2017 STATE OF THE SOFTWARE SUPPLY CHAIN

Sonatype's 3rd annual report on managing
open source components to accelerate innovation.

Sonatype

Introduction

WAYNE JACKSON

Chief Executive Officer, Sonatype

We live in an application economy where software has shifted from being a driver of nominal efficiency gains to an enabler of new customer experiences and markets. Innovation is king, speed is critical, and open source is center stage.

To compete effectively on a global playing field, companies aren't just writing software — they're manufacturing it as fast as they can using an infinite supply of open source component parts, machine automation, and supply chain-like processes.

These trends have been documented in previous versions of our State of the Software Supply Chain report, and this year is no exception. The 2017 State of the Software Supply Chain Report blends a broad set of public and proprietary data with expert research and analysis to reveal the following:

- An insatiable appetite for innovation is fueling the ever expanding supply and demand of open source components
- Components of varying quality are flowing through development lifecycles and landing in production applications
- DevOps-native development teams are leveraging trusted software supply chains to improve quality and productivity

This year's report has similarities to previous years, but there are three differences worth noting. First, the analysis in this year's report extends beyond Java and includes supply chain findings for JavaScript, NuGet, Python, and Docker. Second, this year's paper includes a stronger emphasis on the emergence of DevOps and reflects on the evolution of modern IT organizations as they seek to transform from waterfall-native to DevOps-native software development. Lastly, this year's research delves deeper into the rapidly evolving role of regulation, legislation, and litigation with respect to open source governance and software supply chain management.

We're grateful for your interest in software supply chains. We hope you find the information useful and we welcome your feedback.

EXECUTIVE SUMMARY

Fear of Death:
The Driving Force Behind DevOps Adoption

Whether you're a bank, a drug maker, an auto maker, or a retailer, survival in today's application economy depends on your ability to innovate. Don't believe it? Ask broadcast and cable television companies about Netflix. Ask Ford and General Motors about Tesla. Ask Hilton and Marriott about Airbnb. Ask every taxi company in the world about Uber. Truth be told, every business in the world is subject to disruption from upstart competitors that seemingly emerge overnight. This stark reality is why 84% of enterprises are actively embracing DevOps and dedicating themselves to automating and scaling software innovation.[1]

The Market for Open Source:
Infinite Supply Meets Massive Demand

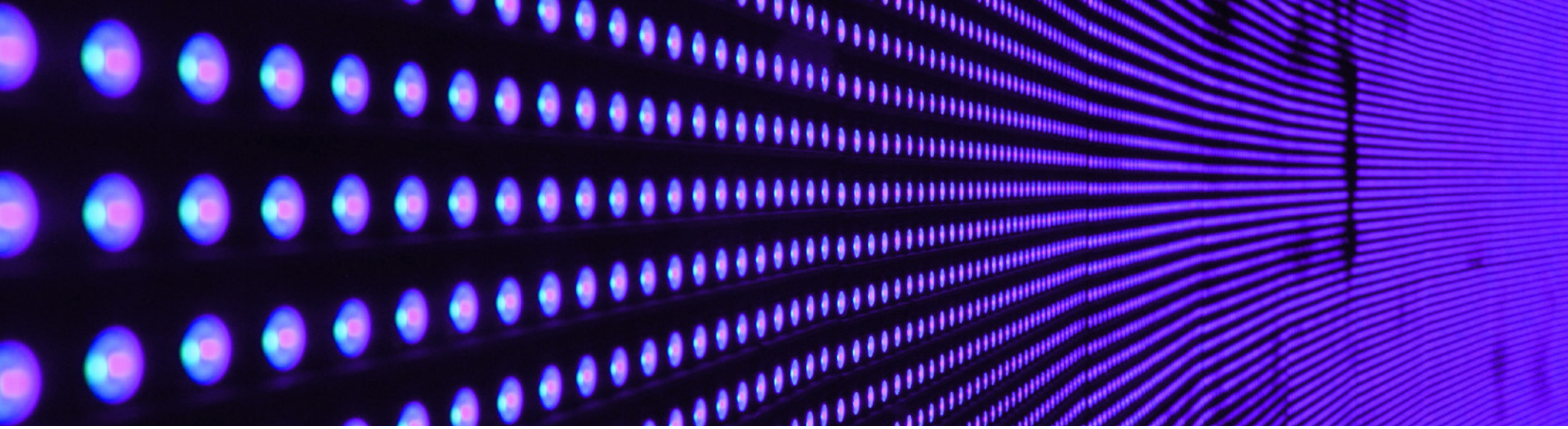
One hundred years ago -- long before the advent of reusable software components -- Robert Collier wisely observed, "that supply always comes on the heels of demand". Today, demand for open source components is growing exponentially. In the Java ecosystem alone, developers requested 17 billion components from the Central Repository in 2014, 31 billion in 2015, and 52 billion in 2016.[2] Demand for JavaScript components is even stronger. **In 2016, developers requested 59 billion components from the npm repository compared to 22 billion in 2015 -- a 262% year-over-year growth.[3]**

The byproduct of massive demand is massive supply. Each and every day the supply of open source across all ecosystems increases by about 1,100 new projects and 10,000 new versions.[4] With each new project releasing on average 14 times per year [5] -- the global supply of open source is increasing at an astonishing rate.

Better Parts and Better Quality:
Why You Should Care

Faced with an infinite supply of open source components, organizations must come to grips with three simple facts: (1) components are not created equal, (2) production applications use components of varying ages and quality, and (3) younger components are 3X healthier than older components.

Today's average application contains over 190 open source components. Data from over 17,000 applications reveals that applications built by teams **utilizing automated governance tools reduced the percentage of defective components by 63%.**



Taming the Beast with DevOps: How to Automate and Scale Your Software Supply Chain

In order to transform from waterfall-native to DevOps-native innovators, organizations must be do three things: (1) view software development as a single supply chain (not a collection of silos) and commit to never passing known defects downstream, (2) embed knowledge throughout the supply chain to create instant feedback loops so errors and defects can be continuously corrected, and (3) create a culture comfortable with continuous experimentation, risk taking, and learning from failure.

For organizations who tame their supply chains, the rewards are impressive: **28% improvement in developer productivity, 30% reduction in overall development costs, and 48% increase in application quality**. In one case, a large financial services firm eliminated 136,000 hours of manual governance and generated $13 million in annual savings. Those who don't will suffer the consequences of reduced quality, higher costs, and disruption from competitors.

Sticks and Carrots: The Inevitable Role of Regulation and Risk

In addition to "fear of death", a collection of regulatory, legislative, and judicial drivers are emerging in a parallel universe and beginning to influence how organizations approach DevOps and the practice of software supply chain automation. **Embedding security early in development** and using a **software Bill of Materials** continue to emerge as industry best practices.

Software Supply Chains by the Numbers

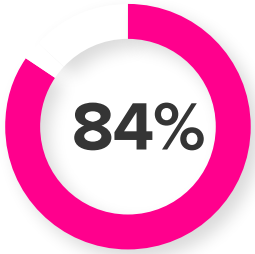
THE FACTS

page 15

Year over year growth	Annual download records
68%	52B
262%	59B
100%	12B

1,096 New OSS projects launched every day page 13

THE CHALLENGES



page 14

OSS Projects DON'T FIX known security defects

page 6

1-in-18 downloads contained at least one known security vulnerability

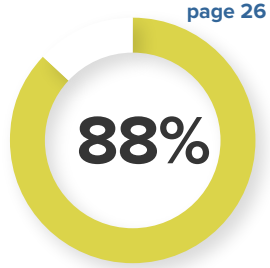
page 38

12 FTC cases revealed failure to assess applications for vulnerabilities

page 36

£100,000 FINE for not preventing a cyber attack exploiting an OSS vulnerability

page 26



are concerned about container security

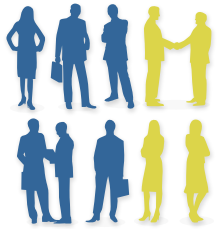
THE OPPORTUNITY

page 32

Managed software supply chains boost quality outcomes by

48%

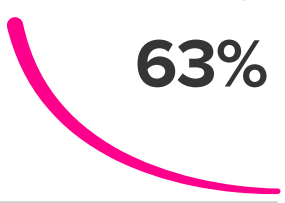
page 21



6-in-10 have policies to guide quality decisions

page 24

Automated governance reduced defects by



page 31

Reduced deploys from 25 to 2.5 DAYS with trusted supply chain methodology

page 30

Automated governance saved 34,000 hours over manual workflows

page 32

Review processes went from weeks — to → zero time

TABLE OF CONTENTS

Introduction 2

Executive summary 4

Chapter 1 - Fear of Death: The Driving Force Behind DevOps Adoption 8

The DevOps-native world of continuous everything 9

Open source: the miracle drug of choice for modern software innovation 10

Save time and money 10

Improve quality 10

Deliver business agility 10

Mitigate business risk 10

Chapter 2 - The Market for Open Source: Infinite Supply Meets Massive Demand 11

Software supply chains are ubiquitous 12

An infinite and ever-growing supply of parts 13

Suppliers are not created equal 14

Insatiable demand for component parts 15

Be aware of what you eat 17

Chapter 3 - Better Parts and Better Quality: Why You Should Care 18

80 - 90% of every application is built from open source components 19

Components age like milk, not like wine 20

Emergence of open source governance programs 21

Local warehouses and inventory controls 22

Defective components swim downstream 23

Building software the Toyota way 24

Questionable Consumption: Bouncy Castle and Struts2 25

The rise of containers, clouds, and infrastructure as code 25

Containers and security: more questions than answers 26

Chapter 4 - Taming the Beast: How to Automate and Scale Your Software Supply Chain 27

Appreciation of the System 28

SPOTLIGHT: Stop bad components at the front door 30

Fast feedback loops 31

Continuous experimentation and learning 32

SPOTLIGHT: PayPal's remediation at scale 33

Chapter 5 - Sticks and Carrots: The Inevitable Role of Regulation and Risk 34

The White House 35

U.S. Commission on Enhancing National Cybersecurity 35

Britain's national cybersecurity strategy 36

U.S. Department of Homeland Security 37

U.S. Department of Health and Human Services 37

U.S. Federal Trade Commission 38

U.S. Department of Commerce 38

U.S. Automotive Industry 38

Cyber insurance premiums - mounting existential risk 39

Summary 40

Footnotes 41

Appendix 43

CHAPTER 1

Fear of Death

Chapter 1 - Fear of Death: The Driving Force Behind DevOps Adoption

In the modern economy if you're not innovating fast enough, you'll get run over by someone else who is.

The DevOps-native world of continuous everything

For every company in every industry, competition today is as likely to come from a startup that you've never even heard of as it is from long established rivals. In the modern economy, if you're not innovating fast enough, you'll get run over by someone else who is.

Time and time again we've seen examples of once-dominant companies -- with waterfall-native development processes -- being completely disrupted by innovative upstarts. The biggest challenge to incumbents are deeply embedded cultural norms and long-standing silos erected by software development, application security, and IT operations teams which create friction, decrease velocity, and diminish innovation.

So, whether you're a bank, a drug maker, an auto maker, or a retailer, survival depends on your ability to innovate. This stark reality is why many organizations no longer view software development as a cost of doing business, but rather as a core competency and strategic imperative that defines the business. It's also why organizations around the world are increasingly embracing DevOps and dedicating themselves to innovating faster than their competitors.

Evidence of this trend is seen in the Annual State of the Cloud Survey [6] conducted by Right Scale which reveals the following:

- DevOps adoption increased from 66% in 2015 to 74% in 2016.
- DevOps adoption is strongest in the enterprise -- 81% of enterprises adopting compared to 70% of SMBs.
- DevOps adoption is occurring from the bottom up -- 29% of teams, 31% of business units, and 21% company wide.

Separately, the 2017 State of DevOps Report [7] conducted by Puppet Labs and DORA provides strong evidence that organizations adopting DevOps practices are experiencing remarkable results, including:

- DevOps teams deploy 46X more frequently -- meaning they deploy multiples times per day instead of once a week or less.
- DevOps teams deploy 440X faster -- meaning they have lead times of less than an hour instead of more than a week.
- DevOps teams recover from downtime 96X faster -- meaning they recover in less than an hour instead of days.
- DevOps teams have a 5X lower change failure rate -- meaning changes to production fail 7.5% of the time instead of 38.5%.

Perhaps the venerable General Electric's recent decision to move the company's headquarters from suburban Connecticut to downtown Boston offers the best example of how "fear of death" is driving big change. When asked by reporters to explain the move, CEO Jeff Immelt said, "we're moving the company to Boston because I want some 29-year-old MIT graduate to punch me right in the nose and say all of GE's technologies are wrong and you're about to lose."

Open source: the miracle drug of choice for modern software innovation

Software innovation is the primary mechanism by which modern companies are competing and winning on a global playing field. Thus, the pressure to innovate faster and better is incredibly intense. In response to this pressure, software development teams are not only turning to DevOps, but they are also turning to open source software components for four simple reasons.

1. Save time and money

Long before the advent of open source software innovation, Isaac Newton famously said, "I see further by standing on the shoulders of giants and I discover truth by building on previous discoveries." This exact concept is a primary reason why open source is so attractive. Simply stated, free and open access to pre-existing software components eliminates the reinvention of wheels and enables organizations to save significant time and money.

2. Improve quality

Linus's Law formulated in 1999 by Eric S. Raymond in his book *The Cathedral and the Bazaar* states that "given enough eyeballs, all bugs are shallow".[8] In other words, if a particular piece of software is exposed to a large enough community of co-developers and beta-testers, then problems will be easily identified and quickly fixed. This simple concept is why open source components lead to higher quality software applications and why organizations such as General Motors, General Electric, American Airlines and Bank of America readily embrace it.

3. Deliver business agility

Survival in the modern world requires organizations to react quickly to a rapidly changing landscape of competitive threats and strategic opportunities. Open source increases agility for developers and businesses alike by speeding up the pace of software development. Software developers and corporate CEOs are both allergic to waste and instead prefer to invest their time toward innovation. Given the choice of spending 15 hours building something from scratch or 15 minutes polishing a piece of code from the community, both the developer and CEO will almost always choose open source. Furthermore, companies that use open source are not tied to a proprietary vendor. Controlling one's destiny and maximizing flexibility is yet wanother reason why open source maximizes business agility.

4. Mitigate business risk

Another benefit to open-source is reduced business risk. Vendors come and go, and commercial priorities change, while the focus for an entire community can remain constant for years and even decades. The openness and transparency of open source communities mitigate risk -- especially when organizations source the best components from the best suppliers.

CHAPTER 2

The Market for Open Source

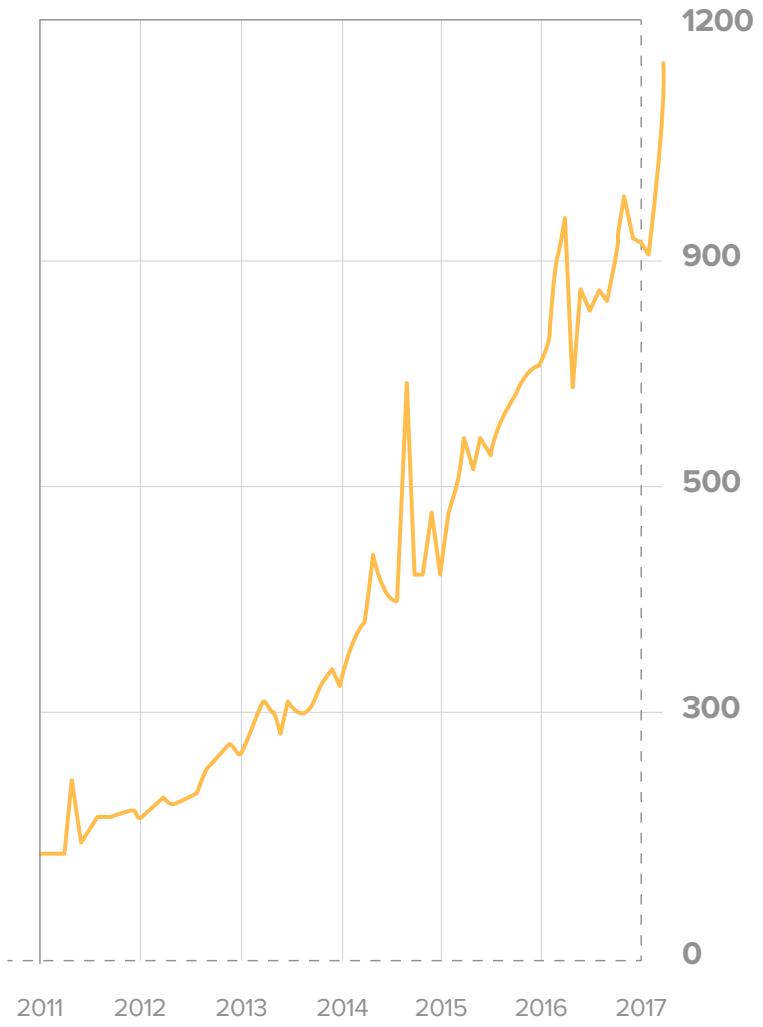
Chapter 2 - The Market for Open Source: Infinite Supply Meets Massive Demand

Consumption of open source is so vast that most organizations can not identify how many components are entering into the software supply chain, where they are flowing through the development lifecycle, or where they might exist in deployed applications.

Software supply chains are ubiquitous

To fuel innovation at DevOps-native velocities, every company -- whether they know it or not -- depends on a software supply chain. These software supply chains are comprised of thousands of open source suppliers (projects) who produce millions of parts (components and versions) each year. These supplier parts are then consumed billions of times each year by software development teams. These parts are then assembled into applications. Finished applications are then transitioned into production environments and managed by IT operations. Once in production, applications deliver value to customers and users in the form of a product or service.

Average number of new OSS Projects coming to market per day



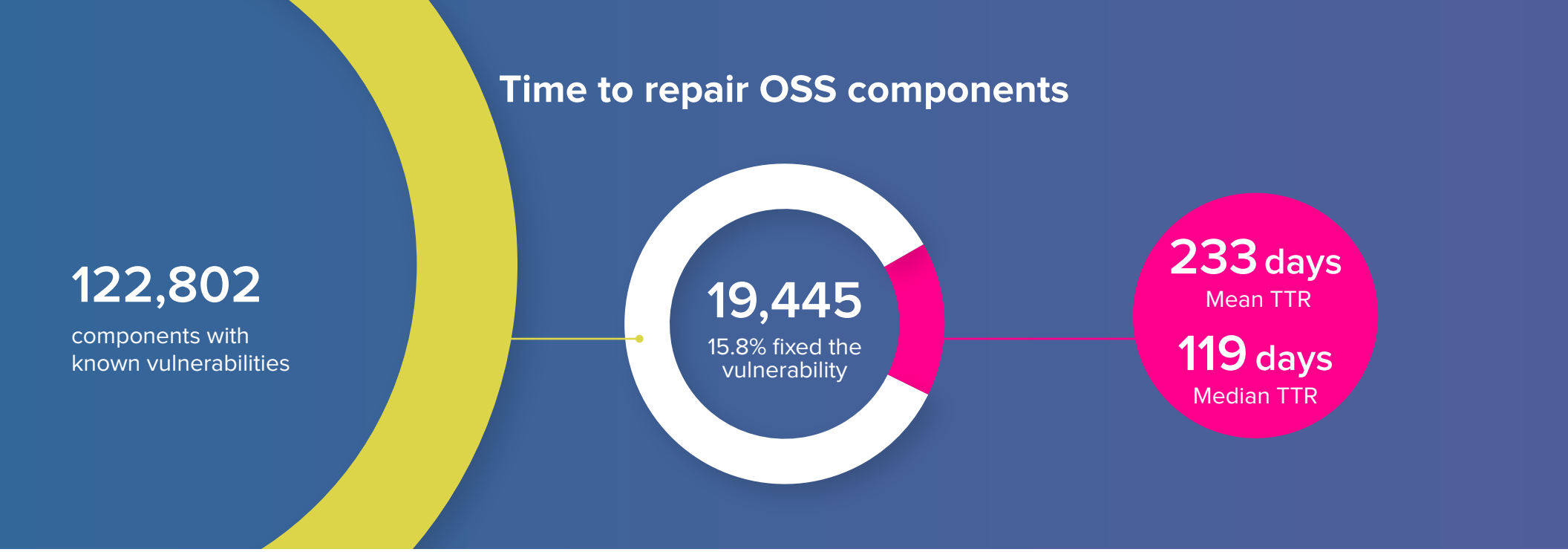
Today 80% to 90% of every modern application is comprised of open source component parts.[9] **Consumption of open source is so vast, that most organizations can not identify how many components are entering into the software supply chain, where they are flowing through the development lifecycle, or where they might exist in production applications.**

Research reveals that open source hygiene is inconsistent and dynamic across supplier projects and individual components. Blindly trusting the quality of open source parts flowing into development lifecycles introduces significant risk for organizations. Conversely, as we will show in Chapter 3, by actively governing the flow of open source components across the software supply chain, organizations can improve developer productivity and reduce vulnerabilities by 63%.[10]

An infinite and ever growing supply of parts

The supply of open source components and containerized applications is massive. There are now more than two million Java unique components in the Central Repository,[11] almost three million unique JavaScript packages in npmjs.org,[12] over 870,000 unique Python components housed in PyPI repository,[13] and over 900,000 .NET components in the NuGet Gallery.[14] There are also more than 900,000 containerized applications [15] housed in Docker Hub -- up from 460,000[16] the previous year.

This massive supply of software parts is rapidly and organically expanding due to new innovations and regular versioning of existing components. Analysis of multiple development ecosystems tracked at modulecounts.com reveals that 1,096 new open source projects (suppliers) are introduced every day. [17] Furthermore, across all open source projects, more than 10,000 new component versions are released daily offering new features, improved performance, bug fixes, and security patches.[18]



Suppliers are not created equal

We recently analyzed 122,802 different open source components with known security vulnerabilities and discovered the following: a small minority of suppliers were quick to remediate known vulnerabilities, some were slower to remediate, and the majority of suppliers simply failed to remediate. Specifically, we found that only 15.8% of suppliers (19,445) actively fix vulnerabilities, while **84% do not actively remediate known security defects.**[19]

Of the 19,445 open source projects that actively fix security vulnerabilities, the **mean time to remediation (MTTR) was 233 days**. While the absolute best suppliers remediated known vulnerabilities in just two days, the median time to repair for these open source projects was 119 days. There were 1,529 projects that remediated vulnerabilities in less than a week. By contrast, over 800 suppliers had MTTRs for known vulnerabilities greater than three years (1,095 days), with 44 projects showing repair times over seven years (2,500 days).[20]

Because suppliers are not equal, organizations would be wise to actively govern which open source projects they work with and which components they ultimately consume. Simply stated, organizations that increase their awareness of supplier performance will inevitably consume higher quality open source parts and build better software applications for less.

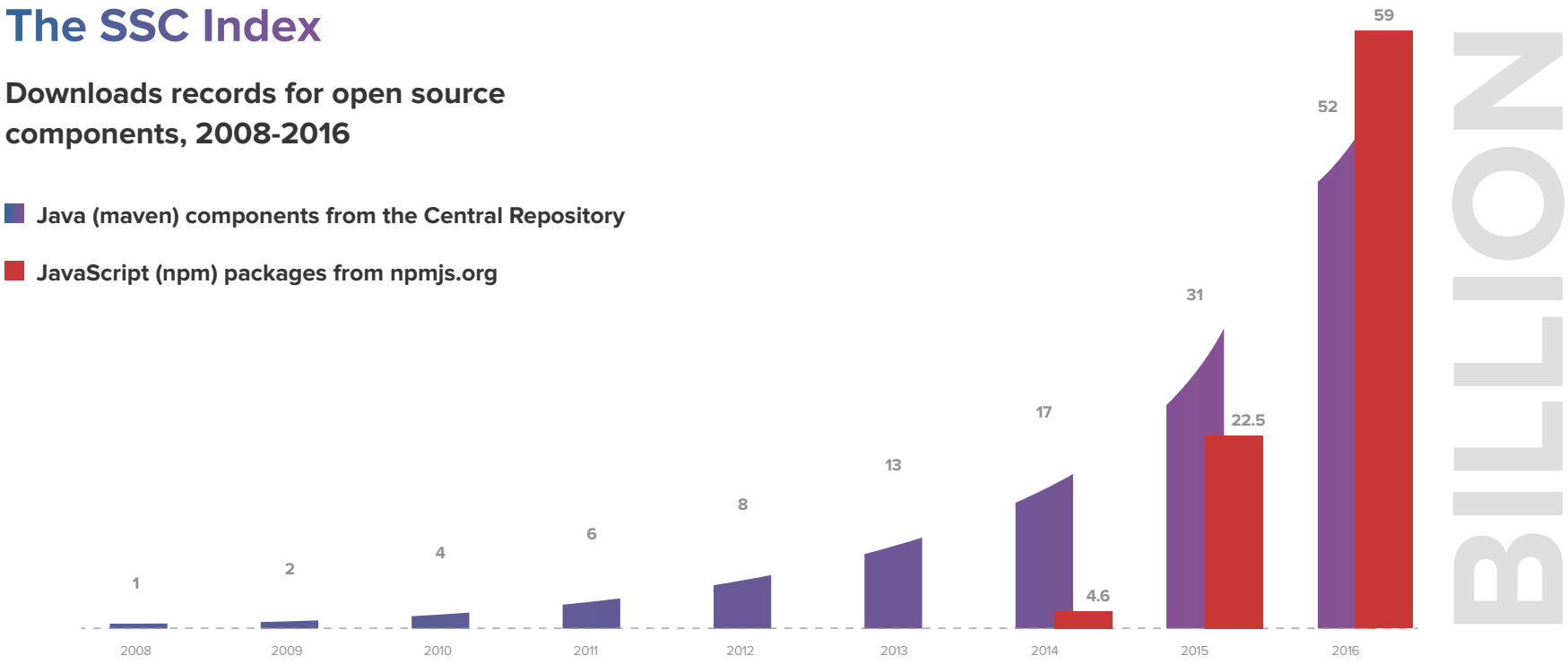
Insatiable demand for component parts

The ever expanding supply of open source projects is driven by an insatiable demand for innovation. Today, a mind-boggling number of components and containers are downloaded and consumed in "all-you-can-eat" fashion from public warehouses like the Central Repository, NuGet Gallery, npmjs.org, ruby-gems.org, Docker Hub and others.

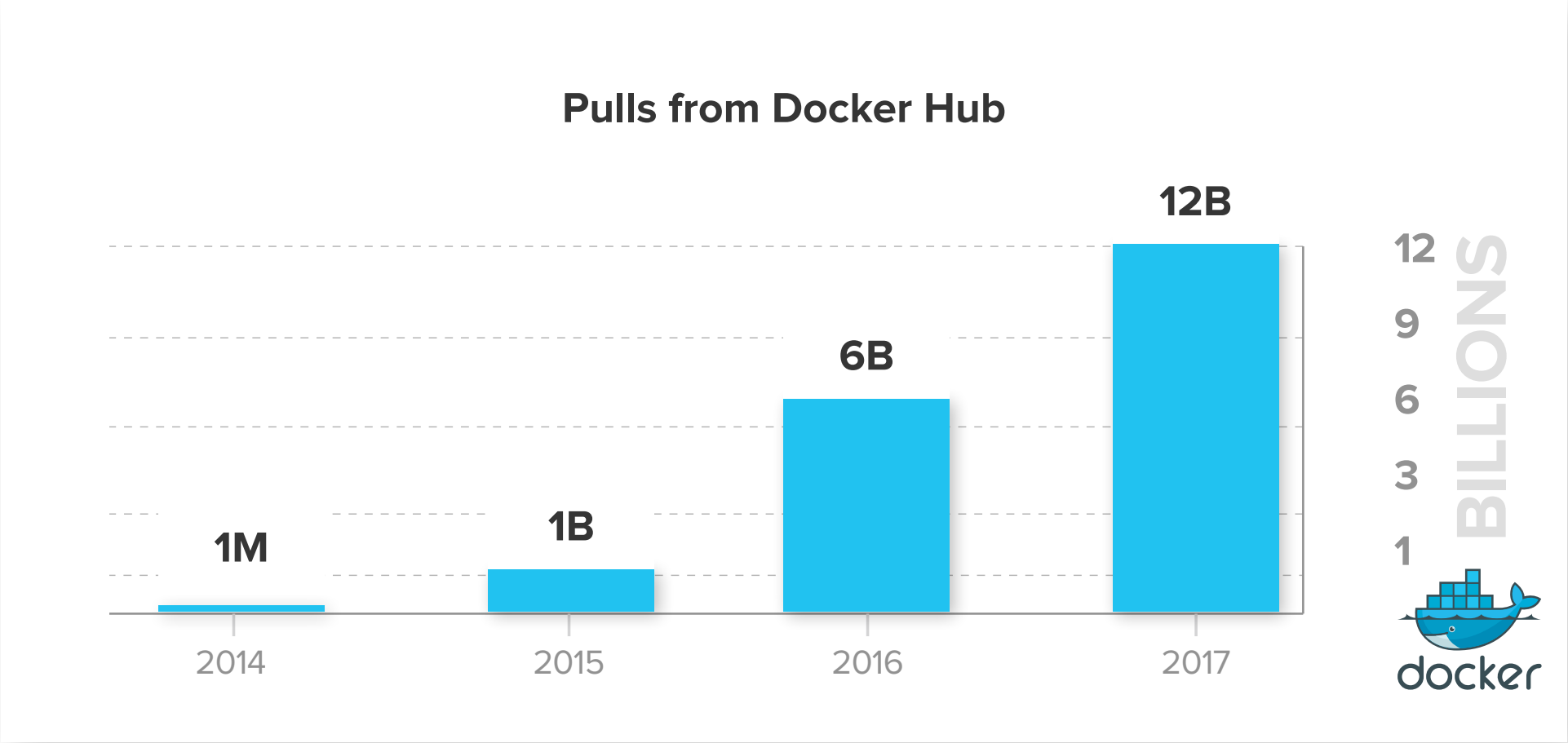
In 2016, the number of Java components downloaded from the **Central Repository grew 68% year-over-year representing an increase to 52 billion from 31 billion.** During this same period, the number of components downloaded from the NuGet gallery was 3.4 billion compared to the previous year of 756 million representing a 347% year over year increase.[21] Similarly, in 2016 the number of

The SSC Index

Downloads records for open source components, 2008-2016



JavaScript components downloaded from the npmjs.org repository grew 262%, representing 59 billion packages served.[22] In February 2017, the PyPI repository reported serving 4.4 billion download requests from its repository.[23] Docker recently forecasted that IT professionals will pull 12 billion containers from the Docker Hub in 2017 compared to just 6 billion in 2016.[24]



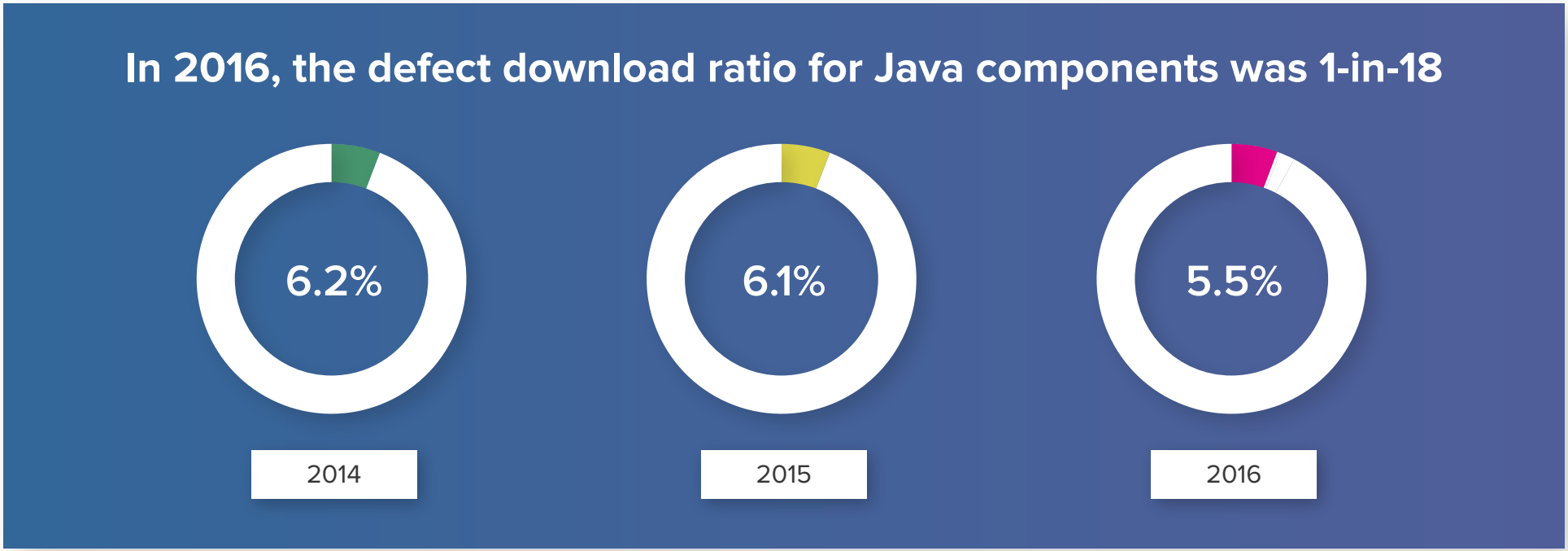
Be aware of what you eat

An analysis of 1.8 million Java components housed in the Central Repository revealed that 5.9% contained a known vulnerability. Similar analysis of 3 million JavaScript components housed in npmjs.org found 6.2% contained a known defect. Finally, an analysis of 873,000 Python components from the PyPI repository discovered that 3.6% were vulnerable.[25]

Because public repositories are immutable by design, vulnerable components are not proactively removed from inventory. Instead, it is incumbent upon development organizations to practice good hygiene when consuming open source components. Teams with suboptimal hygiene inevitably consume open source components with critical vulnerabilities. **In 2016, 5.5% (1 in 18) of Java components downloaded from the Central Repository contained known security vulnerabilities.**[26]

Although a 5.5% defect download ratio is far from perfect, there is empirical evidence that hygiene is beginning to improve with defect download ratios declining slightly in each of the last three years.

For this year's report, we examined 7,500 organizations and studied their consumption of Java components from the Central Repository. **The average enterprise downloaded 125,701 components in 2016.** Further, for Java alone, we found that organizations consumed on average 3,185 unique component versions (parts) from only 1,346 open source projects (suppliers). Deeper analysis of component downloads across 7,500 organizations revealed that 7,248 (5.8%) of their 125,701 downloads from the Central Repository had known security vulnerabilities.[27]



CHAPTER 3

Better Parts and Better Quality

Chapter 3 - Better Parts and Better Quality: Why You Should Care

While the majority of organizations evaluate open source components at some stage of the development lifecycle, the reality is that defects continue to make their way downstream into production applications.

By using younger open source components from higher quality suppliers, development teams can accelerate innovation. Conversely, teams that fail to manage their software supply chains will suffer the consequences of using poor quality or known defective open source components.

Unmanaged software supply chains lead to unforeseen costs. This problem is amplified as infrastructure is converted to code and applications are deployed to production in a continuous fashion. **When defective open source components are permitted to pass downstream within a software supply chain three things happen: vulnerabilities increase, quality degrades, and the pace of innovation dramatically decreases.**

80 - 90% of an application is built from components

Applications are no longer built from scratch. They are assembled from open source and third party components. A recent survey 2,292 IT professionals found that 80 - 90% of an application now consists of component parts.[28]

A Sonatype assessment of 386 applications found similar results with 82% of the applications built from open source components. The typical application contains an average of 182 open source components.[29]

A recent post on the npm blog reveals, "**It's common for a modern JavaScript project to depend on 700–1200 packages.** Ten years ago, the JavaScript world was dominated by a handful of very large libraries, like YUI, Mootools, and jQuery. These "kitchen sink" libraries tried to cover every use case, so you

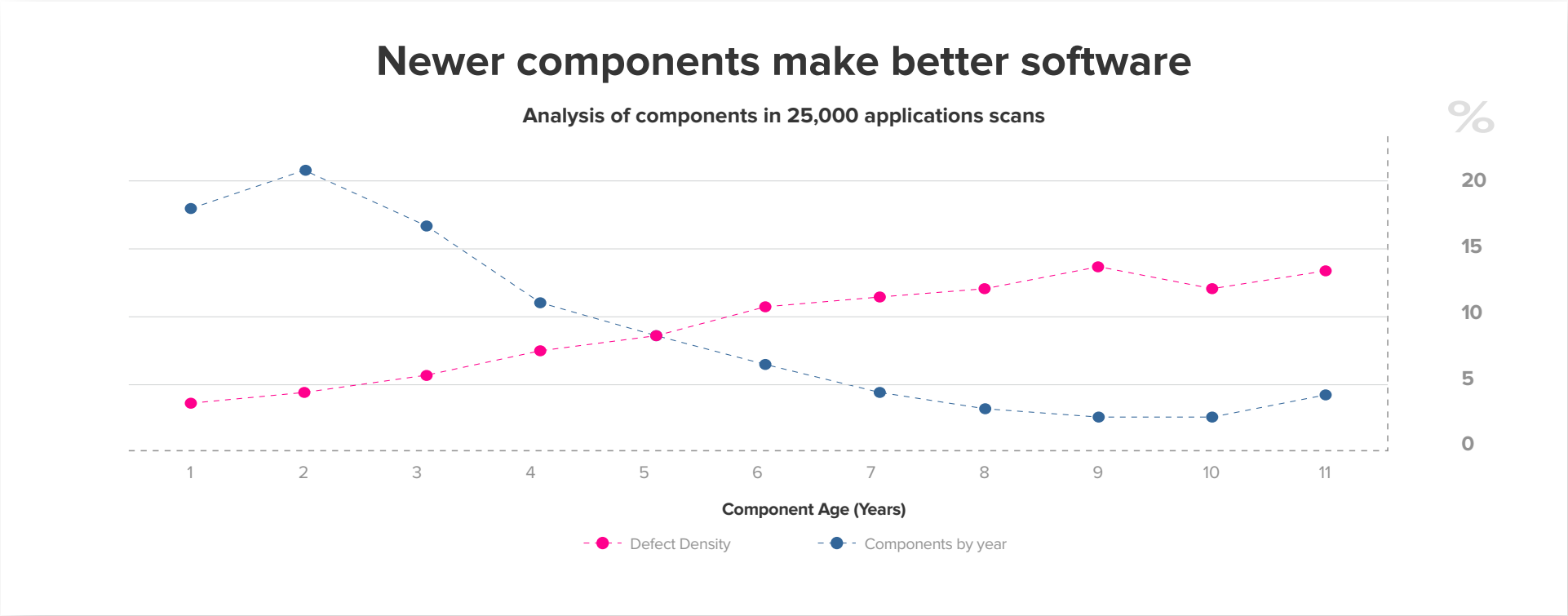
would probably pick one and stick with it. In the past few years, use of semantic versioning (SemVer) enabled easier management of modules. Thus, the use pattern of 'many small modules' became popular".[30]

Components age like milk, not like wine

Analysis of 25,000 applications from last year's report 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 8 and 11 years old had nearly 3x the known security defect rate.[31]

For the components between 8 and 11 years old, further 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. Unfortunately, when defects are discovered in older components, chances of remediating the issue by upgrading to a newer component version are greatly diminished.[32]

Better selection, resulting from open source component analysis and governance, not only improves the quality of the finished application, it also reduces the number of break-fixes and unplanned work.



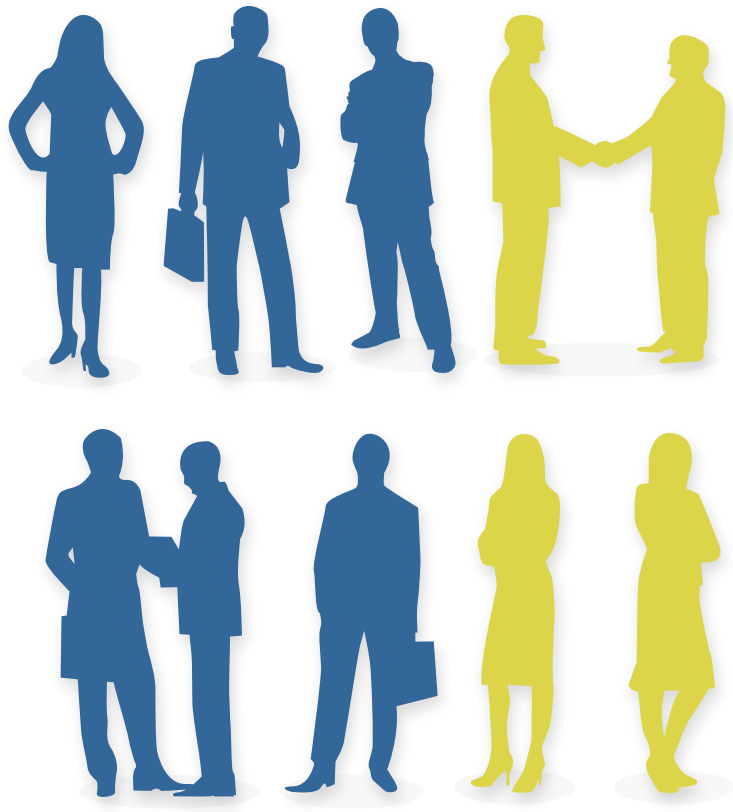
Emergence of open source governance programs

Jeffrey Liker, author of *The Toyota Way* remarked, “Things like ISO-9000, an industry quality standard that calls for all kinds of detailed standard operating procedures, for whatever good they have done, have made companies believe that if they put together detailed rule books the rules will be followed”. In the realm of software supply chains, the rule books are often built by open source governance, security and development teams.

The 2017 DevSecOps Community Survey [34] asked “Does your organization have an open source governance policy?” to which 57% answered affirmatively. Open source governance policies introduce guidelines regarding quality, security, and license attributes similar to ISO standards used in manufacturing.

The good news is that nearly **6-in-10 organizations have policies in place to guide quality decisions** for open source component use. The bad news is that more than 1-in-4 organizations either have no policies in place, or their IT professionals are not aware of the policies; that is, no quality standards are in place to guide the sourcing of components.

Only 6-in-10 organizations have an open source governance policy in place



Local warehouses and inventory controls

Local repository managers and container registries are private assets used as local parts warehouses within software supply chains. Use of local repository managers that act as a conduit between developers and the public repositories is up over 300% in the past three years.[35] Repository managers fetch new components upon request and then cache them locally for reuse. Repository managers also provide a secure and private location to house proprietary components and containers that are developed internally.

In 2016, 2.4% of the downloads from the Central Repository were triggered by repository managers.[36] While the percentage is not very large, keep in mind that a repository manager will only download a component once. Once cached, future downloads of that component are unnecessary. Sonatype analysis of over 40,000 Nexus repositories reveals that **the average repository holds over 1,600 components.**[37]

Analysis of Java component downloads from the Central Repository to repository managers reveals that **7.2% (1-in-14) had at least one known security vulnerability.** Deeper analysis of the 1600 components housed in the average repository manager found 192 security vulnerabilities were present among the components (some components having more than one security vulnerability).[38]

Within software supply chains, repository managers and private container registries represent procurement gates into the development organization. The gates can be left wide open where component flows are not governed or they can represent opportunities for quality and security checkpoints that ensure defects are not passed downstream.

Defective components swim down stream

While the majority of organizations evaluate open source components at some stage of the development lifecycle, the reality is that defects continue to make their way downstream into production applications.

Using data from over **133,000 websites, researchers from Northeastern University showed that "37% include at least one library with a known vulnerability.** From a per-library perspective, at least 36.7% of jQuery, 40.1% of Angular, 86.6% of Handlebars, and 87.3% of YUI inclusions use a vulnerable version.

Alarmingly, many sites continue to rely on libraries like YUI and SWFObject that are no longer maintained. In fact, the median website in our dataset is using a library version 1,177 days older than the newest release, which explains why so many vulnerable libraries tend to linger on the Web".[39]

Defect percentages for JavaScript packages

87%

of Handlebars inclusions were known vulnerable

37%

of jQuery inclusions were known vulnerable

40%

of Angular inclusions were known vulnerable

This year, Sonatype examined 386 applications containing more than 70,000 open source components. Analysis revealed 355 of 386 (92%) had at least one known vulnerable component. Of the 355 applications, 4.6% of the open source components in use had at least one known security vulnerability; the average application in this set has 20 known security vulnerabilities, 6 of which had a Common Vulnerability Scoring System (CVSS) rating between 7 and 10.[40]

Building software the Toyota way

Evidence clearly shows that defective and vulnerable open source components swim downstream from one end of the software supply chain to another. In order to prevent this from happening, teams are embracing automated open source governance tools to continuously evaluate applications and monitor open source hygiene early and everywhere across the the development lifecycle.

On average, applications built by teams utilizing manual governance processes have a defective component ratio of 4.6%. Conversely, an examination of **17,000 applications built by teams utilizing automated open source governance tools revealed a defective component ratio of only 1.7% -- which equates to a 63% reduction in the use of vulnerable components.**[41]

Warehouses

5.5%
component downloads are vulnerable

Manufacturers

7.2%
components downloaded to repository are vulnerable

Finished Goods

4.6%
components in applications are vulnerable in **unmanaged** supply chains

1.7%
components in applications are vulnerable in **managed** supply chains

Questionable consumption: Bouncy Castle and Struts2

Although it's concerning when development teams utilize open source components with known security vulnerabilities, it's downright shocking when developers utilize a vulnerable component despite the fact that a healthy and safe version of the same component is readily available.

Consider the example of Bouncy Castle, a popular and well known cryptographic library for Java developers. In 2016, 197 versions of the Bouncy Castle library were available -- which were downloaded 23.4 million times by development teams. Of those 197 versions, 61 (34%) had known security vulnerabilities -- while 136 (66%) were perfectly healthy. Despite healthy versions of Bouncy Castle being readily available -- development teams downloaded defective versions 48% of the time -- 11.2 million in total.[42]

Similar download behavior was seen for the Commons Collection component in 2016 when 78% of 23.5 million downloads were known vulnerable versions.[43] Catastrophic ransomware attacks at San Francisco's Mass Transit Authority (SMFTA),[44] MedStar,[45] and at Hollywood Presbyterian Hospital were attributed to vulnerabilities in Commons Collection.[46]

Another well-publicized security vulnerability was announced in the Struts2 component in March 2017. Analysis of the Struts2 vulnerability (CVE-2017-5638) revealed that 2,731 organizations had downloaded the vulnerable versions of the project 279,796 times over the prior 12 months.[47] Shortly after this vulnerability was announced, related site outages occurred at the Canadian Revenue Agency,[48] Statistics Canada,[49] GMO Payment Gateway,[50] Japan Post,[51] and Okinawa Electric Power.[52]

While organizations derive tremendous value from open source software components, they also suffer painful consequences of failing to manage their software supply chains.

The rise of containers, clouds, and infrastructure as code

As consumption of open source software components grows exponentially, various other forces are simultaneously causing tectonic shifts on the IT landscape, including:

- **Infrastructure as code:** the rise of "infrastructure as code" means operational environments are increasingly controlled and orchestrated by software.
- **Containers:** the march toward componentization and microservices continues with containers -- which conveniently serve as the "production dumping ground" for good and bad open source software components.
- **Cloud:** cloud-based operational patterns, such as immutable infrastructure, require us to rethink how we package software and directly impact how we build it, deploy it, and operate it.

Effectively, code has inserted itself in between our own software products and the hardware and operational environments on which it runs. Infrastructure as code, containers, and other code sourced from outside an organization are evidence of software supply chains at work. With greater accessibility to these new components comes greater efficiency but also the responsibility for managing them properly.

Containers and security: more questions than answers

The 2017 DevSecOps community survey revealed **88% of respondents were concerned about security when deploying containers into production.**[53] Security in the container realm is a constant concern with even the largest technology providers taking notice: "We've made it our goal to secure the global software supply chain from development, to test, to production," said Nathan McCauley, Director of Security at Docker.[54]

Research from Banyon Ops in 2015 found that security flaws in Docker images are common. Analysis of images hosted on Docker Hub revealed that "more than 30% of official repositories contain images that are highly susceptible to a variety of security attacks (e.g., shellshock, heartbleed, poodle, etc.). For general images (images pushed by Docker users, but not explicitly verified by any authority) this number jumps up to ~40% with a sampling error bound of 3%". Banyon Ops research went on to say "3 out of every 4 images created have vulnerabilities that are relatively easy to exploit with a potentially high impact." [55]

A June 2017 report indicates that container security has improved somewhat since the Banyon Ops data was released. According to Cloud Technology Partners, "The native Docker runtime has made significant improvements over the last year, including the ability to invoke out-of-the-box Seccomp (secure computing mode) profiles. These container profiles can disable 52 system calls by default. But you still have 313 system calls on x64 machines. Do the math; that leaves 261 system calls still open for attack." [56]

CHAPTER 4

Taming the Beast with DevOps

Chapter 4 - Taming the Beast with DevOps: How to Automate and Scale Your Software Supply Chain

Developers no longer wait weeks for approval. Instead, they continuously incorporate new components, build new features, and ship higher quality software faster.

Poorly managed software supply chains lead to sub-optimal and defective open source components flowing into production applications and expose organizations to cyber attacks, bug fixes, and expensive rework.

Well-managed software supply chains deliver benefit across the entire value stream, including: (1) upstream, where applications are developed, (2) midstream, where applications are tested and staged for release, and (3) downstream, where applications run in production and deliver value to users.

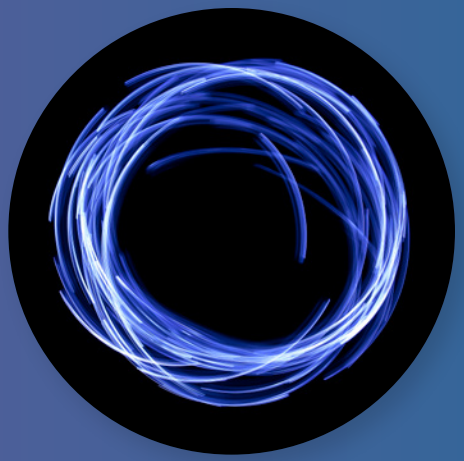
The best way to automate a software supply chain is to embrace DevOps patterns and practices known as “The Three Ways”.[57]

Appreciation of the system

The First Way of DevOps emphasizes the performance of the entire system, as opposed to the performance of a specific silo of work. With regard to software supply chains, the system itself is composed of open source projects, public repositories, open source components, containers, development pipelines, software applications, and the flow of artifacts and information between these elements.

When components flowing into a software supply chain system are not tracked and managed, finished applications will be fragile and unreproducible. Alternatively, if component inputs are monitored across every phase of the entire software supply chain then system visibility is enhanced and application quality improves.

Gene Kim's First Way of DevOps emphasizes "never passing a known defect to downstream work centers, never allowing local optimization to create global degradation, and always seeking to increase flow." Viewing the software supply chain as a complete system enables one to envision a factory-like process by which applications are manufactured using the highest quality components sourced from the highest quality suppliers.



SPOTLIGHT

Stop bad components at the front door

To automatically control the quality of components entering an organization, Gartner recommends that organizations "prioritize OSS software module identification and vulnerability scanning during development" and "implement an 'OSS firewall' to proactively prevent developers from downloading known vulnerable code from Maven, GitHub and other OSS code repositories by policy."[58]

Automating perimeter defenses as part of managing the software supply chain was precisely the challenge faced by a large multinational financial services corporation. During a three month period, developers at the company downloaded 27,000 components from public, internet-based repositories to incorporate into their software application builds.

Beholden to a manual governance process, the company's developers waited anywhere from 7 to 21 days for approval to use a new open source component. Seeking to "increase flow" the company implemented an OSS Firewall to automatically govern the components entering into the software supply chain. In just 90 days and with no human intervention required, the OSS Firewall automatically quarantined 850 components and identified an additional 1,500 as violating defined security and or licensing policy. Compared to manual governance, automating **the component approval process saved the company 34,000 hours in just 90 days.**

By implementing automated perimeter controls for OSS governance, the company created a "real time" approval process. Developers no longer waited weeks for approval. Instead, they could continuously incorporate new components, build new features, and ship higher quality software faster.

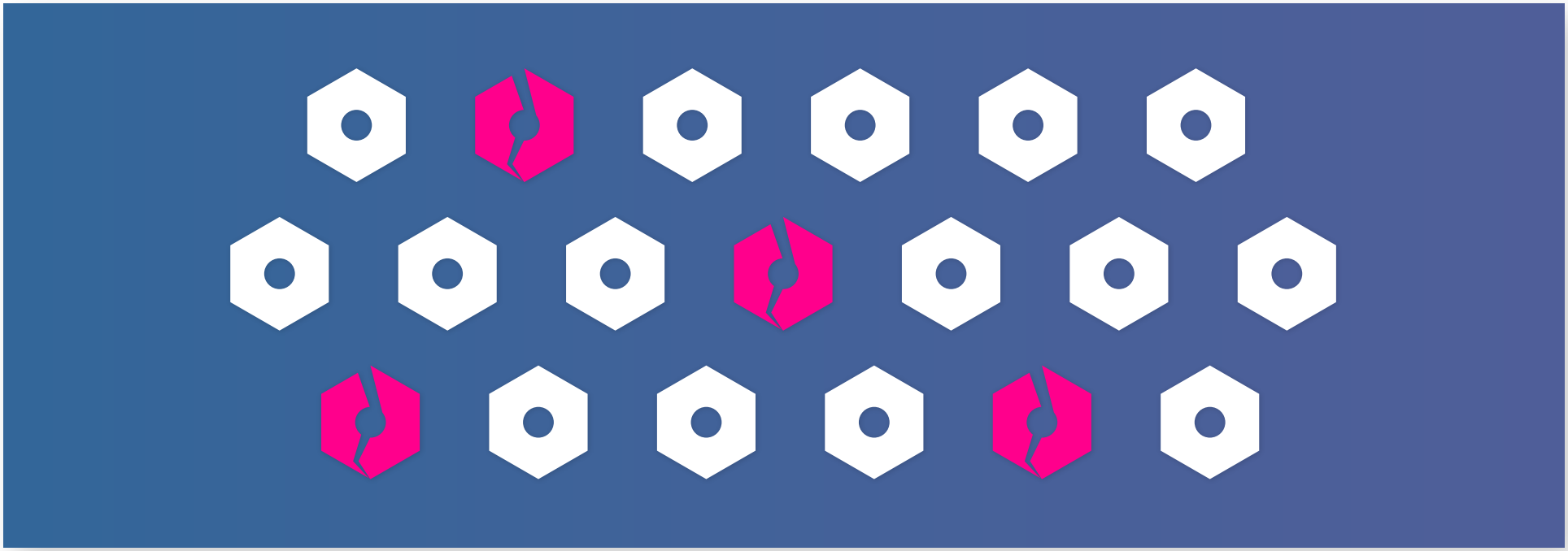
Fast feedback loops

The Second Way of DevOps is to create feedback loops by embedding intelligence directly into the supply chain system. When mistakes are made, developers are informed instantly so corrective action can be taken continuously across every phase of the development lifecycle.

Open source components enable teams to deliver software more efficiently by reducing the amount of code that they need to write. Although these parts provide the fuel for software supply chains, they have two big weaknesses: (1) parts are not created equal, and (2) parts age and grow stale quickly.

To optimize the performance of modern software supply chains, top performing organizations embrace DevOps and tools to create instant feedback loops by embedding intelligence directly into the development lifecycle. Component intelligence is embedded early and everywhere across the entire supply chain; within IDEs, repository managers, source code repositories, version control systems, and continuous integration platforms. Component intelligence informs version adoption rates, software licenses, known security vulnerabilities, age, alternative versions available, and fit within the organization's governance policies. This information enables rapid feedback loops so developers can always select the best component for the job.

By embracing DevOps and creating instant feedback loops, Liberty Mutual was able to reduce the **average time to deploy new applications from 25 to 2.5 days.**[59] Specifically, Liberty Mutual applied automated quality, licensing and security checkpoints throughout their continuous delivery architecture. Their aim was to accelerate time to market by instrumenting and automating "the easiest path to production," and empowering developers with feedback loops.[60]



Similar results were documented at Fannie Mae. Specifically, in Forrester's March 2017 report "Faster Software Delivery Will Accelerate Digital Transformation," Michael Garcia, VP of DevOps Services at Fannie Mae, stated, "One of our most critical business processes to fix in our DevOps journey was managing the lifecycle of production components and open source libraries to minimize vulnerabilities. Through automation, we were able to reduce our review process from weeks to zero time. **This automation, coupled with the complementary nature of Agile and DevOps practices, has led to a significant, measured upward to 48%, improvement in code quality".**[61]

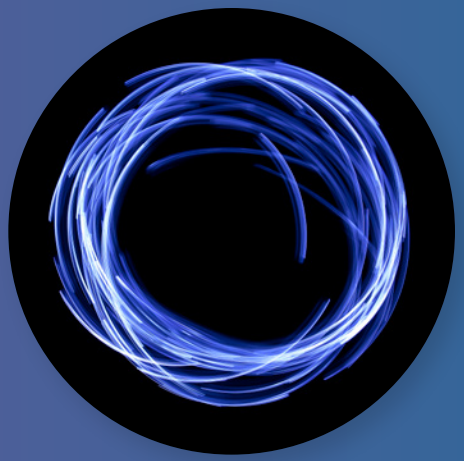
Continuous experimentation and learning

The Third Way of DevOps is to create culture that fosters two things: continual experimentation and understanding that practice is prerequisite to mastering any and all skills.

Experimentation and uncomfortable risk taking ensures that teams are constantly striving to improve. Continual practice ensures that team members have mastered relevant skills to confidently push the envelope.

Today, top performing DevOps teams are embracing software supply chain intelligence to (1) procure open source components from fewer and better suppliers, (2) procure only the best components from those suppliers, and (3) continuously track and trace the precise location of every component throughout every phase of the development lifecycle.

Use of component intelligence across the supply chain also facilitates safe-to-fail experimentation. DevOps teams can experiment faster with new suppliers and components armed with software supply chain intelligence and rapid feedback loops. Should the experiment fail, the original components or new alternatives can be sourced to achieve the best outcome.



SPOTLIGHT

PayPal's remediation at scale

In January 2016, PayPal security researcher Laksh Raghavan published a detailed report of how his company responded to a major security vulnerability in the Commons Collection component. Raghavan's blog[62] described PayPal's Secure Product Lifecycle (SPLC) -- an assurance process to reduce and eliminate security vulnerabilities in PayPal's products over time by building repeatable, sustainable, and proactive security practices that were embedded within their product development process.

Raghavan's report described how cross-functional teams worked together to best change, streamline, and scale their processes by managing their software supply chains.

Raghavan detailed PayPal's remediation actions in four steps: "(1) we quickly patched known vulnerabilities to protect our customers; (2) we invested in tools and technologies that help inventory our applications, libraries and their dependencies in a much faster and smarter way; (3) we streamlined our application upgrade process to be even more agile in responding to such large-scale threats; and (4) we changed our policy on applying security bug fixes during moratorium – we now mandate fixing not just P0 and P1 bugs – but also P2 bugs (e.g., internal apps that are not exposed to the Internet)".[63]

Raghavan reflected on the realities of today's software supply chains, stating, "We understand that today's application infrastructure is complex and you don't own/control all the code that runs in your environment".[64] For PayPal, awareness of their the volume and variety of open source components residing across their software supply chain helped them execute a plan that would minimize business disruption.

CHAPTER 5

Sticks and Carrots

Chapter 5 - Sticks and Carrots: The Inevitable Role of Regulation and Risk

Businesses and organizations decide where and how to invest in cyber security based on a cost-benefit assessment, but they are ultimately liable for the security of their data and systems.

National governments, federal agencies and industry associations are taking action to help organizations improve open source hygiene with regard to software supply chains. In the past year, new guidelines have been introduced from multiple sectors of government and industry to improve the quality, safety and security of software supply chains.

The White House

In May 2017, President Trump issued a new Executive Order declaring, "The executive branch has for too long accepted antiquated and difficult-to-defend IT", believing it is imperative that the United States modernize its IT infrastructure in order to better defend it. The Trump administration is taking a proactive role in assessing vulnerabilities currently in the government's IT infrastructure, software, and supply chains. The Executive Order revealed that **"known but unmitigated vulnerabilities are among the highest cybersecurity risks faced by executive departments and agencies"**. Within 90 days of the order, agencies were required to report to the President on "cybersecurity risks facing the defense industrial base, including its supply chain."[65]

U.S. Commission on Enhancing National Cybersecurity

In December 2016, a nonpartisan presidential commission led by 12 current and former executives from IBM, Uber, MasterCard, Merrill Lynch, and a host of other firms delivered a set of actionable recommendations for strengthening cybersecurity in the public and private sectors.[66]

The Commission's recommendations included assessments of software liability and identification of known security vulnerabilities. The 90 page report recommended the President direct the "The Department of Justice to lead an interagency study with the Departments of Commerce and Homeland Security and work with the Federal Trade Commission, the Consumer Product Safety Commission, and interested private-sector parties to assess the current state of the law with regard to liability for harm caused by faulty IoT devices and provide recommendations within 180 days."[67]

The Commission then recommended the development of a "cybersecurity 'nutritional label' for technology products and services" in order to help the public better understand what known security vulnerabilities have been associated with purchases they make. Several parts of this Commission's report informed the Executive Order issued by the White House in May 2017.

Britain's national cybersecurity strategy

The need for improved cyber hygiene in the UK reached new heights in 2017 following large scale ransomware attacks on its nation's hospital system and an increased focus on software liability. The U.K.'s National Cyber Security Strategy 2016 - 2021 report remarked, "**Businesses and organizations decide on where and how to invest in cyber security based on a cost-benefit assessment, but they are ultimately liable for the security of their data and systems.** Cyber attacks are not necessarily sophisticated or inevitable and are often the result of exploited – but easily rectifiable and, often, preventable – vulnerabilities. In most cases, it continues to be the vulnerability of the victim, rather than the ingenuity of the attacker, that is the deciding factor in the success of a cyber attack."[68]

Shedding a spotlight on the cyber hygiene and software liability, Britain's Information Commissioner's Office (ICO) -- the country's data regulator, said a hacker exploited a well-known security flaw on a Gloucester City Council website months after the vulnerability had been widely reported on and updated component versions had been made available. The ICO fined the **Gloucester City Council £100,000 in June 2017 for not preventing a cyber attack** exploiting the OpenSSL Heartbleed vulnerability.[69]

U.S. Department of Homeland Security

In 2016, the U.S. Department of Homeland Security (DHS) introduced a number of software supply chain management principles to better secure software being developed. These included **(1) building security into the design phase of development, (2) conducting end-to-end risk assessments to bring greater transparency to third-party vulnerabilities, and (3) developing a software bill of materials.**

DHS advised organizations to seize an opportunity for market differentiation, stating "Building security in at the design phase reduces potential disruptions and avoids the much more difficult and expensive endeavor of attempting to add security to products after they have been developed and deployed." The agency went on to recommend using a software bill of materials to help manage risk following any incident when "manufacturers may be faced with the decision between costly recalls and leaving devices with known vulnerabilities in circulation."[70]

U.S. Department of Health and Human Services

In May 2017, HHS's Healthcare Cybersecurity Task Force released a set of recommendations including the need to improve manufacturing and development transparency among developers and users. Similar to the DHS recommendation HHS advised constituents to create a software bill of materials in order "to manage their assets because they must first understand what they have on their systems before determining whether these technologies are impacted by a given threat or vulnerability. **A bill of materials describes its components (e.g., equipment, software, open source, materials), as well as any known risks associated with those components** to enable health care delivery organizations to more quickly determine if they are impacted."[71]

Interestingly, the HHS recommendation came right on the heels of a WhiteScope IO report revealing 8000 known security vulnerabilities across four popular machines used to program pacemakers.[72] For each of the four medical device manufacturers impacted by the research, the findings triggered further investigations of the known vulnerable components that would lead to new releases of software for the programming machines. Externally, the report likely triggered suggestions of new regulations, potential for software liability, threats of competitive displacements, and loss of shareholder value. **Publicly, it also sparked immediate concern over the potential for loss of human life** related to known security vulnerabilities.

© 2016 Ford Motor Company **https://www.ford.ca/performance/gt/**

U.S. Federal Trade Commission

The Federal Trade Commission (FTC) released a report entitled "Start with Security" that highlighted the value of software supply chain transparency and secure coding practices. The report revealed that **"in more than a dozen FTC cases businesses failed to adequately assess their applications for well-known vulnerabilities".**[73] The FTC explained that many of these vulnerabilities stemmed from lack of proper training for developers.

"In cases like MTS, HTC America, and TRENDnet, the FTC alleged that the **companies failed to train their employees in secure coding practices**" explained the report. "The upshot: questionable design decisions, including the introduction of vulnerabilities into the software. For example, according to the complaint in HTC America, the company failed to implement readily available secure communications mechanisms in the logging applications it pre-installed on its mobile devices. As a result, malicious third-party apps could communicate with the logging applications, placing consumers' text messages, location data, and other sensitive information at risk. The company could have reduced the risk of vulnerabilities like that by adequately training its engineers in secure coding practices".[74]

The FTC went on to say that "outdated software undermines security", recommending organizations "prioritize patches by severity" and **employ a reasonable process to update and patch third-party software in order to reduce the risk of a compromise".**[75]

U.S. Department of Commerce

The increasing number of vulnerabilities in software and the risk associated with them caught the attention of the Commerce Department. In 2016, the DOC issued a research report that warned, "As software and technology systems become increasingly interconnected and complex, the likelihood they will contain vulnerabilities increases. As these systems become integrated into a vast array of products and services, **the potential for those vulnerabilities to negatively impact users in profound ways is becoming more significant.** Vulnerabilities create opportunities for malicious attackers to commit cybercrime or disrupt user activity".[76]

U.S. Automotive Industry

According to Motor Authority[77] a Ford GT has over 10 million lines of code, that is much more than what an aircraft needs to fly (2 million lines of code for the Lockheed F-22 Raptor and 14 million lines for the 787 Dreamliner).[78] For consumers, more software delivers a better user experience within their cars, but that same software can introduce more risk. An October 2016 automotive industry report details the need for manufacturers to fully assess the quality or vulnerability of code working its way through their software supply chains. "**The automotive industry should develop and use a risk-based approach to assessing vulnerabilities and potential impacts and should consider the entire supply-chain of operations.** This approach should involve an ongoing risk management framework to assess and mitigate risk over time. At a minimum, organizations should consider cybersecurity risks to safety-critical vehicle control functions and Personally Identifiable Information (PII)".[80]

Cyber insurance premiums - mounting existential risk

“Cyber incidents were ranked as the third-highest global business risk in 2016, Allianz's Risk Barometer determined. The average cost of a breach in the United States reached $7 million in 2016, a Ponemon Institute survey cited in an I.I.I. report. Most traditional commercial general liability policies do not cover cyber risks”.[81]

The growth in consumption of software is matched only by the growth of technological risks businesses now face as a result of utilizing that software. As more regulations and liability concerns related to products employing software come into play, stocking up on insurance against fines and penalties is high on the list. More than 60 carriers now offer stand-alone cyber insurance policies, and it is estimated the U.S. market, worth over $3.25 billion in gross written premiums in 2016, has the potential to grow to $7.5 billion.[82]

In order to deliver safer software faster, many organizations are employing DevOps practices where multiple checkpoints are embedded into the application delivery lifecycle. **Building a holistic understanding into the quality and flow of components through software supply chains is key to reducing an organization’s potential liability.** Tracking and tracing the components used over time, coupled with intelligence about those components also expedites an organization’s ability to recover from some forms of cyber attack.

Summary

We live in an application economy where software innovation is king, speed is critical, and open source is center stage. To compete effectively, modern companies aren't just writing software — they're manufacturing it as fast as they can using an infinite supply of open source component parts, machine automation, and supply chain-like processes.

Software supply chain management is an enterprise imperative for all DevOps organizations. Those embracing the principles of supply chain management are seeing significant improvements in quality and productivity. Organizations choosing to ignore the feast of components being delivered through their supply chains will suffer from growing technical and security debt along with mounting liability concerns.

This is a world defined by key trends, including:

- DevOps practices and tools helping to automate software supply chains
- Developers consuming a massive volume and variety of open source components
- Open source components of varying quality
- Rapid feedback loops embedded within the development lifecycle enabling continuous improvement
- Regulatory and industry initiatives aimed at protecting end users and consumers

To learn more about the trends please visit **www.sonatype.com**.

Our Sources

[1] Right Scale 2017 - State of the Cloud Report: http://assets.rightscale.com/uploads/pdfs/RightScale-2017-State-of-the-Cloud-Report.pdf

[2] Analysis of annual downloads of Java components from the Central Repository in 2008 - 2016

[3] Analysis of annual downloads of npm packages from npmjs.org in 2014 - 2016

[4] Analysis of data sets available at modulecounts.com

[5] Analysis of Java open source projects housed in the Central Repository and Sonatype's 2016 State of the Software Supply Chain Report

[6] http://www.rightscale.com/blog/cloud-industry-insights/new-devops-trends-2016-state-cloud-survey

[7] https://puppet.com/resources/whitepaper/state-of-devops-report

[8] https://en.wikipedia.org/wiki/The_Cathedral_and_the_Bazaar

[9] 2017 DevSecOps Community Survey, www.sonatype.com/2017survey

[10] Sonatype analysis of over 17,000 software applications, 2016 - 2017.

[11] https://search.maven.org/#stats

[12] Sonatype research of JavaScript components, 2017

[13] Sonatype research of PyPI components, 2017

[14] https://www.nuget.org

[15] DockerCon 2017 https://youtu.be/hwkqju_BXEo?t=3m13s

[16] DockerCon 2016 https://youtu.be/vE1iDPx6-Ok?t=10m28s

[17] Analysis of data sets available at modulecounts.com, 2011 - 2017.

[18] Analysis of data sets available at modulecounts.com, 2011 - 2017.

[19] Sonatype primary research into data from the Central Repository, 2017

[20] Sonatype primary research into data from the Central Repository, 2017

[21] Analysis of downloads from the NuGet Gallery, https://www.nuget.org/stats

[22] Analysis of annual downloads from npmjs.org in calendar years 2014 - 2016.

[23] Analysis of downloads from the PyPI.python.org repository in February 2017 by Donald Stufft. https://twitter.com/dstufft/status/831497839608004610

[24] DockerCon 2017, https://www.slideshare.net/Docker/dockercon-2017-general-session-day-1-solomon-hykes-75362520?qid=f8fd1047-01d5-41a9-86cc-1c066f64ad2a&v=&b=&from_search=2

[25] Sonatype analysis of Java, JavaScript and Python components available from public repositories, 2016 - 2017.

[26] Analysis of annual downloads of Java components from the Central Repository in 2016.

[27] Analysis of average Java component downloads at 7500 organizations from the Central Repository in Q4'2016, then annualized.

[28] 2017 DevSecOps Community Survey of 2,292 IT professionals. https://www.sonatype.com/2017survey

[29] Sonatype Application Health Check data analysis, 2017

[30] Why use SemVer? http://blog.npmjs.org/post/162134793605/why-use-semver

[31] Sonatype, 2016 State of the Software Supply Chain Report. www.sonatype.com/ssc2016

[32] Sonatype, 2016 State of the Software Supply Chain Report. www.sonatype.com/ssc2016

[33] The Toyota Way, by Jeffrey Liker, McGraw-Hill, 2004.

[34] 2017 DevSecOps Community Survey. https://www.sonatype.com/2017survey

[35] Sonatype, https://www.sonatype.com/300-per-cent-growth-in-nexus-repository-manager-use

[36] Sonatype research for Central Repository downloads for calendar year 2016.

[37] Sonatype research into Nexus Repository Managers using the Repository Health Check features, April 2017

[38] Sonatype research into Nexus Repository Managers using the Repository Health Check features, April 2017

[39] Thou Shall Not Depend On Me, https://www.sonatype.com/thou-shall-not-depend-on-me

[40] Sonatype Application Health Check data analysis, 2017

[41] Sonatype analysis of over 17,000 software applications, 2016 - 2017..

[42] Sonatype analysis of Central Repository downloads for Bouncy Castle in 2016.

[43] Sonatype analysis of Central Repository downloads for Commons Collection in 2016.

[44] SFMTA, https://www.sfmta.com/about-sfmta/blog/update-sfmta-ransomware-attack

[45] Security Affairs, http://securityaffairs.co/wordpress/45974/malware/samsam-ransomware.html

[46] Cybersecurity Trend, http://www.cybersecuritytrend.com/topics/cyber-security/articles/420473-samsam-ransomware-campaign-exploits-backdoor-jboss-servers.htm

[47] Sonatype analysis of Central Repository downloads for Struts2 from February 2016 to February 2017.

[48] Security Affairs, http://securityaffairs.co/wordpress/57130/hacking/cra-apache-struts-hack.html

[49] Hacker Exploits Apache Struts2 Vulnerability in Statistics Canada Site http://bit.ly/2njlDiX via @ Motherboard http://metacurity.com/#298562

[50] Financial Feeds, http://financefeeds.com/gmo-payment-gateway-confirms-data-leakage-two-client-websites/

[51] Excite.co.jp, http://exci.to/2mqMAwU

[52] ITPro, http://dlvr.it/Ndv4XY

[53] 2017 DevSecOps Community Survey, www.sonatype.com/2017survey

[54] DevOps Digest, May 2016. http://www.devopsdigest.com/docker-security-scanning-released

[55] Banyon Ops https://www.banyanops.com/pdf/BanyanOps-AnalyzingDockerHub-WhitePaper.pdf, 2015

[56] https://techbeacon.com/state-containers-5-things-you-need-know-now

[57] The Three Ways: The Principles Underpinning DevOps, https://itrevolution.com/the-three-ways-principles-underpinning-devops/, 2012

[58] DevSecOps: How to Seamlessly Integrate Security into DevOps, by Neil MacDonald and Ian Head, Gartner, 30 September 2016

[59] All Day DevOps 2016: Building Quality and Security into the Software Supply Chain w/ Eddie Webb https://youtu.be/FsfKsqI07jM

[60] Building Quality and Security into the Software Supply Chain, https://youtu.be/FsfKsqI07jM?t=23m5s

[61] Forrester's March 2017 report "Faster Software Delivery Will Accelerate Digital Transformation" By Diego Lo Giudice, Christopher Condo with Christopher Mines, Luis Deya

[62] Lessons Learned from the Java Deserialization Bug, https://www.paypal-engineering.com/2016/01/21/lessons-learned-from-the-java-deserialization-bug/

[63] Lessons Learned from the Java Deserialization Bug, https://www.paypal-engineering.com/2016/01/21/lessons-learned-from-the-java-deserialization-bug/

Our Sources

[64] Lessons Learned from the Java Deserialization Bug, https://www.paypal-engineering.com/2016/01/21/lessons-learned-from-the-java-deserialization-bug/

[65] https://www.whitehouse.gov/the-press-office/2017/05/11/presidential-executive-order-strengthening-cybersecurity-federal

[66] https://www.nist.gov/sites/default/files/documents/2016/12/02/cybersecurity-commission-report-final-post.pdf

[67] https://www.nist.gov/sites/default/files/documents/2016/12/02/cybersecurity-commission-report-final-post.pdf

[68] https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/567242/national_cyber_security_strategy_2016.pdf

[69] http://news.sky.com/story/council-fined-163100000-for-not-preventing-heart-bleed-cyberattack-10913333

[70] Strategic Principles for Securing the Internet of Things, https://www.dhs.gov/sites/default/files/publications/Strategic_Principles_for_Securing_the_Internet_of_Things-2016-1115-FINAL....pdf

[71] Health Care Industry Cybersecurity Taskforce, June 2017, https://www.phe.gov/Preparedness/planning/CyberTF/Documents/report2017.pdf

[72] http://blog.whitescope.io/2017/05/understanding-pacemaker-systems.html?m=1

[73] https://www.ftc.gov/system/files/documents/plain-language/pdf0205-startwithsecurity.pdf

[74] https://www.ftc.gov/system/files/documents/plain-language/pdf0205-startwithsecurity.pdf

[75] https://www.ftc.gov/system/files/documents/plain-language/pdf0205-startwithsecurity.pdf

[76] Vulnerability Disclosure Attitudes and Actions: A Research Report, https://www.ntia.doc.gov/files/ntia/publications/2016_ntia_a_a_vulnerability_disclosure_insights_report.pdf

[77] http://www.motorauthority.com/news/1098308_the-ford-gt-has-more-lines-of-code-than-a-boeing-passenger-jet

[78 79] Guess what requires a 150 million lines of code, January 2016, https://www.eitdigital.eu/news-events/blog/article/guess-what-requires-150-million-lines-of-code/

[80] Cybersecurity Best Practices for Modern Vehicles - NHTSA, ttps://www.nhtsa.gov/staticfiles/nvs/pdf/812333_CybersecurityForModernVehicles.pdf

[81] https://www.yahoo.com/tech/u-cyber-insurance-market-grows-amid-data-breach-204100136.html

[82] Cyberrisk: Threat and Opportunity, Insurance Information Institute, October 2016, http://www.iii.org/sites/default/files/docs/pdf/cyber_risk_wp_102716-92.pdf

Appendix

Acknowledgments

Each year, the State of the Software Supply Chain report is produced to shed light on the patterns and practices associated with open source software development. The report is made possible thanks to a tremendous effort put forth by many team members at Sonatype, including: Derek Weeks, Matt Howard, Joel Orlina, Bruce Mayhew, Gazi Mahmud, Dariush Griffin, Mike Hansen, Brian Fox, Jessica Dodson, David Barritt, and Janie Gelfond. Additionally, we could not have produced this report without contributions --big and small -- from the DevOps and Open Source Software Community, including: J. Paul Reed (Release Engineering Approaches), Josh Corman (Atlantic Council), Michael Garcia and Barry Snyder (Fannie Mae), Laksh Raghavan (PayPal), Grant Larsen (American Express), Donald Stufft (Python Software Foundation, Amazon), Jonathan Cowperthwait and Laurie Voss (npm), Eddie Webb (Liberty Mutual) and Eric Bourget (Insyders).

A very special thanks goes out to Clara Charbonneau (Insyders) who created the awesome design for this year's report.

About the Analysis

The authors have taken great care to present statistically significant sample sizes with regard to component versions, downloads, vulnerability counts, and other data surfaced in this year's report. While Sonatype has direct access to primary data for Java, JavaScript, Python, .NET and other component formats, we also reference third-party data sources as documented.

Similar to previous years, all of the applications and repository managers analyzed for this report are utilizing Sonatype's Nexus products or related free services.

Copyright © 2017 - present, Sonatype Inc.
All rights reserved. Sonatype and Sonatype Nexus are trademarks of Sonatype, Inc. All other trademarks are the property of their respective owners.