2019 State of the Software Supply Chain

The 5th annual report on global open source software development

presented by

sonatype

in partnership with

galois

IT REVOLUTION

Table of Contents

Introduction

Now in its fifth year, Sonatype's annual State of the Software Supply Chain Report examines the rapidly expanding supply and continued exponential growth in consumption of open source components. Our research also reveals best practices exhibited by exemplary open source software projects and exemplary commercial application development teams.

This year, for the first time, we've collaborated with research partners Gene Kim from IT Revolution and Dr. Stephen Magill, Principal Scientist at Galois and CEO of MuseDev, to objectively examine and empirically document, release patterns and hygiene practices across 36,000 open source project teams and 3.7 million open source releases.

Separately, we observed 12,000 commercial engineering teams to document their consumption of open source and third party libraries. We also conducted two surveys, with a combined participation of over 6,200 development professionals, to better understand the current state of DevSecOps and software supply chain management practices. We compared teams with, and without, automated open source governance capabilities (in an attempt) to reveal the baseline benefit of building applications that utilize higher quality open source components.

The 2019 State of the Software Supply Chain Report blends a broad set of public and proprietary data with survey results, expert research, and analysis to reveal the following:

- ▸ 75% growth in supply of open source component releases over the past two years (Chapter 1)
- ▸ 68% year over year growth in download requests from the Central Repository to 146 billion (Chapter 2)
- ▸ 18x faster median time to update dependencies for exemplary open source components (Chapter 3)
- ▸ 55% reduction in the use of vulnerable open source component releases within managed software supply chains (Chapter 4)
- ▸ 71% increase in confirmed or suspected open source related breaches since 2014 (Chapter 5)

At Sonatype, we've long been active contributors to, and maintainers of, numerous open source efforts, including Apache Maven and Nexus Repository Manager. Since 2005, we've curated and operated the Central Repository, which last year alone serviced 146 billion download requests for component releases from developers around the world. Commercially, our interest lies in helping developers and organizations accelerate innovation and minimize risk by continuously sourcing third-party libraries from the highest quality open source projects.

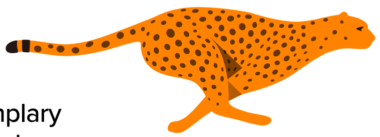
Together with our partners, we are proud to share this research. We hope that you find it valuable.

Exemplary Projects are **3.4x faster** at remediating known vulnerabilities pg. 29

Secure Dev Practices are **9.3x more likely** to proactively remove troublesome dependencies pg. 31

pg. 38

71% increase in open source related breaches over the past five years



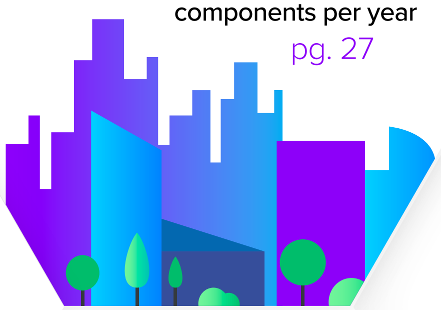
Exemplary Projects are **18x faster** at updating dependencies pg. 13

The top 5% of projects remediate security vulnerabilities **within 21 days** pg. 14

146B Java download requests in 2018, represented a **68% year over year growth** pg. 9

Exemplary Dev Teams are **10x more likely** to schedule dependency updates pg. 29

313,000 average enterprise downloads of OSS components per year pg. 27



pg. 36

1 in 10 downloads of Java component releases have known vulnerabilities

Exemplary Dev Teams experience a **55% reduction** in the use of vulnerable OSS component releases in automated environments pg. 34

Exemplary Dev Teams are **12x more likely** to have automated tools to manage open source dependencies pg. 31

51% of JavaScript components have a known security vulnerability pg. 37

21,448 new open source releases per day pg. 6

pg. 42

PCI introduces **new standards** for development teams using open source components

CHAPTER 1

Global Supply of Open Source

A World of Infinite Choice

1.1 Supply of Open Source is Massive

There are now more than 3.7 million unique Java open source software component releases in the Central Repository, 800,000 unique JavaScript packages in npm, 1.2 million unique Python component releases housed in the PyPI repository, and 1.6 million .NET component releases in the NuGet Gallery.[1] There are also more than 2.2 million containerized applications housed in Docker Hub — up from 900,000 the previous year.[2]

This massive supply of software parts is rapidly and organically expanding due to constant innovations and regular versioning of existing components. These new versions not only offer enhanced features, but also provide improved performance, bug fixes, and security patches.[3]

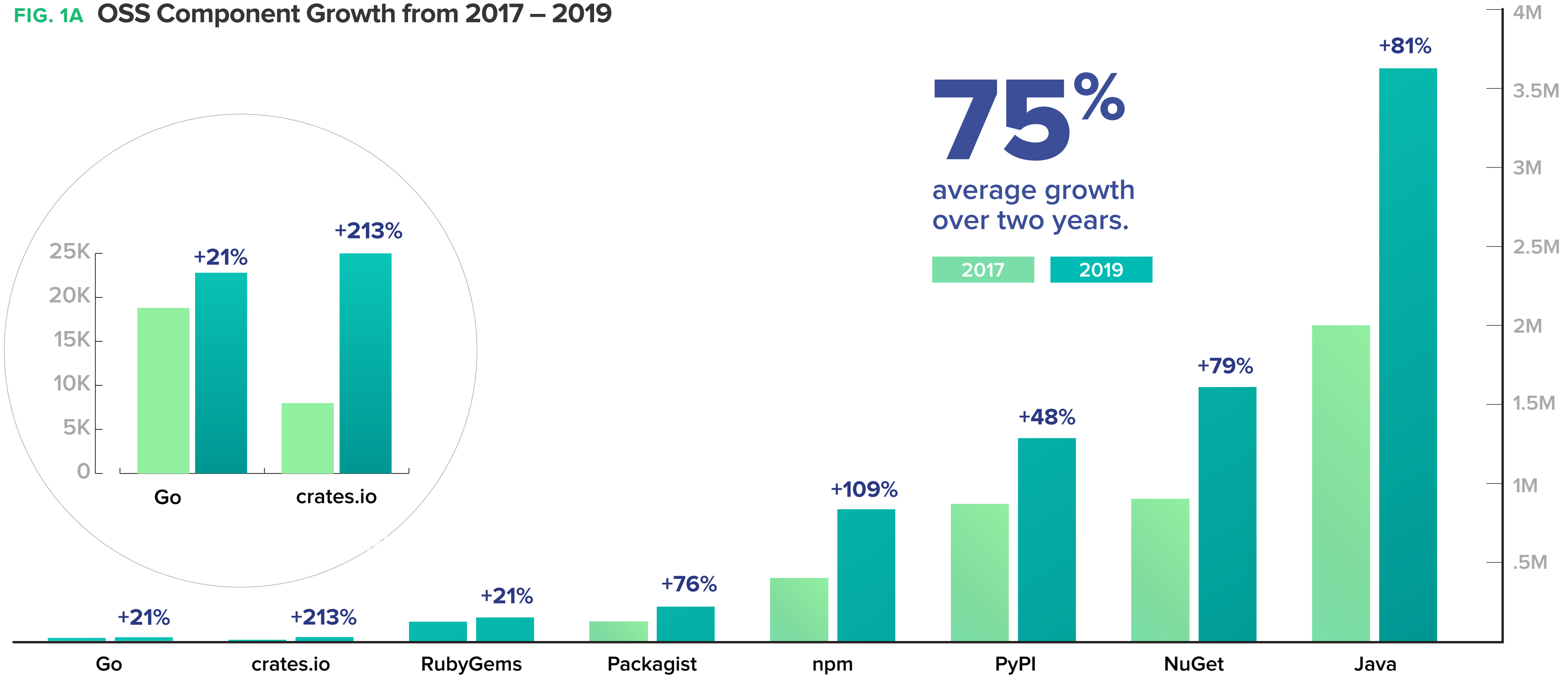
1.2 Supply of Open Source is Expanding Rapidly

Sonatype's study across several open source component ecosystems reveals number of releases housed within public repositories increased from 16.6 million to 28.4 million from January 2018 through today. On average, developers had access to more than 21,448 new open source component releases every day, since the beginning of 2018.

KEY POINT

- **On average, developers had access to more than 21,448 new open source component releases every day, since the beginning of 2018.**

FIG. 1A **OSS Component Growth from 2017 – 2019**



While Java components continue to dominate by sheer number of component releases available, package types like npm and crates.io demonstrated the highest growth rates. npm packages experienced 109% growth and now total 836,000 component releases. Newcomer crates.io (Rust) increased 213% during the period and now offers more than 25,000 component releases (see **FIGURE 1A**).[4]

Open source growth is robust across numerous ecosystems, but npm has grown particularly fast due to JavaScript's emergence as a universal web application programming language. For JavaScript developers looking for a library, tool, or adapter, odds are very good that someone else has already created it and published it to npm where it can easily be borrowed. According to the stewards of the npm repository, "Every week roughly 160 people publish their first package in the registry."[5]

New component versions are released for a variety of reasons, including supporting new functionality, fixing defects, or supporting a new API. New releases may also address non-feature-related concerns, such as improving performance, adding or updating their dependencies, or remediating security vulnerabilities.

Although brand new projects are constantly being created and introduced, growth in open source supply is driven mostly by new versions of existing projects being published. While the growth in supply fuels rapid innovation, it does pose significant questions for organizations wanting to better manage their software supply chains:

- How often do projects publish new versions?
- Do certain projects release updates more frequently?
- Do other projects release updates less frequently?
- What are the implications?
- Who are the best component suppliers?

This year, State of the Software Supply Chain researchers set out to answer these questions and many more.

1.3 Suppliers, Components and Releases

To match terminology in the previous State of the Software Supply Chain reports and provide consistency across research findings, we will follow the Maven terminology using the following definitions:

- A ***supplier*** is a Maven Group (e.g., org.apache.httpcomponents)
- A ***component*** is a Maven Group and Artifact (e.g., org.apache.httpcomponents, httpclient)
- A ***component release*** is a specific Maven Group, Artifact and Version (e.g., org.apache.httpcomponents, httpclient, 4.5.6). For other ecosystems, we sometimes refer to releases as packages.

Fueling Rapid Innovation Consumption of open source is so vast that most organizations cannot identify how many components are entering into their software supply chains, how those components are flowing through development lifecycles, the relative quality and security of those components, or which components exist within production applications.

According to International Data Corporation (IDC), "there were 22.30 million software developers in the world at the outset of 2018. IDC estimated that 11.65 million are full-time developers, 6.35 million are part-time developers, and 4.30 million are nonprofessional developers."[6] In 2018, developers around the world consumed hundreds of billions of open source software component releases.

CHAPTER 2

Global Demand for Open Source

Fueling Rapid Innovation

2.1 Accelerating Demand for Open Source Libraries

The growing demand for innovation has accelerated implementations of automated software development pipelines while also driving open source consumption to new heights across all major ecosystems.

2.1.1 Demand for Java

In 2018, the aggregate number of download requests for Java component releases from the Central Repository grew 68% year over year to 146 billion. With an estimated 12 million Java developers around the world, this equates to 12,166 per person.[7]

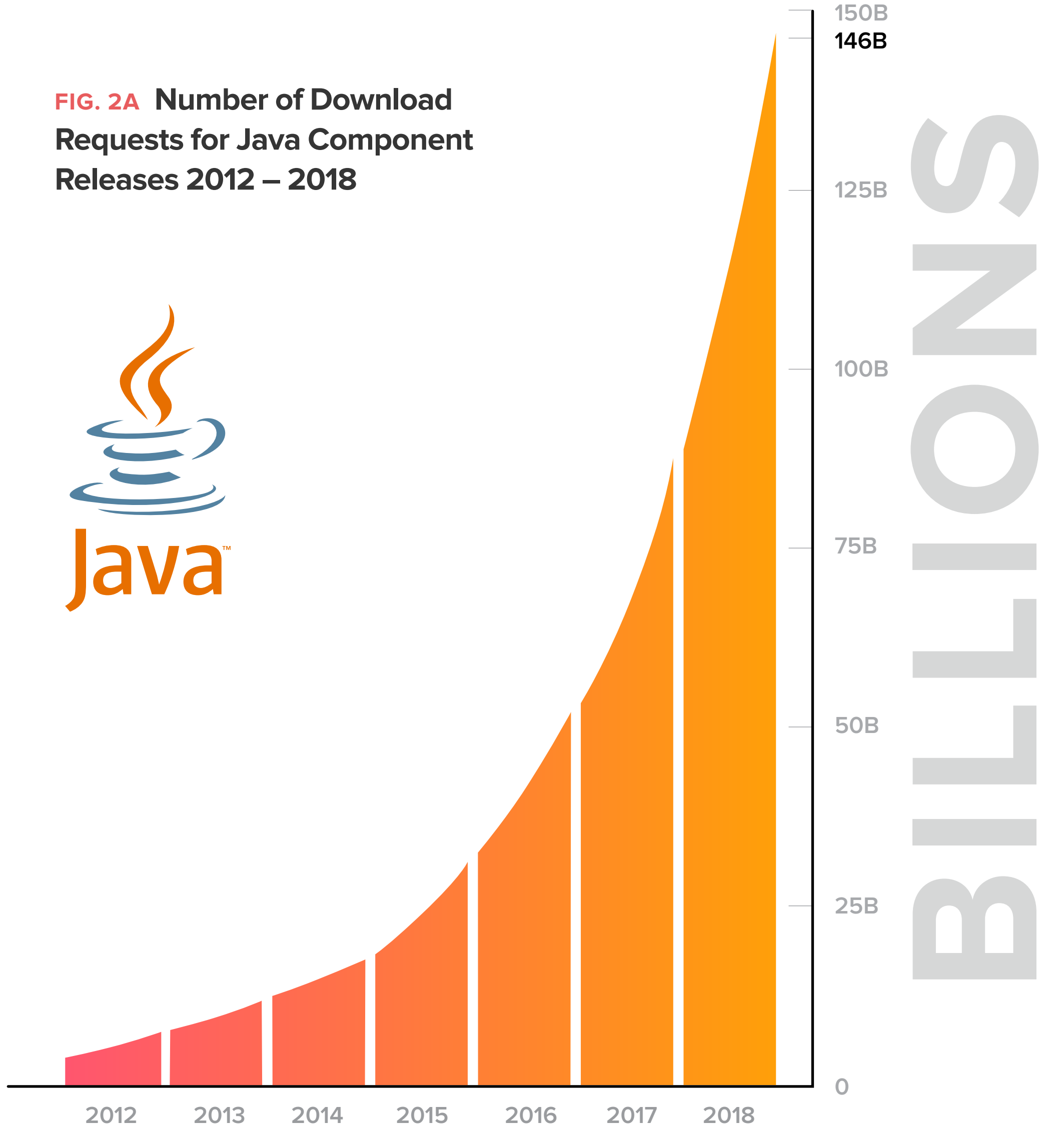
2.1.2 Demand for JavaScript

While growth in demand for Java components is remarkable, a view into JavaScript package downloads demonstrates even greater growth in developer demand. In 2018, average weekly npm package downloads increased from approximately 3.5 billion to 10 billion — an increase of 185%.[8] To add further context, there are an estimated 9.7 million JavaScript developers in the world — meaning the average JavaScript developer is sourcing 1,030 packages per week or 53,608 packages per year.[9]

2.1.3 Others

Other public repositories have experienced similar levels of developer demand. For example, downloads of RubyGems packages have grown from 8 billion to 33 billion over the past three years.[10] NuGet package downloads have increased from 756 million in 2015 to an annual run rate of 16.2 billion in 2019.[11]

FIG. 2A **Number of Download Requests for Java Component Releases 2012 – 2018**



A view into JavaScript component downloads demonstrates even greater growth in developer demand. In 2018, **average weekly npm package downloads increased from approximately 3.5 billion to 10 billion** — an increase of 185%.

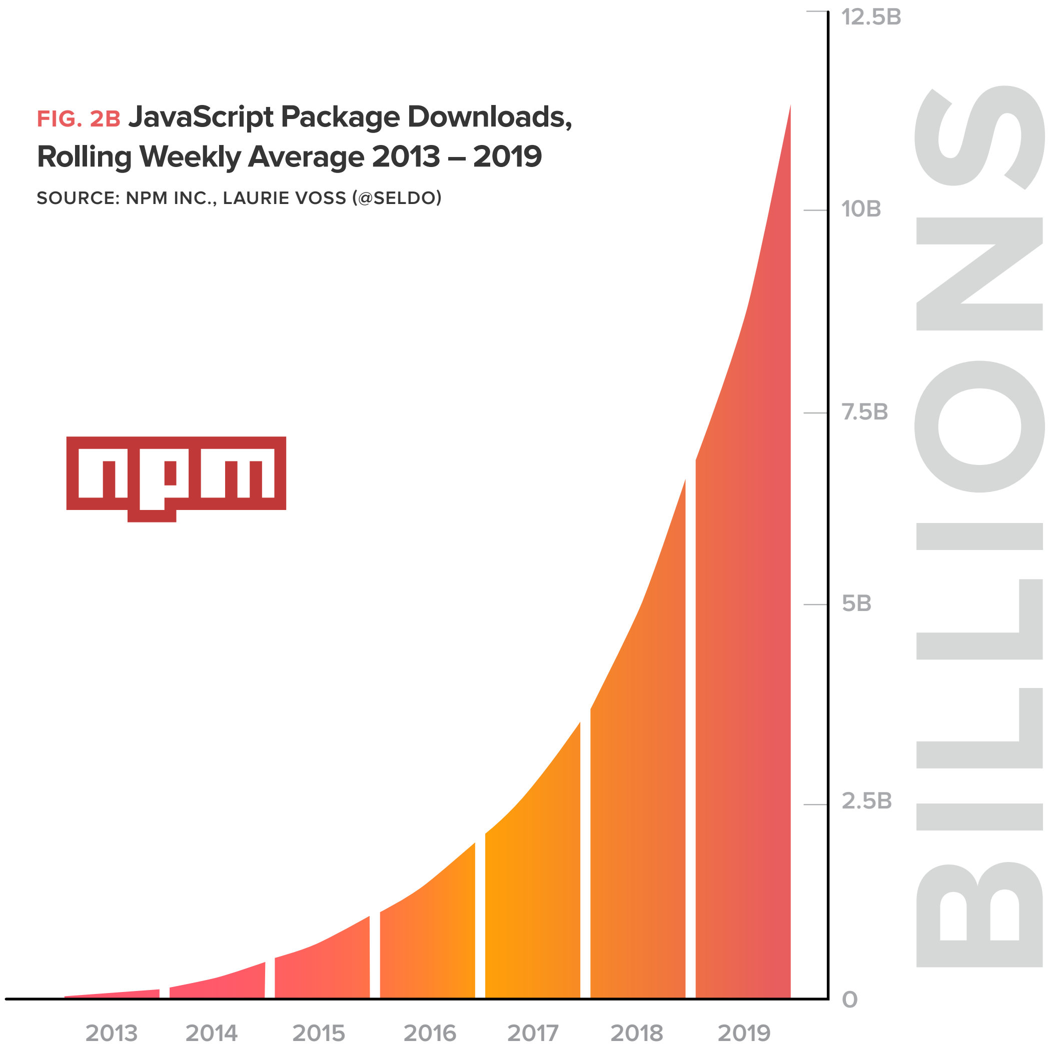
2.2 Automated Pipelines and DevOps Are Key Drivers

Exponential growth in the consumption of open source component releases and containers is a proxy for the adoption of automated software development tools and DevOps pipelines. Automated tooling can generate hundreds or thousands of download requests per build.

In the context of software supply chain management, each download equates to a procurement effort by development teams. Each open source software component release is chosen from an OSS project that acts as a supplier to developers who assemble tens, hundreds, and sometimes thousands of component releases into a finished application.

FIG. 2B **JavaScript Package Downloads, Rolling Weekly Average 2013 – 2019**

SOURCE: NPM INC., LAURIE VOSS (@SELDO)



CHAPTER 3

Exemplary Project Teams

Open Source Projects
Are Not Created Equal

3.1 Research Goals

We wanted to better understand the health and habits of the open source component ecosystem, and how software developers choose which OSS components to use in their own projects. To do this, we studied all the Java artifacts stored in The Central Repository (often referred to as "Maven Central"). At the time of this writing, The Central Repository has over 266,000 unique components with over 3.7 million component releases.

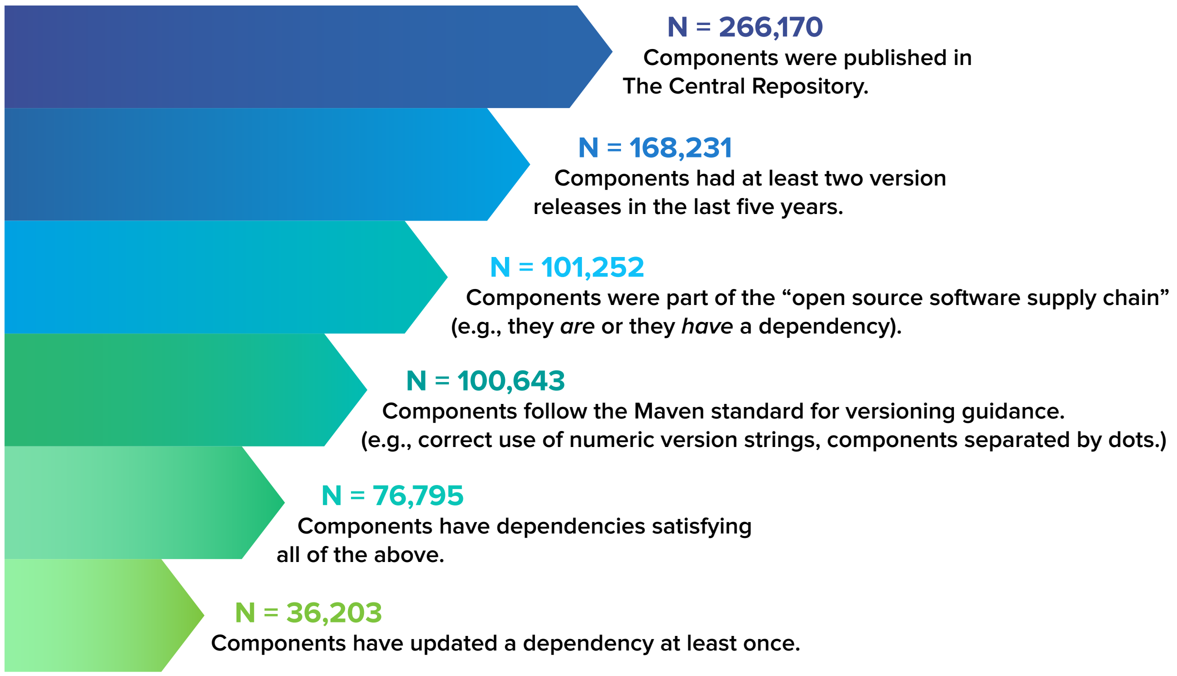
The research set out to answer:

- Do differences exist in how effectively OSS projects update their dependencies and fix vulnerabilities? Are there exemplary components that do this better than others?
- Are exemplary components more widely-used than "non-exemplary" components?
- What factors correlate with exemplary components?
- What advice can be offered to producers of OSS components and the developers that consume them?

For the purpose of this study, we restricted our analysis to components that met six qualifying criteria (see **FIGURE 3A**), resulting in a data set of 36,203 components. Given this data set, we examined a number of attributes to measure responsiveness to security vulnerabilities and determine what properties exemplary component project teams shared. In addition to the security relevant attributes — described in the next section — the primary attributes tracked were:

Number of Dependencies: the maximum count of dependencies for any given component across all versions in the study period, as measured by the dependencies in the Maven pom.xml file.

FIG. 3A **Constructing the Study Dataset (N = 36,203)**



Stale Dependencies (fewer is better): the average percentage of out-of-date component dependencies (i.e., a newer version has been released) present when the component has a new release.

Release Period (shorter is better): average time in days each component version spends as the "current" release. A shorter average release period equates to more frequent releases.

Popularity: the average number of downloads per day from The Central Repository.

SCM Commits per Month: average number of commits to the GitHub repo per month — a measure of development activity (i.e., developer velocity).

Developer Team Size: as measured by the average number of unique developers committing each month.

Presence of Continuous Integration (CI): as measured by the detection of any CI-related configuration files in the source code repository (e.g., Travis, Jenkins, CircleCI, etc.).

Support Type: support for the component comes from an open source foundation, a commercial organization, or is not officially supported by any organization (e.g., a personal project).

To gather these attributes, we augmented the data as follows:

- **CHAOSS:**[12] using the perceval utility to gather GitHub commit data, we gathered the number of commits per month for twelve months, as well as the number of unique developers committing during each month.
- **Libraries.io:**[13] using this dataset, we were able to gather the number of GitHub stars, forks, and pull requests.
- **Sonatype Nexus IQ Server:**[14] using Sonatype data, we are able to gather vulnerability information and a derived component popularity score (based on how frequently components are seen by the Nexus IQ repository scanning service).

3.2 Time to Remediate Vulnerabilities

The first outcome metric we focused on was time to remediate (TTR). TTR is the time required for a component team to remediate any security vulnerabilities reported against their dependencies. For a vulnerable dependency, remediation occurs when it is upgraded to a new version that fixes the vulnerability. We combined data on component updates with data on vulnerabilities, and tagged updates as *security relevant* if there was a known vulnerability targeting the old version at the time the new version was released. We then measure mean time to remediate (MTTR) as the average time required for a component to adopt security relevant updates to dependencies.

Note that the TTR clock starts when a component version fixing a vulnerability is released. Specifically, the TTR clock is not started when the vulnerability is reported or published; early efforts used vulnerability publish times from various sources as the TTR starting time led to many problems. For example, responsible disclosure often delays the publishing of the vulnerability until developers have released a fixed component version, or vulnerabilities may have CVE numbers associated with the date they were reserved rather than published. These factors make it difficult to obtain precise dates of when component teams were advised of vulnerabilities. On the other hand, data on which components fixed a known vulnerability with a new version release is widespread and much more reliable.

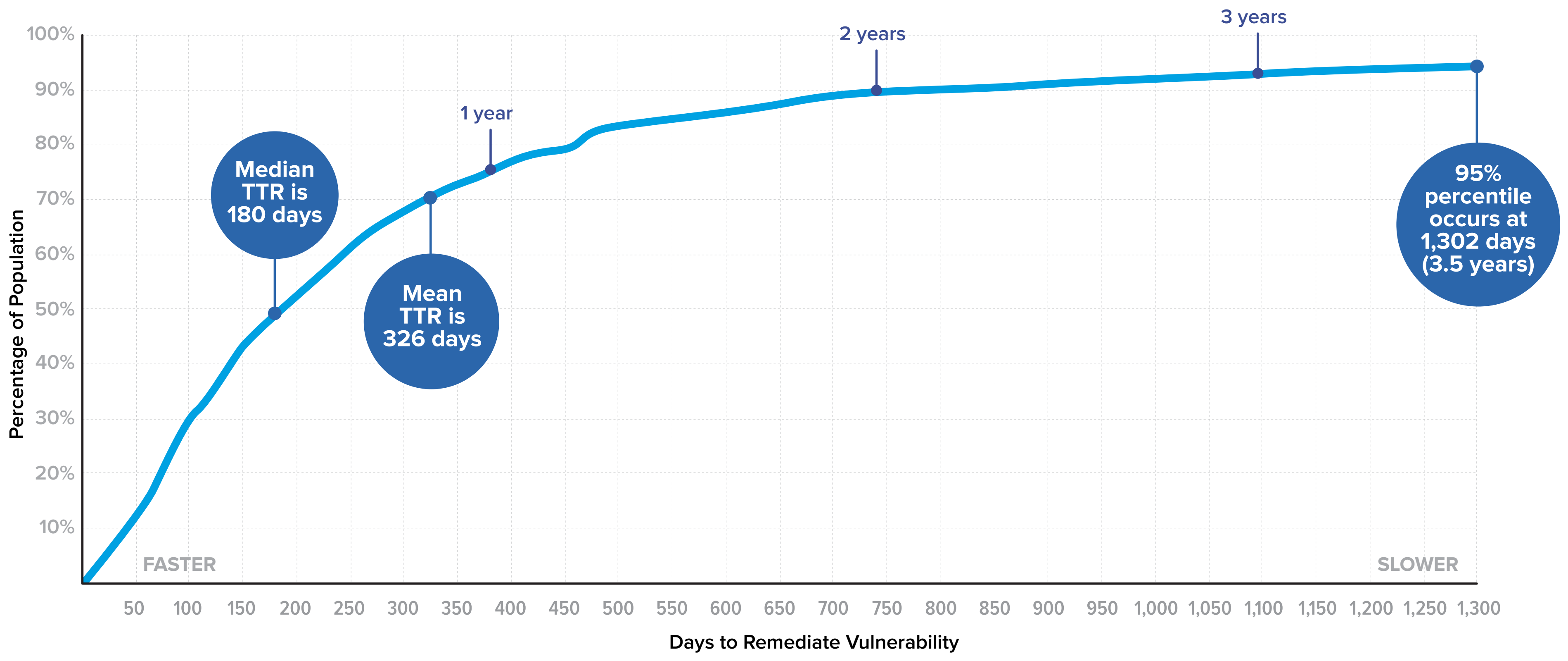
For developers wanting to use more secure components, those exhibiting faster MTTR are desirable. FIGURE 3B (we cropped the x-axis at the 95th percentile) shows a histogram of MTTR across all components, demonstrating the long tail of components that apply security updates very slowly, often years after they are released. We observed:

- In the study, 47% of components — after releasing a new version — had a vulnerability discovered in one of its dependencies, during the period in which that version was current. (N = 36,203)
- The median TTR was 180 days (similar to numbers reported in previous versions of the State of the Software Supply Chain Report). The top 5% of components remediated vulnerabilities within 21 days.

KEY POINTS

- **TTR is the time required for a component team to remediate any security vulnerabilities reported against their dependencies.**
- **For developers wanting to use more secure components, those exhibiting faster MTTR are desirable.**
- **In the study, 47% of components — after releasing a new version — had a vulnerability discovered in one of its dependencies, during the period in which that version was current.**

FIG. 3B **Mean and Median Time to Remediate Vulnerabilities** (cumulative percentage)



- The TTR has an extremely "long tail" — the 95% percentile occurs at 1,302 days (3.5 years), with the maximum TTR at 3,388 days (9.3 years). The mean of the TTR is 326 days.
- 59% of components have had at least one non-up-to-date and known vulnerable dependency at the time of release (e.g., a newer dependency was available that fixed a known vulnerability).

During the period studied, many components had to remediate a security vulnerability caused by a dependency, but over half of the sample set did not have vulnerabilities in any of their direct dependencies. Furthermore, we focused further study on components that were published on GitHub, where we could gather developer activity metrics. When we restrict the population to GitHub-hosted components, the portion that had to deal with a security-relevant dependency update dropped to 16%. In other words, our TTR data set was much smaller than our component data set. This, combined with the hypothesized correlation between median time to update (MTTU) and MTTR, motivated us to look at MTTU as a primary project health metric.

KEY POINTS

- **59% of components have released at least one version that contained a dependency with a known vulnerability.**
- **Over half of the sample set did not have vulnerabilities in any of their direct dependencies.**

3.3 Time to Update Dependencies

To enable a broader analysis of dependency update hygiene, we defined a time to update (TTU) attribute for all components in our dataset.

For each component in our data set, we constructed the dependency graph of the set of components (and their versions) that each component release depends on.

Then for the given period, for every component, whenever a new version of one of its dependencies was released, we measured the time required for the component to update to the newer dependency. We then computed the median of all those time to update data points to obtain the MTTU metric.

FIGURE 3C shows three of our key metrics — time to update (TTU), time to remediate (TTR), and stale dependencies. Suppose:

- Component C depends on Component A and B.
- Component B (version 2.2) has a vulnerability published against it that is fixed in version 2.3.
- The release of B (version 2.3) starts the Component C TTR and TTU clock.
- When Component C updates Component B from version 2.3 to 2.4, the clock stops and we record the TTR and TTU.
- Since the release of A (version 2.4) is not security relevant (no vulnerability known against A), upgrading this component only contributes to TTU.
- When C (version 2.2) is released it is using an old version of A (2.2 rather than 2.3); this causes A to be regarded as a *stale dependency* for the release of C.

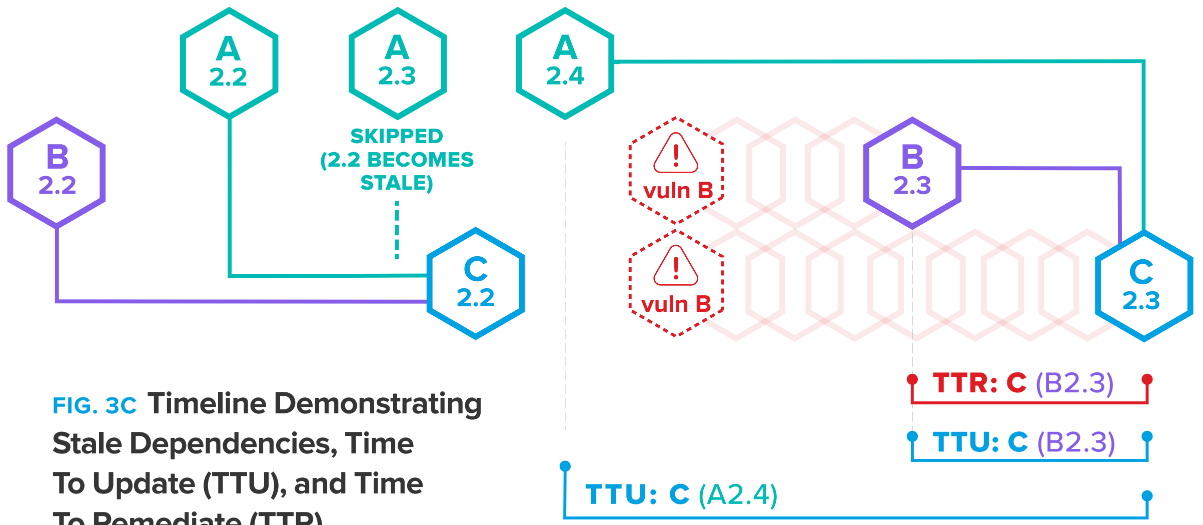


FIG. 3C **Timeline Demonstrating Stale Dependencies, Time To Update (TTU), and Time To Remediate (TTR)**

There are several reasons we suspected that MTTU would be an effective indicator of MTTR performance. One of the most important was phrased by Jeremy Long, founder of the OWASP Dependency Check project who recommends the best security patching strategy is to remain current on all dependencies. Long speculates that "only 25% of organizations report vulnerabilities to users, and only 10% of vulnerabilities are reported as Common Vulnerabilities and Exposures (CVE)."[15] Furthermore, the publication of a CVE is often for a vulnerability that was fixed in an earlier version of a component.

As an example, Long cites a security vulnerability discovered in PrimeFaces — a Java UI framework. The PrimeFaces project became aware of the vulnerability and fixed it in February 2016. A CVE for this vulnerability (CVE-2017-1000486) was subsequently assigned in 2017. Then, the CVE was published into the national catalogue on

KEY POINT

- **Jeremy Long, founder of the OWASP Dependency Check project speculates that "only 25% of organizations report vulnerabilities to users, and only 10% of vulnerabilities are reported as Common Vulnerabilities and Exposures (CVE)."**

January 3, 2018. Upon the publication of the CVE, crypto-miners actively started exploiting vulnerable versions of the component. Developers who made a practice of updating to the latest released versions of PrimeFaces were less at risk than developers who relied upon publication of a CVE to trigger remediation efforts.

To summarize:

- MTTU data was derivable for all components, whereas MTTR data is more sparse. (N=36,203 for MTTU vs. N = 16,997 for MTTR).
- Fast MTTU makes it more likely that components are already protected when new CVEs are published.
- There is a general sense in the security community that "having better security" is achieved by better technical practices. In this case, better practices means integrating updated dependencies into the daily work of the development team.

We therefore explored TTU in our population to better understand how effectively component teams were updating their dependencies.

3.4 Stale Dependencies

Almost every developer has updated their dependencies only to discover that it introduced breaking changes: either compile-time failures, or worse, run-time errors because the dependency functionality has changed. Because of these situations, updating dependencies and patching vulnerabilities becomes an arduous and painful activity, which often leads to developments becoming so behind in updates that upgrading will surely break application functionality (see the survey results in Chapter 4).

To capture the distinction between "fully up to date" and the more conservative "within one version of up to date," we introduced an attribute called stale dependency percentage. This measures the percentage of dependencies, on average, which are not fully up to date when a component releases a new version. In other words, a component that always releases with all their dependencies up to date will have a perfect stale dependency ratio of 0%.

Projects that stick to the "bleeding edge" of using the latest N or N-1 dependencies will have fast MTTU and a low stale dependency ratio. Projects that tend to stay one version behind will have relatively fast MTTU, but high stale dependency ratio. Projects that do not update frequently will have even slower MTTU and higher stale dependency ratios.

One of the papers we were inspired by was *Why and How Java Developers Break APIs*[16] that monitored 400 Java libraries for 116 days, and found 282 commits that were breaking changes. Not all of these were in the released components, but it shows that the risk of breaking changes is real, and potentially introduces a huge economic cost of staying current with OSS components.

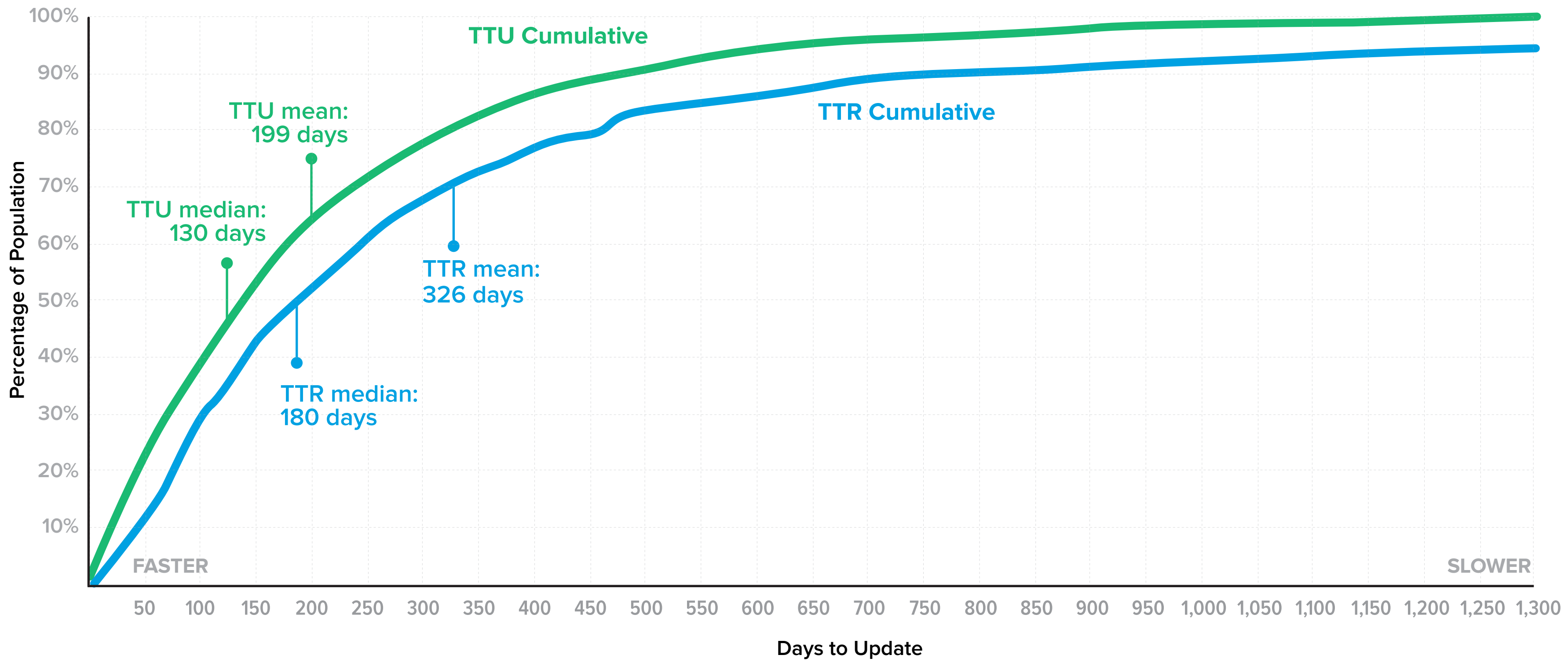
3.5 Exploring the Link Between MTTR and MTTU

Across the entire population, the adoption curve for upgrading dependencies and remediating vulnerabilities are similar, as shown in **FIGURE 3D**. When comparing MTTR with MTTU for non-security-relevant updates on a per-component basis, we see a correlation between update behavior for security relevant updates (MTTR) and non-security-relevant updates (Pearson correlation[17] was 0.6 with N = 17,017). In all, 55% of components had values for MTTR and non-security-relevant MTTU that were within 20% of each other.

KEY POINTS

- **Fast MTTU makes it more likely that components are already protected when new CVEs are published.**
- **We introduced an attribute called stale dependency percentage.**
- **400 Java libraries were monitored for 116 days. Researchers found 282 commits that were breaking changes.**
- **When comparing MTTR with MTTU for non-security-relevant updates on a per-component basis, we see a correlation between update behavior for security relevant updates (MTTR) and non-security-relevant updates.**

FIG. 3D **Time to Remediate (TTR) vs. Time to Update (TTU)** (cumulative percentage)



However, we saw a group of components that focused much more on upgrading vulnerable dependencies than applying other updates — achieving good security outcomes while ignoring (deliberately or accidentally) other component updates. The analysis found that 15% of components have a better than average MTTR, while having below average MTTU for non-security relevant updates.

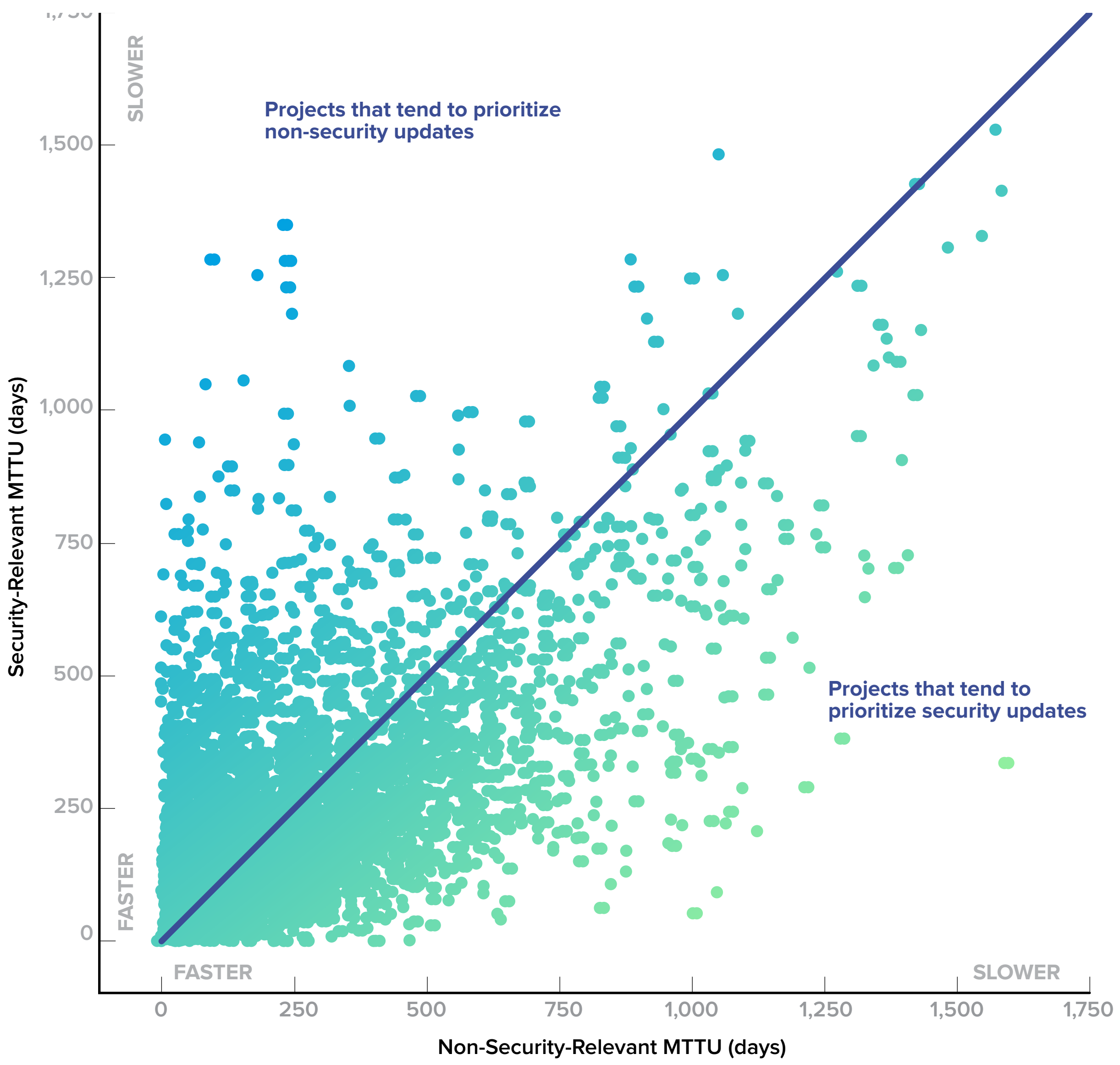
In general, developers staying up to date on dependencies will also stay up to date on security updates, because security updates are a subset of general updates. We observed that many teams follow this practice, exhibiting very similar MTTR and MTTU values. To replicate this practice, security managers can improve vulnerability updating practices by partnering with their development manager counterparts to improve their general dependency management practices.

3.6 Hypothesis Testing

Over several months, we came up with a series of hypotheses of what factors might be associated with better upgrade hygiene (i.e., faster MTTU and

In general, **developers staying up to date on dependencies will also stay up to date on security updates**, because security updates are a subset of general updates.

FIG. 3E **Correlation Between Security-Relevant and Non-Security-Relevant Update Times**



MTTR). Some of the factors included: number of dependencies, developer commit frequency, number of active developers, continuous integration (CI) usage, and component popularity. We hoped to find specific behaviors associated with exemplary component teams and outcomes. This could provide guidance for other component teams (component producers), and provide developers (consumers in the software supply chain) with a list of attributes to look for in OSS components.

To perform this analysis, we combined our dataset with statistics on development behaviors collected from GitHub via the CHAOSS project's Perceval tool. Since not all components are hosted on GitHub, this reduced our dataset size to 10,573 components.

KEY POINT

- **Over several months, we came up with a series of hypotheses of what factors might be associated with better upgrade hygiene (i.e., faster MTTU and MTTR).**

Our primary hypotheses were:

1. Popularity: More popular components will have better update hygiene, due to the pressure to stay secure when a component is widely used.

2. Number of Dependencies: Components with fewer dependencies will have better update hygiene, as it is easier to keep current with fewer dependencies.

3. Size of Team: Large development teams will have better update hygiene.

4. Development Activity: Components that release faster or commit more frequently will have better update hygiene.

5. Use of Continuous Integration: Projects that used continuous integration would have better update hygiene.

6. Institutional Support: Components that are supported by an open source foundation or commercial entity will have better update hygiene, due to having more resources.

While we found evidence for hypotheses 3 and 4, and weak evidence of 5, the most interesting results were the hypotheses that we could not confirm (1 and 2). For each of our hypotheses regarding component attributes that would be associated with faster MTTU, we obtained the following results.

3.6.1 Popularity

QUESTION: Do popular projects, as measured by The Central Repository downloads, have better MTTU?

FINDINGS: No. When deciding to choose components to use in a development project, popularity is often used as a proxy for quality (i.e., "everyone else is using it, so it must be safe, secure, and reliable"). Across the entire population studied, analysis showed popularity did not correlate with fast MTTU (Pearson correlation coefficient of -0.07, Kendall Tau[18] of -0.1). Even when our researchers selected the most popular components (e.g., the top 10 or 20 percent of components by popularity) and compared their MTTU to the rest of the population, no statistically significant differences in MTTU were found.

3.6.2 Number of Dependencies

QUESTION: Do fewer dependencies correlate with faster MTTU?

FINDINGS: No. We had expected components with fewer dependencies to have better MTTU. One would expect that the difficulty of keeping dependencies up to date should grow as the number of dependencies grow. However, our analysis found very little correlation (Pearson = -0.09). What little correlation there was showed that components with larger dependency counts having slightly faster TTU.

Even more surprising, the analysis showed that components with the most dependencies (i.e., the top 10%) had 39% faster TTU (p = 1.2e-29 on Mann-Whitney U test[19]) than the rest of the population. They also had 18% larger development teams than the rest of the population, and this ratio grows as the number of dependencies increase (e.g., at the 95th percentile, the teams were 23% larger). The top 10% by dependency count had 4.8 times more dependencies on average than the bottom 90% and yet had 39% faster TTU. This led us to investigate further the connection between number of dependencies and size of development team. FIGURE 3F shows the plot of number of dependencies versus average size of development team (smoothed over a sliding window of 10 data points).

We were surprised and delighted to find that fast TTU is possible even with large numbers of dependencies. It was particularly interesting that the size of the development team tends to increase as the number of dependencies increases, bringing up an interesting question of causation: when development teams grow, does each developer bring in their favorite dependencies, adding to the dependency count? Or is it that as a project grows, new functionality requires new dependencies, which increase the workload, which requires bringing in new developers. These are questions that we hope to address in a future study.

KEY POINTS

- **Across the entire population studied, analysis showed popularity did not correlate with fast MTTU.**
- **The top 10% by dependency count had 4.8 times more dependencies on average than the bottom 90% and yet had 39% faster TTU.**
- **Fast TTU is possible even with large numbers of dependencies.**

3.6.3 Size of Development Team

QUESTION: Do components with more active developers correlate with faster MTTU?

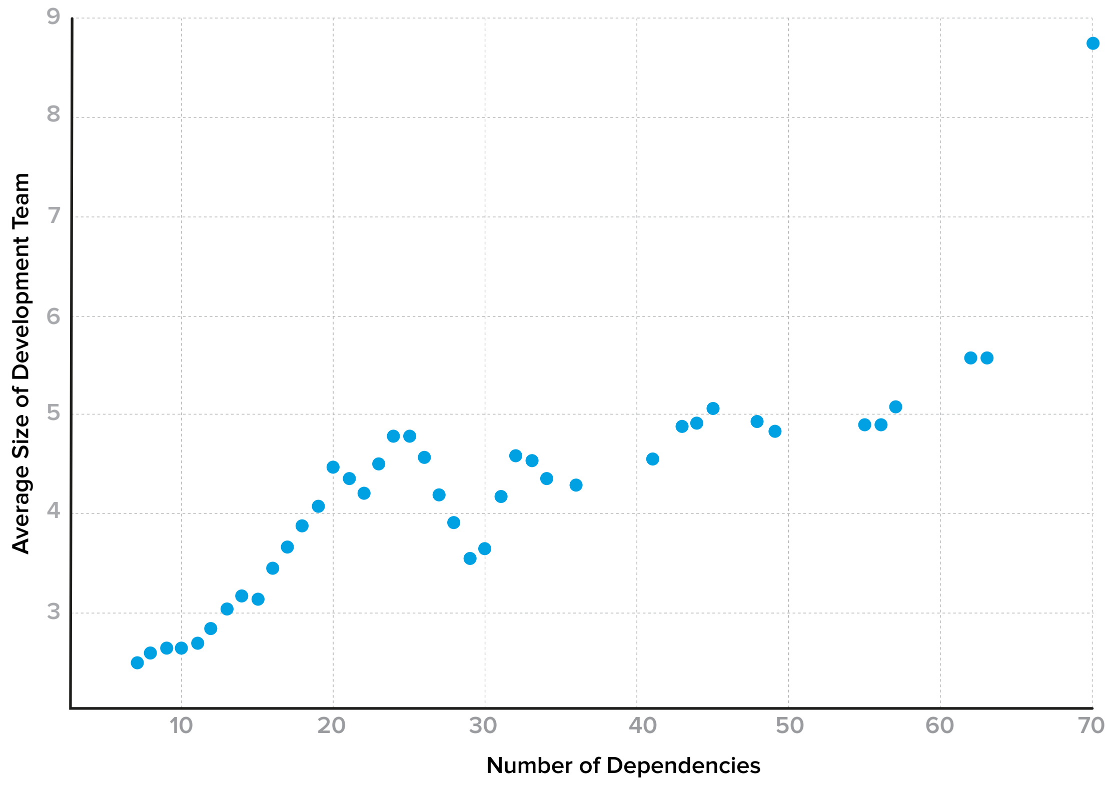
FINDINGS: Yes. Selecting directly for size of development team reveals that across all projects, the top 20% of teams by size (11 or more developers contributing per month) have 50% faster MTTU and release 2.6 times more frequently. They are also 37% more likely to be foundation supported. All results are statistically significant (Mann-Whitney U test) and these trends hold for the top 15% and 10% of teams as well (13 and 16 developers involved per month respectively).

3.6.4 Development Activity and Velocity

QUESTION: Do components with higher release frequency and higher monthly commits correlate with faster MTTU?

FINDINGS: Yes. Clearly, high development activity has the potential to enable a project to stay more up to date. In particular, a project cannot update dependencies without releasing a new version. Therefore, frequent releases are generally required for fast MTTU of dependencies. However it is also possible to spend development effort solely on new features, bug fixes, etc., and ignore dependency management. Furthermore, development and release velocity have been shown to be highly predictive of project outcomes as noted in the 2018 Accelerate: State of DevOps Report.[20] While we cannot determine causation, we found correlations that match these prior results — the top 20% of teams by commits per month had 26% faster MTTU and 83% faster release frequency. Release frequency itself is strongly correlated with MTTU (Pearson correlation of 0.48). This was the strongest

FIG. 3F **More Dependencies Correlate with Larger Development Teams** (smoothed)



KEY POINTS

- **The top 20% of teams by size (11 or more developers contributing per month) have 50% faster MTTU and release 2.6 times more frequently.**
- **The top 20% of teams by commits per month had 26% faster MTTU and 83% faster release frequency. This was the strongest correlation we found among the attributes we considered.**
- **The top 20% of projects by release frequency have 2.3 times faster TTU on average.**

correlation we found among the attributes we considered. The top 20% of projects by release frequency have 2.3 times faster TTU on average.

3.6.5 Use of Continuous Integration

QUESTION: Does the use of continuous integration correlate with faster MTTU?

FINDINGS: No. We were surprised that continuous integration (CI) did not correlate with MTTU. CI was used in the release processes for 68.3% of the components we studied. Usage of CI across all the groups were similar, +/-5%. We speculate that perhaps CI adoption is widespread enough, and that expectations of submitting automated tests with contributed code is high enough, that it is now a mandatory practice for any OSS components.

Because we scanned for the presence of CI configuration files in the source code repository, we were able to establish which tools they used. Of the 7,682 repositories, 90.1% were using TravisCI.

3.6.6 Institutional Support

QUESTION: Is a project supported by an open source foundation or by a commercial entity correlated with faster MTTU?

FINDINGS: Yes and no. We hypothesized that the additional resources available to commercially or foundation-supported projects would generally enable better performance. We based commercial support on group IDs that begin with "com," and foundation support on group IDs associated with multiple components. To improve the labeling, researchers filtered out manually identified outliers such as "com.github," "org.eclipse," and "org.webjars," which all host and aggregate components developed by other groups.

The analysis revealed that Foundation-supported projects had 7.4% faster MTTU in general (p = 0.0002), where researchers also observed highly significant differences in team size (32% larger) and commit frequency (77% faster). By comparison, Commercial projects had 8% slower MTTU (p = 0.0001) and 29% slower commit frequency.

KEY POINTS

- **We were surprised that continuous integration did not correlate with MTTU.**
- **Foundation-supported projects had 7.4% faster MTTU in general (p = 0.0002), where researchers also observed highly significant differences in team size (32% larger) and commit frequency (77% faster).**

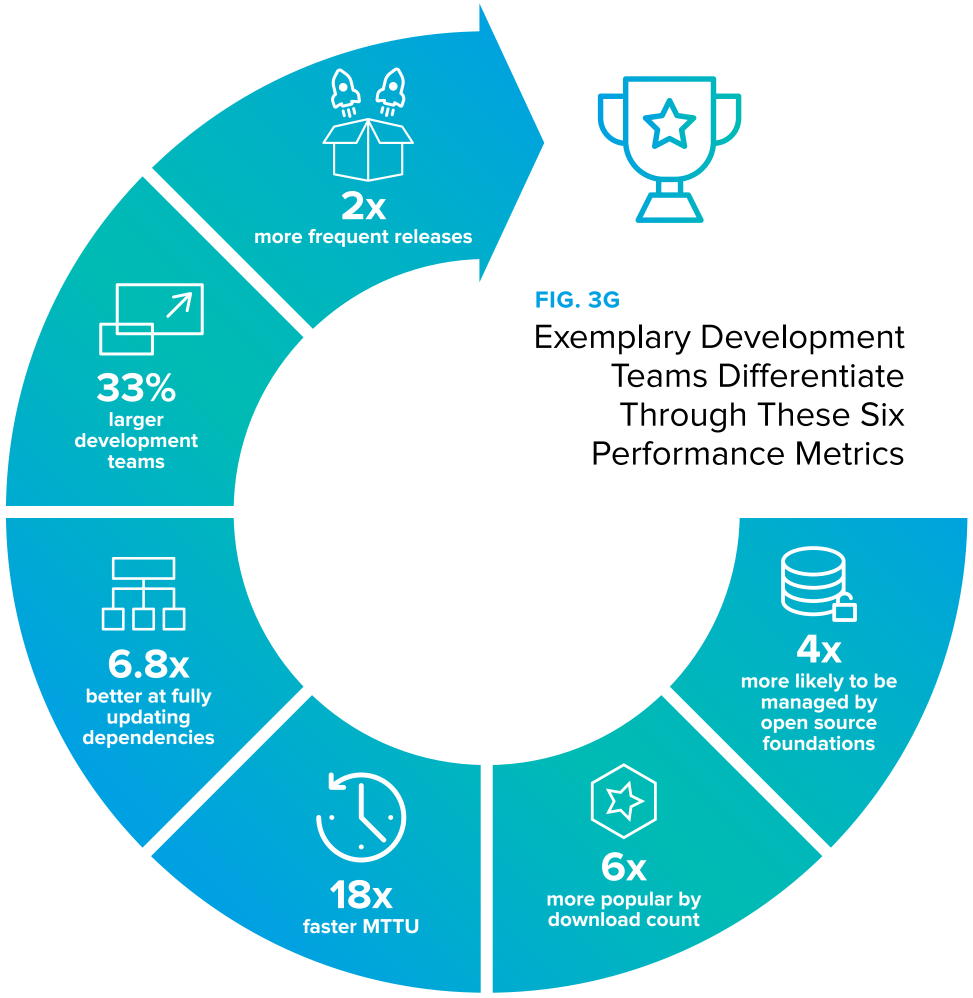

2x
more frequent releases
33%
larger development teams
6.8x
better at fully updating dependencies
18x
faster MTTU
6x
more popular by download count
4x
more likely to be managed by open source foundations

FIG. 3G
Exemplary Development Teams Differentiate Through These Six Performance Metrics

3.7 Finding Different Behavioral Groups

Based on what we learned from our hypothesis testing regarding trends in team composition and performance, we examined five sub-populations of the data to characterize differences in approach to open source software development. The identification and characterization of the sub-populations was driven by a combination of automated clustering, domain, and attribute knowledge gained from the hypothesis testing.

3.7.1 Exemplars

We defined Exemplars to be those teams in the fastest 20% by MTTU, and in the best (lowest) 20% by stale dependency count. Exemplars demonstrate statistically significant differences as compared to the rest of the data set in the following attributes:

- 18x faster MTTU
- 6.8x better at releasing components where all dependencies are up to date
- 6x more popular (as measured by average monthly The Central Repository download counts)
- 2x more frequent component releases
- 33% larger development teams
- 4x more likely to be managed by open source foundations than by commercial stewards

Within the Exemplars, there were two groups with significantly different development team sizes:

3.7.1.1 LARGE EXEMPLARS

Large exemplary teams (top 50% by size, with an average of 8.9 developers committing code on at least a monthly basis), commit code frequently, release frequently, and do an excellent job of managing their dependencies. We can see the effect of open source foundation support in this group, as 91% of these projects are associated with an open source foundation.

3.7.1.2 SMALL EXEMPLARS

The smallest 50% of exemplary teams by number of developers have an average of less than two developers, but still manage to run popular, widely used, and high quality projects. However small in team size, they still update dependencies 14 times faster than the rest of the population and stay fastidiously up to date (91% of dependencies are brought up to date with each release).

3.7.2 Laggards

The teams in the bottom 20% in MTTU and stale dependencies are the furthest behind in terms of update hygiene. These teams release infrequently (around twice each year) and take on average almost two years to adopt updates to dependencies. Almost all of their dependencies (98% on average) are out of date, even after a new release. They are generally less popular (downloaded as often as other projects on average). However there are 67 projects in this group that are among the top 10% most downloaded projects from The Central Repository.

3.7.2.1 FEATURES FIRST LAGGARDS

These teams release frequently (top 50%) but otherwise fall into the Laggard category (bottom 20% MTTU and stale dependencies). They have larger than average (29% larger) development teams, but do not prioritize upgrading dependencies. They release a new version every 50 days on average but take an average of 603 days to upgrade dependencies when new versions are released. As a result, 96% of dependencies are out of date at release time. This was a small group, with 2.6% of the population exhibiting this behavior.

3.7.3 Cautious Teams

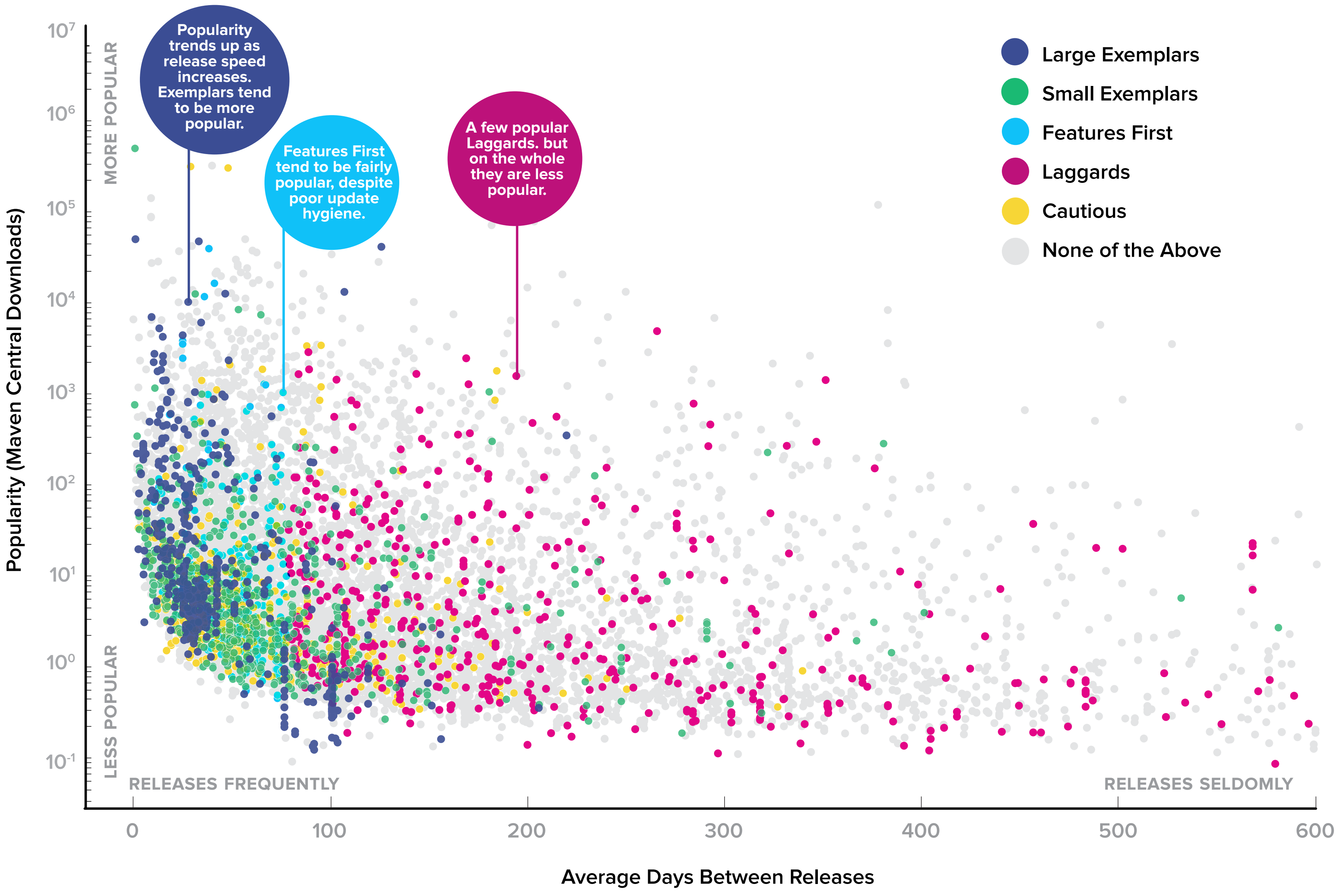
We checked to see how many teams were in the top 50% with respect to MTTU, but the bottom 20% with respect to stale dependencies. These teams maintain better-than-median update cadence, yet do not immediately adopt new versions of dependencies, choosing instead to wait a few months before moving to a new dependency release. This was not a sizable group, with only 4% of our dataset falling into this category.

continues on page 24

KEY POINTS

- **91% of Large Exemplars are associated with an open source foundation.**
- **Small Exemplars update dependencies 14 times faster than the rest of the population and stay fastidiously up to date (91% of dependencies are brought up to date with each release).**
- **For Laggards, almost all of their dependencies (98% on average) are out of date, even after a new release.**
- **Features First release a new version every 50 days on average but take an average of 603 days to upgrade dependencies when new versions are released.**
- **Cautious teams do not immediately adopt new versions of dependencies, choosing instead to wait a few months before moving to a new dependency release.**

FIG. 3H **Cluster Popularity and Release Speed**



3.8 Guidance for Open Source Project Owners and Contributors

Given its association with good security practices and outcomes, we recommend a focus on accelerating and maintaining rapid MTTU. In addition to investing development effort on new features, bug fixes, etc., projects should commit similar resources to dependency management. This means that developers maintaining OSS projects who are considering adding a new dependency, and looking for a metric to guide that choice, would do well to focus on those dependencies with fast MTTU. Since remediating a vulnerable dependency typically involves upgrading to a new dependency version, components with fast TTU values naturally exhibit faster response to dependency vulnerabilities.

To progress comfortably into the status of Exemplar (top 80% of Exemplars), teams should aim for a minimum of four releases annually, and aim to upgrade at least 80% of their dependencies with every release. A higher frequency of dependency updates statistically results in higher quality and more secure code.

3.9 Guidance for Enterprise Development Teams

Enterprise development teams working with software supply chains often rely on an unchecked variety of supply from OSS projects where each developer or development team can make their own sourcing and procurement decisions. The effort of managing 2,778 different projects and 8,200 unique releases **(SEE SECTION 4.2.1)** can introduce significant drag on development and is contrary to an enterprise's need to develop faster as part of any agile, continuous delivery or DevOps practice.

Choosing open source projects should be considered an important strategic decision for enterprise software development organizations. Different components demonstrate healthy or poor performance that impacts the overall quality of their releases. Therefore, MTTU should be an important metric when deciding which components to utilize within your software supply chains. Rapid MTTU is associated with lower security risk and is also more accessible from public sources than other metrics enterprises might want to rely upon such as vulnerability data.

Just as traditional manufacturing supply chains intentionally select parts from approved suppliers and rely upon formalized procurement practices — enterprise development teams should adopt similar criteria for their selection of OSS components. This practice ensures the highest quality parts are selected from the best and fewest suppliers — a practice Deming recommended for decades. Implementing selection criteria and update practices will not only improve quality, but can accelerate mean time to repair when suppliers discover new defects or vulnerabilities. Chapter 4 will further explore the impact of OSS component selection on overall application quality.

KEY POINTS

- **We recommend projects focus on accelerating and maintaining rapid MTTU.**
- **Teams should aim for a minimum of four releases annually, and aim to upgrade at least 80% of their dependencies with every release.**

FIG. 31 The Exemplars: Components Demonstrating the Fastest MTTU and Lowest Stale Dependency Counts

Truncated: for complete list, see Appendix D, page 50

- bz.tsung.android:objectify
- com.ahome-it:ahome-tooling-server-core
- com.amazon.device.tools.build:builder
- com.amazon.device.tools.build:gradle
- com.amazon.device.tools.lint:lint-checks
- com.github.japgolly.fork.scalaz:scalaz-concurrent_sjs0.5_2.11
- com.github.japgolly.fork.scalaz:scalaz-xml_sjs0.5_2.11
- com.github.japgolly.fork.scalaz:scalaz-iterv_sjs0.5_2.11
- com.github.japgolly.fork.scalaz:scalaz-core_sjs0.5_2.11
- com.aranea-apps.android.libs:android-rest
- com.ariht:config-generation-maven-plugin
- com.arasthel:swissknife
- com.asayama.docs.gwt.angular:gwt-angular-pages
- com.asayama.gwt:gwt-util
- com.asayama.gwt.angular:gwt-angular-resources
- com.asayama.gwt.angular:gwt-angular-masonry
- com.asayama.gwt.angular:gwt-angular-http
- com.asayama.gwt.angular:gwt-angular-user
- com.asayama.gwt.angular:gwt-angular-prettify
- com.asayama.gwt.angular:gwt-angular-ng
- com.asayama.gwt.bootstrap:gwt-bootstrap
- com.asayama.gwt.jquery:gwt-jquery
- com.automattic:elasticsearch-statsd
- com.autoscout24.gradle:gradle-monkey-plugin
- com.damnhandy:handy-uri-templates
- com.badlogicgames.gdx:gdx-backend-robovm
- com.badlogicgames.gdx:gdx-backend-lwjgl
- com.badlogicgames.gdxpay:gdx-pay-android-googleplay
- com.badlogicgames.gdxpay:gdx-pay
- com.erinors:xtend-ioc-core
- com.bartoszlipinski:parsemodel-compiler
- com.bazaarvoice.dropwizard:dropwizard-webjars-bundle
- com.bazaarvoice.dropwizard:dropwizard-configurable-assets-bundle
- com.github.advantageous:qbit-spring
- com.github.advantageous:qbit-consul-client
- com.github.advantageous:qbit-eventbus-replicator
- com.github.advantageous:qbit-vertx
- com.github.advantageous:qbit-service-discovery
- com.github.advantageous:qbit-test-support
- com.github.advantageous:qbit-admin
- com.github.advantageous:qbit-core
- com.github.advantageous:qbit-servlet
- com.github.akarnokd:ixjava
- com.github.almondtools:rexlex
- com.github.andreptb:fitnesse-selenium-slim
- com.github.andrewoma.kommon:kommon
- com.github.andrewoma.kwery:fetcher
- com.github.andrewoma.kwery:core
- com.github.andrewoma.kwery:transactional
- com.github.andrewoma.kwery:mapper
- com.github.aro-tech:tdd-mixins-core
- com.github.aro-tech:extended-mockito
- com.github.ben-manes.caffeine:guava
- com.github.chandu0101.scalajs-react-components:macros_sjs0.6_2.11
- com.github.czyzby:gdx-lml
- com.github.danielgindi:helpers
- com.github.davidmoten:bigsort
- com.github.davidmoten:rxjava-extras
- com.github.dblock:oshi-core
- com.github.ddth:ddth-osgikafka
- com.github.ddth:ddth-zookeeper
- com.github.dnvriend:akka-persistence-jdbc_2.10
- com.github.doctoror.rxcursorloader:library
- com.github.fbertola:mother-docker
- com.github.finagle:finch-oauth2_2.10
- com.github.finagle:finch-demo_2.11
- com.github.fracpete:screencast4j-weka-package
- com.github.gabrielemariotti.cards:library-extra
- com.github.heuermh.adamexamples:adam-examples_2.11
- com.github.heuermh.adamplugins:adam-plugins
- com.github.heuermh.adamplugins:adam-plugins_2.11
- com.github.heuermh.adamplugins:adam-plugins_2.10
- com.github.j-fischer:rest-on-fire
- com.github.jinahya:simple-file-back
- com.github.jodersky:flow_2.11
- com.github.joschi:dropwizard-elasticsearch
- com.github.jsonld-java:jsonld-java-sesame
- com.github.jsurfer:jsurfer-simple
- com.github.jszczepankiewicz:dynks
- com.github.jtakakura:gradle-robovm-plugin
- com.github.kentyeh:sd4j
- com.github.kzwang:elasticsearch-river-dynamodb
- com.github.kzwang:elasticsearch-transport-redis
- com.github.kzwang:elasticsearch-osem
- com.github.kzwang:elasticsearch-repository-gridfs
- com.github.mhshams:core
- com.github.michaelruocco:wso2-api-publisher-plugin
- com.github.mictaege:doozer
- com.github.nkzawa:engine.io-client
- com.github.nwillc:contracts
- com.github.oscerd:camel-cassandra
- com.github.pengrad:java-telegram-bot-api
- com.github.persapiens:jsf-undertow-spring-boot-starter
- com.github.persapiens:jsf-undertow-bootsfaces-spring-boot-starter
- com.github.persapiens:jsf-jetty-bootsfaces-spring-boot-starter
- com.github.pwittchen:reactivenetwork
- com.github.pwittchen:reactivebeacons
- com.github.ratrecommends:gdx-utils
- com.github.richard-ballard:arbee-test-utils
- com.github.richard-ballard:arbee-utils
- com.github.salomonbrys.kodein:kodein
- com.github.salomonbrys.kodein:kodein-android
- com.github.scala-blitz:scala-blitz_2.11
- com.bladecoder.engine:blade-engine-spine-plugin
- com.bladecoder.engine:blade-engine
- com.bladejava:blade-jetbrick
- com.bloidonia:groovy-stream
- com.github.sd4324530:fastweixin
- com.github.seratch:ltsv4s_2.11
- com.github.seratch:scalikesolr_2.10
- com.github.seratch:jslack
- com.github.sogyf:goja-qrcode
- com.github.sv244:torrentstream-android
- com.braintreepayments.api:braintree
- com.braintreepayments.api:braintree-api
- com.github.sviperll:metachicory
- com.github.sviperll:adt4j-core
- com.github.thomasnield:rxkotlinfx
- com.github.tibolte:agendacalendarview
- com.github.tkurz.sesame:vocab-builder-cli
- com.github.tkurz.sesame:vocab-builder-maven-plugin
- com.github.triceo.splitlog:splitlog-core
- com.github.vmironov.jetpack:jetpack-bindings-arguments
- com.github.webdriverextensions:webdriverextensions
- com.github.wnameless:smartcard-reader
- com.github.xuwei-k:httpz-scalaj_2.11
- com.github.xuwei-k:msgpack4z-java07
- com.github.xuwei-k:play23scalacheck111_2.11
- com.github.xuwei-k:applybuilder71_2.11
- com.github.xuwei-k:msgpack4z-java
- com.github.xuwei-k:play23scalaz71_2.11
- com.github.xuwei-k:play23scalaz70_2.11
- com.github.xuwei-k:play-twenty-three_2.11
- com.digitalpebble:storm-crawler-tika
- com.cedarsoft.commons:configuration
- com.digitalpebble:storm-crawler
- com.cedarsoft.commons.history:core
- com.mailosaur:mailosaur-java
- com.pengyifan.bioc:pengyifan-bioc
- com.cloudhopper:ch-smpp
- com.cocosw:framework
- com.codeborne:selenide
- com.codebullets.saga-lib:saga-lib-guice
- com.puppycrawl.tools:checkstyle
- com.dimafeng:testcontainers-scala
- com.redowlanalytics:swagger2markup-maven-plugin
- com.rtstatistics:api-client
- com.craterdog.java-security-framework:java-digital-notary-api
- com.cyngn.vertx:vertx-kafka
- com.ea.orbit:orbit-rest-client
- com.ea.orbit:orbit-actors-spring
- com.valchkou.datastax:cassandra-driver-mapping
- com.ea.orbit:orbit-actors-redis
- com.eharmony:aloha-vw-jni
- com.englishtown.vertx:vertx-httpservlet
- com.englishtown.vertx:vertx-zookeeper
- com.englishtown.vertx:vertx-guice
- com.englishtown.vertx:vertx-when
- info.cukes:cucumber-picocontainer
- com.erudika:para-dao-cassandra
- com.eventsourcing:eventsourcing-core
- com.evernote:android-sdk
- com.facebook.presto:presto-teradata-functions
- com.facebook.presto:presto-cli
- com.facebook.presto:presto-orc
- com.facebook.presto:presto-blackhole
- com.facebook.presto:presto-server
- com.floragunn:search-guard-5
- com.floragunn:search-guard-ssl
- com.flozano.statsd-netty:statsd-netty
- com.gabrielittner.auto.value:auto-value-cursor
- com.getbase.android.autoprovider:library
- com.giffing.wicket.spring.boot.starter:wicket-spring-boot-starter-example
- de.leanovate.doby:doby_2.11

CHAPTER 4

Exemplary Dev Teams

Benefits of DevSecOps and Automated Open Source Governance

4.1 The Enterprise Continues to Accelerate

Software innovation has become the last path to differentiation in most competitive industries. Companies must transform their approach to digital innovation or face loss of market share. There is $2 trillion spent annually in software development, but most is labor-oriented and there is a growing need to automate more of the software development process. Thus, exemplary engineering teams are embracing DevOps practices and automated tools to manage third-party dependencies and minimize open source risk.

Forty-seven percent of development teams now deploy to production multiple times a week.[21] Furthermore, DORA's 2018 State of DevOps Report provides strong evidence that organizations adopting DevOps practices are experiencing remarkable results, including:

- DevOps teams deploy code 46x more frequently — meaning they deploy multiple times per day instead of once a week or less.
- DevOps teams have a 7x lower change failure rate — meaning changes to production fail 7.5% of the time instead of 38.5%.
- High performing DevOps teams are 1.75x more likely to extensively use open source software and 1.5 times more likely to expand open source usage in the future.[22]

In order to survive and thrive in today's application economy the best development teams are actively embracing open source innovation, dependency management practices, and automated tooling for open source governance.

4.2 Analysis of 12,000 Large Enterprises

This year's research analyzed the Java open source consumption patterns of 12,000 enterprise development teams to understand average consumption patterns, the diversity of OSS components they relied upon and the number of releases of those projects they consumed. Open source component release download patterns were observed for calendar year 2018. Downloads were observed across enterprises representing 173 countries.

4.2.1 Analysis of Open Source Downloads

QUESTIONS: How many component releases are downloaded by companies each year? How many OSS projects and releases are represented?

FINDINGS: In 2018, the average enterprise downloaded 313,000 open source component releases — representing an increase of 84% year over year.

On a country-by-country basis, downloads patterns revealed interesting variation. For example, the average organization in Germany downloaded 436,000 component releases, followed by France with 324,000, the United States with 309,000, and the United Kingdom with 248,000 downloads.

For the study population, downloads represented an average of 2,778 open source components, including 8,200 unique component releases. This means the enterprises consumes an average of three releases per component. One the high end, 243 (2%) organizations used over almost five (4.61) release versions. On the low end, 1,470 (12%) organizations averaged below two (1.82) release versions.

KEY POINTS

- **There is $2 trillion spent annually in software development, but most is labor-oriented and there is a growing need to automate more of the software development process.**
- **In 2018, the average enterprise downloaded 313,000 open source component releases.**

Researchers noted that it is not uncommon to see downloads of 50 – 60 versions of a specific project over the year long observation period.

4.2.2 Utilization of Repository Managers

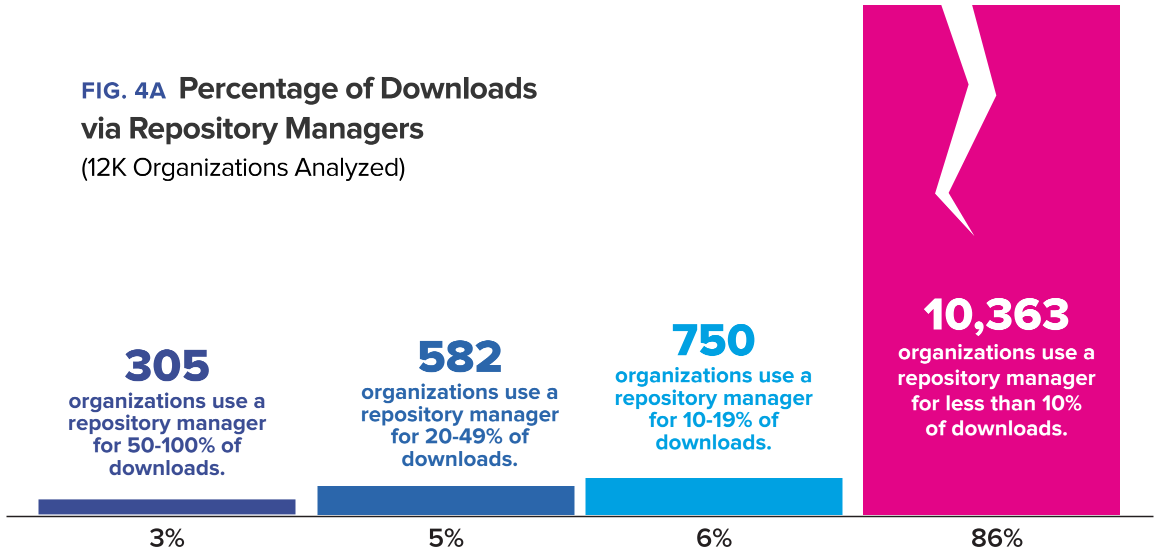
QUESTION: How frequently are repository managers used in the download process?

FINDINGS: Software developers doubled their use of repository managers in 2018 as they sought more efficient and higher velocity practices for OSS consumption. Over 9 million developers now use repository managers as part of their development tool set.[23] Even as the number of repository manager instances grow, their use as a primary download path within development teams has not reached significant levels. Across the 12,000 organizations analyzed, component release downloads via repository managers totaled 5.3% (compared to 1.8% globally).

By contrast, 305 (3%) of the 12,000 organizations demonstrated high utilization of repository managers for more than 50% of their downloads. This group was utilizing repository managers 14.2 times more on average the other organizations observed.

Repository managers not only accelerate the development process by caching component releases locally, but they also can be used to limit the number of paths releases can travel to make their way into an enterprise. Limiting the number of paths is one of the first steps toward controlling and auditing software supply chain behaviors for an enterprise.

FIG. 4A **Percentage of Downloads via Repository Managers**
(12K Organizations Analyzed)



4.3 Component Releases Make Up 85% of a Modern Application

Open source components are pervasive in software development today. An analysis of over 500 applications revealed the average application contains over 460 software component releases, of which 85% were open source. In the same study, it was not uncommon to see applications assembled from 2,000 – 4,000 OSS component releases.

In JavaScript development, the number of npm packages per application is even greater. According to npm.org, “the average modern web application has over 1,000 modules, and trees of over 2,000 modules are not uncommon. In fact, 97% of the code in a modern web application comes from packages downloaded from the npm repository. An individual developer is responsible only for the final 3% that makes their application unique and useful.”[24]

KEY POINTS

- **Across the 12,000 organizations analyzed, component release downloads via repository managers totaled 5.3%**
- **The Exemplars were utilizing repository managers 14.2 times more on average the other organizations observed.**
- **An analysis of over 500 applications revealed the average application contains over 460 software component releases, of which 85% were open source.**
- **97% of the code in a modern web application comes from packages downloaded from the npm repository.**

As development teams strive to deploy new software faster, the practice of assembling open source component releases into the form of an application screams of efficiency. Developers no longer need to code every line from scratch. Developers can download component releases in seconds that deliver new capabilities, built by experts outside of their organizations who make their code freely available to others.

While component use proliferates every development organization today, management of components in these organizations varies considerably. The best organizations follow software supply chain management principles to ensure the best quality component releases are assembled into their applications.

4.4 Characteristics of Exemplary Development Teams

Jeremy Long, founder of the OWASP Dependency Check project once shared, "Good development teams consider out-of-date libraries a code quality issue. They build time into their schedule to upgrade their dependencies. On the other hand, development teams who do not do this regularly are often afraid to break their build."[25]

When it comes to software development, exemplary teams are more likely to embrace the following patterns and practices when it comes to open source dependency updates.

4.4.1 Dependency Update Behaviors for Developers

In order to better understand the practices surrounding open source components within development, this year's researchers surveyed 658 developers in April 2019. The developers were asked to describe their organization's practices as well as to describe their feelings of how those practices impacted their productivity and enjoyment of work.

Researchers performed an analysis of all the respondent answers and three distinct clusters emerged. Broadly speaking, the clusters demonstrated high, medium, and low (29.3%, 48.6%, and 22.0% of respondents, respectively) degrees of reported pain associated with updating dependencies and patching. Researchers then compared the high and low pain clusters to determine what percentage of respondents answered "strongly agree" to the survey questions. Stark differences were found between them (**SEE FIGURE 4B**).

4.4.1.1 UPDATING OPEN SOURCE DEPENDENCIES

QUESTION: Is updating dependencies scheduled as part of your daily work?

FINDINGS: Researchers found that 38% of the developers surveyed updated dependencies as part of their daily work, yet Exemplars were 10x more likely to schedule dependency updates as part of their daily work.

4.4.1.2 USING THE LATEST VERSIONS OF DEPENDENCIES

QUESTION: Do you strive to use the latest version (or latest-N) of all your dependencies?

FINDINGS: From the overall survey population, 46% of developers strove to use the latest version of all of their open source dependencies. Further analysis revealed that Exemplars were 6.2x more likely to use the latest version (or latest-N) of all of their dependencies.

Using the latest versions of dependencies is its own reward. As noted previously in Chapter 3, Exemplar components demonstrated 3.4 times faster MTTR and were 27% more likely to already be protected when new vulnerabilities were discovered.

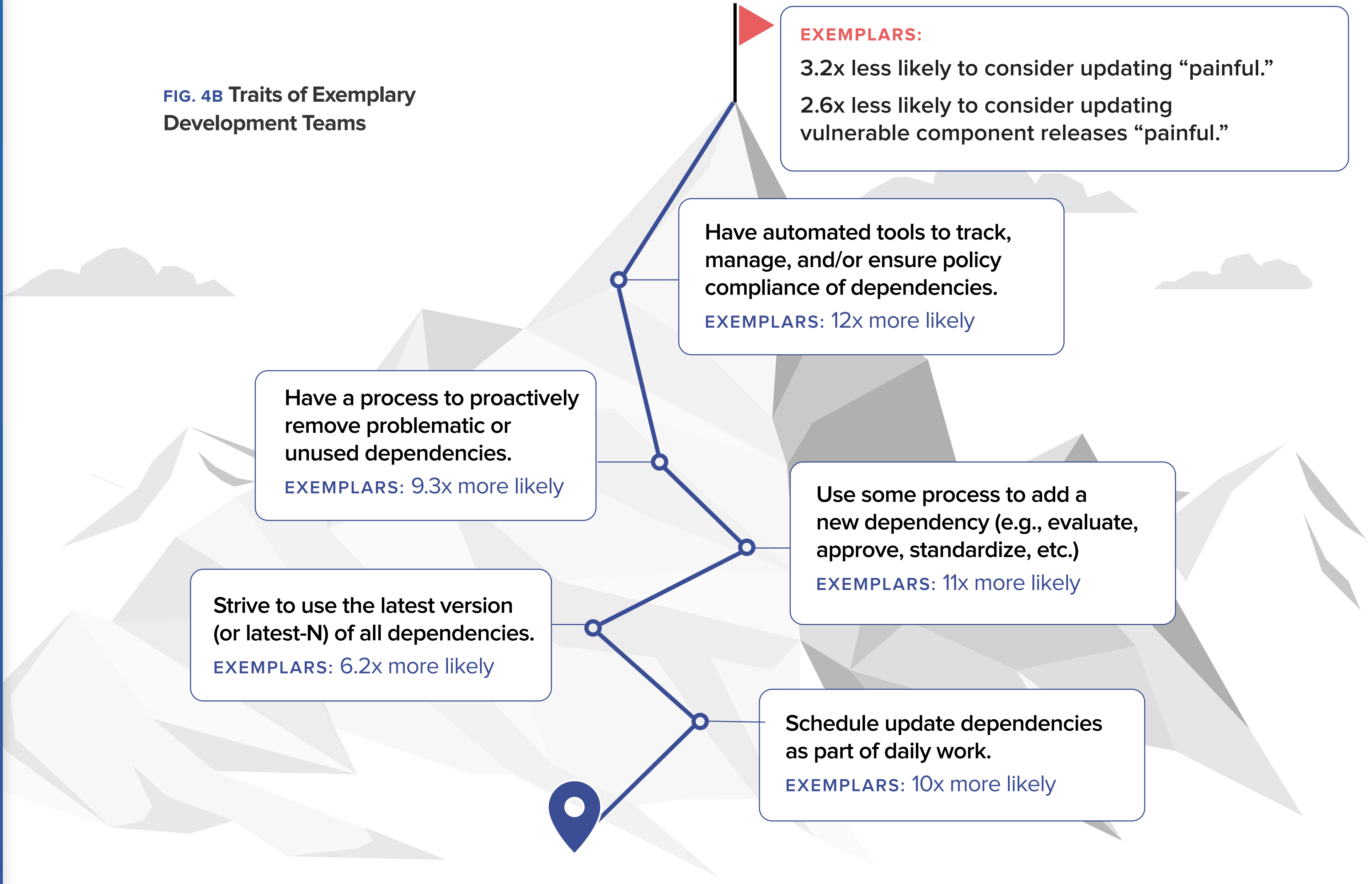
Then, as will be revealed below in section 4.5, teams using the latest component releases can reduce the presence of vulnerable component releases in their applications by 55%. Therefore, development teams that simultaneously incorporate the latest versions of component releases and procure them from exemplary open source projects can improve the overall quality of their applications.

continues on page 31

KEY POINTS

- **Jeremy Long, founder of the OWASP Dependency Check project once shared, "Good development teams consider out-of-date libraries a code quality issue. They build time into their schedule to upgrade their dependencies."**
- **Exemplars are 10x more likely to schedule dependency updates as part of their daily work.**
- **Exemplars are 6.2x more likely to use the latest version (or latest-N) of all their dependencies.**
- **Exemplar components demonstrated 3.4 times faster MTTR and were 27% more likely to already be protected when new vulnerabilities were discovered.**

FIG. 4B **Traits of Exemplary Development Teams**


EXEMPLARS:
3.2x less likely to consider updating "painful."
2.6x less likely to consider updating vulnerable component releases "painful."
Have automated tools to track, manage, and/or ensure policy compliance of dependencies.
EXEMPLARS: 12x more likely
Have a process to proactively remove problematic or unused dependencies.
EXEMPLARS: 9.3x more likely
Use some process to add a new dependency (e.g., evaluate, approve, standardize, etc.)
EXEMPLARS: 11x more likely
Strive to use the latest version (or latest-N) of all dependencies.
EXEMPLARS: 6.2x more likely
Schedule update dependencies as part of daily work.
EXEMPLARS: 10x more likely

4.4.1.3 USING PROCESSES TO ADD A NEW DEPENDENCY

QUESTION: Do you use some process to add a new dependency (e.g., evaluate, approve, standardize, etc.)?

FINDINGS: From the overall survey population, 50% of developers said they relied on some process to add a new open source dependency. When examining the Exemplar cluster, researchers found them to be 11x more likely to have a process in place to add a new dependency (e.g., evaluate, approve, standardize, etc.).

4.4.1.4 PROACTIVELY REMOVING DEPENDENCIES

QUESTION: Do you have a process to proactively remove problematic or unused dependencies?

FINDINGS: When it came to removing problematic component releases (e.g., those with security vulnerabilities), only 30% admitted to having a process in place. Exemplars were 9.3x more likely to proactively remove problematic or unused dependencies.

4.4.1.5 USING AUTOMATION TO MANAGE DEPENDENCIES

QUESTION: Do you have automated tools to track, manage, and/or ensure policy compliance of your dependencies?

FINDINGS: Use of automated solutions can expedite dependency management. This year's survey revealed 37% of the overall population relied on automation to manage dependencies. A closer look at Exemplars demonstrated that they were 12x more likely to have automated tools to track, manage, and/or ensure policy compliance of their dependencies.

4.4.1.6 CHARACTERIZING THE EFFORT TO UPDATE DEPENDENCIES

QUESTION: Do you consider updating dependencies painful?

FINDINGS: Updating dependencies is not a favorite past time of developers. In the overall survey population, 51% of developers agreed that updating was considered painful. The benefit of applying processes and updating to the latest versions of dependencies paid off for the Exemplars. Researchers found Exemplars to be 3.2x less likely to consider updating dependencies to be "painful."

4.4.1.7 CHARACTERIZING THE EFFORT TO UPDATE VULNERABLE COMPONENTS

QUESTION: Do you consider updating vulnerable component releases to be painful?

FINDINGS: In the survey, 52% reported the practice of updating vulnerable component releases as "painful." Again, the benefit of applying processes, updating to the latest N-versions, and regularly removing problematic dependencies paid off for the Exemplars. Exemplars surveyed were 2.6x less likely to consider updating vulnerable components to be "painful."

4.4.2 Component Management in the Enterprise

This year, our researchers surveyed 5,558 development and DevOps professionals as part of the 2019 DevSecOps Community Survey. Survey participants were asked to self identify their DevOps maturity level into different categories, where three clusters emerged. Broadly speaking, there were those with high levels of DevOps maturity, those with improving maturity, and those with low maturity or no DevOps practice.

Stark differences emerged between the high maturity (Exemplar) and low maturity (No DevOps) clusters as they responded to questions about open source component controls, policies, and breaches as seen across the following practices.

KEY POINTS

- **When examining the Exemplar cluster, researchers found them to be 11x more likely to have a process in place to add a new dependency.**
- **Exemplars were 9.3x more likely to proactively remove problematic or unused dependencies.**
- **A closer look at Exemplars demonstrated that they were 12x more likely to have automated tools to track, manage, and/or ensure policy compliance of our dependencies.**
- **Researchers found Exemplars to be 3.2x less likely to consider updating dependencies to be "painful."**
- **Exemplars surveyed were 2.6x less likely to consider updating vulnerable components to be "painful."**

4.4.2.1 GENERATION OF A SOFTWARE BILL OF MATERIALS (SBOM)

QUESTION: Does your organization keep a complete SBOM?

FINDINGS: More rigorous control of open source component releases used in development and operations leads to the use of a Software Bill of Materials (SBOM). SBOMs are used to track and trace which open source component releases have been assembled into an application. The survey revealed that only 53% of Exemplars keep a complete software bill of materials.[26] When paired with vulnerability data about the components, SBOMs can be used to quickly identify where defective component releases have been used in applications — whether under development or in production, thereby accelerating remediation efforts.

4.4.2.2 CONTROLLING OPEN SOURCE USED IN DEVELOPMENT

QUESTION: How well does your organization control which open source component releases are used in development?

FINDINGS: 2019 saw more organizations investing in controls that start with keeping an inventory of all component releases used. When asked how well their organizations controlled which open source component releases were used in development, 74% of Exemplars in DevOps practices remarked that their organization was "completely locked down" or had "some standards" in place. By contrast, 48% of developers in organizations without a DevOps practice claimed to have "no standards" in place.[27]

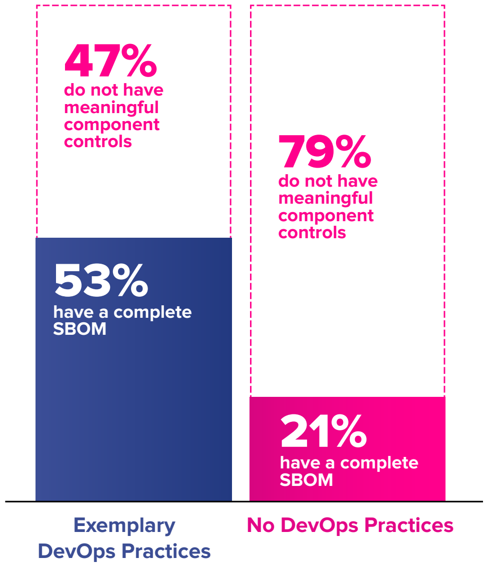
4.4.2.3 EMBRACEMENT OF AUTOMATED TOOLS TO ENFORCE COMPLIANCE AND INFORM DEVELOPERS

QUESTION: Does your organization have an open source policy and do you follow it?

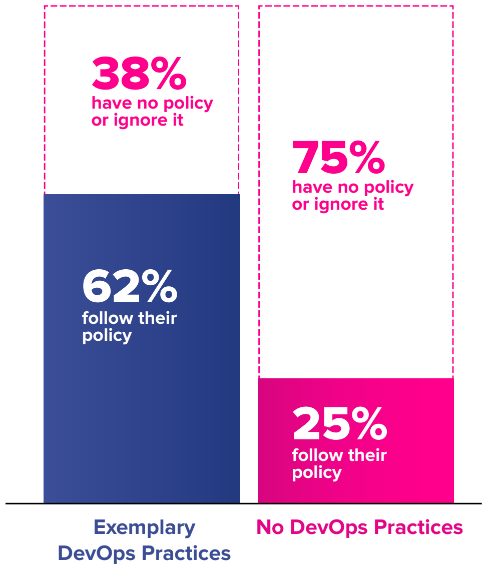
FINDINGS: The survey revealed that 57% of organizations have an open source governance policy in place.[28] These organizations rely on policies supported by a mix of formal documentation, OSS governance committees, automated tooling, and tribal knowledge.

The survey also revealed that automation had a significant impact on whether or not open source policies were followed. In Exemplars DevOps practices where more automated OSS governance solutions have been deployed, 62% of developers

FIG. 4C **Adoption of a Software Bill of Materials**



Adoption of Open Source Governance Policies



SOURCE: 2019 DEVSECOPS COMMUNITY SURVEY (SONATYPE)

KEY POINTS

- **The survey revealed that only 53% of Exemplars keep a complete software bill of materials.**
- **74% of Exemplars in DevOps practices remarked that standards were in place for controlling use of open source components.**
- **57% of organizations have an open source governance policy in place.**

remarked that their organization had a policy and that it was followed. By comparison, developers in organizations with little to no DevOps practices, only 25% of their developers were aware of and followed the policy. When automation of OSS policies is present, developer adherence is 150% stronger.[29]

4.4.2.4 INFORMING DEVELOPERS OF SECURITY RELATED ISSUES

QUESTION: How are you informed of InfoSec and AppSec issues?

FINDINGS: One of the reasons why adherence is stronger in exemplary organizations is that OSS component attributes (e.g., version numbers, vulnerability information, license descriptions) and policy information are delivered inside developer tooling. For developers in Exemplars DevOps practices, 63% are informed of application security issues from within their own development tools — meaning they don't have to leave their common tool sets to receive alerts and information.

Developers inside exemplary teams were 62% more likely to receive notice of application security issues from within their tool sets compared to developers with little to no DevOps practice in place.[30]

KEY POINTS

- **When automation of OSS policies is present, developer adherence is 150% stronger.**
- **For developers in Exemplars in DevOps practices, 63% are informed of application security issues from within their own development tools.**

FIG. 4D Age of Components Used in Managed Software Supply Chains
(Analysis of Java Components Across 68,000 Applications)

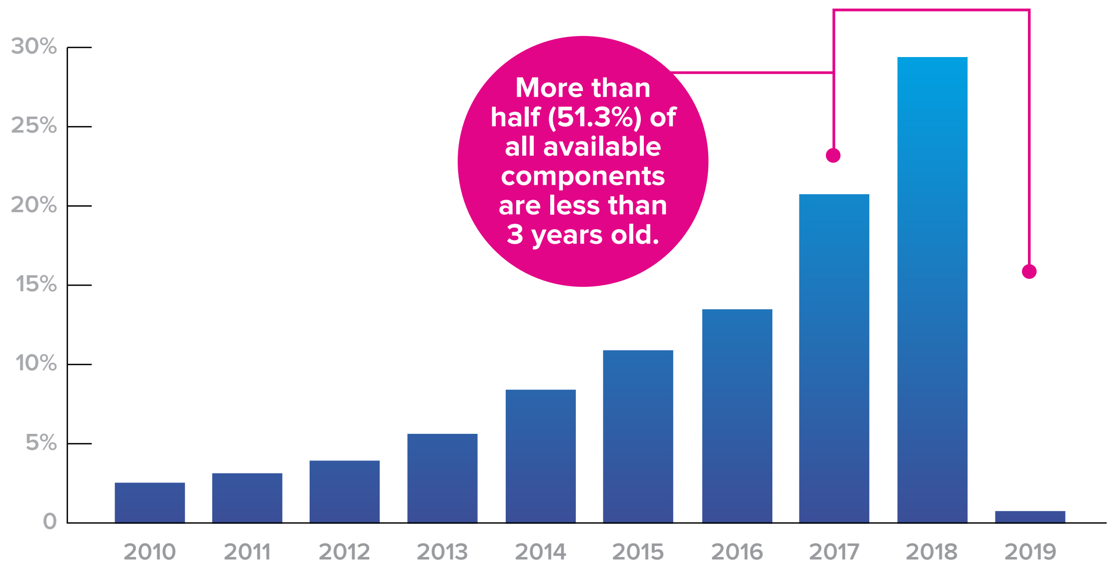
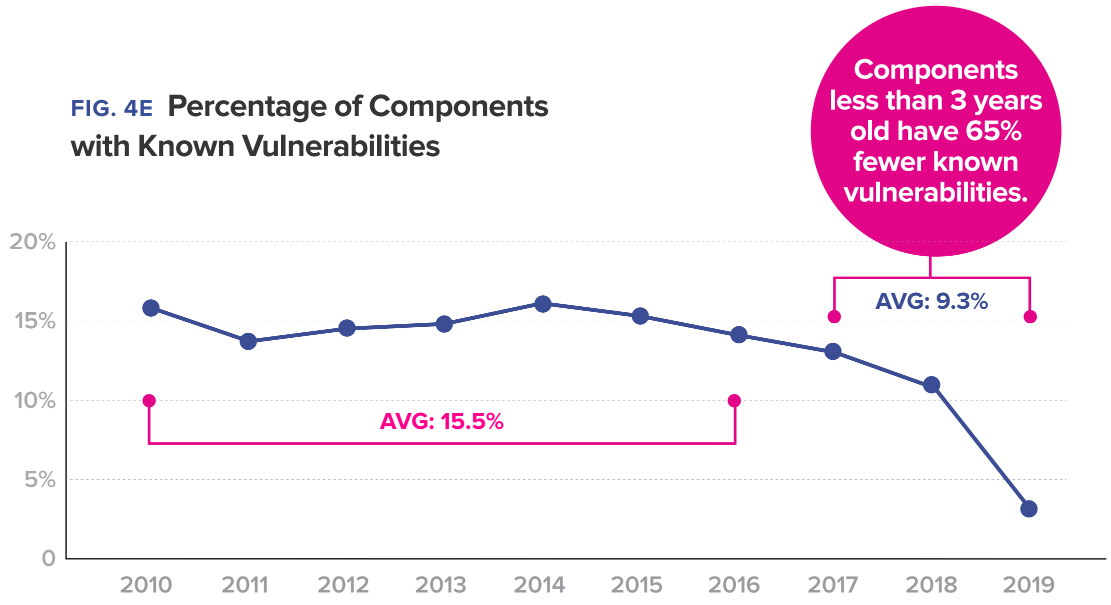


FIG. 4E Percentage of Components with Known Vulnerabilities



4.5 Rewards for Exemplary Development Teams

Development teams who regularly update their open source dependencies and manage their software supply chains can dramatically reduce their risk exposure from the use of vulnerable component releases. For 2019, researchers performed an extensive analysis of open source component releases used in *managed* and *unmanaged* supply chains.

The first set of analyses focused on managed software supply chains. The team looked at open source component releases that were used within 85,000 applications. Analysis of the Java open source component releases revealed the latest versions had the lowest percentage of known defects. Component releases under three years in age reflected security defect rates of 9.3% — representing 45% of OSS parts used in the applications.

Researchers also evaluated older component releases used within managed software supply chains. Analysis reveals that component releases over three years in age demonstrated security defect rates at 15.3% — or 65% higher defect rates compared to the newer component. The older component releases represented 55% of the parts used within managed software supply chains.

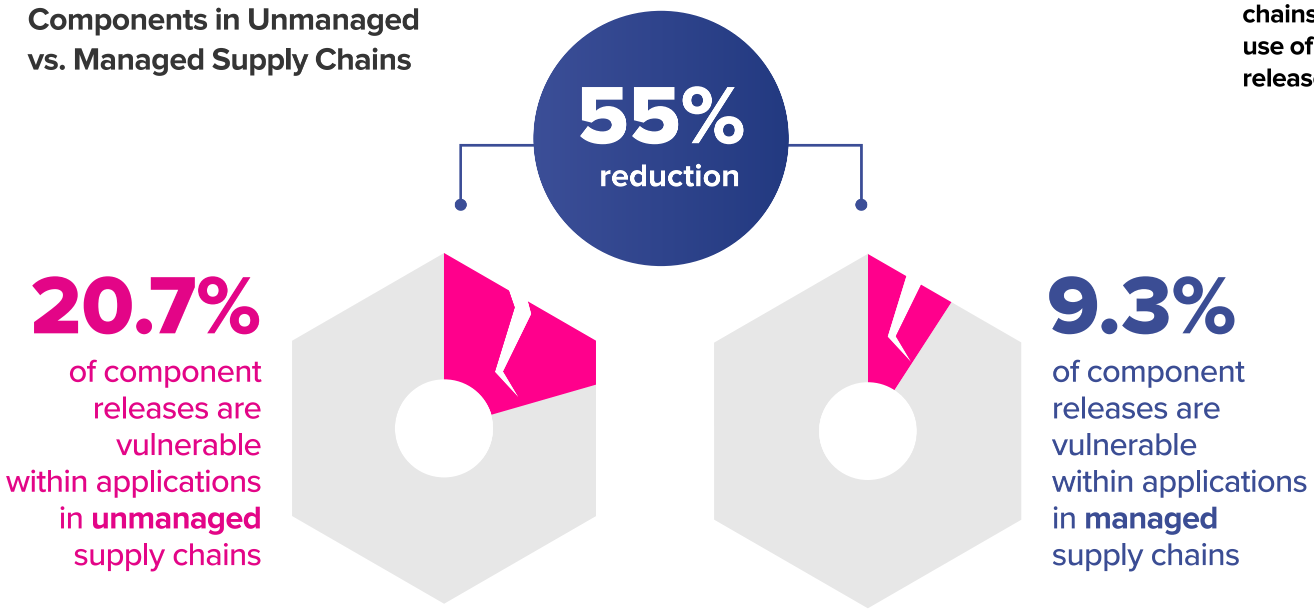
The second set of analyses focused on unmanaged software supply chains where automated open source governance solutions were not present. The average application assembled within this set revealed a security defect rate of 21%. While researchers were not able to assess the age of component releases across the population, defect rates here were 40 – 100% higher when compared to components used within managed software supply chains.

Managed supply chains make better software. For organizations who tamed their supply chains, the rewards were impressive: use of known vulnerable component releases was reduced by 55%.

KEY POINTS

- **Component releases under three years in age reflected security defect rates of 9.3%.**
- **Analysis reveals that component releases over three years in age demonstrated security defect rates at 15.3%.**
- **Applications in unmanaged software supply chains revealed security defect rates of 21%.**
- **For organizations who tamed their supply chains, the rewards were impressive: use of known vulnerable component releases was reduced by 55%.**

FIG. 4F **Proportion of Vulnerable Components in Unmanaged vs. Managed Supply Chains**



CHAPTER 5

The Changing Landscape

Vulnerabilities, Adversaries, and Government Influence

5.1 Deming Emphasizes Building Quality In

In 1945, W. Edwards Deming started advising Japanese manufacturers to detect and fix defects at the beginning of the manufacturing process. By the 1960s, Deming's TQM practices were an intrinsic part of the Japanese culture and were playing rise to their global dominance. Now tied into high-performance production processes, six-sigma manufacturing today aims a defect rate goal of 3.4 parts per million.[31]

Following on the lessons of Deming, Jez Humble and Dave Farley, in their seminal book *Continuous Delivery (2010),* advised development teams to "build quality in." Further echoing those remarks three years later, Gene Kim advised readers to "emphasize performance of the entire system and never pass a defect downstream" as he introduced "the three ways of DevOps" inside *The Phoenix Project.*

These recommendations would come to form the basis for the DevSecOps movement: release faster, build quality into your applications, and automate security controls with minimal friction.

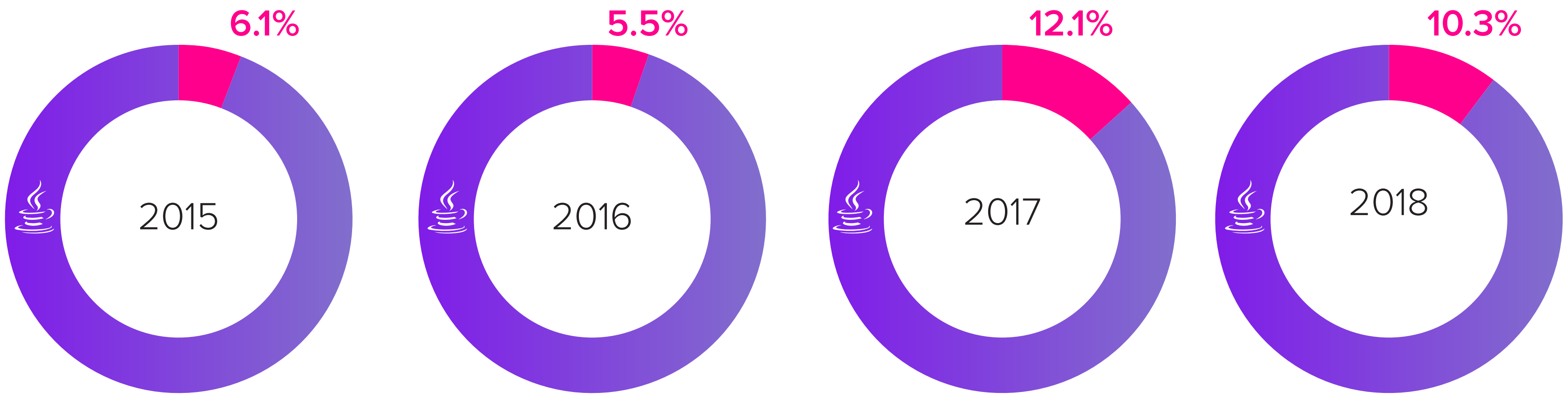
In order to survive and thrive in today's application economy the best development teams are actively pursuing open source fueled innovations, DevSecOps transformations, and automation-based controls of component releases flowing through their software supply chains.

To better understand how defective and known vulnerable component releases flow through software supply chains, we first have to look at public open source repositories. In general, a public repository is created to serve components for a given development language. For example, Python, Java, JavaScript, Ruby, and Rust components are all served from repositories specific to their development language.

KEY POINT

- **Now tied into high-performance production processes, six-sigma manufacturing today aims a defect rate goal of 3.4 parts per million.**

FIG 5A **The Percentage of Vulnerable Java Components Downloaded Over Four Years**



5.2 Tracing Vulnerable Component Release Downloads Across Software Supply Chains

Public open source repositories are immutable by design. This means, any given public repository is the home to all good (and bad) versions of open source component releases. When new vulnerabilities are discovered in component releases, open source projects cannot simply remove the defective component release from the repository, and for good reason. In order for developers to remediate a vulnerable component release, they need access to the defective component release. They first rebuild the application to its original state using the defective version, and replace the vulnerable component release with the safe version.

5.2.1 Mutuum Cavete (Borrower Beware)

It is important to understand that no metadata about a component release downloaded from the public repositories is made available to developers by default (e.g., known security vulnerabilities, release age, observed licenses, interoperability changes, etc.).

Without sufficient due diligence by developers on their component release selections, a vulnerable component download behaves exactly like a safe download. Over the past five years that Sonatype has been tracking downloads from the Central Repository for this report, the percentage of vulnerable Java component releases consumed has ebbed and flowed. In 2018, across billions of open source component release downloads, 1 in 10 (10.3%) had known security vulnerabilities.

This represents a slight decrease from 2017's peak of 12.1% (1 in 8) but when considered as a quality benchmark for managed supply chains it falls far short of expectations. As mentioned earlier, six sigma manufacturing goals aim for defect rates of 3.4 parts per million. At today's defect rate of 1 in 10 downloads, software component releases procured by development teams now sit at one sigma.

Poor hygiene practices are also well documented within the JavaScript community. In a 2018 npm survey of over 33,000 worldwide developers, 83% expressed concern about whether the open source software they use is secure (up from 77% in 2017), and 58% believe that there aren't satisfactory methods for evaluating whether code is safe.[32] Furthermore, an October 2018 report from npm revealed that 51% of JavaScript package downloads contained known vulnerabilities. Eleven percent (11%) of the downloaded component releases analyzed were rated critical and 35% demonstrated high vulnerability ratings.[33]

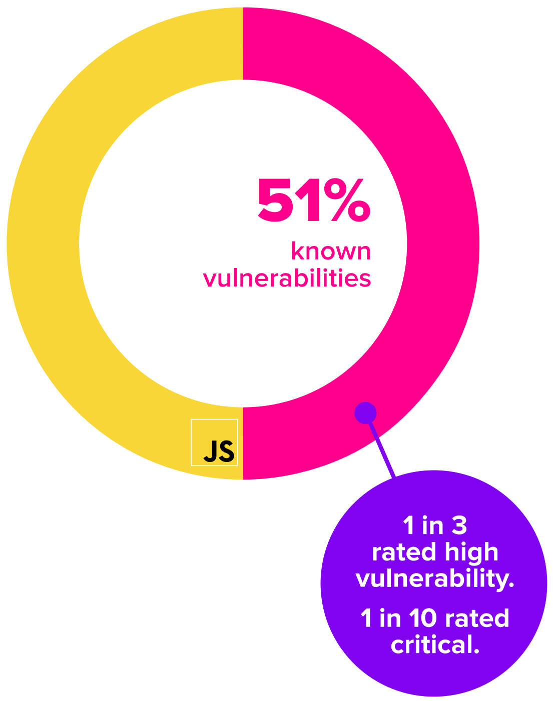
At the next stop along the software supply chain, researchers analyzed downloads to repository managers. Analysis of downloads to Nexus and Artifactory repository managers (i.e., the local warehouses of software supply chains) by development teams reveals that 7.5% were known vulnerable. Once cached in a private, local repository manager, the component release can be served an unlimited number of times to developers within that enterprise.

5.2.2 A Faustian Bargain?

The capabilities that make open source libraries so attractive to development organizations, also make them risky. While component releases are free for developers to download, they are not created equal, and all of them have a cost with respect to maintenance over time.

FIG 5B **Half of JavaScript Package Downloads Contain Known Vulnerabilities**

SOURCE: NPM QUICK AUDIT (4M SCANS PER WEEK)

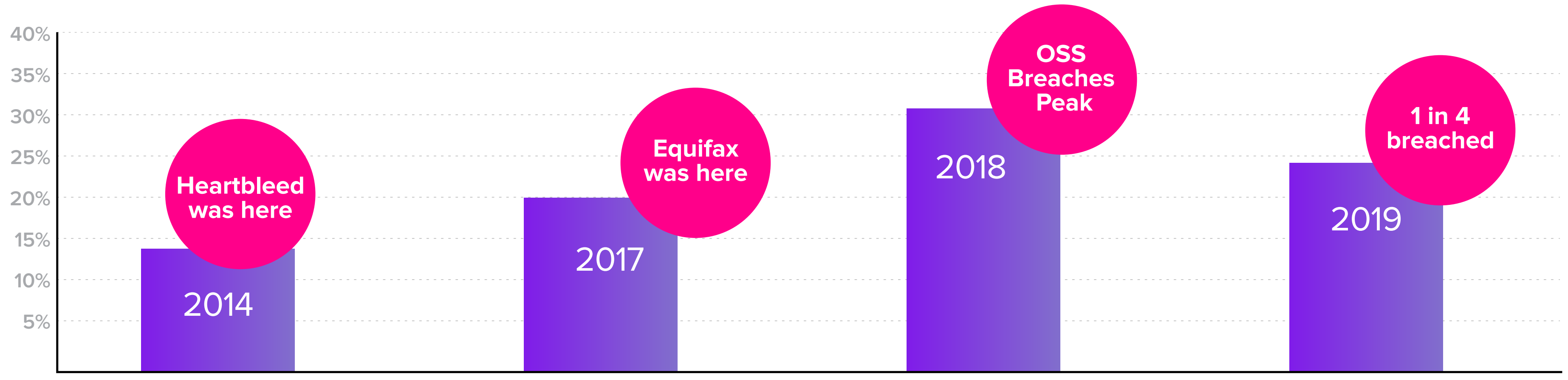


KEY POINTS

- **In 2018, across billions of open source component release downloads, 1 in 10 (10.3%) had known security vulnerabilities.**
- **At today's defect rate of 1 in 10 downloads, software component releases procured by development teams now sit at one sigma.**
- **51% of JavaScript package downloads contained known vulnerabilities.**

FIG 5C **Suspected or Verified Open Source Related Breaches Over Four Years**

SOURCE: DEVSECOPS COMMUNITY SURVEY (SONATYPE)



14%
suspect or have verified a breach related to open source components in the **2014 survey.**

20%
suspect or have verified a breach related to open source components in the **2017 survey.**

31%
suspect or have verified a breach related to open source components in the **2018 survey.**

24%
suspect or have verified a breach related to open source components **in the last 12 months.**

Our study of 12,000 enterprise software development organizations revealed average annual component release downloads of 313,000. Further analysis of those downloads reveals that 27,704 (8.8%) included at least one known security vulnerabilities. Just as well, not all security vulnerabilities are created equal. Of the 27,704 vulnerable downloads, 67% had Common Vulnerability Scoring System (CVSS) at 7.0 or above on a 10 point scale. Thirty percent (30%) had CVSS scores above 9.0 on a 10 point scale.

Minor fluctuations in the percentage of vulnerable download were seen on a country by country basis: United States (8.9%), France (8.8%), United Kingdom (8.8%), and Germany (8.1%).

5.3 Adversaries Increasingly Target Open Source Components

The 2019 DevSecOps Community Survey of 5,558 development professionals, highlighted a 71% increase in confirmed or suspected open source related breaches since 2014 (the same year the notorious OpenSSL Heartbleed vulnerability).[34]

The percentage of open source related breaches dropped 7% compared to the 2018 survey responses. The slight decline in breaches between the two surveys may be attributed to improved open source hygiene and investments made by some organizations following the Equifax breach

KEY POINTS

- **Further analysis of downloads in one study revealed that 8.8% included at least one known security vulnerabilities.**
- **Minor fluctuations in the percentage of vulnerable download were seen on a country by country basis: United States (8.9%), France (8.8%), United Kingdom (8.8%), and Germany (8.1%).**
- **The 2019 DevSecOps Community Survey highlighted a 71% increase in confirmed or suspected open source related breaches since 2014.**

— made public in late 2017. Still, with 1 in 4 survey participants in this year's reporting breaches in the past 12 months — breaches remain at epidemic levels.[35]

5.3.1 A Post-Equifax Look at Apache Struts Vulnerabilities

According to Fortinet's Q4'2018 Threat Report, vulnerable instances of Apache Struts remain the most prevalent exploit. "Demonstrating that the internet never forgets, the Apache Struts exploit (associated with CVE-2017-5638) has been a top detection since its role in the infamous Equifax breach back in 2017. More recently, attackers have been using this exploit as a way to implement cryp-to-jacking functions on compromised machines."[36]

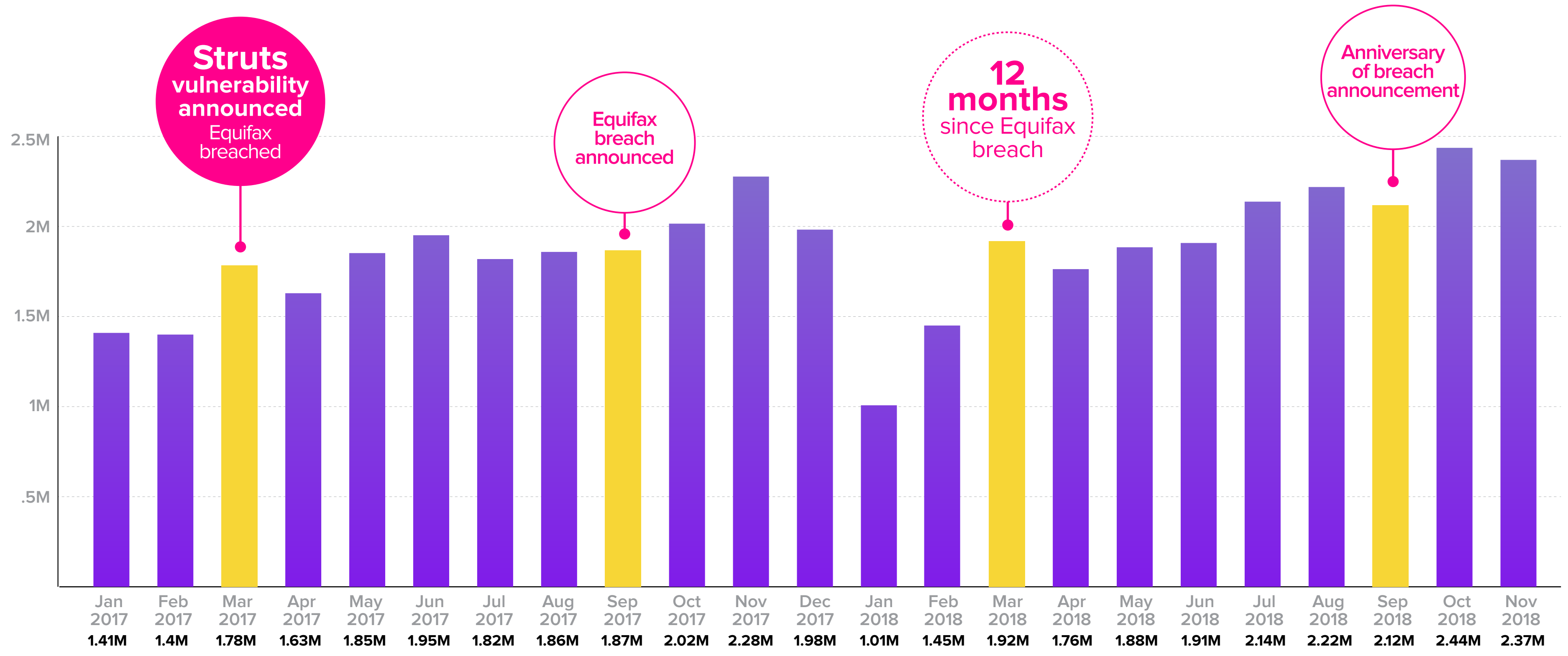
Even more alarming is the trend in elective down-loads of vulnerable Struts component releases. According to Sonatype's analysis of Struts down-loads from the Central Repository, the volume of monthly vulnerable downloads continued to increase following its link to the breach at Equifax.

KEY POINT

- **One year after the breach announcement, monthly vulnerable Struts downloads had increased 11% to 2.1 million.**

FIG 5D **Vulnerable Struts Download Counts January 2017 – November 2018**

SOURCE: SONATYPE



One year after the breach announcement, monthly vulnerable Struts downloads had increased 11% to 2.1 million.

Over the past two years, a dangerous new trend has emerged. **Adversaries are taking advantage of a new attack vector** where they are directly injecting vulnerabilities into open source project releases and container images.

5.3.2 Event-Stream: Malicious Code Injection Targeting Cryptocurrency

In the 2018 State of the Software Supply Chain Report, we advised that adversaries had compressed the time between vulnerability announcement and exploit from 45 to 3 days. In some instances, by injecting malicious code into open source projects on the supply side, time to exploit was further reduced to zero.

Such was the case with the socially engineered code injection for the JavaScript npm package known as event-stream in November 2018. The injection of malicious code into event-stream was accomplished when its developer unwittingly handed his credentials to an adversary who offered to take over maintenance responsibilities. The npm package is downloaded 2 million times per week.[37]

Further investigation of this exploit determined it was targeted as a CoPay's Bitcoin wallet and "designed to harvest account details and private keys from accounts having a balance of more than 100 Bitcoin or 1000 Bitcoin Cash."[38]

5.3.3 Bootstrap-Sass: Malicious Injection of a Back Door

Bootstrap-sass is an open source framework that enables web designers to quickly build a site. With over 28 million downloads through March 2019, it is extremely popular among developers.[39]

On March 27 of this year, Derek Barnes, a software developer whose code relied on the popular Ruby Gems bootstrap-sass component had a build fail. Suspicious, Derek decided to do some research and noticed that "someone" had removed a version of the library (Bootstrap-Sass v3.2.0.2) and immediately released a new version, moments later, v3.2.0.3. He was suspicious why "someone" would modify the library on RubyGems — but not in GitHub, where the library's source code is managed. What Barnes uncovered was sobering: another attack on open source and the software supply chain that underpins so much of modern innovation.[40]

Alarmed by the abnormal release, he alerted the project which led to a backdoor being discovered in the code. The malicious component version (3.2.0.3) was then removed from the RubyGems repository — and the Bootstrap-Sass team revoked access to RubyGems for the developer whose account they believed was compromised and used to push the malicious code.

Before being noticed, the vulnerable version of bootstrap-sass was downloaded over 1,400 times.[41]

5.3.4 Agama Malicious Code Injection

In June 2019, the npm, Inc. security team, in collaboration with Komodo, helped protect over $13 million USD in cryptocurrency assets by finding and responding to a malicious code injection vulnerability targeting the users of a cryptocurrency wallet called Agama. The attack focused on getting a malicious package into the build chain for Agama and stealing the wallet seeds and other login passphrases used within the application. According to npm, "the attack was carried out by using a pattern that is becoming more and more popular; publishing a 'useful' package (electron-native-notify) to npm, waiting until it was in use by the target, and then updating it to include a malicious payload."[42]

5.3.5 Two Years of Malicious Code Injection

The mechanics to the bootstrap, event stream, and electron-native-notify attacks are tricky and yet effective.

While limited in scope, the potential impact of similar attacks through a very popular component can be far-reaching and immediate. When malicious code is injected into software supply chains, adversaries can attack immediately after the code is deployed.

Over the past two years, a dangerous new trend has emerged. Specifically, a series of 16 events triangulate a serious escalation of software supply chain attacks. Adversaries are taking advantage of a new attack vector where they are directly injecting vulnerabilities into open source project releases and container images (see **FIGURE 5E**).

continues on page 42

FIG 5E **A Shifting Battlefront of Attacks: Malicious Code Injection**

July 2017 – June 2019

JUL 2017

npm credentials published online.

Affects access to 14% of the npm repo (79,000 packages).

Malicious npm packaged typosquatted.

40 packages harvested over two weeks, collecting credentials used to publish to the npm repository itself.

docker123321 images created on Docker Hub.

Later accused of poisoning a Kubernetes honeypot (Jan 2018), and equated to a crypto-mining botnet (May 2018).

SEP 2017

PyPI typosquat: 10 malicious Python packages found.

Evidence of the fake packages being incorporated into software was noted multiple times between June and Sept 2017.

JAN 2018

"I'm harvesting credit card numbers and passwords from your site. Here's how."

David Gilbertson writes a fictional tale on his blog about creating a malicious npm package.

FEB 2018

Deleted go-bindata account resurrected by an unknown user.

After a developer deleted their GitHub account, someone immediately grabbed the ID — inheriting the karma instilled in that id and calling into question packages and sources.

Back-doored PyPI package discovered.

Python module ssh-decorator back-doored to enable theft of private ssh keys.

MAR 2018

npm credentials intentionally compromised.

A malicious version of a package from a core contributor to the conventional-changelog ecosystem is published. The package was installed 28,000 times in 35 hours and executed a Monero crypto miner.

MAY 2018

Back-doored npm package discovered.

npm security team responds to reports of a malicious back door in the get-cookies module, published in March. Despite being deprecated, mailparser still receives about 64,000 weekly downloads.

JUN 2018

Linux distro hacked on GitHub.

Unknown individuals gain control of the Github Gentoo organization, and modified the content of repositories as well as pages within. All code considered compromised.

JUL 2018

Compromised JavaScript package caught stealing npm credentials.

A hacker gains access to a developer's npm account and injects malicious code into a popular JavaScript library called eslint-scope, a sub-module of the more famous ESLint, a JavaScript code analysis toolkit.

AUG 2018

Homebrew repository compromised.

Accessed in under 30 minutes through an exposed GitHub API token.

NOV 2018

Malicious package injected into event-stream, a popular npm package.

The injected code targets the Copay application and was designed to harvest account details and private keys from accounts having a balance of more than 100 Bitcoin or 1,000 Bitcoin Cash.

MAR 2019

Back-doored Gems bootstrap-sass RCE package discovered.

A malicious version of the popular bootstrap-sass package, downloaded a total of 28 million times to date, and with 1.6K dependencies, is published to the RubyGems repository.

JUN 2019

Cryptocurrency attack via malicious code injection.

Malicious code targets users of a cryptocurrency wallet called Agama, focusing on getting into the build chain and stealing the wallet seeds and other login passphrases used within the application.

5.4 Government and Industry Apply New Standards to Secure Software Development

While Exemplars are reducing the use of known vulnerable open source component releases in the development application, this does not exonerate them from duties to manage components over time. Secure software practices extend from early development through the active life of an application in the market.

With an ever increasing number of application breaches occurring, standards bodies and government are stepping in to hold development organizations accountable for the quality and security of code they assemble and develop.

5.4.1 New PCI Secure Software Development Standards

In January 2019, the Payment Card Industry Security Standards Council introduced important new standards for software development:

- PCI Secure Software Standards,[43] and
- PCI Secure Software Lifecycle (Secure SLC) Standard[44]

The new standard requires organizations to govern their use of open source software, and it states that any application utilized as part of the payment process, must be secure by design. Specifically, when it comes to the use of open source components and third-party libraries organizations are responsible for ensuring they, as well as any of their vendors, have:

- An up to date inventory of open-source component releases utilized in the software
- A process for identifying known vulnerabilities within open source component releases
- 360 degree monitoring of open source component releases throughout the SDLC
- A policy and process to immediately remediate vulnerabilities as they become known.

To effectively utilize open source components at scale, the PCI standard also advises organizations to generate a software bill of materials (SBOM) so they can easily track and trace the location of every single component release embedded within their production software applications.

5.4.2 National Telecommunications and Information Administration's (NTIA) SBOM Initiative

The Commerce Department's National Telecommunications and Information Administration (NTIA) is considering requiring companies to list their sources of software parts to protect the U.S. software supply chains.

According to the NTIA, their "cybersecurity multistakeholder process will focus on Software Component Transparency. Participants will explore how manufacturers and vendors can communicate useful and actionable information about the third-party software components that comprise modern software and IoT devices, and how this data can be used by enterprises to foster better security decisions and practices."[45]

The NTIA effort comes as the government agencies grow more suspicious of known vulnerable software components being used in application development and increasing awareness of malicious code injection attacks. The NTIA initiative has broad support across Federal agencies and the private sector, as they work together to define standards around a software bill of materials (SBOM).

KEY POINTS

- **The SBOM permits organizations to make informed risk decisions about which technologies to purchase and use based on known vulnerability information.**
- **When new vulnerabilities are discovered, an SBOM allows organizations to quickly identify their exposure and to take appropriate steps in response."**

5.4.4 U.S. House Energy and Commerce Committee

In December 2018, the U.S. House Energy and Commerce Committee released its Cybersecurity Strategy Report. The report details the importance and priority for utilizing Software Bill of Materials (SBOM) for minimizing supply chain risks related to the use of open source software components in modern application development.[46]

The report states, the "SBOM becomes an ingredients list for a given piece of technology, listing the hardware, software, and other relevant components that it contains or relies upon. This creates two primary benefits. First, it permits organizations to make informed risk decisions about which technologies to purchase and use based on known vulnerability information. Second, when new vulnerabilities are discovered, it allows organizations to quickly identify their exposure and to take appropriate steps in response."

Next the authors from the Energy and Commerce Committee pointed out that it "was not that organizations did not know which software was vulnerable… it was that they did not know which pieces of technology that *they depended on* included it. The SBOM minimizes the number of unknown unknowns with which organizations must contend, and greatly increases their ability to protect themselves, their users, and ultimately society."

5.4.5 U.S. Food and Drug Administration

In October 2018, the FDA released guidance for cybersecurity management of medical devices. Similar to recommendations delivered by the House Energy and Commerce Committee, the FDA's report called for a Cybersecurity Bill of Materials (CBOM).

The aim of the CBOM is to provide a list of "commercial, open source, and off-the-shelf software and hardware components to enable device users (including patients, providers, and healthcare delivery organizations (HDOs) to effectively manage their assets, to understand the potential impact of identified vulnerabilities to the device (and the connected system), and to deploy countermeasures to maintain the device's essential performance."

The FDA report also suggested that MDMs may be subject to legal liabilities tied to the distribution of a medical device with a known vulnerability. The report offered, "If a medical device cybersecurity vulnerability allegedly causes bodily harm, the victim may bring product liability claims against the MDM." The FDA warned that a plaintiff's claim could be based on the "failure to provide appropriate warnings about the risk of a cybersecurity vulnerability, or failure to appropriately monitor for the existence of vulnerabilities and appropriately address them once identified."[47]

KEY POINTS

- **In October 2018, the FDA released guidance for cybersecurity management of medical devices. The FDA's report called for a Cybersecurity Bill of Materials (CBOM).**
- **The FDA report suggests that MDMs may be subject to legal liabilities tied to the distribution of a medical device with a known vulnerability.**

Conclusion

Decades ago, W. Edwards Deming taught key principles to significantly improve the effectiveness and quality of business manufacturing processes. Deming deftly advocated selecting the best suppliers and emphasized continuous improvement. He advised eliminating the need for inspection on a mass basis by building quality into products.[48]

Businesses racing to deliver better value to their customers — and differentiate from competitors — are embracing Deming's principles within their open source based software development practices. Software development is evolving from artisan based creations to practices that more closely resemble high velocity parts assembly. This is reflected in the exponential growth of supply and demand for open source components. We've observed double and triple digit growth in open source component ecosystems for a decade, and there is no slowdown in sight.

The purpose of this report was to share with you what we observed across software supply chains. Our findings are clear. Velocity does not have to come at the cost of reduced security.

Exemplary open source project initiatives benefit tremendously from higher code commit and release frequencies. They also do an outstanding job of managing their dependencies. At the same time, Exemplars in enterprise are benefiting from processes that support using the latest component versions. They also embrace automated practices to reduce the presence of known vulnerabilities.

Our deep examination of consumption patterns, development practices, and cybersecurity hygiene revealed:

- 18x faster median time to update dependencies for exemplary open source components
- 6.2x more likely for exemplary enterprise development teams to use the latest open source component version (or latest-N)
- 12x higher use of automation to manage open source dependencies in exemplary enterprise development teams
- 55% reduction in the use of vulnerable open source components within managed software supply chains

Management of software supply chains is not simply ensuring quality at velocity. Our supply chains are being attacked by adversaries in new and creative ways. The result: open source related breaches have jumped 71% over the past five years.

As enterprises look for guidance to improve their software development and security practices, industry standards groups are introducing open source awareness policies. Simultaneously, we've seen government agencies racing to employ new policies and legislation to protect citizens, businesses, and critical infrastructure.

One thing is clear, exemplary software development practices that deliver high quality, improved security at high velocity are not rare. They are being employed today in large numbers and serve as benchmarks for others to strive for and achieve.

Thank you for reading our 5th annual State of the Software Supply Chain Report. We hope you found it useful. And, we welcome your feedback.

Sources

1 Sonatype, npmjs.org, pypi.org, NuGet.org

2 hub.docker.com

3 Sonatype and modulecounts.com

4 npmjs.org and modulecounts.com

5 developers.slashdot.org/story/17/01/14/0222245/nodejss-npm-is-now-the-largest-package-registry-in-the-world

6 www.idc.com/getdoc.jsp?containerId=US44363318

7 www.oracle.com/java/java9.html

8 twitter.com/seldo/status/1105987692305604608

9 9.7M developers use JavaScript: appdevelopermagazine.com/9.7m-developers-use-javascript/

10 Sonatype and rubygems.org

11 Sonatype and nuget.org

12 https://chaoss.community

13 https://libraries.io

14 Sonatype.com/nexus-intelligence

15 The (Application) Patching Manifesto (Jeremy Long): www.youtube.com/watch?time_continue=14&v=qVVZrTRJ290

16 Why and How Java Developers Break APIs (Aline Brito, Laerte Xavier, Dr. Andre Hora, Dr. Marco Tulio Valente): arxiv.org/pdf/1801.05198.pdf

17 A Pearson correlation is a number between -1 and 1 that indicates the extent to which two variables are linearly related. The Pearson correlation is also known as the "product moment correlation coefficient" (PMCC) or simply "correlation." en.wikipedia.org/wiki/Pearson_correlation_coefficient

18 Kendall's Tau is a non-parametric measure of relationships between columns of ranked data. The Tau correlation coefficient returns a value of 0 to 1, where: 0 is no relationship, and 1 is a perfect relationship. www.statisticshowto.datasciencecentral.com/kendalls-tau

19 statisticssolutions.com/mann-whitney-u-test

20 "Announcing Accelerate: State of DevOps 2018: Strategies for a New Economy" (Dr. Nicole Forsgren, Jez Humble, Gene Kim): devops-research.com/2018/08/announcing-accelerate-state-of-devops-2018

21 Sonatype 2019 DevSecOps Community Survey

22 "Announcing Accelerate: State of DevOps 2018: Strategies for a New Economy" (Dr. Nicole Forsgren, Jez Humble, Gene Kim): devops-research.com/2018/08/announcing-accelerate-state-of-devops-2018

23 Sonatype and jfrog.com/about/press/jfrog-secures-165-million-investment-to-lead-universal-devops-in-the-enterprise

24 This year in JavaScript: 2018 in review and npm's predictions for 2019: www.medium.com/npm-inc/this-year-in-javascript-2018-in-review-and-npms-predictions-for-2019-3a3d7e5298ef

25 The (Application) Patching Manifesto (Jeremy Long): www.youtube.com/watch?time_continue=14&v=qVVZrTRJ290

26 Sonatype 2019 DevSecOps Community Survey

27 Sonatype 2019 DevSecOps Community Survey

28 Sonatype 2019 DevSecOps Community Survey

29 Sonatype 2019 DevSecOps Community Survey

30 Sonatype 2019 DevSecOps Community Survey

31 www.isixsigma.com/new-to-six-sigma/statistical-six-sigma-definition

32 https://javascriptsurvey.com

33 npm and the future of JavaScript: slides.com/seldo/npm-future-of-javascript#/18

34 Sonatype 2019 DevSecOps Community Survey and 2014 Open Source and Application Security Survey

35 Sonatype 2019 DevSecOps Community Survey

36 Fortinet Threat Report, Q4'2018

37 Details about the event-stream incident: blog.npmjs.org/post/180565383195/details-about-the-event-stream-incident

38 github.com/urbit/urbit-wallet-generator/issues/4

39 github.com/twbs/bootstrap-sass

40 Corrupting the Software Supply Chain: Lessons from the Bootstrap-sass Hack: blog.sonatype.com/corrupting-the-software-supply-chain-lessons-from-the-bootstrap-sass-hack

41 github.com/twbs/bootstrap-sass

42 Plot to steal cryptocurrency foiled by the npm security team" blog.npmjs.org/post/185397814280/plot-to-steal-cryptocurrency-foiled-by-the-npm

43 New PCI Software Security Standards: blog.pcisecuritystandards.org/just-published-new-pci-software-security-standards

44 New PCI Software Security Standards: blog.pcisecuritystandards.org/just-published-new-pci-software-security-standards

45 NTIA Software Component Transparency: www.ntia.doc.gov/SoftwareTransparency

46 House Energy and Commerce Cybersecurity Strategies Report: www.hsdl.org/?view&did=819388

47 Content of Premarket Submissions for Management of Cybersecurity in Medical Devices: www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM623529.pdf

48 Dr. Deming's 14 Points for Management: deming.org/explore/fourteen-points?apartner=aarp

Appendix A

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. We began collecting data for our 2019 report from the moment our 2018 report was published.

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, Ilkka Turunen, Elissa Walters, Daniel Sauble, Adam Cazzolla, Alex Metry, Andrew Stein, Ken Duck, Kevin Hayen, Kevin Witten, Shade Solon, Alvin Gunkel, and Aaron Massey.

We would also like to offer thanks for contributions big and small from: Hasan Yasar (Carnegie Mellon University Software Engineering Institute), Laurie Voss (npm), Brian Dawson (CloudBees), DJ Schleen (Aetna CVS), Nichole Schimanski (Galois) and Eric Davis (Galois), James Wickett and others across the DevOps and open source development community.

A very special thanks goes out to Melissa Schmidt who created the incredible design for this year's report.

Finally, we could not have produced this report without the amazing contributions and countless hours of deep analysis from our research partners Gene Kim from IT Revolution and Dr. Stephen Magill, Principal Scientist at Galois & CEO of MuseDev.

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. Specifically to Chapter 3, all reported differences are statistically significant ($p < 0.05$) according to a Mann–Whitney U test. 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.

Appendix B

23 Metrics Associated with Open Source Project Health

This year's analysis includes datasets representing development velocity, team size, continuous integration (CI) usage, known security vulnerabilities, component popularity, and other attributes. The objective of this research was to look for top performing projects and characterize their dimensions of excellence. The research also suggests factors that architects, developers, and organizations should consider when making choices about which open source components to use. Our analysis also took into account over 23 metrics associated with open source project health including:

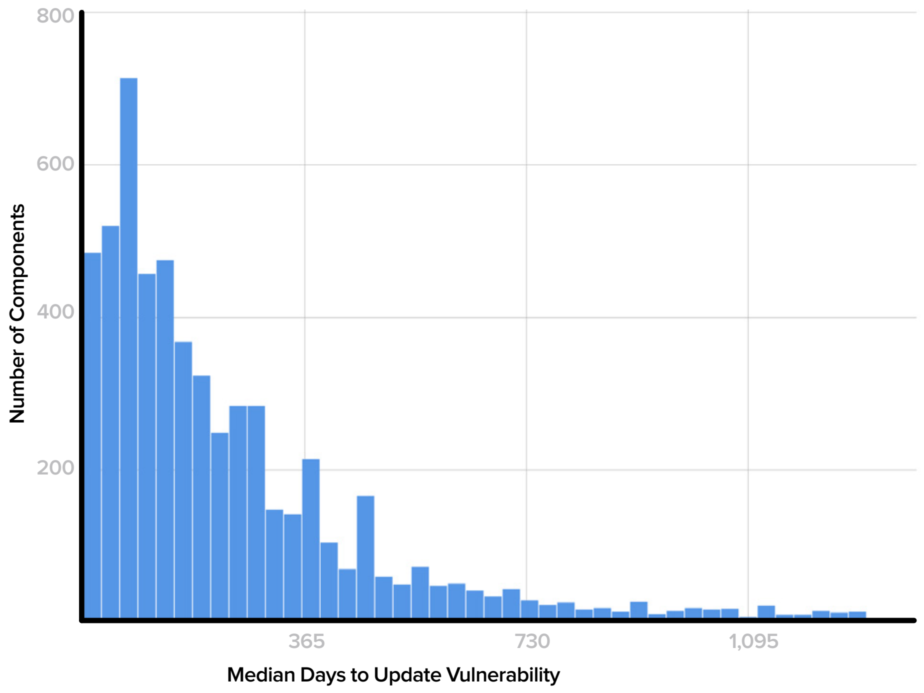
1. **Component_key:** The Maven Group ID + Artifact ID, e.g. org. spring-framework:spring-core
2. **Median_ttu_median (lower is better):** The median time to upgrade a dependency. For each dependency D, start the clock when an update to some new version V of the dependency is released. Stop it when the component has a release where the version of D is >= V.
3. **Median_ttu_sec_rel_median (lower is better):** The median TTU for updates that had a known vulnerability against them at the time the new version was released. This is the best TTR data we have.
4. **Median_ttr_median (lower is better):** A version of TTR that accounts for vulnerabilities discovered long after a component is released. Clock starts when a vulnerability is reported against a dependency of a component. Clock stops when that component updates the dependency.
5. **Already_protected_percentage (higher is better):** Records the percentage of security vulnerabilities that don't apply to this component when they come out because the component was up-to-date with dependencies.
6. **Max_dependency_count:** The number of dependencies of the component.
7. **Avg_not_adopted_avg (lower is better):** The number of dependency updates that were never applied.
8. **Avg_stale_avg (lower is better):** The average percentage of component dependencies that are out of date when a new release of that component comes out.
9. **Avg_period_when_current (lower is better):** Average time each release spends as the "current" release. Reciprocal of release frequency.
10. **Avg_stale_time_avg (lower is better):** Average period when at least one dependency is out of date. We can probably drop this one. I'm not sure it's that meaningful.
11. **Stale_time_proportion (lower is better):** The percentage of time that a component spends being out of date. Can probably drop this too.
12. **Scm_url:** The URL where the code lives.
13. **Two_dots:** The first two components of the group ID.
14. **Three_dots:** The first three components of the group ID.
15. **Central_popularity:** The average number of downloads per day from The Central Repository.
16. **NexusIQ_popularity:** The average number of IQ scans per day.
17. **Stars_count:** Number of stars.
18. **Forks_count:** Number of forks.
19. **Ci-found:** Whether we found a CI script.
20. **Avg-commits-per-month:** Average number of commits per month.
21. **Avg-commits-uniq-devs:** Average number of unique developers committing each month.
22. **Is_commercially_supported:** True if group ID starts with "com"
23. **Is_foundation_supported:** True if there are several projects managed by the same Group ID.

Appendix B

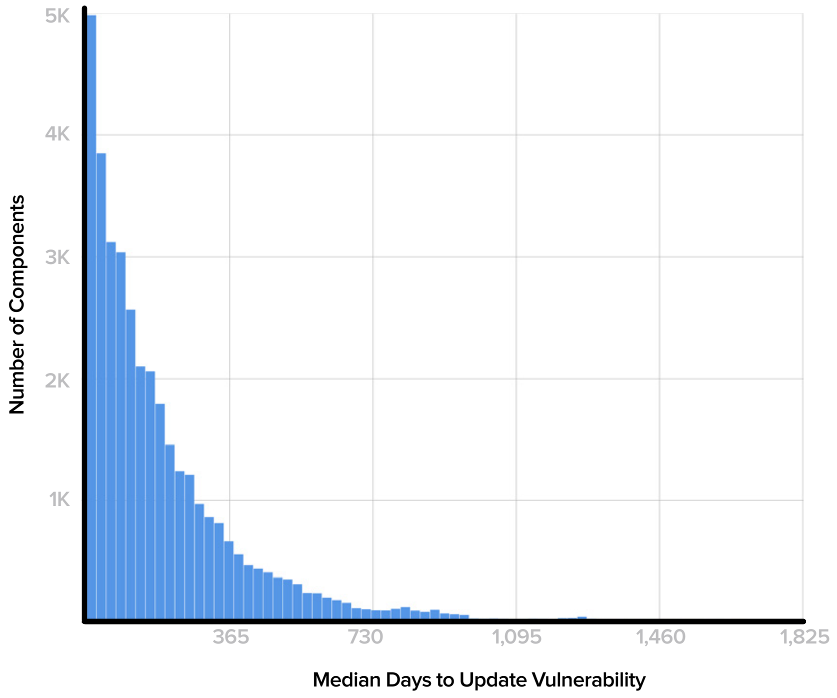
Cumulative Histogram of MTTR

For developers wanting to use secure components, having faster MTTR is desirable. The figures below show a simple histogram of MTTR and MTTU across all components.

Frequency of Median Time to Remediate Vulnerability



Frequency of Median Time to Update Vulnerable Dependencies



Appendix C

Measures of Different Behavior Clusters

Averages

	None (8142)	Small Exemplars (606)	Large Exemplars (595)	Laggards (521)	Features First (280)	Cautious (429)
TTU (Days)	245.22	18.74	10.14	692.82	602.94	103.44
Stale Dependency Percentage	0.64	0.11	0.09	0.98	0.96	0.97
Central Popularity (Average Daily Downloads)	425.5	803.67	4681.28	62.4	297.65	1349.16
Stars	907.31	191.07	1150.31	452.92	1289.35	1009.85
Forks	343.84	41.78	651.37	171.51	434.42	351.81
Number of Dependencies	3.91	3.12	6.14	3.11	3.8	4.11
Percent Using CI	0.73	0.57	0.69	0.69	0.71	0.78
Commit Frequency	42.16	19.51	69.34	24.43	56.47	51.25
Number of Developers (Unique Devs Commiting Per Month)	3.89	1.64	8.91	2.87	5.19	4.67
Average Period Between New Versions (Days)	128.43	87.62	45.64	244.25	49.99	72.14
Percent Commercially Supported	0.3	0.25	0.06	0.37	0.31	0.33
Percent Foundation Supported	0.59	0.57	0.91	0.55	0.65	0.6

Average Multiples

(as compared to "None" class)

	None (8142)	Small Exemplars (606)	Large Exemplars (595)	Laggards (521)	Features First (280)	Cautious (429)
TTU	1	0.08	0.04	2.82	2.45	0.42
Stale Dependency Percentage	1	0.17	0.14	1.55	1.52	1.53
Central Popularity (Average Daily Downloads)	1	–	11	0.15	0.7	–
Stars	1	0.21	1.27	0.5	1.42	–
Forks	1	0.12	1.89	0.5	1.26	–
Number of Dependencies	1	0.8	1.57	0.8	–	1.05
Percent Using CI	1	0.78	0.94	0.94	–	1.06
Commit Frequency	1	0.46	1.64	0.58	1.34	1.22
Number of Developers (Unique Devs Commiting Per Month)	1	0.42	2.29	0.74	1.34	1.2
Average Period Between New Versions (Days)	1	0.68	0.35	1.88	0.39	0.57
Percent Commercially Supported	1	0.85	0.19	1.24	–	–
Percent Foundation Supported		–	1.54	0.92	1.1	–

A dash (–) indicates that the observed difference between groups was not statistically significant (p > 0.05 with a Mann–Whitney U test)

Appendix D

FIG. 31 **The Exemplars: Components Demonstrating the Fastest MTTU and Lowest Stale Dependency Counts**

» bz.tsung.android:objectify
» com.ahome-it:ahome-tooling-server-core
» com.amazon.device.tools.build:builder
» com.amazon.device.tools.build:gradle
» com.amazon.device.tools.lint:lint-checks
» com.github.japgolly.fork.scalaz:scalaz-concurrent_sjs0.5_2.11
» com.github.japgolly.fork.scalaz:scalaz-xml_sjs0.5_2.11
» com.github.japgolly.fork.scalaz:scalaz-iterv_sjs0.5_2.11
» com.github.japgolly.fork.scalaz:scalaz-core_sjs0.5_2.11
» com.aranea-apps.android.libs:android-rest
» com.ariht:config-generation-maven-plugin
» com.arasthel:swissknife
» com.asayama.docs.gwt.angular:gwt-angular-pages
» com.asayama.gwt:gwt-util
» com.asayama.gwt.angular:gwt-angular-resources
» com.asayama.gwt.angular:gwt-angular-masonry
» com.asayama.gwt.angular:gwt-angular-http
» com.asayama.gwt.angular:gwt-angular-user
» com.asayama.gwt.angular:gwt-angular-prettify
» com.asayama.gwt.angular:gwt-angular-ng
» com.asayama.gwt.bootstrap:gwt-bootstrap
» com.asayama.gwt.jquery:gwt-jquery
» com.automattic:elasticsearch-statsd
» com.autoscout24.gradle:gradle-monkey-plugin
» com.damnhandy:handy-uri-templates
» com.badlogicgames.gdx:gdx-backend-robovm
» com.badlogicgames.gdx:gdx-backend-lwjgl
» com.badlogicgames.gdxpay:gdx-pay-android-googleplay
» com.badlogicgames.gdxpay:gdx-pay
» com.erinors:xtend-ioc-core
» com.bartoszlipinski:parsemodel-compiler
» com.bazaarvoice.dropwizard:dropwizard-webjars-bundle
» com.bazaarvoice.dropwizard:dropwizard-configurable-assets-bundle
» com.github.advantageous:qbit-spring
» com.github.advantageous:qbit-consul-client
» com.github.advantageous:qbit-eventbus-replicator
» com.github.advantageous:qbit-vertx
» com.github.advantageous:qbit-service-discovery
» com.github.advantageous:qbit-test-support
» com.github.advantageous:qbit-admin
» com.github.advantageous:qbit-core
» com.github.advantageous:qbit-servlet
» com.github.akarnokd:ixjava
» com.github.almondtools:rexlex
» com.github.andreptb:fitnesse-selenium-slim
» com.github.andrewoma.kommon:kommon
» com.github.andrewoma.kwery:fetcher
» com.github.andrewoma.kwery:core
» com.github.andrewoma.kwery:transactional
» com.github.andrewoma.kwery:mapper
» com.github.aro-tech:tdd-mixins-core
» com.github.aro-tech:extended-mockito
» com.github.ben-manes.caffeine:guava
» com.github.chandu0101.scalajs-react-components:macros_sjs0.6_2.11
» com.github.czyzby:gdx-lml
» com.github.danielgindi:helpers
» com.github.davidmoten:bigsort
» com.github.davidmoten:rxjava-extras
» com.github.dblock:oshi-core
» com.github.ddth:ddth-osgikafka
» com.github.ddth:ddth-zookeeper
» com.github.dnvriend:akka-persistence-jdbc_2.10
» com.github.doctoror.rxcursorloader:library
» com.github.fbertola:mother-docker
» com.github.finagle:finch-oauth2_2.10
» com.github.finagle:finch-demo_2.11
» com.github.fracpete:screencast4j-weka-package
» com.github.gabrielemariotti.cards:library-extra
» com.github.heuermh.adamexamples:adam-examples_2.11
» com.github.heuermh.adamplugins:adam-plugins
» com.github.heuermh.adamplugins:adam-plugins_2.11
» com.github.heuermh.adamplugins:adam-plugins_2.10
» com.github.j-fischer:rest-on-fire
» com.github.jinahya:simple-file-back
» com.github.jodersky:flow_2.11
» com.github.joschi:dropwizard-elasticsearch
» com.github.jsonld-java:jsonld-java-sesame
» com.github.jsurfer:jsurfer-simple
» com.github.jszczepankiewicz:dynks
» com.github.jtakakura:gradle-robovm-plugin
» com.github.kentyeh:sd4j
» com.github.kzwang:elasticsearch-river-dynamodb
» com.github.kzwang:elasticsearch-transport-redis
» com.github.kzwang:elasticsearch-osem
» com.github.kzwang:elasticsearch-repository-gridfs
» com.github.mhshams:core
» com.github.michaelruocco:wso2-api-publisher-plugin
» com.github.mictaege:doozer
» com.github.nkzawa:engine.io-client
» com.github.nwillc:contracts
» com.github.oscerd:camel-cassandra
» com.github.pengrad:java-telegram-bot-api
» com.github.persapiens:jsf-undertow-spring-boot-starter
» com.github.persapiens:jsf-undertow-bootsfaces-spring-boot-starter
» com.github.persapiens:jsf-jetty-bootsfaces-spring-boot-starter
» com.github.pwittchen:reactivenetwork
» com.github.pwittchen:reactivebeacons
» com.github.ratrecommends:gdx-utils
» com.github.richard-ballard:arbee-test-utils
» com.github.richard-ballard:arbee-utils
» com.github.salomonbrys.kodein:kodein
» com.github.salomonbrys.kodein:kodein-android
» com.github.scala-blitz:scala-blitz_2.11
» com.bladecoder.engine:blade-engine-spine-plugin
» com.bladecoder.engine:blade-engine
» com.bladejava:blade-jetbrick
» com.bloidonia:groovy-stream
» com.github.sd4324530:fastweixin
» com.github.seratch:ltsv4s_2.11
» com.github.seratch:scalikesolr_2.10
» com.github.seratch:jslack
» com.github.sogyf:goja-qrcode
» com.github.sv244:torrentstream-android
» com.braintreepayments.api:braintree
» com.braintreepayments.api:braintree-api
» com.github.sviperll:metachicory
» com.github.sviperll:adt4j-core
» com.github.thomasnield:rxkotlinfx
» com.github.tibolte:agendacalendarview
» com.github.tkurz.sesame:vocab-builder-cli
» com.github.tkurz.sesame:vocab-builder-maven-plugin
» com.github.triceo.splitlog:splitlog-core
» com.github.vmironov.jetpack:jetpack-bindings-arguments
» com.github.webdriverextensions:webdriverextensions
» com.github.wnameless:smartcard-reader
» com.github.xuwei-k:httpz-scalaj_2.11
» com.github.xuwei-k:msgpack4z-java07
» com.github.xuwei-k:play23scalacheck111_2.11
» com.github.xuwei-k:applybuilder71_2.11
» com.github.xuwei-k:msgpack4z-java
» com.github.xuwei-k:play23scalaz71_2.11
» com.github.xuwei-k:play23scalaz70_2.11
» com.github.xuwei-k:play-twenty-three_2.11
» com.digitalpebble:storm-crawler-tika
» com.cedarsoft.commons:configuration
» com.digitalpebble:storm-crawler
» com.cedarsoft.commons.history:core
» com.mailosaur:mailosaur-java
» com.pengyifan.bioc:pengyifan-bioc
» com.cloudhopper:ch-smpp
» com.cocosw:framework
» com.codeborne:selenide
» com.codebullets.saga-lib:saga-lib-guice
» com.puppycrawl.tools:checkstyle
» com.dimafeng:testcontainers-scala
» com.redowlanalytics:swagger2markup-maven-plugin
» com.rtstatistics:api-client
» com.craterdog.java-security-framework:java-digital-notary-api
» com.cyngn.vertx:vertx-kafka
» com.ea.orbit:orbit-rest-client
» com.ea.orbit:orbit-actors-spring
» com.valchkou.datastax:cassandra-driver-mapping
» com.ea.orbit:orbit-actors-redis
» com.eharmony:aloha-vw-jni
» com.englishtown.vertx:vertx-httpservlet
» com.englishtown.vertx:vertx-zookeeper
» com.englishtown.vertx:vertx-guice
» com.englishtown.vertx:vertx-when

» info.cukes:cucumber-picocontainer
» com.erudika:para-dao-cassandra
» com.eventsourcing:eventsourcing-core
» com.evernote:android-sdk
» com.facebook.presto:presto-teradata-functions
» com.facebook.presto:presto-cli
» com.facebook.presto:presto-orc
» com.facebook.presto:presto-blackhole
» com.facebook.presto:presto-server
» com.floragunn:search-guard-5
» com.floragunn:search-guard-ssl
» com.flozano.statsd-netty:statsd-netty
» com.gabrielittner.auto.value:auto-value-cursor
» com.getbase.android.autoprovider:library
» com.giffing.wicket.spring.boot.
starter:wicket-spring-boot-starter-example
» de.leanovate.doby:doby_2.11
» com.gocardless:gocardless-pro
» com.godmonth:godmonth-commons
» com.googlecode.jmapper-framework:jmapper-core
» es.litesolutions:sonar-sslr-grappa
» com.hack23.cia:service.external.worldbank
» com.hack23.cia:model.external.worldbank.data.impl
» com.hack23.cia:service.data.impl
» com.hack23.cia:model.external.vdem.indicators.impl
» com.hack23.cia:citizen-intelligence-agency
» com.hack23.cia:model.external.val.
landstingvalkrets.impl
» com.hack23.cia:model.external.riksdagen.
dokumentlista.impl
» com.hack23.cia:model.external.val.
riksdagsvalkrets.impl
» com.hack23.cia:model.external.riksdagen.
utskottsforslag.impl
» com.hack23.cia:service.impl
» com.hack23.cia:service.component.agent.impl
» com.hack23.cia:model.external.worldbank.topic.impl
» com.hack23.cia:model.internal.application.user.impl
» com.hack23.cia:jms-broker
» com.hack23.cia:model.external.
worldbank.countries.impl
» com.hack23.cia:model.external.riksdagen.
dokumentstatus.impl
» com.hack23.cia:service.external.val
» com.hack23.cia:model.external.val.
kommunvalkrets.impl
» com.hack23.cia:service.api
» com.hack23.cia:model.common.impl
» com.hack23.cia:model.external.riksdagen.
voteringlista.impl
» com.hack23.cia:service.external.riksdagen
» com.hack23.cia:model.external.riksdagen.
documentcontent.impl
» com.hack23.cia:model.external.
riksdagen.personlista.impl
» com.hack23.cia:model.external.val.partier.impl
» com.hack23.cia:model.external.
riksdagen.votering.impl
» com.hack23.cia:testfoundation
» com.hannesdorfmann.sqlbrite:dao
» com.holidaycheck:marathon-maven-plugin
» com.ibasco.agql:agql-coc-webapi
» com.intel.jndn.mock:jndn-mock
» com.jakewharton.espresso:espresso-runner
» com.jakewharton.espresso:espresso
» com.jakewharton.sdkmanager:gradle-plugin
» com.jdroidframework:jdroid-java-firebase
» com.jetdrone:yoke-extras
» com.joyent.http-signature:google-http-client-signature
» com.jtransc:jtransc-rt-core-kotlin
» com.jtransc:jtransc-gen-haxe
» com.jtransc:jtransc-utils
» com.jtransc:jtransc-rt
» com.khubla.antlr:antlr4test-maven-plugin
» com.kotcrab.vis:vis-ui
» com.lapis.jsfexporter:export-type-xml
» com.leacox.dagger:dagger-servlet
» io.fabric8:process-spring-boot-starter-activemq
» com.lihaoyi:utest-runner_2.11
» com.lihaoyi:utest_sjs0.5_2.11
» com.lihaoyi:upickle_sjs0.5_2.11
» com.linecorp.armeria:armeria-retrofit2
» com.liulishuo.filedownloader:library
» com.madgag:bfg-parent_2.11
» com.merapar:spring-boot-starter-graphql
» io.github.clickscript:clickscript_2.10
» io.github.gpein:jcache-jee7
» com.michaelpardo:ollie-compiler
» io.github.morgaroth:navigator-import-core_2.11
» io.github.msdk:msdk-spectra-isotopepattern
» io.github.seleniumquery:seleniumquery
» com.mobilesolutionworks:works-bolts
» com.netflix.denominator:denominator-cli
» com.netflix.evcache:evcache-client
» com.netflix.evcache:evcache-core
» com.netflix.hystrix:hystrix-codahale-metrics-publisher
» com.netflix.iep:iep-governator
» com.netflix.iep:iep-module-karyon
» com.netflix.iep:iep-module-eureka
» com.netflix.iep:iep-module-rxnetty
» com.netflix.iep:iep-guice
» com.netflix.iep:iep-module-jmxport
» com.netflix.iep-shadow:iepshadow-iep-rxhttp
» com.netflix.nebula:nebula-kotlin-plugin
» com.netflix.rxjava:rxjava-quasar
» com.netflix.spectator:spectator-reg-metrics2
» com.nicta:rng_2.11
» com.nitorcreations:willow-utils
» com.nrinaudo:tabulate-cats_2.11
» com.nrinaudo:tabulate-cats_2.10
» com.octo.android.robospice:robospice-cache
» com.octo.android.robospice:robospice-
google-http-client
» com.octo.android.robospice:robospice-okhttp
» com.optimaize.soapworks.server.
implgrizzly:soapworks-server-implgrizzly
» com.oracle.truffle:truffle-tck
» com.oracle.truffle:truffle-api
» com.orange.redis-protocol:netty4
» com.outr.query:outrquery-search_2.11
» com.palomamobile.android.sdk:core
» com.palominolabs.http:jetty-http-server-wrapper
» com.palominolabs.metrics:metrics-new-relic
» com.paypal:cascade-examples_2.11
» com.paypal:parent_2.11
» com.pholser:junit-quickcheck-generators
» com.r0adkll:postoffice
» com.sandinh:couchbase-akka-extension_2.11
» com.sandinh:play-hikaricp_2.11
» com.sandinh:couchbase-scala_2.10
» com.scalarx:scalarx_sjs0.5_2.11
» com.scalatags:scalatags_sjs0.5_2.11
» com.segment.analytics.android:
analytics-integration-amplitude
» com.semanticcms:semanticcms-view-tree
» com.semanticcms:semanticcms-core-view-content
» com.semanticcms:semanticcms-autogit-servlet
» com.semanticcms:semanticcms-file-taglib
» com.semanticcms:semanticcms-view-all
» com.semanticcms:semanticcms-core-sitemap
» com.semanticcms:semanticcms-section-taglib
» com.semanticcms:semanticcms-section-servlet
» com.semanticcms:semanticcms-dia-model
» com.semanticcms:semanticcms-news-servlet
» com.semanticcms:semanticcms-news-model
» com.semanticcms:semanticcms-view-what-links-here
» com.semanticcms:semanticcms-autogit-taglib
» com.semanticcms:semanticcms-file-servlet
» com.semanticcms:semanticcms-
theme-documentation
» com.semanticcms:semanticcms-core-taglib
» com.semanticcms:semanticcms-news-taglib
» com.semanticcms:semanticcms-file-view
» io.zipkin.java:zipkin-autoconfigure-
storage-elasticsearch
» com.sksamuel.scrimage:scrimage_2.11
» com.sksamuel.scrimage:scrimage-canvas_2.11
» com.smb-tec.neo4j:neo4j-community
» com.smb-tec.xo:xo-tinkerpop-blueprints
» com.smoketurner:dropwizard-swagger
» com.smoketurner.dropwizard:consul-core
» com.smoketurner.dropwizard:zipkin-example
» com.smoketurner.dropwizard:zipkin-core
» com.smoketurner.dropwizard:dropwizard-riak
» com.softwaremill:reactive-kafka_2.10
» com.softwaremill.events:core_2.11
» com.soywiz:korio-ext-amazon-common
» com.soywiz:korge-ext-particle
» jp.vmi:selenese-runner-java
» com.squareup.burst:burst-android
» net.danlew:android.joda
» com.tascape.qa:thx-webservice
» com.thoughtworks.tools:dependency-check
» com.threerings:tripleplay-java-swt
» com.timcharper:cassandra-talks-scala_2.11
» com.tinkerpop.blueprints:blueprints-sparksee-graph
» net.osgiliath.framework:net.osgiliath.helpers.camel.
cdi.configadmin
» com.uwetrottmann:trakt-java
» com.vaadin:vaadin-client
» com.vilt-group.minium:minium-core
» com.vilt-group.minium:minium-script
» com.vmware.photon.controller:photon-model-security
» com.vmware.xenon:xenon-slf4j
» com.wandoulabs.avro:astore_2.11
» com.wandrell:java-patterns
» com.weicoder:dao
» nl.bstoi.jersey.test-framework:jersey-spring-
exposed-test-framework-core
» no.difi.vefa:validator-core
» com.semanticcms:semanticcms-autogit-view
» org.codehaus.sonar-plugins.
java:sonar-findbugs-plugin
» org.ddogleg:ddogleg
» org.hypoport:mockito-mockinjector
» org.jbpm:jbpm-console-ng-generic-forms-api
» org.passay:passay
» org.requs:requs-demo
» org.wicketstuff:wicketstuff-restannotations-examples
» org.wicketstuff:wicketstuff-selectize
» org.woodylab.boot:spring-boot-starter-pebble
» pl.wkr:fluent-exception-rule
» ru.systemate:morpholog-client
» su.litvak.chromecast:api-v2
» am.ik.home:uaa-client
» am.ik.home:uaa-integration-test
» at.chrl:chrl-spring
» at.chrl:chrl-orm
» biz.gabrys.maven.plugins:css-splitter-maven-plugin

» ch.cern.dirq:dirq
» ch.rasc:embeddedtc
» ch.rasc:constgen
» ch.sbb.releasetrain:webui
» ch.sbb.releasetrain:director
» ch.sbb.releasetrain:utils
» ch.sbb.releasetrain:action
» ch.sbb.releasetrain:mavenmojos
» ch.sbb.releasetrain:config
» ch.softappeal.yass:yass
» cn.dreampie:jfinal-mailer
» cn.org.zeronote:commondao
» co.cask.cdap:cdap-notifications-api
» co.cask.cdap:cdap-explore-client
» im.chic.crypto:crypto-utils
» im.chic.weixin:weixin-utils
» info.android15.satellite:satellite
» de.ahus1.keycloak.dropwizard:keycloak-dropwizard
» de.codecentric:spring-boot-admin-starter-client
» de.codecentric:spring-boot-starter-admin-client
» de.codecentric:spring-boot-admin-sample
» de.codecentric:spring-boot-starter-batch-web
» de.javakaffee:kryo-serializers
» de.learnlib:learnlib-counterexamples
» de.learnlib:learnlib-reuse
» de.learnlib:learnlib-drivers-basic
» de.learnlib:learnlib-basic-eqtests
» de.learnlib:learnlib-core
» de.learnlib:learnlib-algorithm-features
» de.learnlib:learnlib-parallelism
» de.learnlib:learnlib-nlstar
» de.learnlib:learnlib-dhc
» de.learnlib:learnlib-ttt
» de.learnlib:learnlib-examples
» de.learnlib:learnlib-lstar-generic
» de.learnlib:learnlib-mapper
» de.learnlib:learnlib-kearns-vazirani
» de.learnlib:learnlib-acex
» de.learnlib:learnlib-cache
» de.learnlib:learnlib-lstar-baseline
» de.learnlib:learnlib-discrimination-tree
» de.learnlib.testsupport:learnlib-learning-examples
» de.otto.edison:togglz
» de.saly:javamail-mock2-fullmock
» de.saly:javamail-mock2-halfmock
» de.svenkubiak:embedded-mongodb
» de.svenkubiak:jpushover
» de.svenkubiak:mangooio-mongodb-extension
» de.taimos:spring-dao-hibernate
» de.taimos:dvalin-dynamodb
» de.taimos:daemon-framework-spring
» de.taimos:spring-cxf-daemon
» de.taimos:spring-dao-mongo
» edu.stanford.protege:org.protege.editor.core.application
» edu.stanford.protege:protege-owlapi-extensions
» eu.michael-simons:java-akismet
» fr.iscpif.gridscale:gridscaleslurm_2.11
» fr.iscpif.gridscale:gridscalessh_2.11
» fr.iscpif.gridscale:gridscaleoar_2.11
» fr.iscpif.gridscale:gridscalepbs_2.11
» fr.iscpif.gridscale:gridscalesge_2.11
» fr.iscpif.gridscale:glitesrmexample_2.11
» fr.iscpif.gridscale:gridscalehttp_2.11
» fr.iscpif.gridscale:gliteexample_2.11
» fr.zebasto:spring-postinitialize
» gr.grnet:pithosj
» info.ganglia.gmetric4j:gmetric4j
» info.johtani:elasticsearch-extended-analyze
» io.advantageous.qbit:qbit-servlet
» io.advantageous.qbit:qbit-eventbus-replicator
» io.advantageous.qbit:qbit-boon
» io.bigio:bigio-benchmark
» io.buji:buji-pac4j
» io.sniffy:sniffy
» io.dropwizard.modules:dropwizard-flyway
» io.dropwizard.modules:dropwizard-protobuf
» io.fabric8:gitective-core
» io.fabric8:fabric-webapp-agent
» io.fabric8:process-spring-boot-registry
» io.fabric8:watcher-dozer
» io.fabric8:fabric-jolokia
» io.fabric8:swagger-annotator
» io.fabric8:fabric-camel
» io.fabric8:console
» io.fabric8:fabric-git-hawtio
» io.fabric8:kubernetes-jolokia
» io.fabric8:gateway-api
» io.fabric8:fabric-dynamic-jaxb
» io.fabric8.examples:fabric-camel-cxf
» io.fabric8.examples:fabric-loanbroker-rateservice
» io.fabric8.forge:kubernetes
» io.fabric8.insight:insight-kibana3
» io.fabric8.insight:insight-eshead
» io.fabric8.jube:console
» io.fabric8.jube:core
» io.fabric8.jube:process-manager
» io.fabric8.jube.images.fabric8:quickstart-karaf-camelcbr
» io.fabric8.jube.images.fabric8:quickstart-karaf-camellog
» io.fabric8.jube.images.fabric8:quickstart-karaf-camelwiki
» io.fabric8.runtime:fabric8-runtime-container-tomcat-registration
» io.gatling:gatling-http
» io.gatling:gatling-core
» tv.cntt:netcaty_2.10
» io.hawt:hawtio-local-jvm-mbean
» io.hawt:hawtio-web
» io.konik:itext-carriage
» io.mangoo:mangooio-test-utilities
» io.mangoo:mangooio-core
» io.mangoo:mangooio-integration-test
» io.mangoo:mangooio-benchmark
» io.maxthomas:concrete-dictum
» io.springfox:springfox-staticdocs
» io.openscore.content:score-ssh
» io.openscore.content:score-mail
» io.openscore.lang:score-lang-runtime
» io.reactivex:rxkotlin
» io.segment.android:analytics
» am.ik.home:uaa-server
» io.taig.android:soap_2.11
» io.vertigo:vertigo-tempo-impl
» io.zipkin.java:zipkin-storage-cassandra
» io.zipkin.java:zipkin-junit
» io.zipkin.java:zipkin-storage-elasticsearch
» io.zipkin.java:zipkin-autoconfigure-collector-scribe
» io.zipkin.java:zipkin-server
» io.zipkin.java:zipkin-autoconfigure-storage-cassandra3
» io.zipkin.java:zipkin-autoconfigure-metrics-prometheus
» io.zipkin.java:zipkin-autoconfigure-storage-cassandra
» io.zipkin.java:zipkin-autoconfigure-ui
» io.zipkin.java:zipkin-autoconfigure-collector-kafka
» it.unibo.alchemist:alchemist-engine
» it.unibo.alchemist:alchemist-incarnation-protelis
» it.unimi.dsi:webgraph
» javax.cache:spring-annotations-test-harness
» javax.cache:guice-annotations-test-harness
» javax.cache:specific-implementation-tester
» javax.cache:spring-annotations-tester
» javax.cache:guice-annotations-tester
» javax.cache:cdi-annotations-tester
» me.lessis:zoey-core_2.11
» me.lessis:zoey-testing_2.11
» me.mattak:moment
» net.aequologica.neo:geppaequo-cdi
» net.aequologica.neo:geppaequo-web
» net.aequologica.neo:shakuntala-test
» net.aequologica.neo:buildhub-core
» net.aequologica.neo:quintessence-core
» net.aequologica.neo:parole-core
» net.aequologica.neo:buildhub-persist
» net.aequologica.neo:buildhub-web
» net.aequologica.neo:geppaequo-core
» net.aequologica.neo:parole-web
» net.aequologica.neo:dagr-model
» net.aequologica.neo:dagr-web
» net.anotheria:moskito-webui-jersey
» net.bytebuddy:byte-buddy-dep
» net.code-story:http
» net.imagej:minimaven
» net.kemitix:kxssh
» net.kencochrane.raven:raven-appengine
» net.kencochrane.raven:raven-logback
» net.liftmodules:omniauth_2.6_2.11
» net.mostlyoriginal.artemis-odb:contrib-eventbus
» net.mostlyoriginal.artemis-odb:contrib-core
» net.osgiliath.features:net.osgiliath.feature.karaf-enterprise
» net.osgiliath.framework:net.osgiliath.features.karaf-features-validation
» net.osgiliath.framework:net.osgiliath.features.karaf-features-security
» net.osgiliath.hello:net.osgiliath.hello.features
» net.postgis:postgis-jdbc
» net.postgis:postgis-jdbc-java2d
» net.sf.derquinsej:derquinsej-hib3
» net.sf.derquinsej:derquinsej-test-support
» net.sf.derquinsej:derquinsej-core
» net.sf.sprockets:sprockets-android
» net.sf.uadetector:uadetector-core
» net.wessendorf.websocket:simple-client
» net.yslibrary.rxrealm:rxrealm
» nl.komponents.kovenant:kovenant-disruptor
» no.difi.sdp:sikker-digital-post-java-klient
» nz.ac.auckland.composite:composite-ebean
» nz.ac.auckland.composite:composite-jetty
» nz.co.aetheric.maven:composite-jetty
» nz.net.osnz.composite:composite-spring
» nz.net.osnz.composite:composite-spring-jdbc
» nz.net.osnz.composite:composite-spring-aspects
» nz.net.osnz.lmz:lmz-runner
» nz.net.osnz.lmz:lmz-syllabus
» nz.net.osnz.lmz:lmz-stencil
» org.agrona:agrona-agent
» org.akhikhl.gretty:gretty-runner-spring-boot-jetty
» org.akhikhl.gretty:gretty-helper-commons
» org.akhikhl.gretty:gretty-plugin-commons
» org.akhikhl.gretty:gretty-plugin
» org.akhikhl.rooty:rooty
» org.akhikhl.unpuzzle:unpuzzle-eclipse2maven
» org.ansj:ansj_seg
» org.arquillian.cube:arquillian-cube-containerless

» org.arquillian.cube:arquillian-cube-spi
» org.atmosphere:vibe-platform-bridge-vertx2
» org.atmosphere:vibe-platform-bridge-grizzly2
» org.atmosphere:vibe-platform-bridge-play2
» org.atmosphere:vibe-platform-bridge-atmosphere2
» org.atmosphere:vibe-platform-action
» org.atmosphere:vibe-platform-http
» org.atmosphere:vibe-platform-ws
» org.atmosphere:vibe-platform-bridge-servlet3
» org.atmosphere:vibe-platform-bridge-netty4
» org.blocks4j.commons:blocks4j-commons-metrics3
» org.boofcv:recognition
» org.boofcv:evaluation
» org.boofcv:visualize
» org.boofcv:processing
» org.boofcv:applet
» org.boofcv:geo
» org.boofcv:io
» org.boofcv:feature
» org.boofcv:openkinect
» org.boofcv:sfm
» org.boofcv:xuggler
» org.boofcv:android
» org.boofcv:ip
» org.boofcv:calibration
» org.carewebframework:org.carewebframework.shell
» org.codelibs:elasticsearch-solr-api
» org.cogroo:cogroo-nlp
» org.cometd.java:cometd-websocket-jetty
» org.cometd.tutorials:cometd-tutorials-skeleton
» org.cubeengine:pericopist-core
» org.danilopianini:javalib-java7
» org.dashbuilder:dashbuilder-services-api
» org.dashbuilder:dashbuilder-renderer-default
» org.dashbuilder:dashbuilder-validations
» org.dashbuilder:dashbuilder-common-client
» org.dashbuilder:dashbuilder-widgets
» org.dashbuilder:dashbuilder-dataset-api
» org.dashbuilder:dashbuilder-displayer-editor
» org.dashbuilder:dashbuilder-server-all
» org.dashbuilder:dashbuilder-renderer-chartjs
» org.dashbuilder:dashbuilder-renderer-google
» org.dashbuilder:dashbuilder-dataset-client
» org.dashbuilder:dashbuilder-webapp
» org.dashbuilder:dashbuilder-client-all
» org.dashbuilder:dashbuilder-displayer-api
» org.dashbuilder:dashbuilder-dataset-editor
» org.dashbuilder:dashbuilder-displayer-screen
» org.dashbuilder:dashbuilder-dataset-shared
» org.dashbuilder:dashbuilder-displayer-client
» org.dbtools:dbtools-gen
» org.deeplearning4j:dl4j-spark-ml
» org.deeplearning4j:dl4j-caffe
» org.deeplearning4j:deeplearning4j-cli-api
» org.dm.gradle:gradle-bundle-plugin
» org.drools:drools-decisiontables
» org.drools:drools-jsr94
» org.drools:drools-wb-test-scenario-editor-api
» org.drools:default-kiesession
» org.drools:cdi-example-with-inclusion
» org.drools:drools-wb-guided-scorecard-editor-api
» org.drools:drools-benchmark
» org.drools:drools-wb-scorecard-xls-editor-api
» org.drools:droolsjbpm-integration-distribution
» org.drools:cdi-example
» org.drools:drools-pmml
» org.drools:drools-wb-enum-editor-client
» org.drools:drools-wb-guided-dtable-editor-api
» org.drools:kiefilesystem-example
» org.drools:drools-wb-globals-editor-backend
» org.drools:drools-wb-guided-template-editor-api
» org.drools:drools-wb-enum-editor-api
» org.drools:drools-wb-guided-dtable-editor-client
» org.drools:drools-wb-guided-template-editor-client
» org.drools:drools-wb-workitems-editor-client
» org.drools:drools-wb-dsl-text-editor-backend
» org.drools:drools-wb-dsl-text-editor-client
» org.drools:drools-wb-test-scenario-editor-backend
» org.drools:drools-templates
» org.drools:drools-wb-dtable-xls-editor-api
» org.drools:drools-wb-factmodel-editor-api
» org.drools:drools-wb-guided-dtree-editor-api
» org.drools:drools-workbench-models-datamodel-api
» org.drools:kie-module-from-multiple-files
» org.drools:kiebase-inclusion
» org.drools:default-kiesession-from-file
» org.drools:droolsjbpm-tools-distribution
» org.drools:drools-wb-scorecard-xls-editor-backend
» org.drools:drools-workbench-models-guided-template
» org.drools:kiemodulemodel-example
» org.drools:drools-wb-guided-dtable-editor-backend
» org.drools:drools-workbench-models-guided-dtree
» org.drools:drools-wb-test-scenario-editor-client
» org.drools:named-kiesession-from-file
» org.drools:drools-wb-drl-text-editor-backend
» org.drools:drools-wb-drl-text-editor-client
» org.drools:drools-verifier
» org.drools:drools-wb-workitems-editor-backend
» org.drools:drools-wb-factmodel-editor-backend
» org.drools:drools-beliefs
» org.drools:drools-wb-drl-text-editor-api
» org.drools:drools-wb-guided-scorecard-editor-client
» org.drools:drools-wb-workitems-editor-api
» org.drools:drools-wb-dtable-xls-editor-backend
» org.drools:kiecontainer-from-kierepo
» org.drools:droolsjbpm-integration-examples
» org.drools:drools-workbench-models-guided-dtable
» org.drools:jbpm-simulation
» org.drools:drools-examples
» org.drools:drools-android
» org.drools:drools-wb-dsl-text-editor-api
» org.drools:drools-wb-scorecard-xls-editor-client
» org.drools:drools-reteoo
» org.drools:drools-scorecards
» org.drools:drools-wb-jcr2vfs-import
» org.drools:named-kiesession
» org.drools:drools-compiler
» org.drools:drools-wb-globals-editor-client
» org.drools:drools-persistence-jpa
» org.drools:drools-wb-globals-editor-api
» org.drools:drools-wb-guided-rule-editor-backend
» org.drools:drools-wb-dtable-xls-editor-client
» org.drools:drools-wb-guided-rule-editor-api
» org.drools:drools-wb-guided-rule-editor-client
» org.drools:drools-core
» org.drools:drools-wb-enum-editor-backend
» org.drools:drools-workbench-models-test-scenarios
» org.drools:drools-wb-guided-dtree-editor-backend
» org.drools:drools-wb-guided-scorecard-editor-backend
» org.drools:drools-distribution
» org.drools:drools-wb-guided-dtree-editor-client
» org.drools:drools-workbench-models-guided-scorecard
» org.drools:drools-jboss-integration
» org.drools:drools-wb-guided-template-editor-backend
» org.ehcache.modules:ehcache-management
» org.ehcache.modules:ehcache-impl
» org.elasticsearch:elasticsearch-analysis-smartcn
» org.elasticsearch:elasticsearch-analysis-icu
» org.elasticsearch:elasticsearch-analysis-stempel
» org.elasticsearch:elasticsearch-analysis-kuromoji
» org.elasticsearch:elasticsearch-analysis-phonetic
» org.elasticsearch:elasticsearch-cloud-gce
» org.everit.osgi:org.everit.osgi.jdbc.commons.dbcp
» org.fxmisc.flowless:flowless
» org.fxmisc.richtext:richtextfx
» org.gaul:s3proxy
» org.georegression:georegression
» org.georegression:experimental
» org.got5:tapestry5-jquery
» org.greencheek.related:related-indexing
» org.greencheek.related:related-domain
» org.greencheek.related:related-web-indexing
» org.greencheek.related:related-web-searching
» org.greencheek.related:related-searching
» org.greencheek.spray:spray-cache-spymemcached
» org.guvnor:guvnor-asset-mgmt-backend
» org.guvnor:guvnor-asset-mgmt-api
» org.guvnor:guvnor-rest-client
» org.guvnor:guvnor-services-api
» org.guvnor:guvnor-workingset-client
» org.guvnor:guvnor-message-console-backend
» org.guvnor:guvnor-structure-client
» org.guvnor:guvnor-project-api
» org.guvnor:guvnor-m2repo-editor-client
» org.guvnor:guvnor-project-backend
» org.guvnor:guvnor-structure-backend
» org.guvnor:guvnor-m2repo-editor-api
» org.guvnor:guvnor-m2repo-editor-backend
» org.guvnor:guvnor-message-console-api
» org.guvnor:guvnor-services-backend
» org.guvnor:guvnor-rest-backend
» org.guvnor:guvnor-project-builder
» org.guvnor:guvnor-organizationalunit-manager
» org.guvnor:guvnor-asset-mgmt-client
» org.guvnor:guvnor-structure-api
» org.guvnor:guvnor-workingset-api
» org.guvnor:guvnor-message-console-client
» org.guvnor:guvnor-project-client
» org.hawkular.accounts:hawkular-accounts-api
» org.hawkular.accounts:hawkular-accounts-sample
» org.hawkular.inventory:hawkular-inventory-impl-tinkerpop-spi
» org.hawkular.inventory:hawkular-inventory-impl-tinkerpop
» org.hawkular.inventory:hawkular-inventory-impl-tinkerpop-tinkergraph-provider
» org.hawkular.inventory:hawkular-inventory-impl-tinkerpop-sql-provider
» org.hibernate:hibernate-validator
» org.hisrc.w3c:atom-v_1_0
» org.hisrc.w3c:ws-addr-v_1_0-core
» org.hisrc.w3c:xhtml-v_1_0-strict
» org.hisrc.w3c:xlink-v_1_0
» org.hisrc.w3c:xmlschema-v_1_0
» org.hyperscala:hyperscala-service_2.11
» org.hyperscala:hyperscala-connect_2.11
» org.hyperscala:hyperscala-site_2.11
» org.igniterealtime.smack:smack-debug-slf4j
» org.incode.module.note:incode-module-note-dom
» org.infinispan:infinispan-as-client-modules
» org.infinispan:infinispan-lucene-v4
» org.isisaddons.module.audit:isis-module-audit-dom
» org.isisaddons.module.command:isis-module-command-dom
» org.isisaddons.module.devutils:isis-module-devutils-dom

» org.isisaddons.module.docx:isis-module-docx-dom
» org.isisaddons.module.publishmq:
isis-module-publishmq-dom-servicespi
» org.isisaddons.module.settings:
isis-module-settings-dom
» org.isisaddons.wicket.fullcalendar2:
isis-wicket-fullcalendar2-cpt
» org.isisaddons.wicket.summernote:
isis-wicket-summernote-cpt
» org.isisaddons.wicket.wickedcharts:
isis-wicket-wickedcharts-cpt
» org.javamoney:moneta
» org.javamoney:moneta-bp
» org.javers:javers-persistence-sql
» org.jboss.aerogear.test:spacelift-jboss-manager
» org.jboss.aerogear.test.arquillian:arquillian-non-deploying-container-checks-api
» org.jboss.aerogear.test.arquillian:
arquillian-non-deploying-container
» org.jboss.aerogear.test.arquillian:arquillian-non-deploying-container-checks-impl
» org.jboss.cdi.tck:cdi-tck-ext-lib
» org.jboss.cdi.tck:cdi-tck-api
» org.jboss.gwt.elemento:elemento-core
» org.jboss.remotingjmx:remoting-jmx
» org.jboss.resteasy:resteasy-jettison-provider
» org.jboss.threads:jboss-threads
» org.jboss.weld.examples:weld-osgi-paint-api
» org.jboss.weld.module:weld-jta
» org.jboss.windup.decompiler:decompiler-procyon
» org.jbpm:jbpm-console-ng-dashboard-backend
» org.jbpm:jbpm-console-ng-
human-tasks-forms-backend
» org.jbpm:jbpm-human-task-core
» org.jbpm:jbpm-console-ng-
process-runtime-admin-client
» org.jbpm:jbpm-kie-services
» org.jbpm:jbpm-console-ng-executor-service-client
» org.jbpm:jbpm-console-ng-generic-api
» org.jbpm:jbpm-form-modeler-api
» org.jbpm:jbpm-persistence-jpa
» org.jbpm:jbpm-console-ng-showcase
» org.jbpm:jbpm-shared-services
» org.jbpm:jbpm-form-modeler-document
» org.jbpm:jbpm-console-ng-dashboard-api
» org.jbpm:jbpm-services-ejb-api
» org.jbpm:jbpm-document
» org.jbpm:jbpm-console-ng-human-tasks-forms-client
» org.jbpm:jbpm-console-ng-human-
tasks-admin-backend
» org.jbpm:jbpm-form-modeler-bpmn-form-builder
» org.jbpm:jbpm-form-modeler-editor-api
» org.jbpm:jbpm-services-cdi
» org.jbpm:jbpm-console-ng-business-domain-client
» org.jbpm:jbpm-designer-backend
» org.jbpm:jbpm-form-modeler-editor-backend
» org.jbpm:jbpm-console-ng-business-domain-backend
» org.jbpm:jbpm-audit
» org.jbpm:jbpm-console-ng-
human-tasks-forms-modeler-client
» org.jbpm:jbpm-flow-builder
» org.jbpm:jbpm-console-ng-human-tasks-backend
» org.jbpm:jbpm-runtime-manager
» org.jbpm:jbpm-console-ng-executor-service-backend
» org.jbpm:jbpm-console-ng-human-tasks-admin-api
» org.jbpm:jbpm-services-api
» org.jbpm:jbpm-human-task-audit
» org.jbpm:jbpm-form-modeler-renderer-backend
» org.jbpm:jbpm-console-ng-human-tasks-client
» org.jbpm:jbpm-services-ejb-timer
» org.jbpm:jbpm-console-ng-business-domain-api
» org.jbpm:jbpm-console-ng-process-runtime-api
» org.jbpm:jbpm-executor
» org.jbpm:jbpm-console-ng-bpm-home-client
» org.jbpm:jbpm-form-modeler-data-modeler
» org.jbpm:jbpm-console-ng-human-tasks-admin-client
» org.jbpm:jbpm-executor-cdi
» org.jbpm:jbpm-console-ng-process-runtime-backend
» org.jbpm:jbpm-console-ng-human-tasks-api
» org.jbpm:jbpm-flow
» org.jbpm:jbpm-executor-ejb
» org.jbpm:jbpm-console-ng-
process-runtime-forms-client
» org.jbpm:jbpm-designer-api
» org.jbpm:jbpm-form-modeler-editor-client
» org.jbpm:jbpm-human-task-jpa
» org.jbpm:jbpm-services-ejb-impl
» org.jbpm:jbpm-console-ng-generic-forms-client
» org.jbpm:jbpm-console-ng-
workbench-integration-client
» org.jbpm:jbpm-console-ng-documents-backend
» org.jbpm:jbpm-console-ng-generic-client
» org.jbpm:jbpm-examples
» org.jbpm:jbpm-form-modeler-renderer-client
» org.jbpm:jbpm-human-task-workitems
» org.jbpm:jbpm-console-ng-process-runtime-client
» org.jbpm:jbpm-console-ng-executor-service-api
» org.jbpm:jbpm-designer-client
» org.jbpm:jbpm-bpmn2
» org.jbpm:jbpm-console-ng-documents-api
» org.jbpm:jbpm-form-modeler-renderer-api
» org.jbpm:jbpm-console-ng-documents-client
» org.jbpm:jbpm-console-ng-dashboard-client
» org.jbpm:jbpm-form-modeler-showcase
» org.jnario:org.jnario.lib.maven
» org.jodd:jodd-mail
» org.joeyb.undercarriage:grpc
» org.joinfaces:jsf-jetty-myfaces-
bootsfaces-spring-boot-starter
» org.joinfaces:jsf-jetty-butterfaces-spring-boot-starter
» org.joinfaces:jsf-butterfaces-spring-boot-starter
» org.joinfaces:jsf-undertow-
bootsfaces-spring-boot-starter
» org.joinfaces:jsf-myfaces-
butterfaces-spring-boot-starter
» org.joinfaces:jsf-undertow-
butterfaces-spring-boot-starter
» org.joinfaces:jsf-undertow-myfaces-bootsfaces-
spring-boot-starter
» org.junit.contrib:junit-theories
» org.jvnet.ogc:ows-v_1_1_0
» org.jvnet.ogc:kml-v_2_2_0
» org.jvnet.ogc:sld-v_1_0_0-geoserver
» org.jvnet.ogc:wmc-v_1_0_0
» org.jvnet.ogc:gml-v_3_2_1
» org.jvnet.ogc:wfs-v_2_0
» org.jvnet.ogc:sos-v_2_0
» org.jvnet.ogc:ols-nav-v_1_3
» org.jvnet.ogc:owc-v_0_3_1
» org.jvnet.ogc:ows-v_2_0
» org.jvnet.ogc:wmts-v_1_0
» org.jvnet.ogc:ols-v_1_1_0
» org.jvnet.ogc:wcs-v_2_0
» org.jvnet.ogc:iso19139-v_20070417
» org.jvnet.ogc:sps-v_1_0_0
» org.jvnet.ogc:wps-v_1_0_0
» org.jvnet.ogc:iso19139-v_20060504
» org.jvnet.ogc:sampling-v_2_0
» org.jvnet.ogc:wms-v_1_3_0
» org.jvnet.ogc:wcst-v_1_1
» org.jvnet.ogc:sampling-v_1_0_0
» org.jvnet.ogc:sld-v_1_0_0
» org.jvnet.ogc:wcs-v_1_1
» org.jvnet.ogc:om-v_2_0
» org.jvnet.ogc:sld-v_1_1_0
» org.jvnet.ogc:wfs-v_1_0_0
» org.jvnet.ogc:wmc-v_1_1_0
» org.jvnet.ogc:csw-v_2_0_2
» org.jvnet.ogc:gml-v_3_1_1
» org.jvnet.ogc:gml-v_3_2_0
» org.jvnet.ogc:ows-v_1_0_0
» org.jvnet.ogc:filter-v_1_1_0
» org.jvnet.ogc:om-v_1_0_0
» org.jvnet.ogc:swes-v_2_0
» org.jvnet.ogc:arml-v_2_0
» org.jvnet.ogc:wps-v_2_0
» org.jvnet.ogc:indoorgml-v_1_0
» org.jvnet.ogc:gmlcov-v_1_0
» org.jvnet.ogc:wcs-v_1_0_0
» org.jvnet.ogc:omx-v_1_0_0
» org.jvnet.ogc:citygml-v_1_0
» org.jvnet.ogc:omeo-v_1_0
» org.kaazing:robot.all
» org.kie:kie-drools-wb-webapp
» org.kie:kie-wb-webapp
» org.kie:kie-drools-wb-home-page-community
» org.kie:kie-wb-home-page-community
» org.kie:kie-identity-session-provider
» org.kie:kie-ci-osgi
» org.kie.uberfire:kie-uberfire-social-activities-client
» org.kie.uberfire:kie-uberfire-social-activities-backend
» org.kie.uberfire:kie-uberfire-social-activities-api
» org.kie.workbench.screens:kie-wb-common-
social-home-page-backend
» org.kie.workbench.screens:kie-wb-common-
social-home-page-api
» org.kie.workbench.screens:kie-wb-common-
project-imports-editor-api
» org.kie.workbench.screens:kie-wb-common-
search-screen-api
» org.kie.workbench.screens:kie-wb-common-
contributors-client
» org.kie.workbench.screens:kie-wb-common-
default-editor-client
» org.kie.workbench.screens:kie-wb-common-
project-editor-api
» org.kie.workbench.screens:kie-wb-common-
server-ui-client
» org.kie.workbench.screens:kie-wb-common-
project-editor-client
» org.kie.workbench.screens:kie-wb-common-
default-editor-backend
» org.kie.workbench.screens:kie-wb-common-
project-explorer-client
» org.kie.workbench.screens:kie-wb-common-
search-screen-backend
» org.kie.workbench.screens:kie-wb-common-
data-modeller-api
» org.kie.workbench.screens:kie-wb-common-
project-explorer-api
» org.kie.workbench.screens:kie-wb-common-
home-client
» org.kie.workbench.screens:kie-wb-common-
data-modeller-backend
» org.kie.workbench.screens:kie-wb-common-
project-editor-backend
» org.kie.workbench.screens:kie-wb-common-
java-editor-api
» org.kie.workbench.screens:kie-wb-common-
social-home-page-client

» org.kie.workbench.screens:kie-wb-common-contributors-backend
» org.kie.workbench.screens:kie-wb-common-project-imports-editor-client
» org.kie.workbench.screens:kie-wb-common-project-explorer-backend
» org.kie.workbench.screens:kie-wb-common-search-screen-client
» org.kie.workbench.screens:kie-wb-common-default-editor-api
» org.kie.workbench.screens:kie-wb-common-home-api
» org.kie.workbench.screens:kie-wb-common-s erver-ui-backend
» org.kie.workbench.screens:kie-wb-common-server-ui-api
» org.kie.workbench.screens:kie-wb-common-data-modeller-client
» org.kie.workbench.screens:kie-wb-common-java-editor-client
» org.kie.workbench.services:kie-wb-common-services-backend
» org.kie.workbench.services:kie-wb-common-refactoring-backend
» org.kie.workbench.services:kie-wb-common-refactoring-api
» org.kie.workbench.services:kie-wb-common-data-modeller-core
» org.kie.workbench.services:kie-wb-common-datamodel-backend
» org.kie.workbench.services:kie-wb-common-services-api
» org.kie.workbench.services:kie-wb-common-datamodel-api
» org.kie.workbench.widgets:kie-wb-metadata-widget
» org.kie.workbench.widgets:kie-wb-decorated-grid-widget
» org.kie.workbench.widgets:kie-wb-common-ui
» org.kie.workbench.widgets:kie-wb-config-resource-widget
» org.kohsuke.args4j:args4j-maven-plugin
» org.languagetool:language-uk
» org.lastaflute:lastaflute
» org.lumongo:lumongo-storage
» org.mapsforge:mapsforge-map-android
» org.mobicents.diameter:jdiameter-ha-api
» org.monifu:monifu-core-js_2.11
» org.monifu:monifu-rx_2.11
» org.nmdp.ngs:ngs-align
» org.nmdp.ngs:ngs-hml
» org.nmdp.ngs:ngs-reads
» org.ocelotds:ocelot-glassfish
» org.ocelotds:ocelot-wildfly
» org.openehealth.ipf.platform-camel:ipf-platform-camel-ihe-fhir
» org.openfuxml:ofx-xml
» org.openscience.cdk:cdk-inchi
» org.openscoring:openscoring-service
» org.openscoring:openscoring-common
» org.openscoring:openscoring-common-gwt
» org.openurp.code:openurp-code-api
» org.openurp.platform:openurp-platform-ws
» org.openurp.platform.api:openurp-platform-api-web
» org.openurp.platform.kernel:openurp-platform-kernel-core
» org.openurp.platform.kernel:openurp-platform-kernel-ws
» org.openurp.platform.kernel:openurp-platform-kernel-webapp
» org.openurp.platform.security:openurp-platform-security-webapp
» org.openurp.platform.security:openurp-platform-security-core
» org.openurp.platform.security:openurp-platform-security-app
» org.optaplanner:optaplanner-core
» org.optaplanner:optaplanner-wb-solver-editor-api
» org.parceler:parceler
» org.rapidpm.microservice:rapidpm-microservice-modules-persistence-local-hashmap
» org.requs:requs-exec
» org.requs:requs-core
» org.rhq.metrics:rest-servlet
» org.richfaces:richfaces-core
» org.rythmengine:spring-rythm
» org.scala-lang:scala-compiler
» org.scalastuff:json-parser_2.11
» org.scalatest:scalatest-core_sjs0.6_2.10
» org.scalatest:scalatest-core_2.10
» org.scalikejdbc:scalikejdbc-interpolation-core_2.11
» org.scalikejdbc:scalikejdbc-async_2.11
» org.seasar.doma.boot:doma-spring-boot-sample-simple
» org.simpleflatmapper:sfm
» org.skinny-framework:skinny-orm_2.10
» org.skinny-framework:skinny-orm_2.11
» org.skinny-framework:skinny-factory-girl_2.11
» org.skinny-framework:skinny-test_2.11
» org.skinny-framework:skinny-logback
» org.spf4j:spf4j-jmh
» org.springframework.boot:spring-boot-starter-actuator
» org.springframework.boot:spring-boot-gradle-plugin
» org.springframework.boot:spring-boot-starter-velocity
» org.tomitribe:beryllium
» org.uberfire:uberfire-runtime-plugins-api
» org.uberfire:uberfire-widgets-service-api
» org.uberfire:uberfire-nio2-api
» org.uberfire:uberfire-runtime-plugins-backend
» org.uberfire:uberfire-widgets-properties-editor-api
» org.uberfire:uberfire-widgets-core-client
» org.uberfire:uberfire-backend-cdi
» org.uberfire:uberfire-security-api
» org.uberfire:uberfire-runtime-plugins-client
» org.uberfire:uberfire-nio2-model
» org.uberfire:uberfire-workbench-client-views-patternfly
» org.uberfire:uberfire-apps-client
» org.uberfire:uberfire-wires-bpmn-api
» org.uberfire:uberfire-widget-markdown
» org.uberfire:uberfire-wires-bayesian-parser-backend
» org.uberfire:uberfire-client-api
» org.uberfire:uberfire-widgets-properties-editor-client
» org.uberfire:uberfire-widgets-service-backend
» org.uberfire:uberfire-api
» org.uberfire:uberfire-apps-backend
» org.uberfire:uberfire-security-client
» org.uberfire:uberfire-commons-editor-api
» org.uberfire:uberfire-widgets-roperties-editor-backend
» org.uberfire:uberfire-wires-bpmn-client
» org.uberfire:uberfire-backend-server
» org.uberfire:uberfire-commons-editor-backend
» org.uberfire:uberfire-testing-utils
» org.uberfire:uberfire-wires-bayesian-parser-api
» org.uberfire:uberfire-backend-api
» org.uberfire:uberfire-distro
» org.uberfire:uberfire-commons-editor-client
» org.uberfire:uberfire-apps-api
» org.uberfire:uberfire-widgets-commons
» org.walkmod:walkmod-java-formatter-plugin
» org.wicketstuff:wicketstuff-select2-examples
» org.wicketstuff:wicketstuff-stateless-examples
» org.wicketstuff:wicketstuff-datastore-redis
» org.wicketstuff:wicket-mount
» org.wicketstuff:wicketstuff-jwicket-tooltip-wtooltips
» org.wicketstuff:wicketstuff-dropdown-menu
» org.wicketstuff:wicketstuff-context-examples
» org.wicketstuff:wicket-facebook
» org.wicketstuff:tinymce4-examples
» org.wicketstuff:wicket-mount-core
» org.wicketstuff:wicketstuff-progressbar
» org.wicketstuff:wicketstuff-urlfragment-examples
» org.wicketstuff:wicketstuff-security-wicomsec
» org.wicketstuff:wicketstuff-jwicket-examples
» org.wicketstuff:wicketstuff-lazymodel
» org.wicketstuff:wicketstuff-tinymce
» org.wicketstuff:javaee-inject-example-war
» org.wicketstuff:wicketstuff-input-events
» org.wicketstuff:wicketstuff-logback
» org.wicketstuff:wicketstuff-twitter
» org.wicketstuff:wicketstuff-logback-examples
» org.wicketstuff:wicketstuff-objectautocomplete
» org.wicketstuff:wicketstuff-openlayers-proxy
» org.wicketstuff:wicketstuff-servlet3-auth
» org.wicketstuff:lightbox2
» org.wicketstuff:wicketstuff-bundle
» org.wicketstuff:wicketstuff-push-cometd
» org.wicketstuff:wicketstuff-stateless
» org.wicketstuff:wicketstuff-htmlcompressor
» org.wicketstuff:wicketstuff-native-websocket-javax
» org.wicketstuff:wicketstuff-jwicket-ui-effects
» org.wicketstuff:wicketstuff-push-core
» org.wicketstuff:flot-examples
» org.wicketstuff:wicketstuff-inmethod-grid-examples
» org.wicketstuff:wicketstuff-push-timer
» org.wicketstuff:modalx-examples
» org.wicketstuff:wicketstuff-html5
» org.wicketstuff:wicketstuff-jwicket-tooltip-walterzorn
» org.wicketstuff:wicketstuff-restannotations
» org.wicketstuff:wicketstuff-annotation
» org.wicketstuff:wicketstuff-jwicket-core
» org.wicketstuff:wicketstuff-jwicket-ui-dragdrop
» org.wicketstuff:javaee-inject-example-ejb
» org.wicketstuff:wicketstuff-mbeanview
» org.wicketstuff:wicketstuff-whiteboard-examples
» org.wicketstuff:wicketstuff-jwicket-ui-resize
» org.wicketstuff:wicketstuff-serializer-kryo
» org.wicketstuff:wicketstuff-jwicket-ui-tooltip
» org.wicketstuff:wicketstuff-annotationeventdispatcher
» org.wicketstuff:wicketstuff-urlfragment
» org.wicketstuff:modalx
» org.wicketstuff:javaee-inject-example-ear
» org.wicketstuff:wicket-osgi-test-web
» org.wicketstuff:wicket-osgi-test-service
» org.wicketstuff:wicket-shiro-example-base
» org.wicketstuff:wicketstuff-progressbar-spring
» org.wicketstuff:wicketstuff-openlayers
» org.wicketstuff:wicketstuff-plugin
» org.wicketstuff:async-task-demo
» org.wicketstuff:wicketstuff-security-swarm
» org.wicketstuff:wicketstuff-htmlcompressor-examples
» org.wicketstuff:wicketstuff-jwicket-ui-sort
» org.wicketstuff:wicketstuff-security-wasp
» org.wicketstuff:tinymce3-examples
» org.wicketstuff:wicketstuff-editable-grid
» org.wicketstuff:wicketstuff-jslibraries
» org.wicketstuff:wicketstuff-glassfish4-integration
» org.wicketstuff:wicketstuff-servlet3
» org.wicketstuff:wicketstuff-poi
» org.wicketstuff:wicketstuff-jsr303
» org.wicketstuff:wicketstuff-wicket7

» org.wicketstuff:wicketstuff-googlecharts
» org.wicketstuff:wicketstuff-serializer-ui
» org.wicketstuff:wicketstuff-sitemap-xml-examples
» org.wicketstuff:wicketstuff-gae-initializer
» org.wicketstuff:wicketstuff-yui-common
» org.wicketstuff:wicketstuff-closure-compiler
» org.wicketstuff:wicketstuff-serializer-common
» org.wicketstuff:wicket-mount-example
» org.wicketstuff:wicketstuff-jwicket-tooltip-beautytips
» org.wicketstuff:wicketstuff-datastore-memcached
» org.wicketstuff:wicketstuff-ioc-bundle
» org.wicketstuff:wicketstuff-minis
» org.wicketstuff:wicketstuff-portlet
» org.wicketstuff:wicketstuff-simile-timeline
» org.wicketstuff:wicketstuff-yui-calendar
» org.wicketstuff:wicketstuff-autocomplete-tagit
» org.wicketstuff:wicketstuff-context
» org.wicketstuff:wicketstuff-jquery
» org.wicketstuff:wicketstuff-tinymce4
» org.wicketstuff:wicketstuff-datastore-cassandra
» org.wicketstuff:wicketstuff-inmethod-grid
» org.wicketstuff:wicketstuff-serializer-fast
» org.wicketstuff:wicket-shiro-example-realm
» org.wicketstuff:wicketstuff-osgi
» org.wicketstuff:progressbar-example
» org.wicketstuff:wicketstuff-dashboard-widgets-ofchart
» org.wicketstuff:wicketstuff-jwicket-ui-accordion
» org.wicketstuff:wicketstuff-jee-web
» org.wicketstuff:wicketstuff-datastore-common
» org.wicketstuff:async-task-impl
» org.wicketstuff:wicket-shiro-example-spring-jdbc
» org.wicketstuff:wicketstuff-tinymce3
» org.wicketstuff:wicketstuff-javaee-inject
» org.wicketstuff:wicketstuff-select2
» org.wicketstuff:wicketstuff-jwicket-ui-datepicker
» org.wicketstuff:whiteboard-examples
» org.wicketstuff.foundation:wicket-foundation-core
» org.wicketstuff.scala:wicketstuff-scala-archetype
» org.wicketstuff.scala:wicketstuff-scala
» org.wicketstuff.scala:wicketstuff-sample
» org.wildfly:wildfly-connector
» org.wildfly:wildfly-weld
» org.wildfly:wildfly-naming
» org.wildfly:wildfly-clustering-spi
» org.wildfly:wildfly-mod_cluster-extension
» org.wildfly:wildfly-iiop-openjdk
» org.wildfly:wildfly-clustering-service
» org.wildfly:jipijapa-eclipselink
» org.wildfly:wildfly-messaging-activemq
» org.wildfly:jipijapa-hibernate5
» org.wildfly:wildfly-clustering-ejb-spi
» org.wildfly:wildfly-jacorb
» org.wildfly:wildfly-pojo
» org.wildfly:wildfly-web-common
» org.wildfly:wildfly-clustering-common
» org.wildfly:jipijapa-hibernate4-1
» org.wildfly:wildfly-bean-validation
» org.wildfly:wildfly-jpa
» org.wildfly:wildfly-messaging
» org.wildfly:wildfly-webservices-server-integration
» org.wildfly:wildfly-ee
» org.wildfly:wildfly-ejb3
» org.wildfly:wildfly-jsr77
» org.wildfly:wildfly-jaxrs
» org.wildfly:wildfly-jsf
» org.wildfly:wildfly-sar
» org.wildfly.core:wildfly-domain-http-interface
» org.wildfly.core:wildfly-server
» org.wildfly.core:wildfly-jmx
» org.wildfly.core:wildfly-request-controller
» org.wildfly.core:wildfly-controller
» org.wildfly.core:wildfly-patching
» org.wildfly.core:wildfly-cli
» org.wildfly.core:wildfly-host-controller
» org.wildfly.core:wildfly-protocol
» org.wildfly.core:wildfly-embedded
» org.wildfly.core:wildfly-version
» org.wildfly.core:wildfly-discovery
» org.wildfly.core:wildfly-management-client-content
» org.wildfly.core:wildfly-controller-client
» org.wildfly.core:wildfly-io
» org.wildfly.core:wildfly-remoting
» org.wildfly.core:wildfly-process-controller
» org.wildfly.core:wildfly-core-feature-pack
» org.wildfly.core:wildfly-logging
» org.wisdom-framework:resource-controller
» org.wisdom-framework:ehcache-cache-service
» org.wisdom-framework:application-configuration
» org.wisdom-framework:hibernate-validation-service
» org.wisdom-framework:i18n-service
» org.wisdom-framework:default-error-handler
» org.wisdom-framework:wisdom-ipojo-module
» org.xblackcat.sjpu:sjpu-saver
» org.xhtmlrenderer:flying-saucer-pdf
» org.xtext:xtext-gradle-lib
» org.zapodot:jackson-databind-java-optional
» pl.allegro.tech.boot:handlebars-spring-boot-starter
» pl.chilldev.commons:commons-text
» pl.chilldev.commons:commons-jsonrpc
» pl.wavesoftware:eid-exceptions
» ru.stqa.selenium:webdriver-expected-conditions
» ru.stqa.selenium:webdriver-factory
» ru.stqa.selenium:webdriver-wrapper
» ru.stqa.selenium:webdriver-logging-wrapper
» ru.stqa.selenium:webdriver-repeatable-actions
» ru.vyarus:guice-ext-annotations
» ru.yandex.qatools.clay:clay-utils
» ru.yandex.qatools.embed:postgresql-embedded
» se.culvertsoft:mgen-javagenerator
» se.culvertsoft:mgen-idlparser
» se.culvertsoft:mgen-javascriptgenerator
» se.wfh.libs:beencode
» si.uom:si-units-java8
» tech.aroma.banana:banana-thrift
» tec.units:unit-ri
» tv.cntt:xitrum-ko_2.11
» tv.cntt:netcaty_2.11
» tv.cntt:xitrum-hazelcast2_2.11
» tv.cntt:xitrum-hazelcast3_2.11
» tv.cntt:xitrum_2.11
» uk.co.real-logic:aeron-samples
» us.eharning.atomun:atomun-mnemonic
» us.fatehi:schemacrawler-mysql
» us.fatehi:schemacrawler-h2
» uy.klutter:klutter-elasticsearch-jdk7
» uy.klutter:klutter-config-typesafe-jdk6
» uy.klutter:klutter-json-jackson-jdk6
» uy.klutter:klutter-json-jackson-jdk8
» uy.kohesive.injekt:injekt-config-typesafe-jdk7
» uy.kohesive.injekt:injekt-config-typesafe-jdk6
» uy.kohesive.kovert:kovert-vertx
» uy.kohesive.kovert:kovert-vertx-jdk8

sonatype

More than 10 million software developers rely on Sonatype to innovate faster while mitigating security risks inherent in open source. Sonatype's Nexus platform combines in-depth component intelligence with real-time remediation guidance to automate and scale open source governance across every stage of the modern DevOps pipeline. Sonatype is privately held with investments from TPG, Goldman Sachs, Accel Partners, and Hummer Winblad Venture Partners.

Visit www.sonatype.com to learn more.

Headquarters
8161 Maple Lawn Blvd, Suite 250
Fulton, MD 20759
United States • 1.877.866.2836

Virginia Office
8281 Greensboro Dr Suite 630
McLean, VA 22102

APAC Office
5 Martin Place, Level 14
Sydney 2000, NSW
Australia

European Office
199 Bishopsgate
London EC2M 3TY
United Kingdom