Techstrong Research

PulseMeter

Sponsored by snyk

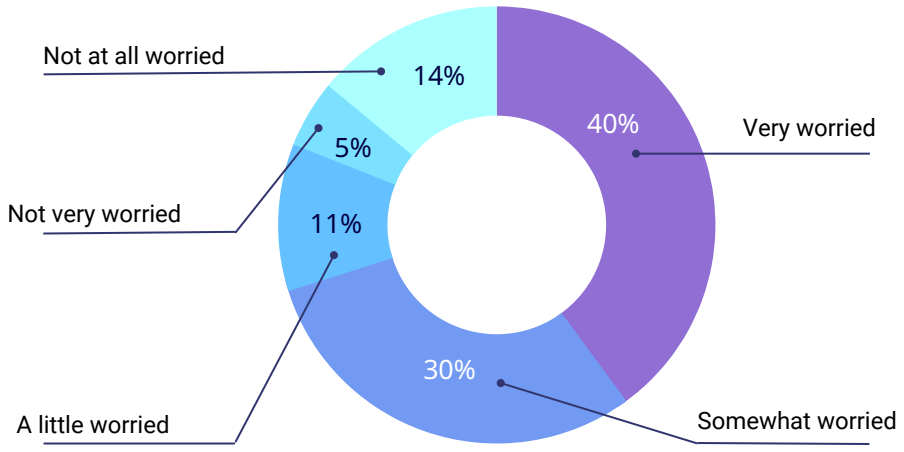
Software supply chain attacks are top of mind and with good reason. Not only are these attacks extremely high profile but they can cause lasting damage. Dealing with the SolarWinds attack and the Log4Shell vulnerability still brings up difficult memories long after the initial incidents. To detect these attacks and defend against them, organizations can scan first-party code, third-party components, containers, and other pipeline tools that manage the code-to-cloud process. Another tactic, recently mandated by the U.S. federal government, is the software bill of materials. SBOMs itemize the components embedded within code, allowing users and consumers of these applications and software packages to track potential exposure to existing and not-yet-discovered vulnerabilities within those components. There are also runtime application security defenses that can detect and block attacks.

In late 2022, Techstrong Research polled our community of DevOps, cloud-native, cybersecurity, and digital transformation readers and viewers to take their pulse on SBOMs. We found that 42% of respondents produce SBOMs for at least 40% of their applications, while only 25% don't produce SBOMs at all. Almost half (49%) indicated they were analyzing first-party code, third-party dependencies, and infrastructure components. Overall, testing is reasonably widespread. This diligence is essential given the increasing adoption of open-source software, which respondents say they select based on functionality first (25% of respondents), while the security of the project is a close second at 22%. But SBOM is not the only tactic used to handle security within an open source project, as respondents monitor vulnerability reports (25%) and whitelisting packages (24%).

Software Supply Chain Attacks Identified as Top Concern

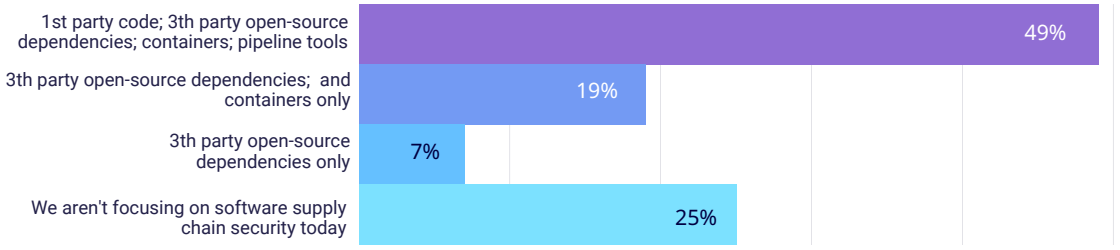
Are you worried about software supply chain attacks?

These attacks are high profile and potentially very damaging.
Over 78% of respondents are at least somewhat worried, as they should be.



Which of these do you include as part of your software supply chain security analysis today?

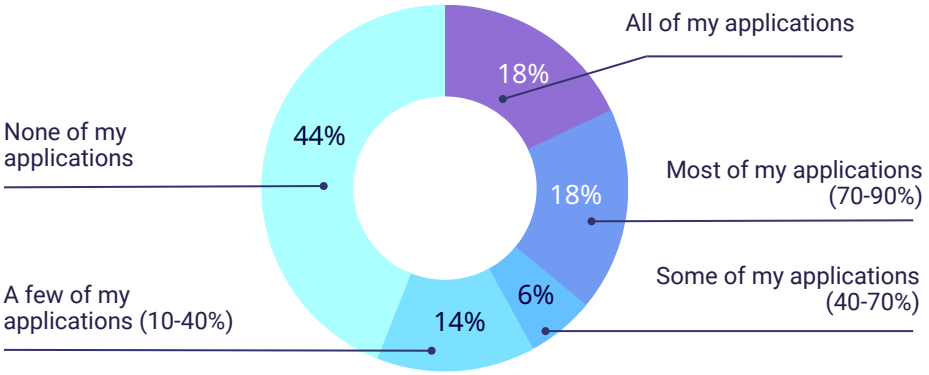
Almost 50% of respondents comprehensively analyze their code and the third-party and open source libraries that they leverage, which gives them a fighting chance given the complexity of today's applications. Yet, 25% don't focus on this today, evidently figuring the problem will disappear. It won't.



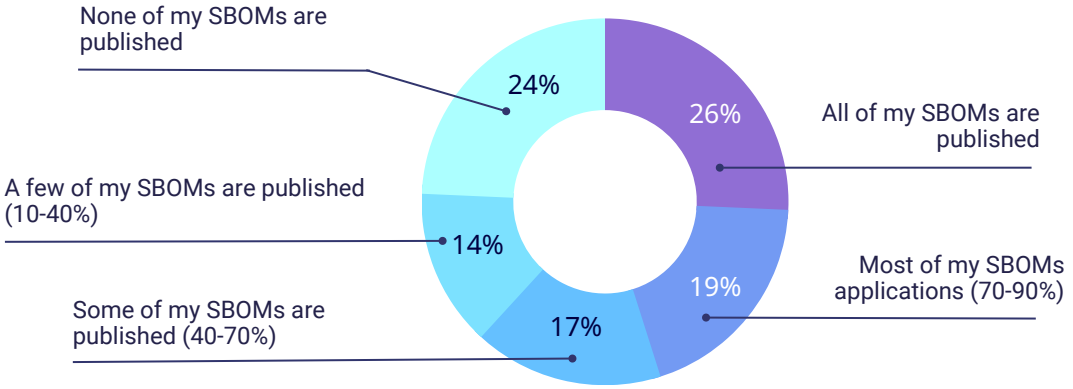
Over Half of Respondents Already Use SBOMs

SBOMs will be required for any software bought by the U.S. government, and over 55% of respondents already produce at least a few, with 18% publishing SBOMs for all applications with another 18% of respondents stating that they produce SBOMs for most apps (70%-90%). Not all the SBOMs produced are published, but we expect production and publishing to increase over the next year as customers increasingly require published SBOMs.

Do you currently produce software bills of materials (SBOMs) for your applications?

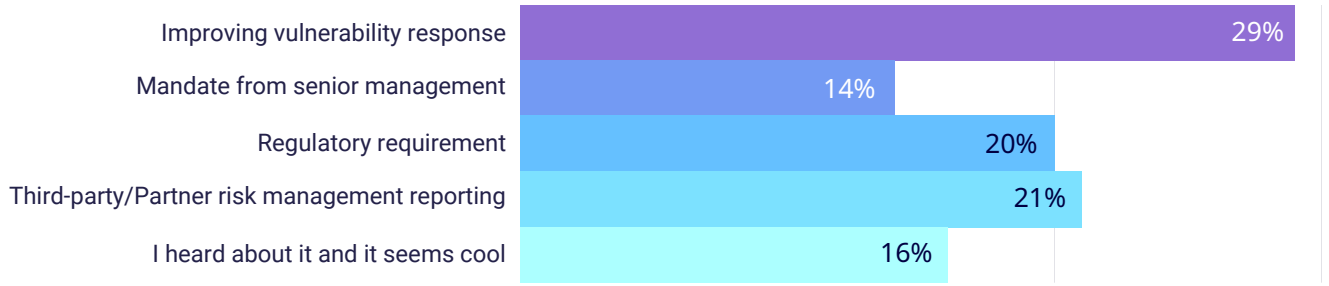


Of those SBOMs produced, how many are published?



What are the main drivers for using SBOM?

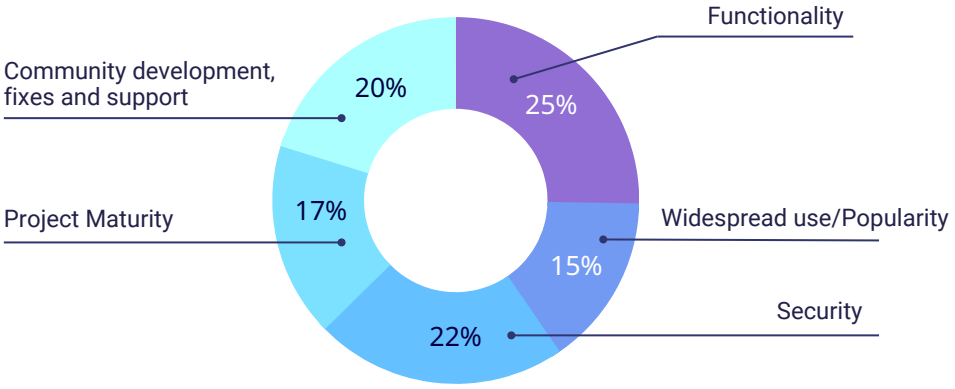
The drivers for SBOM are split between security (improve vulnerability responses) and compliance/reporting, which reflects the historical drivers for security solutions.



Security and Automation Key to Open Source Success

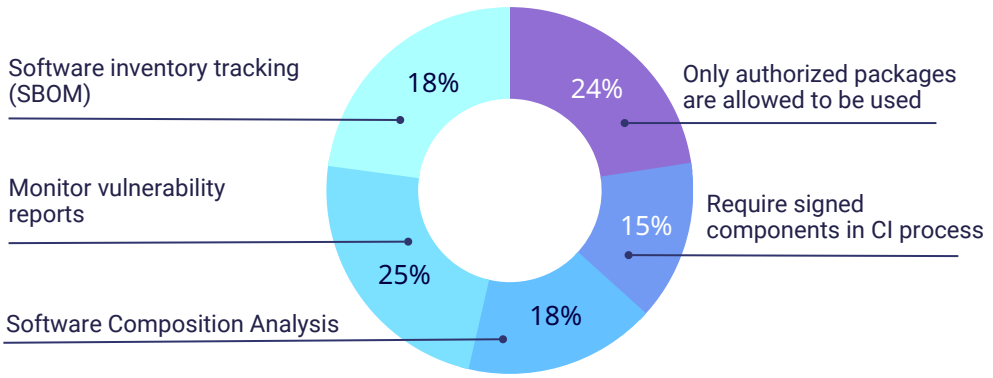
What are the key factors considered in your selection of open-source packages?

You can't use open source software that doesn't work (evidenced by the 25% that prioritize functionality), but it also needs to be secure (22%) and supported by a strong community (20%).



How do you secure the open-source software used in your applications?

Securing open source is still largely a manual process; monitoring for vulns (25%) and only allowing authorized code (24%). More scalable and automated functions like SCA (18%) and requiring signed components in CI (15%) lag a bit.



Techstrong Research Analyst View

Numerous recent high-profile software supply chain attacks brought the issue of vulnerable packages into the spotlight. Regardless of whether you believe components integrated into your code make up a large or a small component of the software you deploy, you need to pay attention to the security of any code you didn't write, whether it's imported into the code as a third-party dependency, or if it's brought in from a base container image. We expect organizations to do much more to scrutinize these components, given recent guidance that mandates SBOMs for US Federal customers, but also enterprises that follow the lead of the Federal Government.

Organizations should use multiple tactics to secure software, including actively scanning the code, monitoring for vulnerabilities in the components listed in the SBOM, and requiring only signed packages from trusted sources to be integrated. The software bill of materials provides critical information about what lurks within the code, allowing the assessment of these components. This improves vulnerability response by allowing developers to quickly identify vulnerable versions of components and in conjunction with the right developer security tooling, to quickly mitigate at-risk components where they are introduced. We believe publishing an SBOM is necessary but not sufficient on its own to allow organizations to secure their code. Developers need awareness and guidance to choose less risky packages as they code, and organizations also need a continuous process to track and fix new vulnerabilities in the components (e.g., Log4J).

The bottom line is that software supply chain security requires a multi-faceted approach. These security controls must happen at all stages of the software development and delivery pipeline and must be a joint effort between security and development. After years of application security being marginalized by developers preoccupied with shipping code faster, we will soon see these organizations mandate the shipping of secure code.