Benchmark Report | 2022

THROUGH THE SIG LOOKING GLASS

softwareimprovementgroup.com

FOREWORD

The SIG annual benchmark reports have always interested me highly, but this year's report is of particular relevance. Going through the findings, I am sure you will see many pieces of information that will make you think.

In fact, I believe in this report our team will present insights that will change the way you deal with the development of software. Or to put it more bluntly, if reading the report doesn't make you change your way of working, we either failed to get our story across, or you're one of the very few exceptions.

Every year, we present how software and software quality are doing globally, how industries perform and how different technologies stack-up. See for yourself where your industry has ended up, and whether you see the need to increase your efforts. Have a look at which technologies will typically give you best quality and reconsider your legacy estate to evaluate if this isn't the moment to start modernizing.

However, I want to draw your special attention to our research on open source.

Estimates state that no less than 80% of the world's production code is actually open source software, but the way open source is managed is frankly up for renewal. Read the research to see how long known vulnerabilities are left untouched, hoping nothing goes wrong. Have a look at our ground-breaking research where we analyzed how the build quality of software impacts the likelihood of a security breach, or a vulnerability occurring. Please read this part carefully, and you will change the way you work with open source, and you will never be surprised again about the Log4J event. This was an accident waiting to happen.

I wish you a truly informative reading experience going through our report. We're proud of the results, and we hope they will help you in creating that healthier digital world we aim for.

Best regards,

Luc Brandts

Group CEO

Dr. Luc Brandts is CEO of SIG Holding. He has worked in the information technology industry since 1994 when he founded his company, BWise, growing it to become a recognized global market leader in the risk management and compliance space. Throughout his career, he has also held various board member and investor roles. Brandts holds a PhD in mechanical engineering from the University of Eindhoven.

Author Benchmark Report

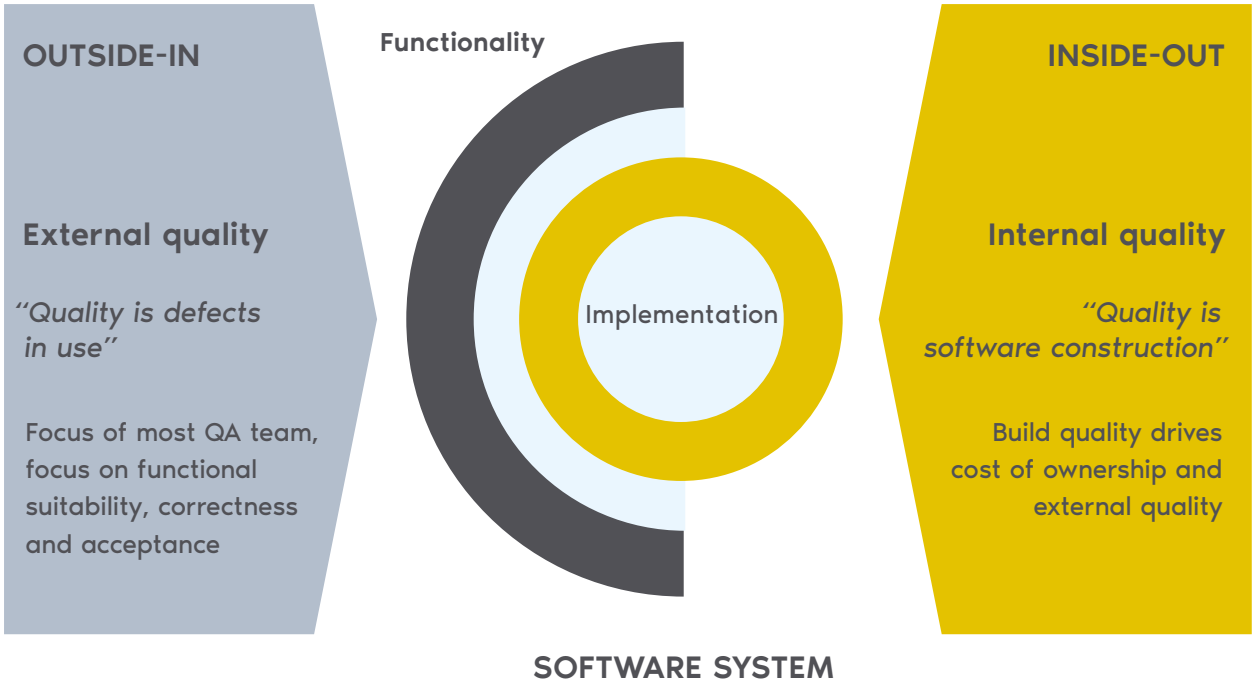
Dr. Magiel Bruntink is Head of Research at Software Improvement Group. He is an internationally published author in the field of software engineering, with 20 years of experience in research, consulting, and education. Bruntink holds a PhD in Computer Science from Delft University of Technology.

CONTENTS

Introduction to the industry trends in build quality

SOFTWARE ASSURANCE REQUIRES BOTH OUTSIDE-IN AND INSIDE-OUT APPROACHES

The annual SIG Benchmark Report examines the software build quality trends SIG has measured in software engineering. The 2022 edition spans more than 13 years of measurements across more than 7,500 systems. Our database is an aggregation of build quality and security measurements of more than 70 billion lines of code.



In order to fully understand the risks of a software system, it is not enough to look at the software from the outside. You really need to look at all the code, only then a full understanding is possible. Seeing a demo, using the software, or trying to break in from the outside will show less than 10% of potential trouble.

Over many years, decades even, software engineering has quietly evolved from its niche and proprietary beginnings to its current state of sprawling ecosystems of (open source) software. At the same time, the maturity of programming language technology, automated tool support, and development process expertise have all increased dramatically.

THE CORE CHALLENGES THAT REMAIN

In previous editions we already turned our attention to technical debt and legacy software weighing down the IT budgets and innovation capacity of organizations across the industry. Since then, legacy software remains the elephant in the room for many organizations and will be for years to come. At the same time, sustained developments in recent years are showing that build quality improvement is indeed possible with the right technology, process, and people.

In this edition however, we are turning our attention to a mostly silent revolution that has taken place in enterprise software of recent years:

80% of modern applications are sourced from software supply chains, often third-party open-source software projects.

Sprawling open-source ecosystems such as Maven (Java), NuGet (C#), Npm (JavaScript), and many others, have become essential to modern software development. It is easier than ever for developers to obtain common library functionality from software supply chains, and to contribute back to the open-source projects that produce them. This reuse capability provides major productivity benefits across the industry.

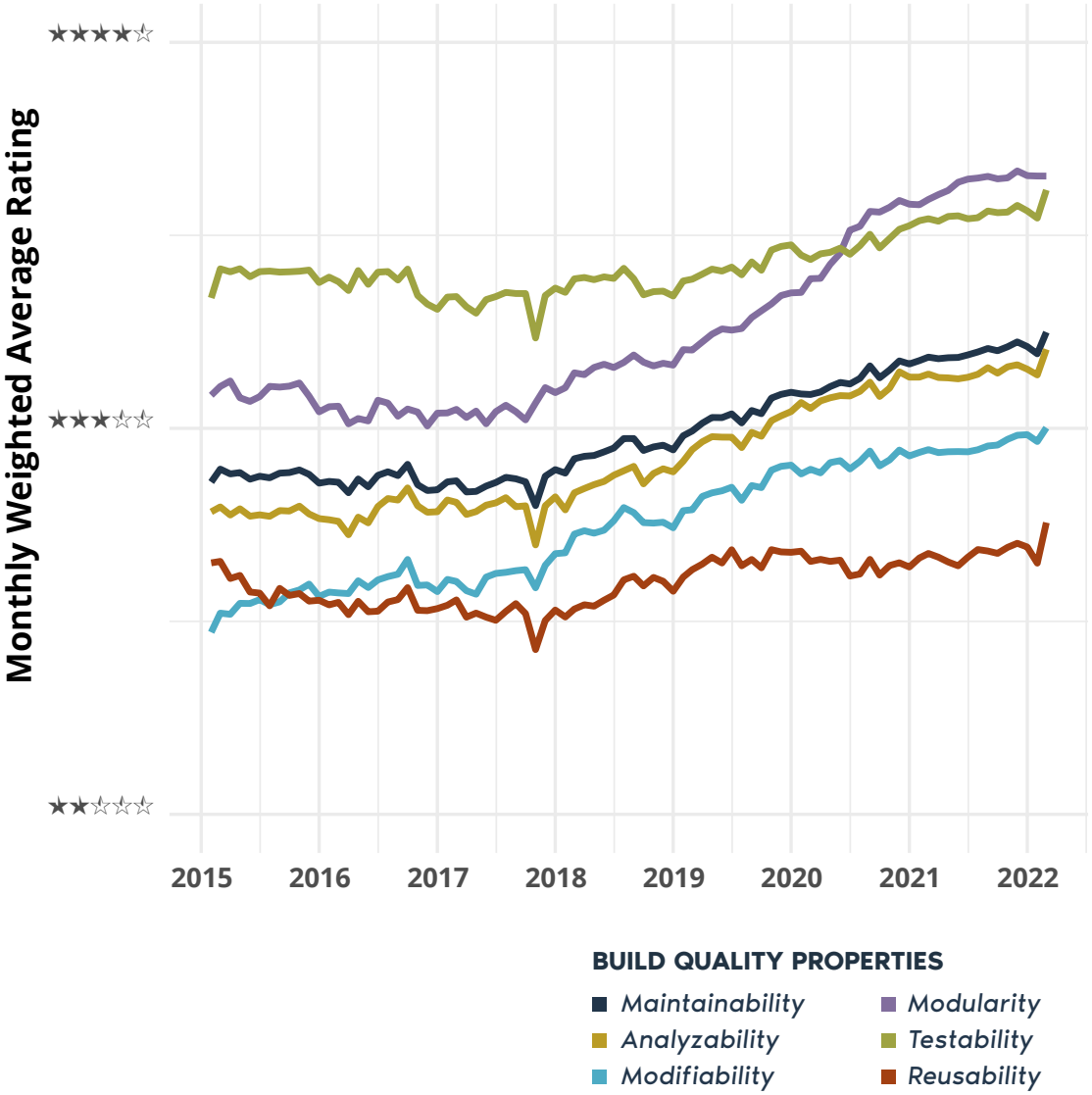
However, maintaining that this revolution happens silently is getting increasingly hard. The large-scale reliance on software supply chains indeed comes with risks, often impacting security or data privacy aspects. A recent and renowned example in a long series of security vulnerabilities, Log4Shell led to widespread coverage in the mainstream media around the 2021 Holidays.

This 2022 edition of the Benchmark Report examines the state of open-source health of thousands of our enterprise software clients.

We follow-up with recent R&D directions for SIG's assurance and benchmarking services that facilitate the industry to *shift-left* on security and software supply chains risks.

GLOBAL BUILD QUALITY AS MEASURED BY SIG

Including 100K person-years worth of enterprise software



Introduction to the Industry Trends in Build Quality

SOFTWARE BUILD QUALITY NEEDS CONTINUES IMPROVEMENT TO KEEP SUPPORTING BUSINESS GOALS

A core element of SIG's software assurance is the measurement of maintainability, an aspect of software build quality as defined by ISO/IES 25010:2011. Maintainability is a major factor in keeping software-related TCO low, and business agility high. With Sigrid, SIG has measured Maintainability and its underlying metrics for 7,500 of our client's software systems over the years.

Looking at the timeline of the past seven years of our core maintainability metrics, we are clearly seeing an upward trend in the overall scores since about 2018. That's indeed positive, because in the enterprise software domain, continuous quality improvement is a necessity. Without it, business falls behind on cost efficiency, innovation, and agility to deliver in the market.

The largest build quality improvement is seen for Modularity, which is the measurement of software component composition. Software applications developed today tend to be smaller and better structured than the builds of the past - a very positive trend. Whether this continues is determined by teams maintaining focus and getting the help they need to safeguard build quality.

INDUSTRY SECTORS AND TECHNOLOGY STACKS ARE HIGH-LEVEL REFERENCE POINTS IN SIG BENCHMARK

7,500+ systems evaluated

800.000+ inspections

70 billion+ Lines of Code
in data warehouse

300+ technologies

SIG HAS THE LARGEST SOFTWARE METRIC DATABASE IN THE WORLD
The SIG benchmark is based on a certified and yearly calibrated subset of the data in our data warehouse.

In addition to build quality metrics, our database includes data on the industrial sectors in which the systems of our clients operate, and on the technology stacks they employ. For yearly reference, we rank the most popular industry sectors and technology stacks in our database, according to the average build quality we see for their systems.

Of course, these ranking are mostly indicative. An actual choice of technology stack depends on many factors that depend on application type, organization, and other factors. Within each industry sector there can also be high- and low performers that may not be reflected by the averages shown here.

- Deltas indicate position change since the 2021 Benchmark report.
- Scores range between 0.5 and 5.5 stars in the SIG Maintainability Model.
- Scores are weighted by their (code) volume, most recent snapshot per system.
- At least 50 systems measured per sector, and 45 per technology stack.

#	Delta	Technology stacks: 2019 through 2021	Score
1		**Low Code**	**3.37**
2	**+**	**Scripting and Mobile**	**3.30**
3	**-**	**Java/JVM**	**3.29**
4		Microsoft/.NET	3.05
5		Packaged Solution Customizations	2.99
6		BPM/Middleware	2.91
7		Legacy/3GL/4GL	2.49

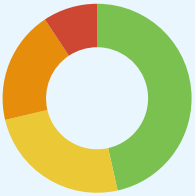
#	Delta	Technology stacks: 2019 through 2021	Score
1		**Industrial Transportation**	**3.44**
2	**++++**	**Energy, Oil & Gas**	**3.40**
3	**–**	**Banking**	**3.32**
4	–	Insurance	3.25
5		Government	3.16
6	––	Financial Services	3.16
7	+	Software & Computer Services	3.06
8	++	Telecommunications	2.87
9	––	Retail	2.83
10	–	Support Services	2.83


GLOBAL
BUILD
QUALITY

MAINTAINABILITY MEASUREMENT IS OUR TOOL TO DETERMINE SOFTWARE BUILD QUALITY

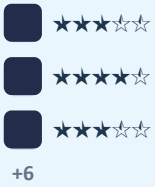
1 - MEASUREMENTS

Perform measurements on the code base



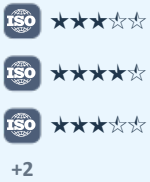
2 - QUALITY PROFILES

Aggregate measurements to **quality profiles**



3 - SYSTEM CHARACTERISTICS

Translate quality profiles to **system characteristic scores**



4 - ISO STANDARD SUB-CHARACTERISTICS

Translate to **ISO 25010 sub characteristic scores**

5 - OVERALL RATING

Translate to **overall rating** of technical quality

THE TICKING TIMEBOMB OF UNCONTROLLED SOFTWARE SUPPLY CHAINS

Healthy usage of third-party open source software (OSS) provides many benefits, like increased development speed through code reuse. Already back in 2014, Contrast Security reported that 80% of applications consist of third-party library code.[1] Now in 2022 this rate is likely much higher, following the strongly growing demand of OSS packages that is being reported by software supply chain vendors like SonaType in the State of the Software Supply Chain (SSSC) report.[2]

1. *cdn2.hubspot.net/hub/203759/file-1100864196-pdf/docs/Contrast_-_Insecure_Libraries_2014.pdf*
2. *sonatype.com/resources/white-paper-2021-state-of-the-software-supply-chain-report-2021*

While it is unthinkable to forego the reuse and development speed-up opportunities of OSS packages, there are fundamental risks that need mitigation. A point clearly made by the President of the United States in February 2021, in the Executive Order on America's Supply Chains, which has since been followed up by organizations and vendors world-wide. With the increased demand, also the rate of reported security vulnerabilities (CVEs) is ever growing (20K CVEs in 2021, compared to 18K in 2020). Perhaps to prove the point, just before the 2021 Holidays, the Log4Shell vulnerability struck and forced security assurance teams across the globe to mitigate.

Since the beginning of 2021 SIG tracks 1,000 client systems specifically to help mitigate software supply chain risks with our Sigrid Open Source Health module. In this edition of the Benchmark Report, we can provide unique insights due to our combination of software supply chain and build quality data, as observed with our enterprise software clients.

SIG Research analyzed a grand total of 5.7 million data points in the period 2021-Q1 until 2022-Q2, across 18 different package management systems, including all the major OSS ecosystems.

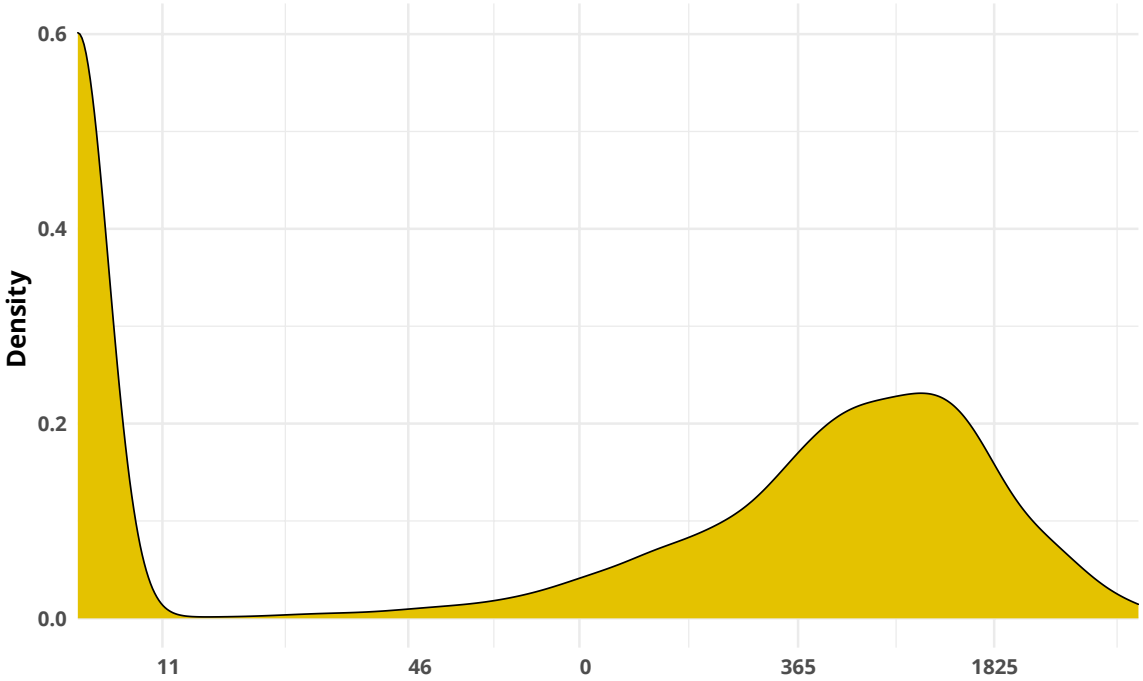
WHAT WE ARE SEEING SUGGESTS THE FOLLOWING:

Overall, the enterprise software domain urgently NEEDS TO IMPROVE its OSS USAGE HEALTH.

OSS libraries of lower build quality have a HIGHER RISK of emerging SECURITY VULNERABILITIES.

Client applications with HIGHER BUILD QUALITY and automated dependency management ARE AHEAD of the pack.

HEALTHY USAGE OF OPEN-SOURCE SOFTWARE IS IN URGENT NEED OF ATTENTION



DEPENDENCY STALENESS DENSITY

Tracking 49K dependency versions of 18 ecosystems in 1000 client systems

Healthy usage of OSS is a multi-factor concept, and often a resource balancing act on the scale of a software portfolio. Typical enterprise software systems have 50 direct OSS dependencies, let alone the dependencies of those dependencies. Still, key risk factors to manage for all OSS dependencies include:

- **FRESHNESS:** are we up-to-date with the latest versions?
- **VULNERABILITY:** are we exposed to known security vulnerabilities?
- **BUILD QUALITY:** does the software we rely on conform to build quality standards?

There are many more factors to consider, such as the predictability of OSS development projects, potentially lurking code licensing risks, issues related to library popularity, API coupling, and solution lock-in.

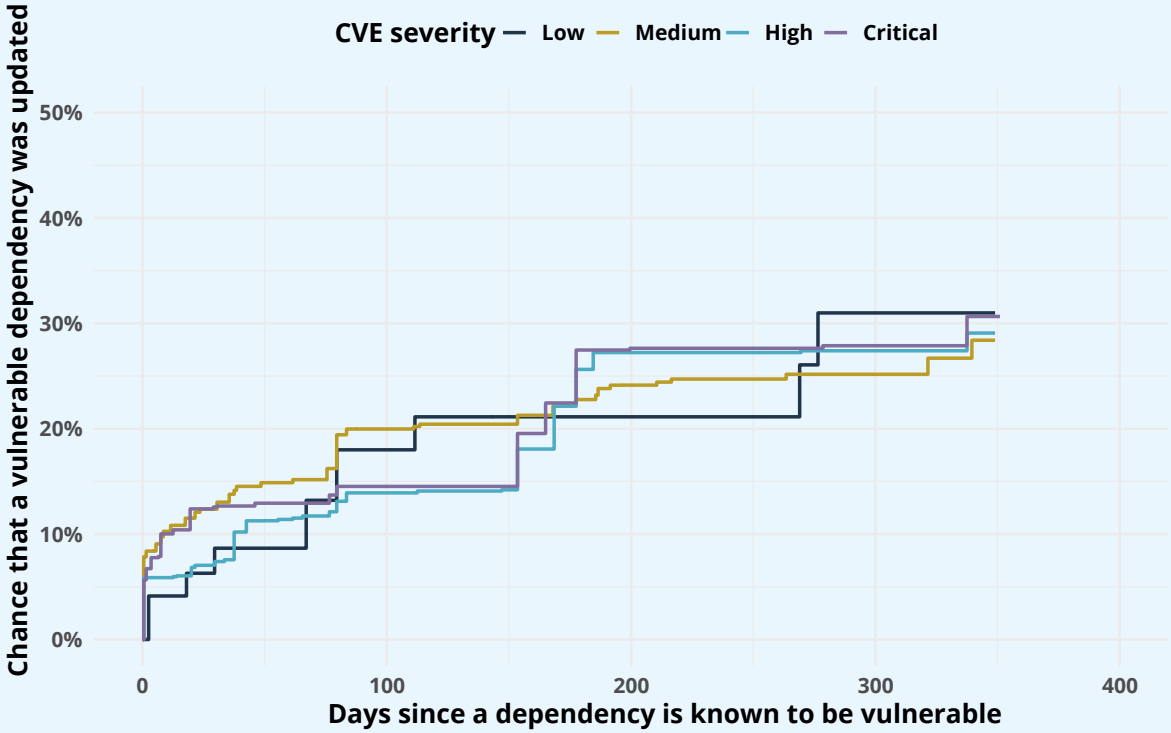
The data we are seeing indicate that, despite the best efforts of developers and Software Composition Analysis (SCA) tool vendors, usage of OSS is still far from healthy. This observation applies to all major technologies we are seeing, from Java (Maven, Gradle), .NET (NuGet), to JavaScript (Npm), PHP (Composer), and others.

In our Dependency staleness density plot we show how staleness is distributed across our dataset. There are two clear ''humps'' visible in the data: the hump on the left-side contains dependencies that are upgraded quickly once new versions becomes available. The right-hand-side hump, representing the majority, are dependencies for which upgrades are postponed; either due to lack of information or because of accepting with the inherent risks.

Clearly, this is a large and accumulating risk factor, given that stale versions miss out on necessary fixes for security vulnerabilities and other issues. SonaType's SSSC report indicates between 6.5% and 29% of OSS packages indeed contain vulnerabilities.

KEY FINDING: Open-source libraries are updated **years** rather than **days** after updates become available. This is a growing hazard: the risk of security vulnerabilities emerging in those stale libraries is building up. **Many of them already contain vulnerabilities.** *Stale libraries can be tracked down automatically and can often be updated easily.*

THE SEVERITY OF VULNERABILITIES IS NOT A MAJOR FACTOR...



TIME-TO-UPDATE: SEVERITY OF VULNERABILITIES

Tracking 8000 vulnerable Maven dependencies in 220 Java systems

...IN WHETHER UPDATES ARE DONE

A common idea is that the severity of vulnerabilities, a combination of their estimated impact and ease of exploit, has a relation with how fast software teams do updates. Vulnerabilities are public reports, often referred to as CVEs (for Common Vulnerability Enumeration), that are listed by organizations such as the National Vulnerability Database (NVD) or GitHub Advisories, and other sites. Once CVEs are published, in most cases a patched version of the affected software is already available, and people are expected to update as soon as possible.

TIME-TO-UPDATE is a measure calculated using survival analysis, a methodology commonly used in medical studies revolving around interventions and event follow-up time. We measure the duration between first seeing a library version appear and the time it is updated to a next version. The method accounts for cases that were not followed-up, so we can get an indication of the chance a version was updated (y-axis) after a certain time has passed (x-axis).

SO, DO TEAMS UPDATE VULNERABILITIES FASTER IF THEY ARE MORE SEVERE? Surprisingly, no, says our joint academic research with the Technical University Delft.[1] Among OSS projects, severity, and other vulnerability metadata, are not convincingly used to prioritize or address vulnerabilities.

Unfortunately, the situation appears to be quite similar for enterprise software development, as shown by analysis of data from our database. To perform this analysis, we joined outside data on CVE publication dates and affected versions with the versions used among a subset of our client's systems. Here we include client systems that use the Maven dependency management tool, one of the most prevalent in the industry.

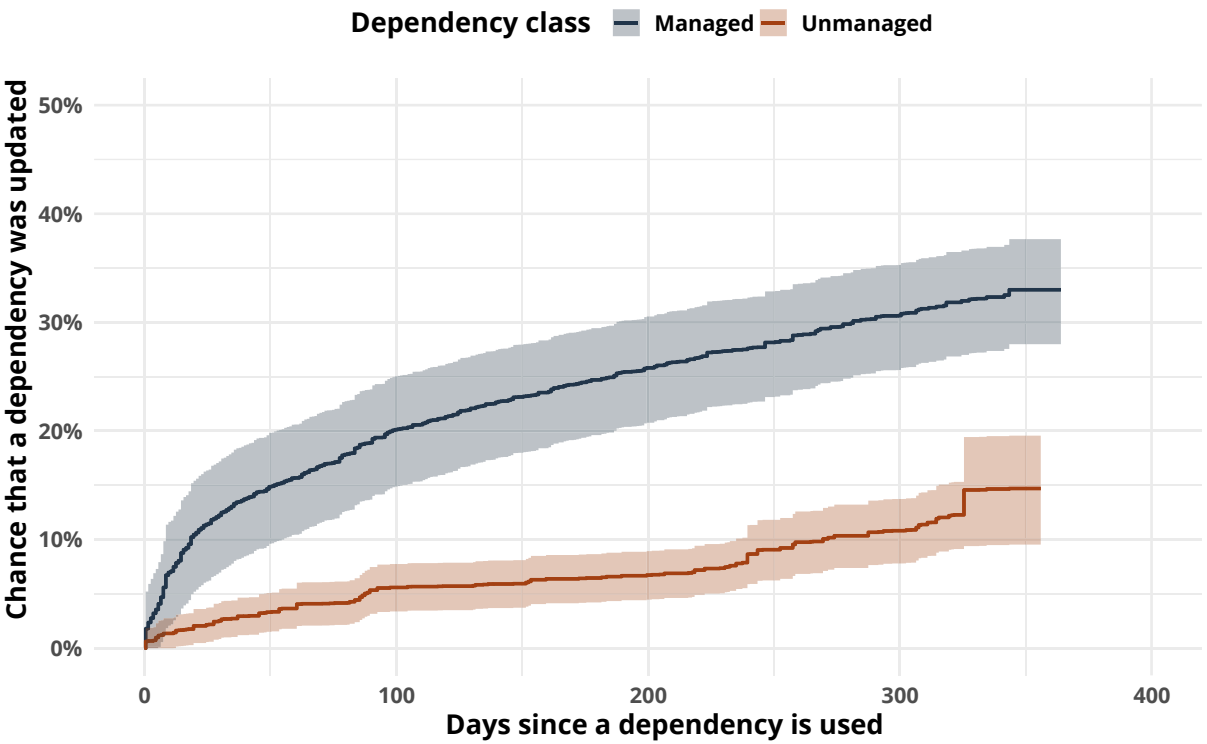
The plot shows that severity of vulnerabilities is only a very minor factor in time-to-update, and only in the first few days after a vulnerability was published. Each level of severity roughly has the same update speed. Our academic research drills down further into the reasons and paths to remediation, based on our observations in OSS project.

For now, we have one key message, which is to immediately increase the urgency in addressing OSS usage health in the enterprise software domain. Developers and software portfolio managers require more visibility of vulnerabilities they are already exposed to and need to be provided with ways to remediate them.

KEY FINDING: Users of known vulnerable open-source libraries are not updating quickly, even if vulnerabilties are critical. **70% are still using known vulnerable Java libraries after a year has passed.** *In many cases, security updates are available that can be implemented by development teams.*

1. R. Heddes, Vulnerability Risk Modelling in Open Source Software Systems, Master's Thesis, TU Delft and Software Improvement Group, 2022.

UNMANAGED DEPENDENCIES ARE A CAUSE OF SLOW UPDATING



TIME-TO-UPDATE: MANAGED VERSUS UNMANAGED DEPENDENCIES

Tracking 151K dependency versions from 9 ecosystems in 900 client systems

Let's zoom in on some factors that may be underlying the overall slow update frequency of OSS dependencies. Broadly speaking, there are two classes of dependency management approaches: managed and unmanaged. In the managed case, an automation tool is used to keep track of required dependencies, resolve transitive dependencies, notify of updates, perform installation, and so on. Examples are Maven, Gradle, NuGet, Npm, and many others.

In contrast, some dependencies are still included in an unmanaged fashion – that is to say, code artifacts (Jars, DLLs, or JavaScript files) are included as-is, typically mixed within the source code structure of an application. These artifacts needs to be tracked and updated by hand, so to speak. This practice, which sounds like something of the past, is still used for 13% of the dependency versions we see in the past year.

In the time-to-update analysis here we are looking at about 19K unmanaged dependency versions versus 151K managed versions for the main technologies Java, C# .NET, and JavaScript. Overall time-to-update is long – as we observed before with many dependencies going stale. Even for the managed dependencies, only about 33% were updated after a year has passed. Clearly though, unmanaged dependencies are the even slower bunch, being updated at half the rate or less. Unmanaged dependencies are often hidden away in large portfolios, making them an unseen risk. Automated tooling such as Sigrid® can detect and help mitigate these cases.

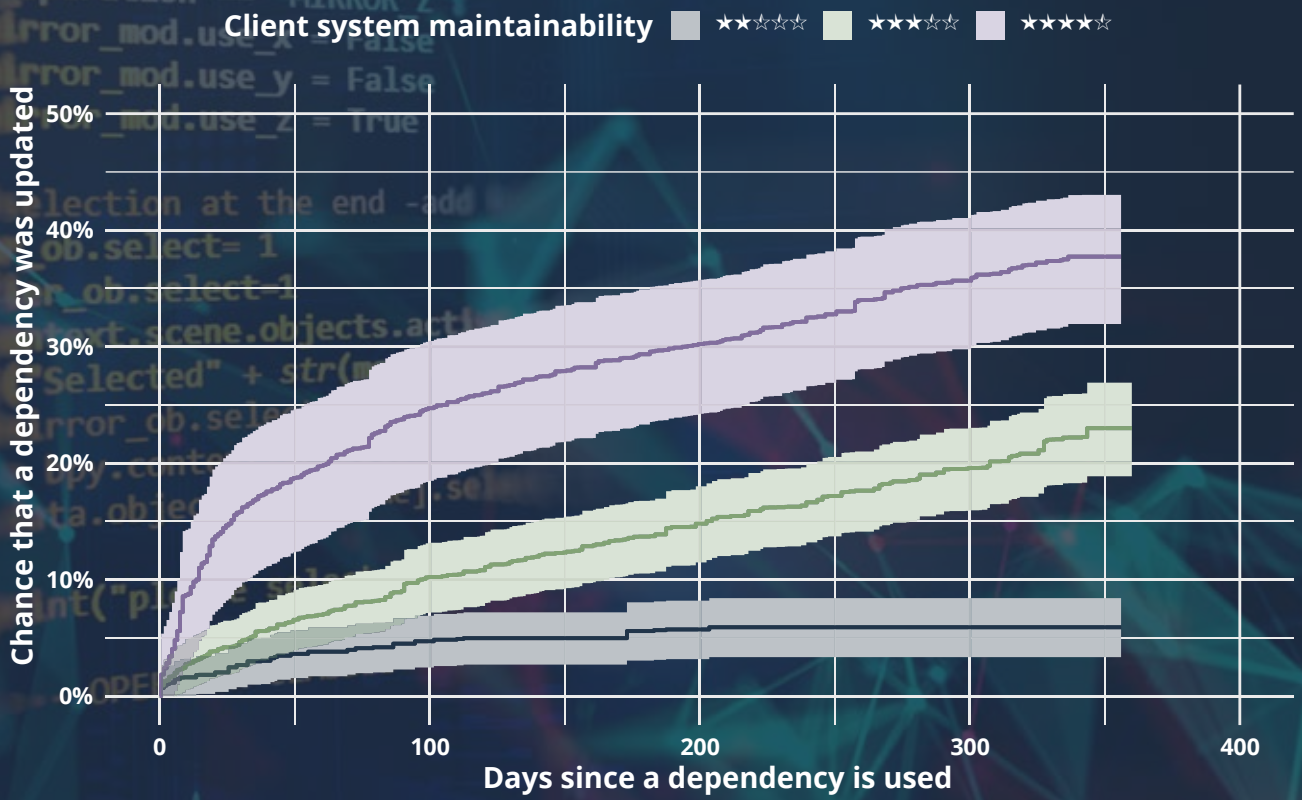
KEY FINDING: Open-source libraries are sometimes included in code bases without automated package management. These unmanaged libraries are updated **more than 2 times slower** than managed libraries. *This practice can be tracked down and replaced by automated package manangement in most cases, reducing the risk of libraries going stale.*

LOWER SYSTEM BUILD QUALITY IS CORRELATED WITH SLOW UPDATING OF DEPENDENCIES

Next, let's turn our attention to the build quality of the enterprise software systems that SIG is tracking. As explained earlier in this report, measuring maintainability is also the bread-and-butter of our software assurance services, making it possible to bring together data from multiple perspectives.

An important task in software maintenance is keeping dependencies up to date. Often, this task consists of incrementing version numbers in configuration files and re-running the test suites, but not always. Sometimes version changes come with changes to an API or to the core functionality, asking for code changes to be implemented in the system.

TIME-TO-UPDATE: MAINTAINABILITY OF CLIENT SYSTEMS



Tracking 161K dependencies from 18 ecosystems in 1000 client systems

In general, better maintainable code is easier and more fun to update, because potential issues are easier to be found since the code is easier to understand, test, and – if needed – fix. We therefore expect a correlation to be visible between maintainability scores for systems and the rate at which they update their dependencies.

In the plot we track the time-to-update for three groups of systems based on their average score of respectively 2, 3 and 4 stars in the SIG Maintainability Model. For 1- and 5-star systems too few datapoints remained, so they are excluded from the analysis. While the 3- and 4-star systems are a strong majority (which is a good thing!) for all three groups a confidently distinct outcome is visible.

Apparently, 4-star systems do tend to update their dependencies faster than 3- and 2-star systems, and 3 stars outperform 2 stars. After 100 days, 25% of the dependency versions of 4-star systems have been updated, compared to just 10% and 6% for the 3- and 2-star system, respectively.

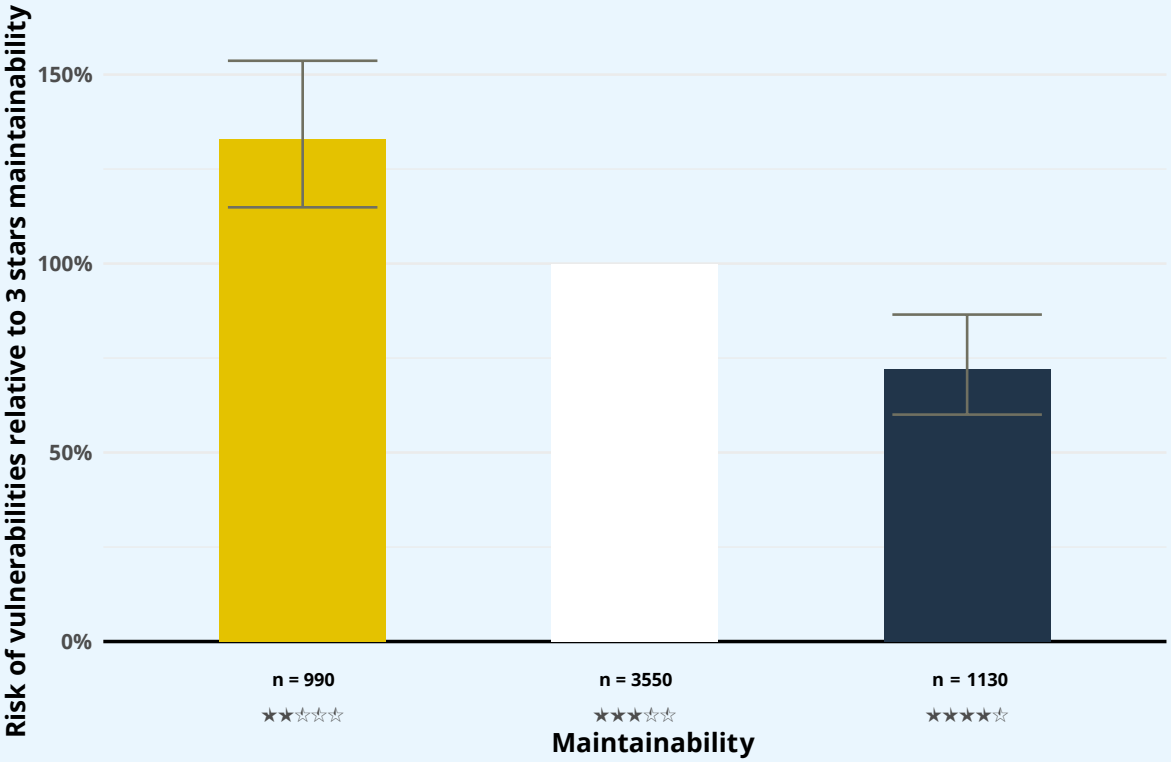
Aiming for high build quality, including maintainability, is recommendable its own right. These data further support that general advice: aim for above average quality (4 stars) and help reduce security risks coming from outdated dependencies. That will lead to fewer cases of panic when security vulnerabilities are published and reach the general press.

KEY FINDING: Enterprise software systems of higher build quality update their open-source libraries much faster than lower quality systems. At the recommended 4-star build quality, updates are **50% faster** than at 3-stars and **more than 300% faster** than in 2-star systems. *Aim for 4 stars in new developments, monitor quality continuously, and refactor existing code as much as feasible.*

BUILD QUALITY AND VULNERABILITY RISK ARE RELATED

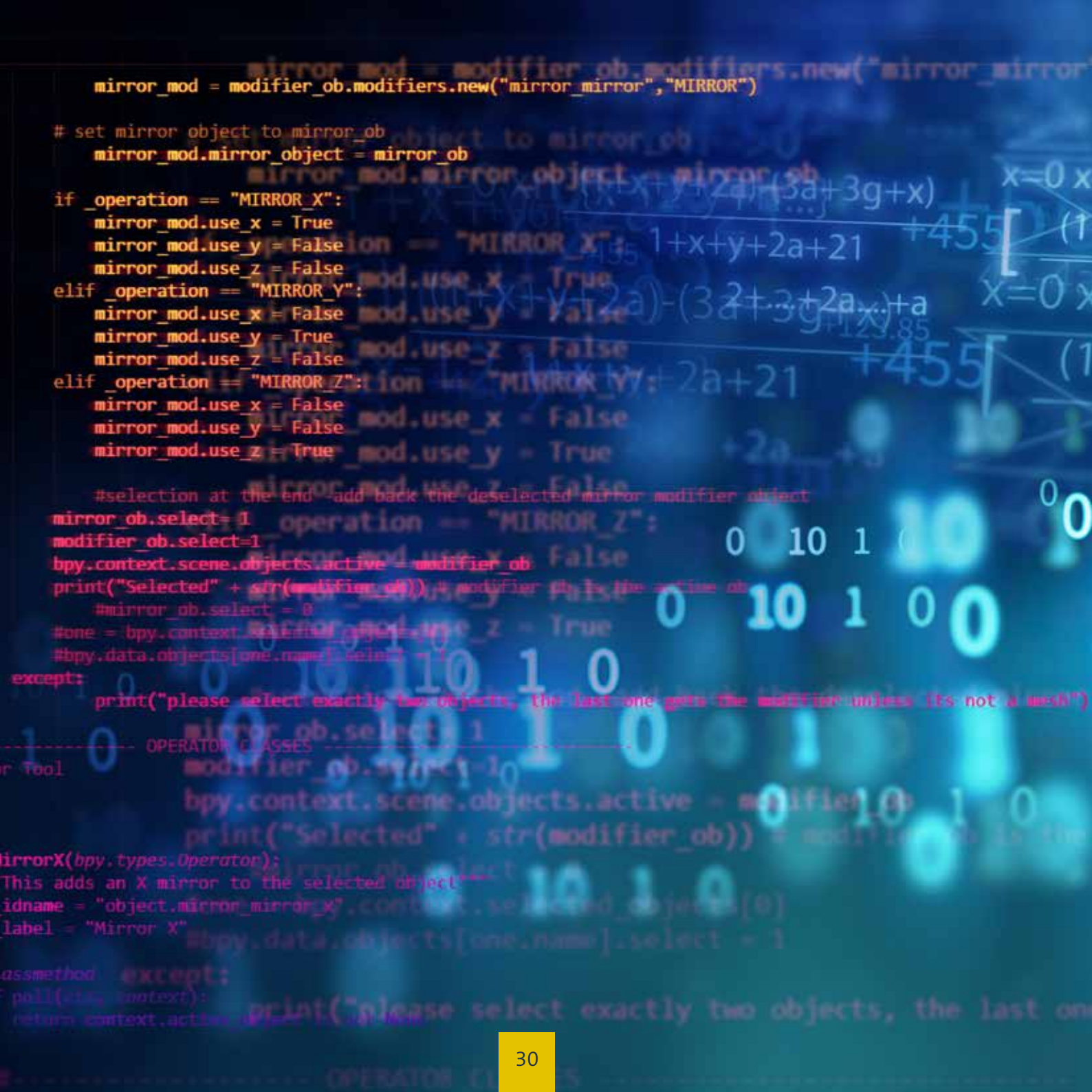
Another common intuition we would like to address is the relation between build quality and the emergence of vulnerabilities. Intuitively, vulnerabilities would be appearing more often in software with lower build quality. Why? Because lower quality software is harder to understand, modify, and test, increasing the potential for error significantly. Lack of quality in design, architecture, and process, can fundamentally increase the risk of future security vulnerabilities as well.

LOWER QUALITY HAS MORE RISK



RELATIVE VULNERABILITY RISK COMPARED TO AVERAGE BUILD QUALITY

Vulnerability and build quality data from 5700 Maven dependency versions


mirror_mod = modifier_ob.modifiers.new("mirror_mirror","MIRROR")
# set mirror object to mirror_ob
mirror_mod.mirror_object = mirror_ob
if _operation == "MIRROR_X":
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
elif _operation == "MIRROR_Y":
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
elif _operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True
#selection at the end -add back the deselected mirror modifier object
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier ob is the active ob
#mirror_ob.select = 0
except:
print("please select exactly two objects, the last one gets the modifier unless its not a mesh")
OPERATOR CLASSES
irrorX(bpy.types.Operator):
This adds an X mirror to the selected object"""
idname = "object.mirror_mirror_x"
label = "Mirror X"
print("please select exactly two objects, the last on

In the bar chart we show the relative risk scores of libraries grouped by their build quality score. In the middle is the 3-star group, which acts as the reference with 100% risk. The proportion of vulnerable dependency versions in that 3-star group is about 15% of 3,550, giving also an idea of absolute risk. To the left and right are the 2- and 4-star groups, respectively, with their risk of having at least 1 vulnerability relative to the 3-star group. There were too few instances of 1-star and 5-star libraries to justify presentation.

The bar chart shows that 2-star build quality is correlated with a higher relative risk of vulnerabilities: about 33% higher than for 3-star build quality. Positively, 4-star build quality has a 28% lower risk of vulnerabilities.

So, 4-star dependencies are less often vulnerable, arguing for a library management strategy that is informed by the build quality of the libraries themselves. Commonly, OSS libraries are added without batting an eye to their internals; that practice will need to change. While our analysis is ongoing, in our underlying data we see branch-point complexity as the strongest predictor of vulnerabilities.

To wrap up, we would like to remark that vulnerabilities do still occur at all levels of build quality, indicating that software is never perfect. Additional tooling, secure software design, and process measures, are required to further reduce security risks.

KEY FINDING: Open-source libraries are sometimes included in code bases without automated package management. These unmanaged libraries are updated **more than 2 times slower** than managed libraries. *This practice can be tracked down and replaced by automated package manangement in most cases, reducing the risk of libraries going stale.*

GETTING SOFTWARE SUPPLY CHAINS BACK IN CONTROL

with these three key findings

1 THE ENTERPRISE SOFTWARE DOMAIN URGENTLY NEEDS TO IMPROVE ITS OSS USAGE HEALTH.

We are seeing way too long response times before OSS dependencies are updated, even in the presence of critical security vulnerabilities. Comparing notes with the SonaType SSSC report, the enterprise software industry is not doing better than the OSS domain in this regard. The tools to implement stronger OSS usage strategies are all available – it is a matter of implementing and enforcing their usage.

2 HIGHER APPLICATION BUILD QUALITY AND AUTOMATED DEPENDENCY MANAGEMENT CORRELATE WITH OSS USAGE HEALTH.

Looking in detail at our data gathered using the Sigrid platform, some correlations are emerging that could help guide improving OSS usage health. First, application code bases of higher build quality tend to update sooner and keep dependencies fresher overall. Second, and perhaps unsurprisingly, the application of automated dependency management tools more than double the overall dependency freshness.

OSS LIBRARIES OF BELOW-AVERAGE BUILD QUALITY HAVE CLOSE TO 2 TIMES MORE RISK OF VULNERABILITIES THAN LIBRARIES AT THE RECOMMENDED 4-STAR LEVEL.

This result calls for integration of build quality measurement of libraries into dependency risk management strategies, in addition to already commonly applied SCA tools and expert-driven security processes.

Finally, it is important to realize that dealing with vulnerabilities is like mopping up the floor with the tap still running. Most software is full of weaknesses that were introduced through error, early design flaws, changing requirements, missed mitigation actions, or even malicious intent. Some weaknesses are more serious than others, and only a few will later cause critical vulnerabilities. Spotting and prioritizing the critical weaknesses earlier on is an important direction in software assurance, helping teams address issues before they manifest in public. We address this need for shifting-left in the next section.

SHIFTING-LEFT ON SECURITY AND SOFTWARE SUPPLY CHAIN RISKS

Shifting-left is an often-heard phrase these days. With some terminology, we can be a bit more concrete in what we think should be accomplished with this shift. To be short about it, software development *shifting-left* implies that it becomes aware, diagnoses, mitigates, or resolves (security) weaknesses at earlier development stage. Such weaknesses are often precursors to more serious problems down the line, for instance security vulnerabilities, performance degradations, or outages.

Many emerging software quality standards and modern industry taxonomies on software use the Mitre CWE – Common Weakness Enumeration – to refer to standardized descriptions of weaknesses in software. Among such standards are, for instance, the OWASP Top 10 listing the most common and problematic security issues, and the new ISO/IEC 5055:2021 standard for *"Automated source code quality measures"* that has chapters on security, performance efficiency, and reliability.

In our view, a lot more needs to happen than creating tools to facilitate automatic checking of source code. For instance, shifting-left on security requires changes in in early design phases, requirements analysis, and training. Many marketplace tools such as Static Analysis Security Testing (SAST), Dynamic Analysis Security Testing (DAST), and several other categories, can indeed help the process become more efficient. However, these tools also have blind spots, and they can generate false alarms. Expertise and eyes-on review will still be a necessary complement.

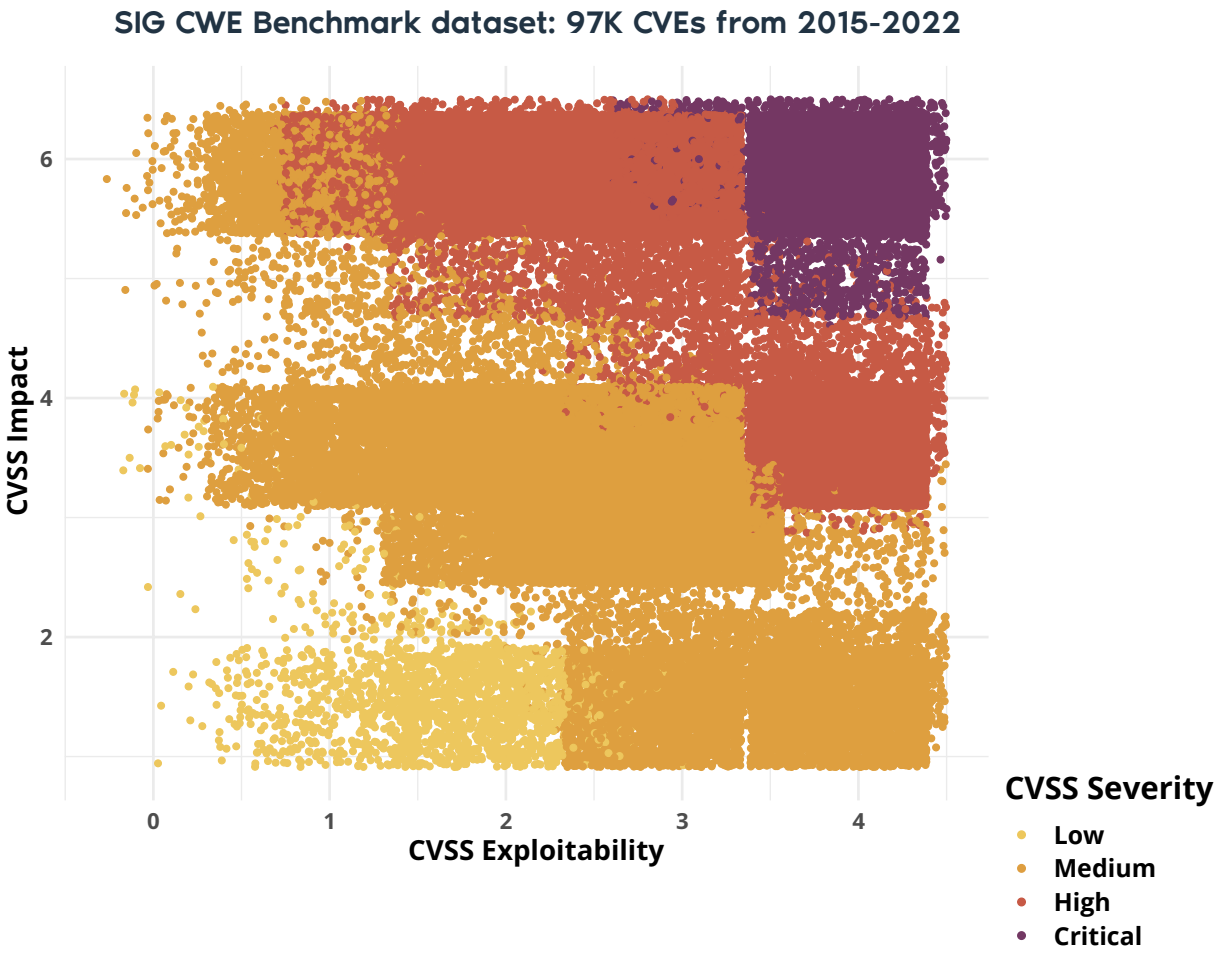
In this section we will highlight three of our solution directions to allow software development to shift-left:

Sigrid® | Software Security: As a new module for Sigrid, SIG's software assurance platform, we are developing a new benchmark that highlights and prioritizes the thousands of weaknesses found by SAST tools. It allows us to highlight weaknesses that commonly lead to critical vulnerabilities, and to filter out those that are exploitable only in niche circumstances.

FASTEN: SIG participates in joint academic-industrial research to create the next generation tools and approaches in software supply chain analysis. These tools spot issue precursors that are lurking in software ecosystems in a fine-grained and accurate manner.

SCRAMBLE: SIG develops support for security code review to help alleviate a pressing concern: extreme shortage of software security expertise. SCRAMBLE will support security code review that both leverages available SAST, DAST, and other tools, and complements the blind spots of automated tools.

SIGRID® | SOFTWARE SECURITY BENCHMARK IS BASED ON THE ANALYSIS OF THOUSANDS OF SECURITY EXPERTS



Nowadays, software vulnerabilities comprise vast public datasets, hosted by institutes such as the US National Vulnerability Database (NVD) and various others. Each vulnerability report, also called CVE, went through a phase of expert analysis, scoring, publication, and potential re-analyses. The information included in such CVE databases is vast. In 2021 alone, 20K CVEs were reported, up from 18K in 2020.

Each CVE is scored using a standardized system called Common Vulnerability Scoring System (CVSS) which incorporates details on attack vector, complexity, impact, and so on. Finally, CVSS provides a score between 1 to 10 reflecting the CVE's overall severity. Commonly, this score is categorized into low, medium, high, and critical severity levels.

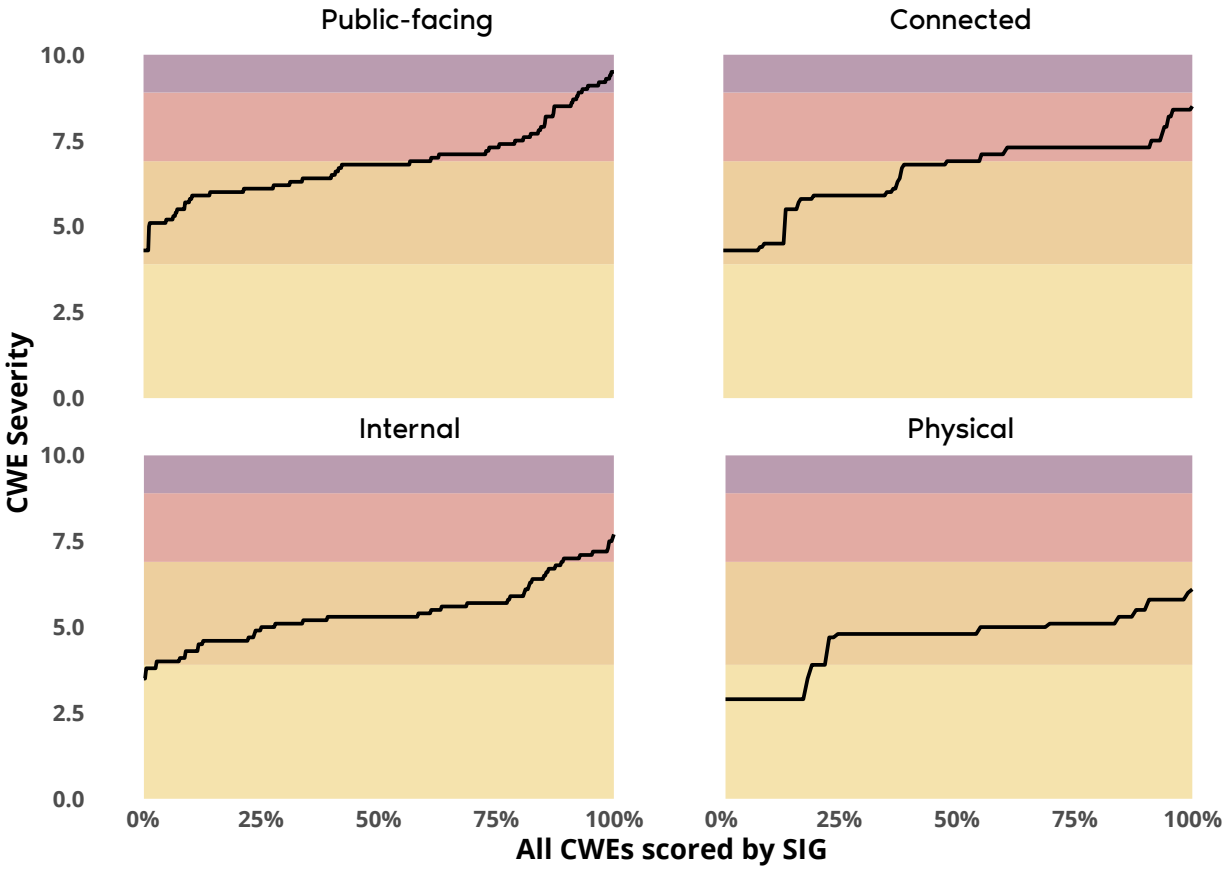
In the dataset plot we illustrate the dataset that we used to develop the SIG CWE benchmark. There are 97K CVEs included, dating from 2015 onwards, which are scored on impact, exploitability, and overall severity.

The key insight is that CVEs contain links to the software weaknesses that enabled the CVE in the first place. If only those weaknesses had been found and fixed before, right? To give an example, the following three weaknesses underly the (first) Log4Shell vulnerability CVE-2021-44228, which is scored 10 for critical severity:

- CWE-502 Deserialization of Untrusted Data
- CWE-20 Improper Input Validation
- CWE-400 Uncontrolled Resource Consumption

Simply speaking, seeing such weaknesses in software is often a cause for alarm. Fixes should be prioritized over weaknesses that are less correlated with severe vulnerabilities. In the full benchmark we weigh the final severity scores by all linked vulnerabilities, to provide a bit more nuanced scoring. In addition, we increase finding relevance by include context data related to the system in question.

DEPLOYMENT TYPE IS USED BY SIGRID® | SOFTWARE SECURITY TO FINE-TUNE WEAKNESS SCORES



SIG CWE Benchmark arranged by deployment type

The SIG CWE Benchmark is grounded on the tens of thousands of CVE scores provided by thousands of security experts. Yet, those CVEs are generic and not yet interpreted in the context of a client system or portfolio. It may turn out that some weaknesses are already mitigated for by specific provisions in the source code, network-level configuration, or through smart design choices.

Even if mitigations are already implemented, weaknesses in software need to be clearly flagged and documented. In the future, mitigations may go out of date and require adaption, which could re-enable existing weaknesses to become exploitable.

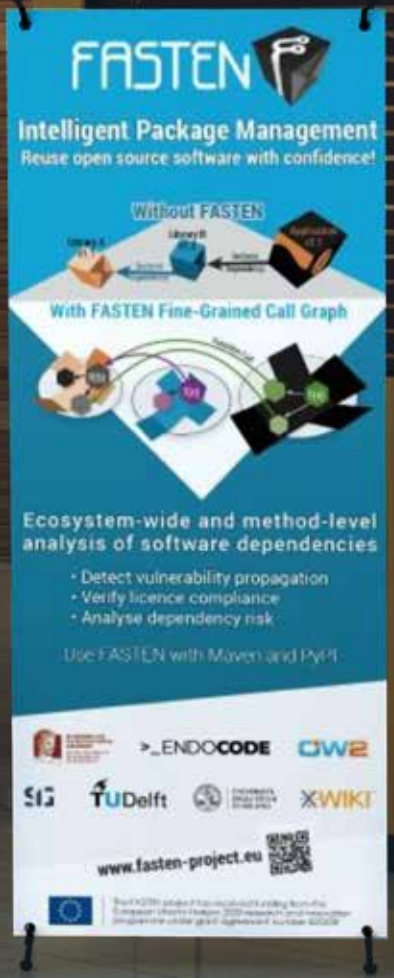
As shown in the graph, the SIG CWE Benchmark accommodates for the selection of a deployment type. That's an essential property of an operational software system; determining whether the public-at-large has access to the system, or just people logged in on the internal network, or whether even physical access to a system is needed to exploit a weakness.

In the Sigrid® | Software Security module, systems-under-analysis are assigned a deployment type to determine the overall severity of weaknesses found. The SIG CWE Benchmark then selects only the relevant CVEs to use in the scoring system, putting emphasis on the critical CVEs that can be exploited with public access, versus the low severity ones for which physical access is needed.

Using the deployment type mechanism, Sigrid® | Software Security can score and prioritize weaknesses across a portfolio of systems. Client teams can then focus on fixing weakness with the highest potential impact and exploitability.

LARGE-SCALE AND FINE-GRAINED SOFTWARE ECOSYSTEM ANALYSIS WITH FASTEN

Software ecosystems are massive evolving collections of (open source) software packages. Popular ones such as Maven (Java), PyPI (Python), and NuGet (.NET) include millions of packages, which each have a project behind them, with their own release schedules and dependency management strategies. Those packages are again re-used by millions of other pieces of software, both libraries and applications. Remember that modern applications rely on (open source) software packages for 80% or more of their actual code? How to find and maintain strong footing on such a sprawling quagmire?



The answer of FASTEN, an EU-funded project in which SIG collaborates with 5 European universities and companies, is to model software ecosystems in full detail, down to the level of individual units of code (e.g., Java methods) and their code level dependencies or call graphs. In the past 3 years, the project has developed a service that tracks dependencies at the method call-graph level and offers:

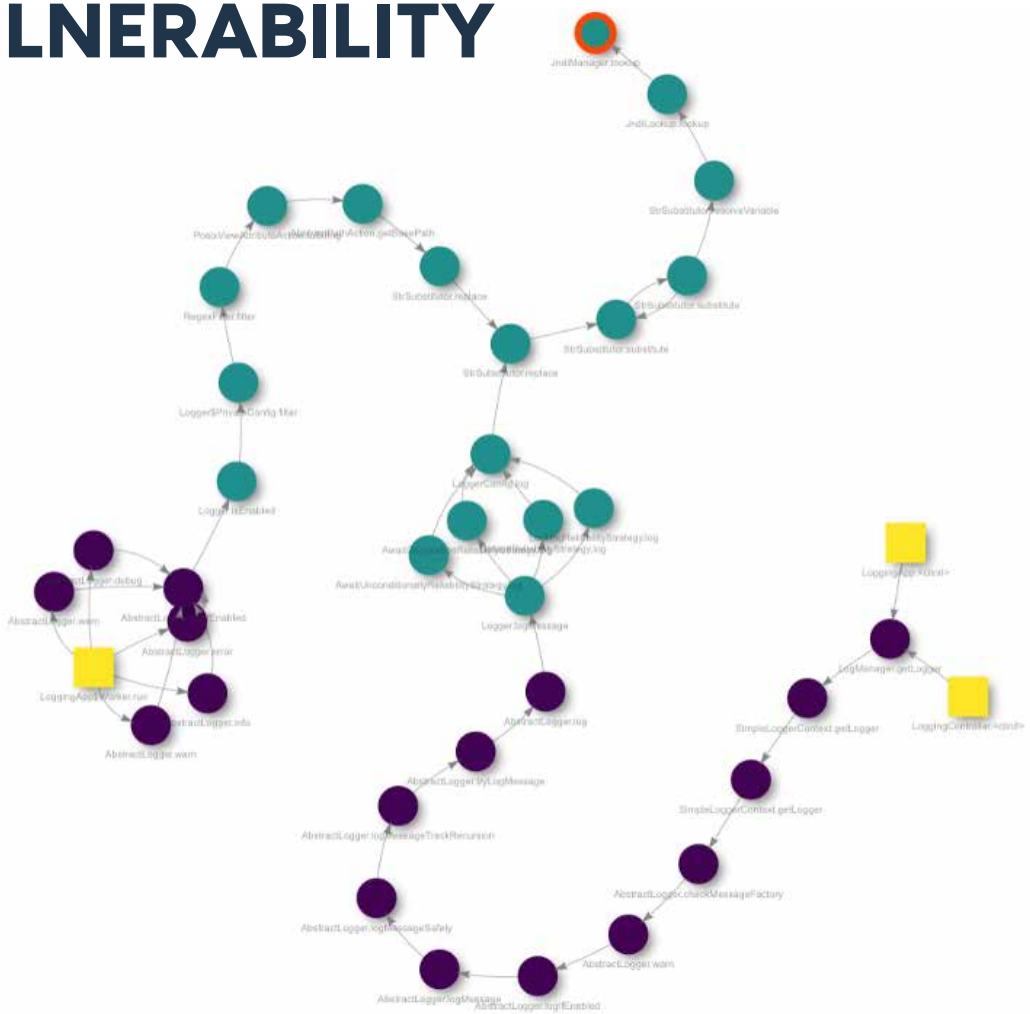
- Security vulnerability propagation across call chains and dependencies to allow focused remediation,
- Licensing compliance checking specific to the OSS code files that are in actual use,
- Quality risk profiles for the source code that an application relies on.

In the last three years, the project has implemented a series of tools and databases to support these analyses for the Java, C, and Python programming languages. Work is ongoing to add further ecosystems to the analysis toolset, such as NuGet (.NET), and Npm (JavaScript).

At SIG, we integrated several components of the FASTEN project into our software assurance platform Sigrid. For instance, the FASTEN vulnerability analysis is currently feeding the Sigrid Open-Source Health and Security modules. This integration helps Sigrid get ahead in reporting the relevant security vulnerabilities to our clients.

The FASTEN project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement number 825328.

FASTEN FINE-GRAINED ANALYSIS CAN TRACE CALL CHAINS FROM APPLICATION TO VULNERABILITY



FASTEN graph showing Log4Shell CVE-44228 vulnerability call chain

Let's showcase an example analysis that FASTEN can provide. In the graph we visualize a code base that was vulnerable to the December 2021 Log4Shell vulnerability. With FASTEN, we can trace the vulnerable call chains from an application (yellow boxes) through various methods in dependencies to the actual vulnerable code unit at the top (marked in red). Such call chains allows us to provide more accurate diagnosis and to suggest remediation efforts in a more actionable and detailed way.

Using the FASTEN tools and databases we also provide new empirical large-scale insights on the impact of low build quality. Starting with 10K of the most popular Java Maven package-versions among SIG's clients, we analyzed 21 million code units (Java methods) with code quality data. In addition, 11K of those units were found to be involved in vulnerabilities by the FASTEN tools.

Given these data, we can now cross-correlate code quality and vulnerability status:

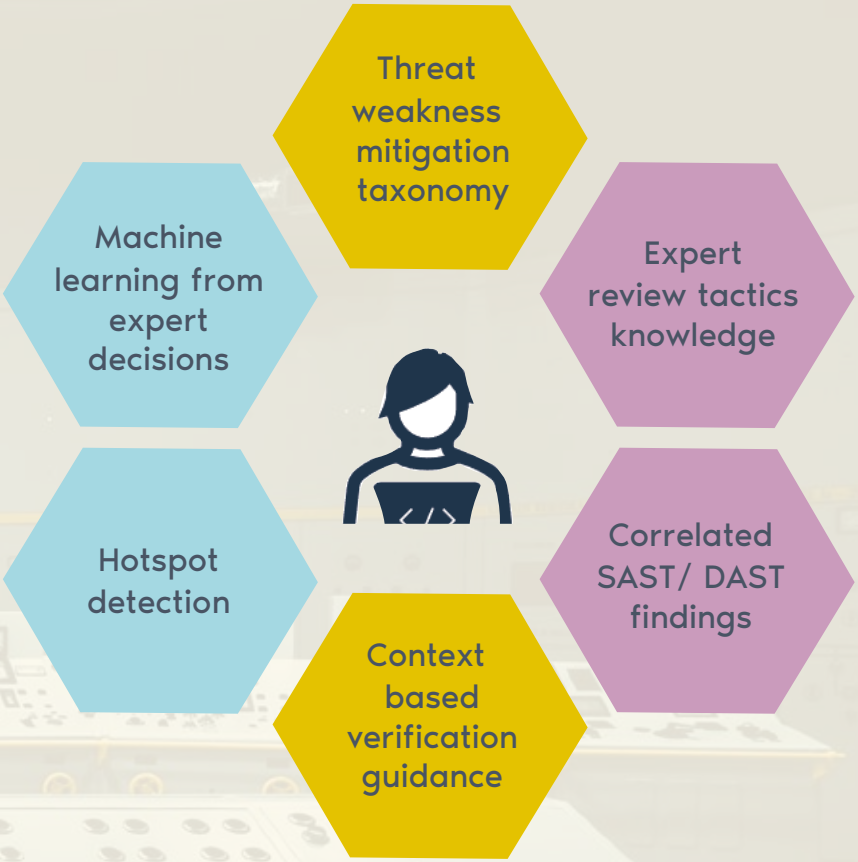
- The average code unit is around 8 lines of code, with 2 branch points and 1 parameter.
- The average vulnerable code unit has 25 lines of code, with 6 branch points and 2 parameters.

So, the average vulnerable code unit is indeed above the SIG recommended low risk thresholds for both code size (15 lines of code) and branch point complexity (5 branch points). Obviously not all units with poor code quality are vulnerable, but they may indeed be at greater risk of becoming vulnerable in the future.

To further test this finding we trained a multinomial logistic regression algorithm that predicts vulnerability based on code quality metrics. This initial attempts reaches 67% accuracy using just the metrics lines of code, branch points, and parameter count. This suggests a promising research direction with potential accuracy gains to be made by adding more quality information.

SCRAMBLE - SMART CODE REVIEW ASSISTANCE MODULE BLENDING LEADING EXPERTISE

- Manual secure code review is an essential part of shifting left and security by design. After all, many types of weaknesses cannot be found using tools alone.
- This requires effort from secure code review experts in SIG's lab. Their expertise is very rare and because of the success of SIG's platform and its services, the scalability of this skill is of the greatest importance.
- Our strategy is to harness the expertise of our experienced code reviewers and the results of our research into an intelligent system to assist code review.
- Under government funding and in collaboration with research institutes and academia, SIG is developing SCRAMBLE - Smart Code Review Asistance Module Blending Leading Expertise, to make code/design review more consistent, efficient, and feasible for a larger group of people – within SIG, but also at clients and at partners. This addresses one of the key problems in software security: shortage of experts.



SCRAMBLE

Smart Code Review Assistance Module
Blending Leading Expertise

HOW TO SHIFT-LEFT ON SECURITY AND SOFTWARE SUPPLY CHAIN RISKS TODAY?

This edition of the Benchmark Report has provided several outcomes that urge for action. In general, there is a strong need for a greater sense of urgency in the enterprise software field regarding software supply chain issues. Security vulnerabilities really are popping up left-and-right at increasing pace. While this report is being written, the Spring4Shell vulnerability emerged, with similarly critical severity as the Log4Shell incident of 2021. And it's only March at the time of writing! It really feels like we are mopping up the floor with the tap still running.

- In the medium to long term, this issue should be addressed at the level of (government) policy. Industry should be required to comply with higher standards – standards which are applied in practice and which the (cyber) security community is working to further improve, but all-too-often still have an optional status in actual development.

- On the short term, besides the general advice for raising urgency, we recommend to act upon the following points today, or maybe tomorrow:

- Implement full transitive dependency analysis to become aware of all publicly reported issues,

- Sharpen up dependency version update policies to prevent unmitigated use of stale and vulnerable versions,

- Review the actual in-use dependencies by considering whether they are essential or largely redundant,

- Discover any low build quality dependencies and consider if they are worth the risk of more vulnerabilities,

- Ensure that build quality of your own software is up to standards (4 stars) to lower the risk of becoming the next vulnerability incident to hit the papers,

- Perform automated code security scanning on all code to find potential weaknesses before they become exploitable,

- Introduce tool-supported security code review for critical systems that addresses security-by-design, common weaknesses in code, mitigation strategies, and unsafe usage of APIs or dependencies.

POINTS OF ACTION

Software engineering is about continuous and never-ending improvement. Software is never perfect, or all-too-often, even good enough. The world changes, but so do the tools and techniques that are available. We are happy to see that many engineering improvements find their way into the enterprise software domain and make a measurable impact.

Having said that, we must remain critical and urge attention for our findings on third-party libraries usage in the enterprise software domain.

1

ORGANIZATIONS MUST INVEST, TODAY, RATHER THAN TOMORROW, IN SHIFTING-LEFT ON THEIR SOFTWARE SUPPLY CHAIN USAGE. THIS CAN BE ACCOMPLISHED BY INCREASING VISIBILITY OF ISSUES DEEP-DOWN IN SOFTWARE ECOSYSTEMS AND BY ENABLING TEAMS TO QUICKLY ADDRESS THEM. THE BUILD QUALITY OF THIRD-PARTY LIBRARIES IS OF EQUAL CONCERN FOR ITS USERS AS FOR ITS OWN DEVELOPERS, AS WE HAVE SHOWN IN THIS REPORT. IT MEANS ALSO THAT INVESTMENT IN THE UPSTREAM (OPEN SOURCE) PROJECTS CAN BRING GREAT BENEFITS; SINCE THAT'S WHERE MOST OF THE SOFTWARE WE RELY ON THESE DAYS IS BEING BUILT.

AS SHOWN IN OUR YEARLY TECHNOLOGY STACK RANKING, **SOFTWARE BUILT IN LEGACY TECHNOLOGY** IS, QUITE LITERALLY, STILL THE ELEPHANT IN THE ROOM. TECHNOLOGIES LIKE LOW CODE AND OTHER MODERN PROGRAMMING TECHNOLOGIES OFFER BUILD QUALITY IMPROVEMENT POTENTIAL YET PROVIDE NO IMMEDIATE SOLUTION TO THE OFTEN-RISKY AND COSTLY TASK OF REPLACING THE OLD SOFTWARE ELEPHANTS. A LOT OF ENGINEERING GRUNT WORK AND CAREFUL MIGRATION PLANNING IS STILL NEEDED TO ACCOMPLISH THIS; WHICH MANY ORGANIZATIONS ARE HARD PRESSED TO EXECUTE BY THEMSELVES.

2

3

This edition of the Benchmark Report is concluded with a brief summary of our Research Vision for the coming years. We will be happy to report our progress to you in the coming year and in the next edition of the Benchmark Report.

AS AN INDUSTRY WE ARE FACING A PRESSING SHORTAGE OF SOFTWARE SECURITY EXPERTISE; THERE IS SIMPLY TOO MUCH SOFTWARE AND TOO FEW SECURITY EXPERTS TO GO AROUND. THE SOFTWARE BEING DEVELOPED TODAY IS THEREFORE AT RISK OF MISSING SECURITY REQUIREMENTS IN DESIGN, PROCESS, CODE CONSTRUCTION, OR IMPLEMENTATION. WITH OUR R&D WE ARE DEVELOPING NEW SUPPORT TO MAKE THE JOB OF SECURITY REVIEW EASIER AND MORE EFFECTIVE – AT THE SAME TIME WE LABOR FOR IMPROVED SECURITY TRAINING AND KNOWLEDGE SHARING WITH THE COMMUNITY.

TOGETHER WITH SIG ORGANIZATIONS CAN BE CONFIDENT IN THE SOFTWARE THEY RELY ON EVERY DAY

From top to bottom, all employees at any organization, irrespective of industry, use software applications during their working day. Technology is now prevalent and directly impacts productivity, efficiency, revenue, and success. This high dependency means complete software assurance is now indispensable. Organizations can only achieve this by considering the full spectrum of software: the quality of the product(s), the development processes, the proficiency of the teams, and the environmental impact. Software Improvement Group (SIG) Research will be investigating these themes to help organizations achieve a healthier digital world.

Gartner recently forecasted the global 2021 spend on enterprise software will surpass 600 Billion USD, with a projected growth rate of 12% for 2022. To put that number into context a 12% growth rate means spending would double every six years – Software is still eating the world!

Projected growth is not just a matter of increased scale, software analysts and technology leaders see an increase of complexity coupled with a higher bar of competition from start-ups and scale-ups. Cloud- and edge computing, microservice architecture, and digital transformation, are continuing challenges. Shifting-left on cybersecurity (DevSecOps), further automation of enterprise (data) architecture, and the integration of machine learning and AI technologies into mainstream production (AIOps) are of increasing priority. Rising energy costs are also forcing organizations to accelerate Green IT initiatives that improve the sustainability of data centres and applications.

SIG Research performs scientific and applied research to increase the capability of our software assurance platform and enable the enterprise to have confidence in their applications to reduce costs and accelerate growth. The increasing complexity of enterprise software demands novel ideas and approaches for offerings to remain competitive. SIG Research's mission is to increase the flow of new ideas, test and validate them in an applied context, and contribute to the public body of knowledge on software engineering.

Get the full document at:

Acknowledgements

This report was compiled through a team effort across the SIG organization. Thank you for all your contributions, helpful suggestions and review comments: Lodewijk Bergmans, Luc Brandts, Clarinda Dobbelaar, Chushu Gao, Corina Kuijlen, Nick Potts, Rob van der Veer, Frédéric Wolff, Femke van Velthoven and Miroslav Zivkovic.

Full-spectrum software assurance research:

- **Integrated models of software product, process, and people**
- **Shared data platform linking software data from all angles**
- **Intelligent automation of machine-learnable software analysis**
- **Precise online monitoring of global software ecosystems**
- **Sustainable software development and efficient operation**

Design
Plushommes.com

About Software Improvement Group

Software Improvement Group (SIG) helps organizations trust the technology they depend on. We've made it our mission to get software right for a healthier digital world by combining our intelligent technology with our human expertise to dig deep into the build quality of enterprise software and architecture - measuring, monitoring, and benchmarking it against the world's largest software analysis database.

With SIG software assurance, organizations can surface the factors driving software total cost of ownership and make fact-based decisions to cut costs, reduce risk, speed time to market, and accelerate digital transformation.

Software Improvement Group is the first fully certified laboratory in the world to measure against the ISO 25010 standard. We make this lab accessible to our clients through our SaaS software assurance platform – Sigrid – which enables them to take a risk-based approach to improving the health of their IT landscapes.

We serve clients spanning the globe in every industry, including DHL, Philips, ING, KLM, BTPN, Weltbild, KPN, as well as leading European governmental organizations.

SIG was founded in 2000 as an independent technology company with embedded consulting services. SIG is headquartered in Amsterdam, with offices in New York, Copenhagen, Antwerp and Frankfurt.

Learn more at www.softwareimprovementgroup.com.

Fred. Roeskestraat 115
1076 EE Amsterdam
The Netherlands

www.softwareimprovementgroup.com
marketing@softwareimprovementgroup.com