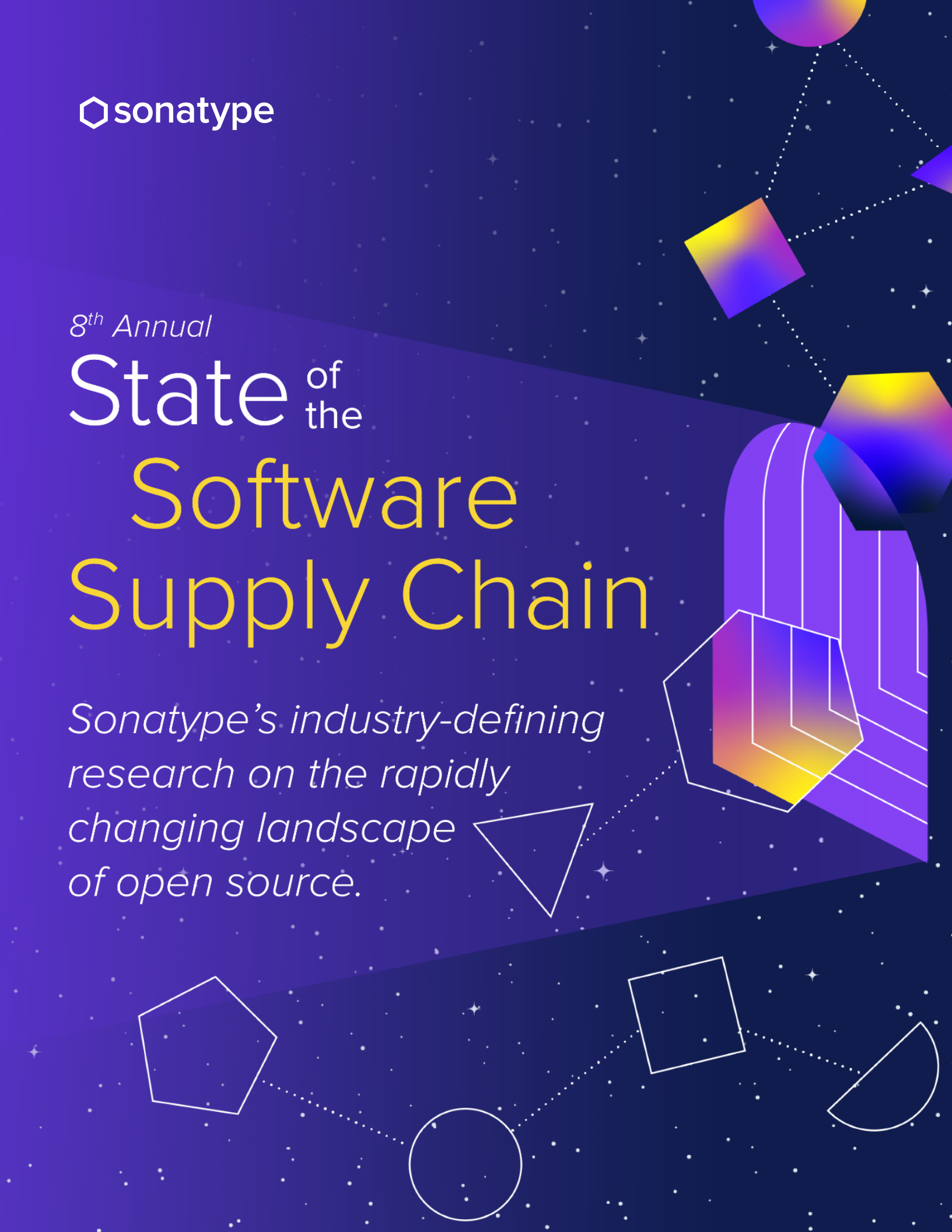
sonatype

8th Annual

State of the Software Supply Chain

Sonatype's industry-defining research on the rapidly changing landscape of open source.

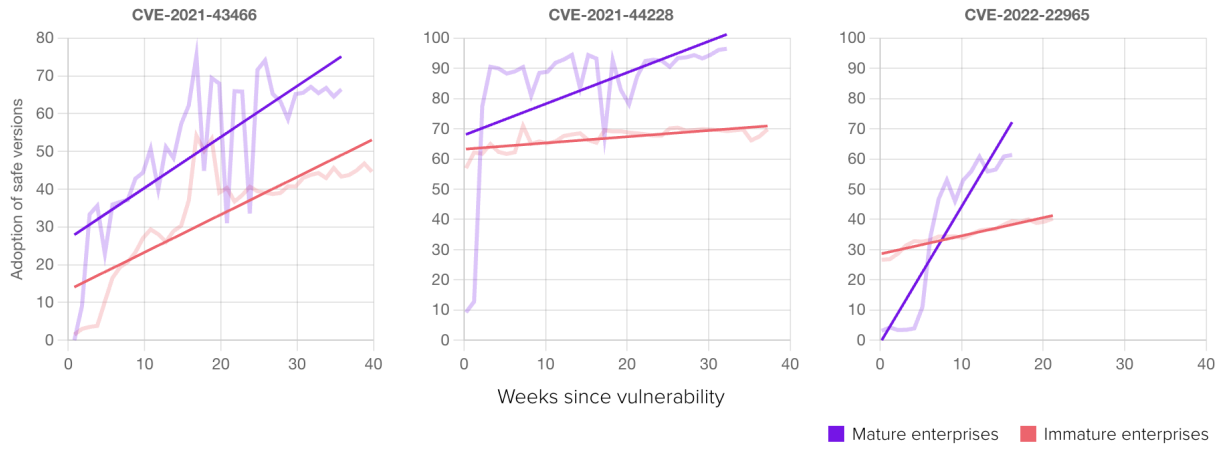


Finding 2 — Consumer vs. Maintainer

As open source supply chain incidents have increasingly made their way into global headlines, questions about where security failures originate have surfaced again and again. Much attention has been paid to open source projects and their maintainers, often labeled as being irresponsible or unwilling to update their software. But who is really to blame?

According to Maven Central repository download data, open source consumers are proliferating the majority of open source risk. Immature consumption behavior is at the root of this – if we change behavior, enormous risk is immediately eliminated from the industry. More than that, there are solutions available today that help solve this problem. As seen in **Figure 3.7**, given the right tools, consumers can change their behaviors and greatly reduce their consumption of open source risk. Enterprises using software supply chain management solutions (Sonatype Nexus Lifecycle in this instance) have noted a 22.6% risk reduction.

FIGURE 3.7. COMPARISON OF MATURE VS. IMMATURE CONSUMPTION BEHAVIOR OVER TIME



Source: Maven Central download statistics and a sampling of enterprise customers (Sonatype Nexus Lifecycle)

Consumption behavior is at the root of this – if we change behavior, enormous risk is immediately eliminated.

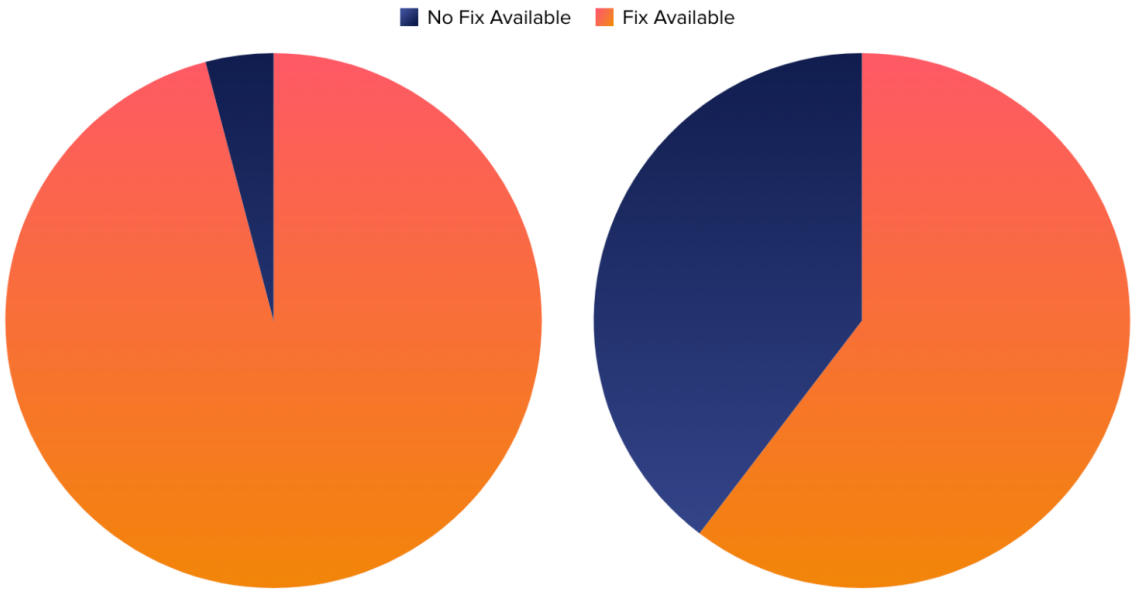
However, these tools are not in widespread use, mostly because of a lack of awareness and perhaps an assumption that such solutions will impair development productivity. Interestingly, the opposite is true. With the right solutions implemented the right way, there is an opportunity for material productivity gains in conjunction with risk reduction.

From the OpenSSF's Open Source Software Security Mobilization Plan to the establishment of community funds for maintainers, we continue to see most open source risk solutions focus heavily on maintainers. However, this one-pronged approach will only help solve part of the problem. While securing production is an excellent first step and very important, even more urgent is the need to secure consumption and raise awareness of the benefits.

What Did We Analyze?

We analyzed how the world consumes open source from Maven Central across 131+ billion downloads over the year. We compared consumers downloading vulnerable dependencies without a fixed version to vulnerable dependencies where a fixed version was available but not chosen. From the 8.6 billion monthly downloads, 1.2 billion vulnerable components were consumed or 14% of all downloads are vulnerable.

FIGURE 3.8. CONSUMER VS MAINTAINER OF VULNERABLE DOWNLOADS



How Common Are Fixed Vulnerabilities?

We found that 55 million or 4.5% of the vulnerable dependencies were due to a vulnerable version with no available fix – meaning 95.5% of known-vulnerable downloads had a non-vulnerable option available. That means 1.18 billion avoidable vulnerable dependencies are being consumed each month. Consumers of these no-fix-available projects were selecting these versions as they had no other choice. Despite the low number of vulnerable versions with no alternatives, a great majority of vulnerable versions that have a fix are still being downloaded.

How Common Are Vulnerable Releases?

Could the sheer volume of vulnerable releases be causing developers to choose vulnerable dependencies? The fact that a monthly average of 1.2 billion billion vulnerable dependencies were downloaded is a very big problem. But, 1.2 billion vulnerable downloads does not equate to 1.2 billion vulnerable releases. How many vulnerable releases does that number actually represent?

There are approximately 10 million releases available for download in the Maven Central Repository. According to our data, only 35% of those releases (3.5 million) included a known-vulnerable issue. Of the vulnerable releases, only 4.2% (147,000) had no available fix. This means that 95.8% of vulnerable downloaded releases had a fixed version available.

Because the total releases with no fix available are a drop in the bucket compared to total vulnerable releases, we cannot assume these vulnerable releases are to blame. Could that 4.2% be improved? Certainly. However, our analysis shows that consumers are disproportionately selecting vulnerable versions when a fixed version is available.

How Many Poor Choices Are Being Made?

We now know that this very big problem comes from a relatively small number of releases. But, how many people are actually culpable in the problem space? There are approximately 26 million developers – or "consumers" – around the world. According to our data, around 14.4 million of those consumers are downloading vulnerable dependencies. Of this group, 5.7 million downloaded a dependency with no fix available–meaning 8.7 million consumers had a fixed choice available to them but still chose a vulnerable version. Even if a fixed version is fix available, perhaps as a result of maintainers better protecting OSS projects–14.4 million consumers are still choosing a compromised version instead. Clearly, we cannot solve the issue of open source security without consumers changing their behaviors.

Why Are Consumers Making Poor Choices?

Why have 8.7 million open source consumers chosen vulnerable versions over non-vulnerable versions? Though there can be reasons for deliberately using vulnerable versions, they should be very rare. For example, using a vulnerable version with no fix available to avoid a critical code break, or if you have a mitigating control in place. If the project maintainer has not fixed the vulnerability, it's time to change the underlying technology. Unfortunately, changes at that level are no easy task, requiring a lot of time, consideration, and investment–things organizations and engineering teams may not prioritize over speed or efficiency.

We're continuing to explore why this may be happening, but we've seen a few themes emerge over the years:

Possible Explanations for Poor Component Choice

- **Popularity** - When deciding which dependencies 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”). Theoretically, this makes sense as, more popular projects should be getting fixed faster. But they aren't. As revealed in our 2019 State of the Software Supply Chain Report, the popularity of a dependency does not correlate with a faster median update time. Developers may feel safe in selecting more popular projects, but just because a dependency is popular, doesn't necessarily mean it's “better.”
- **Clarity** - Oftentimes, developers aren't manually selecting individual versions when building software supply chains–those dependencies are already part of a project that's being used or built upon. As cited in the 2020 State of the Software Supply Chain Report, 80-90% of modern applications consist of open source software. If an SBOM and proper DevSecOps practices are not implemented, developers and software engineering teams may have no way of knowing that those vulnerable components are being used, pulled, or built upon.
- **Automation** - Though there are plenty of open source automation tools, very few have security capabilities built in. Similar to the Clarity issue above, this automation may mask potential vulnerable dependencies, enabling developers to unknowingly build upon projects with known vulnerabilities.
- **Inactive Releases** - There are almost 500,000 projects within Maven Central, but only ~74,000 of those projects are actively used. That means 85% of projects are sitting in this repository and taking up space, potentially overwhelming developers with available options.

As seen in **Figure 3.8** above, removing the small percentage of no-fix-available releases (in blue on the left) would only remove 40% of the no-fix-available issues visible (in blue on the right). This is because – even if there were zero options with no available fix – consumers are still downloading vulnerable versions. Since only a small percentage of the problem stems from open source project maintainers, the focus must shift. The most significant and persistent risks are owned by consumers, who need solutions to consistently choose safe versions.

About the Analysis

The authors have taken great care to present statistically significant sample sizes with regard to component versions, downloads, vulnerability counts, and other data surfaced in this year's report. While Sonatype has direct access to primary data for Java, JavaScript, Python, .NET, and other component formats, we also reference third-party data sources as documented. Further, Sonatype's research analyzed scan data from 185,000 anonymized, validated applications.

Acknowledgements

Each year, the State of the Software Supply Chain report is a labor of love. It is produced to shed light on the patterns and practices associated with open source, development and the evolution of software supply chain management practices.

The report is made possible thanks to a tremendous effort put forth by many team members at Sonatype, including Alex Aklson, PhD, Alexis Del Duke, Alli VanKanegan, Andrew Yorra, Audra Davis-Hurst, Ax Sharma, Bill Healey, Brian Fox, Bruce Mayhew, Eddie Knight, Elissa Walters, Ember DeBoer, Ilkka Turunen, Juan Morales, Katy Hiller, Leina Sanchez, Luke McBride, Maury Cupitt, Mike Hansen, Mitun Zavery, Nicole Lavella, Phil Snare, Stephen Magill, PhD, Steve Poole, Tara Condon, Tiffany Jennings, Todd Baseden and Vlad Drobinin, PhD.

We would also like to offer thanks for contributions, big and small, and for sharing perspectives with our many colleagues across the DevOps and open source development community.

About Sonatype

Sonatype is the software supply chain management company. We empower developers and security professionals with intelligent tools to innovate more securely at scale. Our platform addresses every element of an organization's entire software development life cycle, including third-party open source code, first-party source code, and containerized code. Sonatype identifies critical security vulnerabilities and code quality issues and reports results directly to developers when they can most effectively fix them. This helps organizations develop consistently high-quality, secure software which fully meets their business needs and those of their end-customers and partners. More than 2,000 organizations, including 70% of the Fortune 100, and 15 million software developers already rely on our tools and guidance to help them deliver and maintain exceptional and secure software.