time, 18F acknowledges the potential risk of open source component usage, recommending that developers carefully choose high quality open source projects that have strong contributing communities, reliable review of components they release, and a high level of responsiveness when problems are discovered. [52]

National Institute for Standards and Technology (NIST)

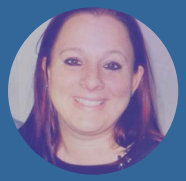
Applications destined to run on U.S. government systems are subject to scrutiny via a Risk Management Framework (RMF). The RMF applies security controls defined in National Institute for Standards and Technology (NIST) special publication 800-53, "Security and Privacy Controls for Federal Information Systems and Organizations". These controls include developing and documenting an inventory of information system components, which could include a bill of material listing open source components used. Another required control is vulnerability scanning, which should include identification of vulnerabilities in open source components used. [53]

INDUSTRY SPOTLIGHT

The importance of continuous improvement:

"What I really love about the recognition of software supply chains is being able to manage processes a certain way so that you can reduce defects. Having worked for Toyota in the past and understanding the supply chain mentality, you get a sense of how you could put something together better, incrementally improving on it, and then sharing that process. It really helps you figure out what things are important. When managing our software supply chain, introducing the notion of fewer, better suppliers was really a core concept."

"I love the idea of transparency. When building things with continuous improvement in mind, you need to look at things from an opportunistic perspective. You're not just looking to make things perfect, you're looking for those opportunities to improve over time." [54]



Shannon Lietz
DevOpsSec Lead
Intuit

Calculating the Cost of Rework

Eliminating mistakes before they happen is much less expensive than fixing mistakes after the fact.

Calculating the cost of component rework

The Phoenix Project, a book written by Gene Kim explains how modern software development closely resembles the same process utilized by manufacturers of physical goods.

Whether you're Toyota manufacturing automobiles, or a software development team assembling components into applications; the simple truth is that the flow of work should move in one direction only: forward. When work moves backward, it leads to a huge amount of waste.

To help software development organizations calculate the amount of waste associated with backward flowing work, we created a free visualization tool (www.sonatype.com/calculator) that quantifies the value of unscheduled, unplanned work stemming from the use of defective, outdated, and risky components. The calculator shows how time and budget allocated to rework and bug fixes reduces investments in innovation and negatively impacts shareholder value.

For example, a company with a portfolio of 2000 applications that dedicates resources to remediating 10% of their defective components would rack up $7,420,000 in waste annually. A company with a portfolio of 500 applications would rob their innovation budget by $1,800,000 annually by remediating 15% of their defective components.

Using fewer, better parts at Capital One

When presenting at DevOps Enterprise Summit 2016, Tapabrata Pal, Director and Platform Engineering Fellow at Capital One, shared "We found it inevitable to do DevOps security the right way....when you are transforming the CI/CD pipeline to start it, we said there will be only open copy of each particular library in the binary code repositories that we can check for security and legal vulnerabilities." [55] His team was able to eliminate large volumes of defective, outdated, and risky components by following the principles of using the highest quality parts from fewer suppliers. In a large organization like Capital One they are profiting from taking control of their software supply chain.

According to Josh Corman, Director of Cyber Statecraft, Atlantic Council, "The way we can get more safety and security into digital infrastructure, is to move to the ultimate evolved posture of software development as a supply chain. This makes you even faster than DevOps–even more efficient and with higher quality and risk mitigation without tradeoffs. If we get better at our component selection and our traceability of what we're using where, we can reduce the number of break-fixes or reduce the amount of unplanned, unscheduled work". [56]

SUMMARY

Conclusion

Decades ago, W. Edwards Deming taught automobile manufacturers the critical importance of building quality into their products by more effectively managing suppliers, sourcing parts, and tracking the precise vocation of every part assembled in every vehicle. Today, these same lessons are being applied to optimize the performance of modern software supply chains.

It's true that open source components enable development organizations to deliver software more efficiently by reducing the amount of code that they need to write. It's also true that every single component offers potential benefits as well as risks. Top performing development organizations manage these risks by identifying the best open source suppliers, selecting only the best components, and tracking the precise location of every component assembled in every application.

As caretakers of the Central Repository, we support millions of software developers from around the world. Last year alone, these developers requested 31 billion downloads of open source components. From this unique vantage point, we strive to do two things everyday; cultivate a deep understanding with respect to the quality of open source components, and study the patterns and practices exhibited by high-performance software development organizations that consume these components to build applications.

The sole purpose of this report has been to share with you the things that we observe, including:

1. Developers are gorging on an ever expanding supply of open source components.

2. Vast networks of open source component suppliers are growing rapidly.

3. The massive variety and volume of software components vary widely in terms of quality.

4. Top performing enterprises, federal regulators and industry associations have embraced the principles of software supply chain automation to improve the safety, quality, and security of software.

We sincerely hope the findings and analysis in this report helped you better understand the impacts of open source components on your software supply chain.

Thank you.

Team Sonatype

OUR SOURCES

[0] State of the Software Supply Chain Report 2015

[1] Analysis of annual download requests from the Central Repository in calendar year 2015.

[2] Oracle's estimate for the number of Java developers worldwide in 2015.

[3] Sonatype's 2014 open source development and application security survey

[4] Analysis of annual downloads from the Central Repository in calendar year 2007 - 2015.

[5] http://thenewstack.io/a-brief-look-at-immutable-infrastructure-and-why-it-is-such-a-quest/

[6] http://www.theregister.co.uk/2016/03/23/npm_left_pad_chaos/

[7] https://www.openhub.net

[8] https://www.usenix.org/system/files/login/articles/15_geer_0.pdf

[9] https://www.usenix.org/system/files/login/articles/15_geer_0.pdfp17

[10] https://adtmag.com/Blogs/WatersWorks/2014/01/Worldwide-Developer-Count.aspx

[11] RubyGems.org

[12] bower.io/stats/

[13] Docker keynote presentation from DockerCon 2015

[14] Docker keynote presentation from DockerCon 2015

[15] https://search.maven.org/#stats and State of the Software Supply Chain Report 2015

[16] Analysis of annual downloads from the Central Repository in calendar year 2015.

[17] Analysis of annual downloads from the Central Repository in calendar year 2015.

[18] Analysis of annual downloads from the Central Repository in calendar year 2014.

[19] Quote comes from video interview at https://youtu.be/zRfRKA178CM?t=12m6s

[20] Analysis of annual downloads from the Central Repository in calendar year 2015.

[21] Analysis of annual downloads from the Central Repository in calendar year 2014.

[22] Analysis of repository manager connections to the Central Repository 2011 - 2016.

[23] Analysis of annual downloads from the Central Repository in calendar year 2015.

[24] Sonatype research andso-co-it.com (http://bit.ly/1UmkcoD)p23 (F1) Analysis of annual downloads from the Central Repository in calendar year 2015.

[25] Sonatype analysis of 1000 repository managers in May 2016.

[26] Sonatype analysis of 1000 repository managers in May 2016.

[27] http://blog.sonatype.com/2015/11/nexus-firewall-quality-at-velocity/

[28] http://blog.sonatype.com/2015/11/nexus-firewall-quality-at-velocity/

[29] Analysis of downloads at 3000 organizations from the Central Repository in Q1'2016, then annualized.

[30] Analysis of annual downloads from the Central Repository in calendar year 2014.

[31] Analysis of downloads at 3000 organizations from the Central Repository in Q1'2016, then annualized.

[32] Analysis of downloads at 3000 organizations from the Central Repository in Q1'2016, then annualized.

[33] Analysis of downloads at 3000 organizations from the Central Repository in Q1'2016, then annualized.

[34] State of the Software Supply Chain Report 2015

[35] Sonatype research, May 2016

[36] Sonatype research, May 2016

[37] Sonatype research, May 2016

[38] Sonatype research, May 2016

[39] Cisco 2015 Midyear Security Report http://www.cisco.com/web/offers/lp/2015-midyear-security-report/index.html

[40] Analysis of annual downloads from the Central Repository in calendar year 2007 - 2015.

[41] A new narrative on cyber security, http://thehill.com/blogs/congress-blog/technology/278712-a-new-narrative-on-cyber-security

[42] Financial Services | Information Sharing & Analysis Center (FS-ISAC) - https://www.sonatype.com/software-security-control-white-paper

[43] [44] [45] http://www.aba.com/Training/Conferences/Documents/NCCB16_Mon_Ins%20and%20Outs%20of%20Cyber%20Insurance_buyers%20guide.pdf

[46] https://www.ftc.gov/system/files/documents/cases/160222asuscmpt.pdf

[47] https://www.ftc.gov/news-events/press-releases/2016/02/asus-settles-ftc-charges-insecure-home-routers-cloud-services-put

[49] FDA: Collaborative Approaches to Medical Device Cybersecurity, January 20, 2016 , http://www.fda.gov/downloads/MedicalDevices/NewsEvents/WorkshopsConferences/UCM489249.pdf

[49] http://www.ul.com/newsroom/pressreleases/ul-launches-cybersecurity-assurance-program/

[50] How to Put Software Assurance into Contracts: An Effort of the Department of Defense (DoD) Software Assurance (SwA) Community of Practice (CoP) Contract Language Working Group. February 2016.

[51] Energy Sector Control Systems Working Group's, "Cybersecurity Procurement Language for Energy Delivery Systems", April 2014.

[52] https://18f.gsa.gov/2014/11/26/how-to-use-more-open-source/

[53] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf

[54] Quote comes from video interview at https://youtu.be/zRfRKA178CM?t=12m6s

[55] https://www.youtube.com/watch?v=bbWFCK-GhxOs&feature=youtu.be&t=21m17s

[56] Quote comes from video interview at https://youtu.be/zRfRKA178CM?t=12m6s

THANKS TO & LEGAL DISCLAIMERS

Thank You

The making of this report was a team effort of employees at Sonatype and friends of ours within the community. Special thanks for their contributions -- big and small -- go to Derek Weeks, Joel Orlina, Bruce Mayhew, Matt Howard, Wayne Jackson, Mike Hansen John Martin, Samantha Mayhew, Nicole Forsgren, J. Paul Reed, Josh Corman, James Wickett, Paula Thrasher, Shannon Lietz, Nigel Simpson, Gareth Rushgrove, Shannon Sking, Jessica Dodson, Eric Bourget, and Clara Charbonneau.

Sample Sizes and Analysis

In any report of this size, there is a risk that sampling issues will arise due to the nature of the way data was collected. For example, all of the applications and repositories analyzed for this report came from organizations interested enough about software supply chain practices to engage Sonatype for an assessment of their components. We have taken care to present analysis on statistically significant sample sizes. Additionally, much of the data about component downloads is specific to Java components and the Central Repository -- representative of what we believe to be one of the most mature and managed sources of components for developers.

Because software supply chain management best practices are still in their early stages of adoption, the report spotlights a number of individual organizations that have shared their experiences for others to learn from.

A Word from our Legal Team


Sonatype