Open-Source **Reliability Leaderboard**

Issue one: Findings from 2022

mend.io

Contents

Introduction

Welcome to the inaugural edition of the Mend.io Open-Source Reliability Leaderboard! Powered by data from Renovate Bot, Mend.io's wildly popular open-source dependency management tool, the Leaderboard presents the top packages in terms of reliability across three of the most widely used languages.

We built the Leaderboard for several reasons, starting with the risk imposed by our increasingly vulnerable software supply chain. The ongoing rise in cyberattacks that target the software supply chain, coupled with a shifting regulatory landscape, highlights the growing urgency of building secure applications.

We also wanted a different lens through which to view application security. While existing technologies like software composition analysis (SCA) and static application security testing (SAST) are vital for detecting and remediating problems, little has been done to build a more holistic strategy of preventing, or at least preparing, for problems. The need to consider a more holistic strategy is akin to adopting fitness and healthy living routines as a way to avoid longer term health problems.

Successful implementation of the strategy hinges on having access to the knowledge necessary to prevent possible open-source vulnerabilities from ever being installed in the first place. For that to happen, companies need to know not only what packages are in use at their companies, but ***how safe they are***. This is becoming more important at larger companies, as we see enterprise customers increasingly take this approach by standardizing on a pre-curated selection of reliable open-source code packages.

And finally, we wanted to leverage and share a valuable resource. The Mend.io team knows that there is no better arbiter of package reliability than Renovate, which has gathered crowd-sourced data on over 25 million dependency updates.

Executive Summary

While evaluating software reliability is a challenge for any development program, the world of open source software adds additional hurdles in the form of variances in how open-source code is created, distributed, and supported. While software reliability should naturally be considered when selecting software components, the reality is that 'should be' does not always translate to actually doing so. Therefore, tapping Renovate's rich trove of data to create some reliability rankings seemed like a worthy project.
By analyzing what packages are consistently releasing good updates, we can arrive at an accurate picture of the package's overall reliability for software engineers trying to balance functional risk with security risk.

Like any data-driven project, selecting filtering criteria proved to be a complex and nuanced process. What languages should we evaluate? Did we want to rank the reliability of packages that were updated individually? Doing so would omit packages that were updated as part of a group, an increasingly common practice. Should we filter by major and minor releases? Major versions are by the nature of the update more apt to cause dependency trouble downstream. The criteria we ultimately settled on produced a pretty comprehensive profile for npm, PyPi, and Maven. However, we also wanted to give a tl;dnr option for curious readers who are strapped for time. The shorthand version is below—a top 25 list aggregated from both our individual and group rankings. You can find more granular details in the General Findings section.

Meet the Top 25 by Language

Rank	Package name	Rank	Package name
1	prettier-eslint	14	tap
2	np	15	react-markdown
3	jest-cli	16	c8
4	commitlint	17	@sendgrid/mail
5	@fortawesome/free-regular-svg-icons	18	@nestjs/common
6	@rollup/plugin-babel	19	ava
7	mocha	20	@fortawesome/fontawesome-free
8	@types/mocha	21	semantic-release
9	@nestjs/core	22	@rollup/plugin-commonjs
10	swagger-ui-express	23	@commitlint/config-angular
11	@nestjs/swagger	24	@rollup/plugin-node-resolve
12	@nestjs/testing	25	sinon
13	@semantic-release/github		

Source: Renovate

Maven

Rank	Package name	Rank	Package name
1	org.apache.maven.scm:maven-scm-provider-gitexe	14	org.apache.maven.plugins:maven-install-plugin
2	com.github.ekryd.sortpom:sortpom-maven-plugin	15	io.swagger.parser.v3:swagger-parser
3	org.apache.maven.plugins:maven-release-plugin	16	org.jetbrains.kotlin:kotlin-maven-serialization
4	com.diffplug.spotless:spotless-maven-plugin	17	org.springframework:spring-aspects
5	org.flywaydb:flyway-maven-plugin	18	org.apache.maven.plugins:maven-shade-plugin
6	org.apache.maven.plugins:maven-scm-plugin	19	org.springframework.boot:spring-boot-actuator
7	io.gravitee.common:gravitee-common	20	redis.clients:jedis
8	org.apache.maven.plugins:maven-javadoc-plugin	21	info.picocli:picocli
9	io.gravitee:gravitee-bom	22	com.google.cloud.tools:jib-maven-plugin
10	com.google.cloud:libraries-bom	23	o.sentry:sentry-spring-boot-starter
11	org.springframework.boot:spring-boot-starter-undertow	24	com.slack.api:slack-api-client
12	org.sonatype.plugins:nexus-staging-maven-plugin	25	com.google.auth:google-auth-library-oauth2-http
13	org.owasp:dependency-check-maven		

Source: Renovate

PyPi

Rank	Package name	Rank	Package name
1	pulumi	14	pytest-mock
2	botocore-stubs	15	setuptools
3	types-python-dateutil	16	types-redis
4	types-pytz	17	types-requests
5	slack-sdk	18	aws-lambda-power tools
6	pulumi-aws	19	platformers
7	pip	20	faker
8	types-setuptools	21	aws-cdk.core
9	typing-extensions	22	flake8-bugbear
10	sentry_sdk	23	ruff
11	google-auth	24	sphinx
12	pytz	25	sagemaker
13	jupyterlab		

Source: Renovate

General Findings

Data was pulled for 2022 across three languages: npm, Maven, and PyPi. Keep in mind that the test results are user tests, and sometimes users write bad tests. That's not related to the package.
As such, there will nearly always be some level of failed tests for every package. Nobody scores 100 percent at scale, because users write bad tests.

Criteria

After quite a bit of discussion, the team employed the following filters to build the tables:

Non-grouped (individual) updates and grouped updates were analyzed separately. While most updates are individual, it is increasingly common to run automated batch updates. With that in mind, we felt it was worthwhile to see what packages performed reliably in groups as well as individually. We were also looking for All-Star packages—those that ranked well in both groups.

Minor updates only. By this, we mean that the previous version and the updated version have the same major semantic version number. Because updates to a different major version are intended to cause breaking changes, we did not include those.

Sourced from reliable repos. We excluded repos deemed somewhat iffy due to consistently failing tests.

Tests run after a prior successful run. This was done to avoid counting a failure that was introduced prior to the current update.

Number of versions. With fewer than three releases, the success data for a given package varies too wildly to be useful. So we limited it to packages with at least three releases.

Package popularity. We defined this as the top packages in each language used by Renovate users. Because the data sets varied considerably by language, we chose what we considered to be reasonable cut-off filters for each language, which are noted in the pertinent section. When possible, we also presented separate leaderboards on what we call the Titans: packages for which Renovate has recommended more than 10,000 minor version updates.

Explaining the Rankings

The team created Reliability Leaderboards for the following categories in each of the three programming languages:

Individual Champions

These leaderboards rank the 20 most reliable packages used in individual updates.

Team Players

These leaderboards list packages ranked in the top 20 for reliability for group updates.

All Stars

These packages appeared in the top twenty for both Individual Champions and Team Players.

Titans

The Top 10 most reliable of the most heavily used set of packages (i.e., those for which more than 10,000 Renovate PRs were created).
We separated these packages into a separate group because they were, to some extent, victims of their own success. Logically, more pulls will lead to more operator errors on the customer side.

Results

We had some general predictions going into this, and while some proved correct, we also busted.

Prediction: Group runs bring down overall reliability. True.
Any fan of the TV show *Survivor* can tell you that in competition, groups are often hurt by their weakest link. The same holds true when it comes to group updates. A group of ten packages is ten times more likely to encounter a failure.

Prediction: Frequent releases improve average success rates. False.
You would think frequent releases would correlate to better reliability through faster bug fixes and an engaged maintainer community, but nope! Release frequency had no effect at all on how reliably a package updated. Maybe those teams that take more time between releases are doing better testing.

The Best of the Best

Looking across the categories, the most reliable package for each language are the following.

Npm: np

Maven: org.apache.maven.scm:maven-scm-provider-gitexe

PyPi: Pulumi

We Still Have Questions

There are always more questions than answers, but that doesn't stop us from asking them. We hope to come up with data-driven answers to some of these in the next issue of the Open-Source Reliability Leaderboard.

Why do some packages update well individually and fall off the group update chart?
Yes, group runs bring everything down, but that's just one aspect. We also wonder about interdependencies in a group—that is, would a package have a better success rate with a different group of packages?

Why are some packages more reliable in a group update?
Some packages need to be updated at the same time as others, so will be more likely to fail (or even destined to fail) if upgraded alone. We also wonder whether some good team players are playing on good teams. That is, a package is being updated with other reliable players.

How can we use this to improve group updates?
We know that the way packages are grouped likely affect the success rate at which some are updated—which means that people need to be more intentional about what goes into group updates. Knowing which packages don't perform well in a group allows companies to to improve the groups already in use by updating the trouble makers individually. Used in conjunction with automated merge confidence tools, this could prove helpful when planning group updates and provide visibility into the dependency update gap. Data like this will allow people to create larger groups that will succeed with higher confidence, which means that companies will spend less time processing updates. Bottom line? Improving the quality of group updates helps an organization improve application security.

Note: Some of these packages are not typically used as software dependencies, but instead are used as pipeline tools. Pipeline tools will run after the testing phase of a build has succeeded. So naturally, the failure rate would be close to zero. In a way, it's cheating a bit, so we thought it worth noting.

Color Coding Explained

In the following charts, these colors denote the following:

- Package appears in Individual Champion list
- Package appears in Team Player list
- Package is ranked in both lists.

npm (NodeJS datasource)

For npm, the filter was limited to the top 1,000 packages. The lint and nestjs communities did well, both individually and in group updates.

Individual Champions

Rank	Package name	Number of versions	Success percentage
1	prettier-eslint	5	100.00%
2	np	5	99.68%
3	jest-cli	19	98.73%
4	commitment	16	98.44%
5	@fortawesome/free-regular-svg-icons	7	98.44%
6	@rollup/plugin-babel	5	98.36%
7	mocha	8	98.31%
8	@types/mocha	4	98.21%
9	@nestjs/core	32	98.08%
10	swagger-ui-express	4	98.02%
11	@nestjs/swagger	12	98.01%
12	@nestjs/testing	32	98.01%
13	@semantic-release/github	5	97.97%
14	tap	11	97.94%
15	react-markdown	6	97.94%
16	c8	3	97.90%
17	@sendgrid/mail	3	97.87%
18	@nestjs/common	31	97.87%
19	ava	9	97.85%
20	@fortawesome/fontawesome-free	6	97.78%

Source: Renovate

Best Team Players

Rank	Package name	Number of versions	Success percentage
1	np	5	99.39%
2	ava	9	97.21%
3	tap	11	96.21%
4	prettier-eslint	5	96.12%
5	@commitlint/config-angular	8	94.73%
6	c8	3	94.62%
7	sinon	9	94.58%
8	@types/bluebird	2	94.40%
9	@rollup/plugin-babel	5	94.25%
10	mocha	8	94.00%
11	semantic-release	6	93.25%
12	daisyui	95	93.10%
13	@rollup/plugin-node-resolve	10	93.02%
14	eslint-plugin-ember	15	92.74%
15	rollup	65	92.67%
16	eslint-plugin-svelte3	4	92.67%
17	@rollup/plugin-commonjs	14	92.51%
18	@jest/globals	19	92.40%
19	svelte-preprocess	6	92.31%
20	eslint-plugin-n	15	92.14%

Source: Renovate

All Stars

Package name	Individual rank	Group rank
prettier-eslint	1	4
np	2	1
@rollup/plugin-babel	6	9
mocha	7	10
tap	14	3
c8	16	6
ava	19	2

Source: Renovate

Titans

Package name	Number of versions	Success percentage
jest	34	96.85%
@commitlint/cli	15	96.82%
@types/jest	25	96.63%
lint-staged	27	96.56%
rollup	65	96.36%
ts-node	8	96.31%
ts-jest	21	96.15%
eslint	27	95.52%
pnpm	106	95.07%
@typescript-eslint/eslint-plugin	75	94.67%

Source: Renovate

Maven (Java datasource)

For Maven, the filter is limited by the number of updates. We were happy to see strong representation from Google, Apache, and SpringFramework packages. Indeed, Apache ended up with four packages on the All-Star list. It's reassuring to see that the big players are producing safe updates.

Individual Champions

Rank	Package name	Number of versions	Success percentage
1	org.apache.maven.scm:maven-scm-provider-gitexe	4	99.15%
2	com.amazonaws:aws-java-sdk-kms	66	98.77%
3	com.amazonaws:aws-java-sdk-iam	94	98.68%
4	io.swagger:swagger-annotations	5	98.68%
5	net.javacrumbs.shedlock:shedlock-provider-jdbc-template	16	98.53%
6	org.apache.maven.plugins:maven-release-plugin	3	98.41%
7	io.netty:netty-tcnative-boringssl-static	8	98.31%
8	io.sentry:sentry-spring-boot-starter	35	98.26%
9	com.github.ekryd.sortpom:sortpom-maven-plugin	4	98.21%
10	info.piccoli:piccoli	2	98.21%
11	de.codecentric:spring-boot-admin-starter-client	18	98.21%
12	org.apache.maven.plugins:maven-scm-plugin	3	98.11%
13	com.slack.api:slack-api-client	24	98.10%
14	org.springframework.boot:spring-boot-actuator	14	97.89%
15	net.bytebuddy:byte-buddy	13	97.84%
16	com.diffplug.spotless:spotless-maven-plugin	25	97.78%
17	com.google.cloud:libraries-bom	16	97.74%
18	org.apache.maven.plugins:maven-javadoc-plugin	3	97.51%
19	com.google.cloud.tools:jib-maven-plugin	5	97.10%
20	com.google.auth:google-auth-library-oauth2-http	13	97.10%

Source: Renovate

Best Team Players

Rank	Package name	Number of versions	Success percentage
1	org.apache.maven.scm:maven-scm-provider-gitexe	5	99.39%
2	com.github.ekryd.sortpom:sortpom-maven-plugin	9	97.21%
3	org.apache.maven.plugins:maven-release-plugin	11	96.21%
4	com.diffplug.spotless:spotless-maven-plugin	5	96.12%
5	org.flywaydb:flyway-maven-plugin	8	94.73%
6	org.apache.maven.plugins:maven-scm-plugin	3	94.62%
7	io.gravitee.common:gravitee-common	9	94.58%
8	org.apache.maven.plugins:maven-javadoc-plugin	2	94.40%
9	io.gravitee:gravitee-bom	5	94.25%
10	com.google.cloud:libraries-bom	8	94.00%
11	org.springframework.boot:spring-boot-starter-undertow	6	93.25%
12	org.sonatype.plugins:nexus-staging-maven-plugin	95	93.10%
13	org.owasp:dependency-check-maven	10	93.02%
14	org.apache.maven.plugins:maven-install-plugin	15	92.74%
15	io.swagger.parser.v3:swagger-parser	65	92.67%
16	org.jetbrains.kotlin:kotlin-maven-serialization	4	92.67%
17	org.springframework:spring-aspects	14	92.51%
18	org.apache.maven.plugins:maven-shade-plugin	19	92.40%
19	org.springframework.boot:spring-boot-actuator	6	92.31%
20	redis.clients:jedis	15	92.14%

Source: Renovate

All Stars

Package name	Individual rank	Group rank
org.apache.maven.scm:maven-scm-provider-gitexe	1	1
org.apache.maven.plugins:maven-release-plugin	6	3
com.github.ekryd.sortpom:sortpom-maven-plugin	9	2
org.apache.maven.plugins:maven-scm-plugin	12	6
org.springframework.boot:spring-boot-actuator	14	19
com.diffplug.spotless:spotless-maven-plugin	16	4
com.google.cloud:libraries-bom	17	10
org.apache.maven.plugins:maven-javadoc-plugin	18	8

Source: Renovate

Titans

Package name	Number of versions	Success percentage
io.sentry:sentry-spring-boot-starter	35	98.26%
com.diffplug.spotless:spotless-maven-plugin	25	97.78%
org.apache.maven.plugins:maven-shade-plugin	3	96.47%
io.quarkus.platform:quarkus-maven-plugin	37	95.65%
io.sentry:sentry-logback	34	95.20%
org.slf4j:slf4j-simple	15	94.96%
org.apache.maven.plugins:maven-compiler-plugin	4	94.69%
software.amazon.awssdk:sqs	234	94.52%
org.springdoc:springdoc-openapi-ui	11	93.18%
org.slf4j:slf4j-api	15	92.91%

Source: Renovate

PyPi (Python datasource)

For PyPi, the filter is limited by the number of updates.

Individual Champions

Rank	Package name	Number of versions	Success percentage
1	pulumi	49	100.00%
2	botocore-stubs	180	99.22%
3	types-python-dateutil	20	98.41%
4	types-pytz	13	97.87%
5	slack-sdk	23	97.54%
6	pulumi-aws	42	97.30%
7	pip	13	97.30%
8	types-setuptools	29	97.26%
9	typing-extensions	5	95.77%
10	sentry_sdk	33	95.56%
11	google-auth	21	95.16%
12	pytz	6	95.11%
13	jupyterlab	21	94.85%
14	pytest-mock	6	94.74%
15	setuptools	53	94.61%
16	types-redis	40	94.41%
17	types-requests	51	94.13%
18	aws-lambda-power tools	33	93.79%
19	platformers	8	93.75%
20	faker	42	93.41%

Source: Renovate

Best Team Players

Rank	Package name	Number of versions	Success percentage
1	pulumi	49	97.09%
2	pulumi-aws	42	96.91%
3	pip	13	96.15%
4	ruff	76	92.96%
5	sagemaker	69	91.60%
6	constructs	298	90.29%
7	flake8-bugbear	12	89.64%
8	pymongo	7	89.11%
9	setuptools	53	88.44%
10	flake8-simplify	16	88.32%
11	aws-cdk.core	45	87.96%
12	aws-lambda-power tools	33	87.80%
13	jupyterlab	21	86.13%
14	pytype	36	85.82%
15	sphinx	11	85.23%
16	pipenv	39	85.09%
17	paramiko	11	84.16%
18	ansible	25	83.55%
19	types-requests	51	82.77%
20	types-setuptools	29	82.52%

Source: Renovate

All Stars

Package name	Individual rank	Group rank
pulumi	1	1
pulumi-aws	6	2
pip	7	3
types-setuptools	8	20
jupyterlab	13	13
setuptools	15	9
types-requests	17	19
aws-lambda-power tools	18	12

Source: Renovate

Preventive Application Security with Mend.io

When it comes to preventive application security, one of the smartest things companies can do is to proactively update open source dependencies to reduce the application attack surface and reduce possible problems from out-of-date libraries if emergency updates are needed.
Granted, there will always be a need to balance new development efforts with security requirements. But if done right, this preventive approach won't negatively impact developers' workload and may even free up development resources. We recommend to following:

Automate dependency management.
Organizations should aim to put in place an automated dependency management routine that checks open source dependencies consistently, flags issues, and assists in the remediation process. A good example is the Smart Merge Control feature within Mend SCA. Mend's Smart Merge Control is essentially like autopilot for component updates. It can examine dependencies within a project and batch only the updates that have a high confidence level that they will pass build tests and not break the build.
Smart Merge Control provides a high degree of automation, including identifying all the high confidence updates, generating the associated pull requests, and then merging them, all automatically but with ultimate developer oversight and control.

Assess confidence levels.
Any time an organization updates open-source components for newer, potentially more secure versions, they risk functional problems with existing applications. That's where confidence levels come in. Mend's Merge Confidence ratings levels, available as a standard feature in Mend SCA, assess the likelihood of whether updating a given component will negatively impact application functionality or cause other issues. . Mend.io's Merge Confidence levels are based on peer crowd-sourced data from over 25 million dependency updates tracked by Mend Renovate. Developers can merge updated components with high confidence levels with assurance that they are unlikely to break the build. And when a component has a lower confidence level, it can flag to the developers that extra work may be required to merge it, so they can plan appropriately.

Create batch updates.
Most open source development projects are getting increasingly complex, with more and more components. That means checking and updating dependencies tends to take more time and effort. That's why the Batch Update capability in Mend SCA is especially important. Mend's Smart Merge ControlBatch Update functionality provides a way for developers to batch update (typically high confidence updates) into a single collection that can be applied all at once. Even though it's not complex work, manually generating 10, 20, or 50 pull requests for components that need updating can be time consuming and boring. Mend's Batch Update functionality eliminates the need to do that manually and helps automate the update process.

Contributors

Justin Clareburt

Justin Clareburt is the Product Owner for Renovate at Mend.io. He has been building software solutions since last century, and most recently for Microsoft, Google, and Amazon. Justin is passionate about developer productivity and is renowned for his love of keyboard shortcuts. He is an avid supporter of open-source development, and is responsible for many free popular productivity tools and keyboard shortcut packs.

Rhys Arkins

Rhys Arkins is Vice President of Product Management, responsible for developer solutions at Mend.io. He was the founder of Renovate Bot – an automated tool for software dependency updating, which was acquired by Mend.io in 2019. Rhys is particularly fond of automation and a firm believer in never sending humans to do a machine's job.

Carol Hildebrand

A veteran of Computerworld and CIO magazine, Hildebrand is an award-winning technology writer who writes extensively about cybersecurity and how it impacts business innovation.

About Mend.io

Mend.io, formerly known as WhiteSource, has over a decade of experience helping global organizations build world-class AppSec programs that reduce risk and accelerate development—using tools built into the technologies that software and security teams already love. Our automated technology protects organizations from supply chain and malicious package attacks, vulnerabilities in open source and custom code, and open-source license risks. With a proven track record of successfully meeting complex and large-scale application security needs, Mend.io is the go-to technology for the world's most demanding development and security teams. The company has more than 1,000 customers, including 25 percent of the Fortune 100, and manages Renovate, the open source automated dependency update projectFor more information, visit www.mend.io, the Mend blog, and Mend on LinkedIn and Twitter.