Software Bill of Materials (SBOMs)

Reducing Open Source Risk Throughout the Development, Delivery and Deployment of Software

Authored By:

Jim Routh
Esteemed cybersecurity expert and former CISO and CSO at MassMutual, Aetna and CVSHealth

Vince Arneja
Chief Product Officer at GrammaTech

Executive Summary

The Apache Log4j vulnerability exposed a massive software supply chain weakness in thousands of software applications. The prevalent use of open source components in software is creating significant risk. Whether you are developing software to support internal organizational use, delivering software for your customers to consume, or deploying third party software across your enterprise to provide business functionality, risk from open source components in software must be managed. If open source risk continues to go unmanaged, your organization could face brand and reputational damage, financial loss and regulatory penalties from increasing cyberattacks targeting vulnerable and popular software components.

For developers, the days of working in the dark by trusting external code being risk free are over. The same goes for software vendors delivering products to customers and organizations deploying software to support their businesses. Vulnerable open source components in software, as a cybersecurity blind spot historically, has to be addressed and fixed. The need to "trust, but verify" software at all stages of its lifecycle is required to reduce open source risk throughout the software supply chain.

This paper will shed light on an emerging critical software supply chain weakness and how generating software bills of materials can be the foundation to improving security at multiple stages of the software lifecycle.

Table of Contents

What is a SBOM and Why Is It Important?

COVID exposed the world to what a supply chain is and the economic issues that arise when the chain is broken. In simplistic terms, a supply chain includes all the activities required to deliver goods or services to the consumer. From sourcing to manufacturing to deliver of goods and services, managing the supply chain is critical to guarantee consumers receive quality goods and services on time.

An important part of supply chain management is the bill of materials (BOM) that defines the raw materials and components – the ingredients – of the finished product typically delivered by the supplier. Organizations rely on the BOM to ensure they are producing and delivering products that meet quality, integrity and safety standards for their customers. If a problem arises in a product, the documented BOM can help the organization pinpoint the root cause of the issues as detailed as the failure of a bolt. Decisions can then be rapidly made to fix the problem, source new parts or change vendors to not disrupt the supply chain process.

So, what is a software bill of materials (SBOM) and why is it important?

The NTIA defines a "Software Bill of Materials" (SBOM) as a nested inventory for software, a list of ingredients that make up software components. NTIA SBOM resources available at www.ntia.gov/SBOM. Additional, SBOM resources from CISA: www.cisa.gov/sbom.

Think of a SBOM in the same way as a physical BOM. It is the comprehensive list of ingredients (software components) that comprise the final software package through the software supply chain from development to delivery to deployment. Essentially, software development organizations are manufacturing software the same way physical goods are produced and delivered. In this case, the raw materials and components are code the developers custom build coupled together with sourced third-party code and open source libraries to create software.
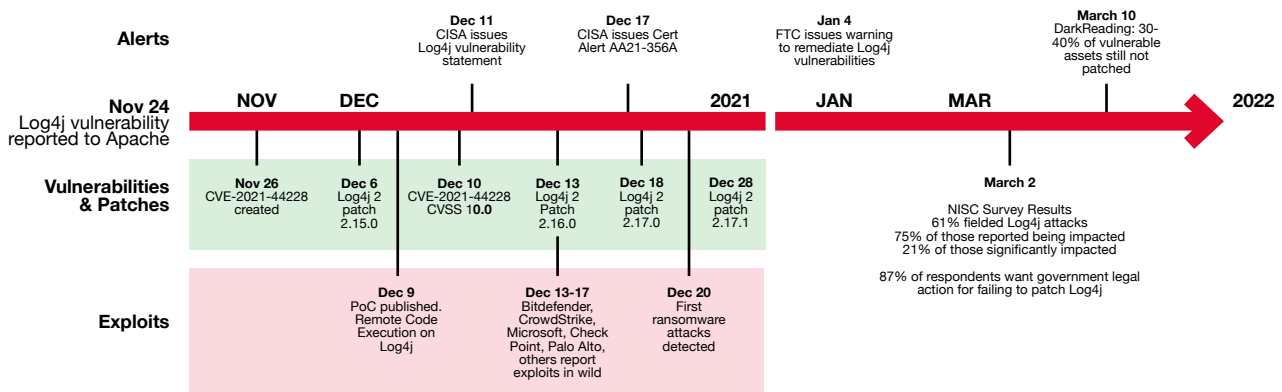
By knowing what is in the software, developers, IT professionals and cybersecurity experts can make more intelligent decisions and take immediate action when vulnerabilities are discovered in certain components used within the software. With software security, speed is of the essence as cyber attackers are quickly ready to exploit known vulnerabilities. Once a critical vulnerability is exposed, it is only a matter of time until it is being actively exploited.

For developers and cybersecurity professionals, the need to manage risk and remediate vulnerabilities becomes a race to protect the software from attack. The recent Apache Log4j open source library vulnerability highlights a software supply chain with a material defect and why SBOMs are now needed more than ever.

Log4j Vulnerability Puts a Spotlight on SBOMs

Without access to SBOMs, development, IT and cybersecurity teams struggled to quickly identify assets affected by the Zero-Day Log4j vulnerability and take immediate actions to defend against attacks. The sense of urgency and the demand for SBOMs to help protect against open source vulnerabilities like Log4j and improve mitigation efforts and incident response times is highlighted in this Log4j timeline.

Log4j: Immediate Impact and a Long Tail Software Supply Chain Problem

Alerts			**Dec 11** CISA issues Log4j vulnerability statement		**Dec 17** CISA issues Cert Alert AA21-356A		**Jan 4** FTC issues warning to remediate Log4j vulnerabilities		**March 10** DarkReading: 30-40% of vulnerable assets still not patched	
Nov 24 Log4j vulnerability reported to Apache	**NOV**	**DEC**				**2021**	**JAN**	**MAR**		**2022**
Vulnerabilities & Patches	**Nov 26** CVE-2021-44228 created	**Dec 6** Log4j 2 patch 2.15.0	**Dec 10** CVE-2021-44228 CVSS 10.0	**Dec 13** Log4j 2 Patch 2.16.0	**Dec 18** Log4j 2 patch 2.17.0	**Dec 28** Log4j 2 patch 2.17.1		**March 2** NISC Survey Results 61% fielded Log4j attacks 75% of those reported being impacted 21% of those significantly impacted 87% of respondents want government legal action for failing to patch Log4j		
Exploits		**Dec 9** PoC published. Remote Code Execution on Log4j		**Dec 13-17** Bitdefender, CrowdStrike, Microsoft, Check Point, Palo Alto, others report exploits in wild	**Dec 20** First ransomware attacks detected					

Compelling Events Driving Need for SBOMs

- *Software Supply Chain Attacks on the Rise*
    The critical Zero-Day vulnerability in the popular open source Apache Log4j library put another spotlight on a broken software supply chain. As organizations struggled to find what software contained this vulnerable open source library, the need for SBOMs significantly increased.
- *Presidential Cybersecurity Executive Order*
    After the SolarWinds attacks, improving software supply chain security became one of the top initiatives of the recent cybersecurity executive order. Software vendors working with the U.S. Government will be required to provide SBOMs.
- *Industry SBOM Compliance Requirements*
    Governing bodies in specific industries are now starting to require SBOMs from software vendors. As an example, the FDA is requiring medical device manufactures to produce SBOMs as go-to-market prerequisite

SBOMs – The Foundation of Software Supply Chain Security

The SBOM is foundational to improving software supply chain security from development to delivery to deployment. By knowing what is in the software, software developers can ensure more resilient software and and cybersecurity and IT professionals can gain insight into the software the organization is using to better identify mitigate risk.

Through every stage of the software lifecycle, generating a machine readable SBOM will provide software visibility enabling you to:

Discover
- Identify open source components in third-party code and COTS/third-party software
- Detect known (N-day) and unknown (Zero-day) vulnerabilities in those components

Manage
- Make more intelligent security decisions based on visibility into code/software
- Adhere to security and vendor risk compliance requirements

Remediate
- Protect against cybersecurity threats with actionable vulnerability intelligence
- Streamline vulnerability remediation to mitigate software risk

Applying SBOMs to Improve Software Security

The most common enterprise cybersecurity practice is a layered approach generally starting with perimeter defenses to keep cyber attackers out. A similar layered approach can be applied to ensure software security starting in development through to delivery and deployment. In this approach, the basis for software security is rooted in development. With a software quality focused approach, software developers following best practices for DevSecOps can better ensure code resiliency. As the software is prepared for release, security assurance testing can verify the software meets security requirements that are part of software quality. And, as software is deployed in organizations, software development leaders, DevOps leaders and cybersecurity professionals can analyze software to gain visibility into the specific software components that are vulnerable and creating risk. In the next section, we will show you how SBOMs can be applied in software development, delivery and deployment to improve software security and quality.
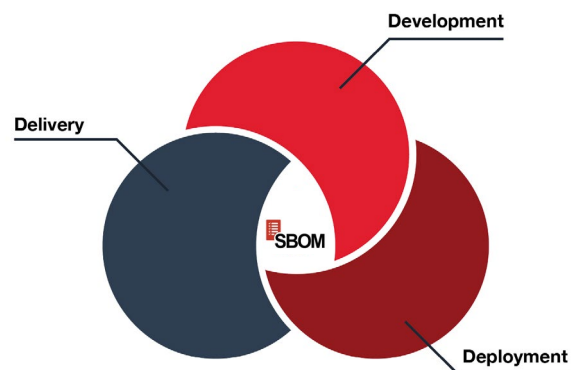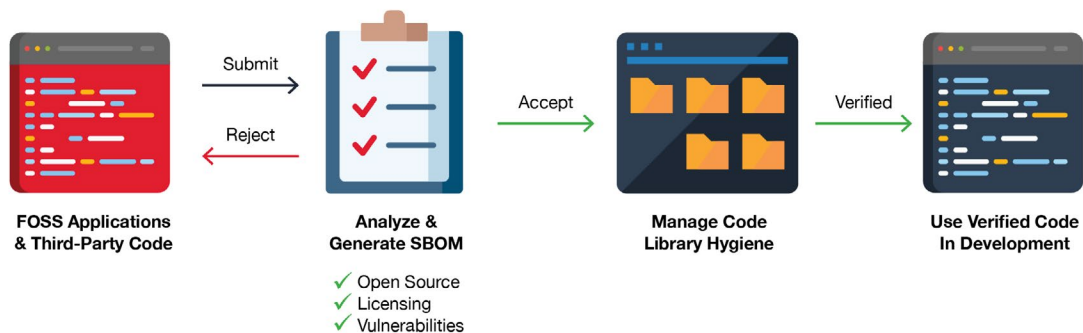
SBOMs for Software You Are Developing

Developing software solely from custom code is not practical nor efficient as developers are under tremendous pressure to build and release software quickly. Because of this, software development teams are increasingly using more open source and third-party code for standard functionality that does not need to be custom created. The use of open source and third-party code helps to improves developer productivity and output, but risk needs to be managed to produce quality and resilient software.

In most cases, software development teams maintain and manage code repositories and libraries from which they reuse open source and third-party code in projects. While DevSecOps best practices integrate static application security testing (SAST) into development workflows to find and fix issues in custom code, many development teams still blindly trust open source and third-party code.

Proper vetting of these components should include generating SBOMs for third-party code and open source libraries to identify the compositions and understand dependencies that may not be apparent at the surface level. Generating SBOMs at this stage will enable development teams to gain more visibility into the components they are using. With this visibility, they can use SBOM results to further detect vulnerabilities that may be hiding in these components and ensure they are using updated and properly licensed versions.

Maintaining good open source and third-party code hygiene is a must to develop software that meets quality and security standards. By regularly analyzing open source and third-party code and generating SBOMs, development teams can confidently manage their component libraries with a focus on improving security.

FOSS Applications & Third-Party Code → Submit / Reject → **Analyze & Generate SBOM** ✓ Open Source ✓ Licensing ✓ Vulnerabilities → Accept → **Manage Code Library Hygiene** → Verified → **Use Verified Code In Development**
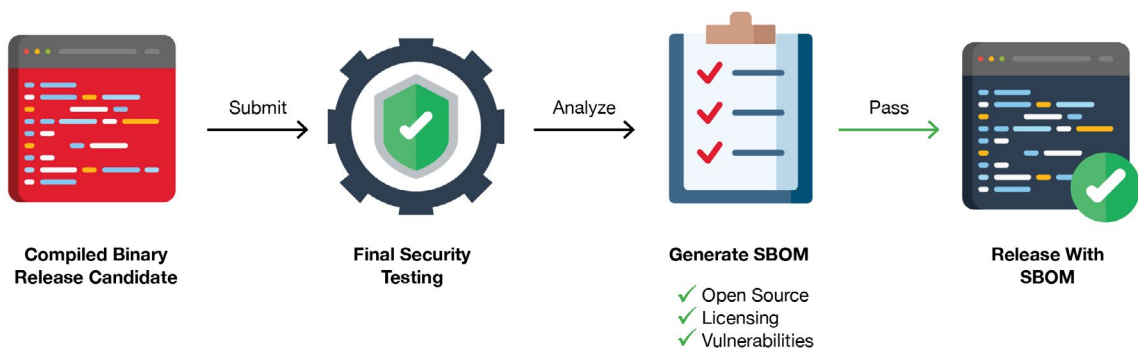
FACT

VDC Research study shows in-house custom code is shrinking. In the Enterprise market, 80% of the decline of custom code is going to commercial/third-party code with 20% is going to open source. For the Embedded market, 56% of the decline has gone to open source with 44% shifting to commercial/third-party code.[1]

SBOMs for Software You Are Delivering

For in-house development teams, independent software vendors and embedded software devices, delivering software that meets security requirements is more essential than ever now. Too much software is being released today with risky components and potentially hidden vulnerabilities that put the business, users and customers at significant risk. Before delivering software, stringent security assurance practices must be implemented to ensure the software being released is free of vulnerabilities as possible and meets security standards.

After software code in development passes through QA testing, the code is then compiled as a binary and prepped for packaging, release and deployment. This is where a final software security assurance check can and should be implemented. With the code in binary form, scanning the compiled code to produce a SBOM can identify the use of open source components in the "final" software package. Additionally, a final vulnerability analysis of the identified open source components can check for any Zero-day and N-day vulnerabilities that may be hiding in these components, so they can be fixed/remediated. This software security assurance step provides a final validation to ensure that software that is heading to the commercial market, released into production or embedded into devices does not contain open source components with exploitable vulnerabilities.

SBOMs will soon become a requirement when delivering software and products to market. With a focus on improving the nation's cybersecurity, the 2021 Presidential Cybersecurity Executive Oder highlighted a software supply chain security challenge and specifically called out the need for SBOMs. If software vendors provide software to the U.S. government, they will soon be mandated to provide an SBOM upon software delivery. Other highly regulated industries are also now beginning to make SBOMs a requirement when delivering products such as medical devices and critical infrastructure controls. Providing a SBOM to customers will deliver visibility into the open source make-up of the software so they can react quicker to vulnerabilities such as Log4j.

Compiled Binary Release Candidate → Submit → **Final Security Testing** → Analyze → **Generate SBOM** (✓ Open Source ✓ Licensing ✓ Vulnerabilities) → Pass → **Release With SBOM**

FACT

A recent Osterman Research report found that 100% of analyzed commercial off the shelf (COTS) software contained open source components with vulnerabilities. 85% of those applications contained critical (CVSS 10.0) vulnerabilities in open source components.[2]
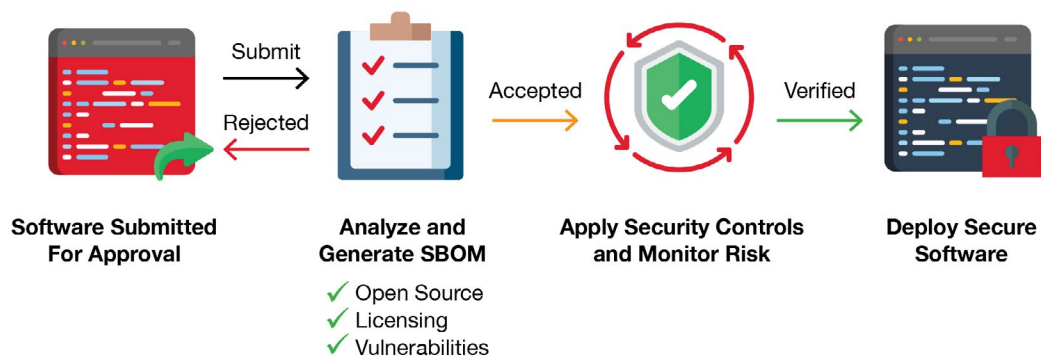
SBOMs for Software You Are Deploying

Enterprise organizations deploy a tremendous amount software to support and power their business. The annual spend on enterprise software is now approaching $600 billion with a growth rate of 11.5%.[3] From critical business software (financial, communication, productivity) to software embedded in devices (printers, routers, IoT), the threat landscape is increasing and with much of today's software now Internet connected it is even easier to exploit software vulnerabilities.

As seen above, the recent Osterman Research report found 100 percent of widely used commercial-off-the-shelf software contained exploitable vulnerabilities[2]. The research results highlight a serious software supply chain blind spot that is exposing organizations to the risk of damaging cybersecurity attacks targeting software vulnerabilities. The Log4j vulnerability is a perfect example of critical vulnerability in an open source component found in many widely used software applications that put thousands of organizations on alert of being cyberattacked.

While software developers and vendors will over time do a better job developing and delivering secure software, enterprise cyber defenders must do a better job ensuring the software being used is meeting security and resiliency requirements. This process can start by trusting, but verifying the security of the software provided by vendors. With the ability to generate their own SBOMs by analyzing software binaries, enterprise information security teams can benefit from visibility they have never had into the make-up of the software the organization is either evaluating or already using and what vulnerabilities exist.

The visibility into software and the information provided in the SBOM can enable cybersecurity teams to proactively improve their software security posture, make more intelligent software security decisions and speed threat forensics when another open source vulnerability like Log4j is discovered.

Software Submitted For Approval → Submit / Rejected → **Analyze and Generate SBOM** → Accepted → **Apply Security Controls and Monitor Risk** → Verified → **Deploy Secure Software**

✓ Open Source
✓ Licensing
✓ Vulnerabilities

FACT

When a Forrester Research Survey asked, "how was the external attack carried out?", 30 percent of cybersecurity professionals answered "software vulnerability" with the number predicted to keep increasing.[4]

Value of Binary Software Composition Analysis to Create SBOMs

Software composition analysis (SCA) is the most common technology used to identify the make up or composition of software and generate a SBOM. There are few different technologies to conduct SCA, mainly source code and binary analysis. While both have their place and advantages, our focus here is on binary SCA, its application and value.

Unlike source code SCA solutions, binary SCA solutions analyze compiled code, or the final software product that is being delivered to and deployed by customers and users. The advantage of binary SCA solutions is that analysis can be run on software without access to the actual source code. This enables new applications to analyze software to identify what is in it and produce a SBOM – complementing the use of source code SCA in software development.

In the SBOM applications described above, binary SCA provides visibility into open source and third-party software that source code SCA simply cannot. The value of binary SCA is that these components, libraries, packages and products can be scanned to create a SBOM—helping to determine composition and risk (vulnerabilities and licensing) that could be hiding in the software. The SBOM can be used by developers, cybersecurity teams and IT professionals to make more intelligent risked-based decisions on whether to accept or reject software to improve the overall security of the software they are developing, delivering or deploying.

New software products available today deliver unprecedented visibility into the software supply chain for applications being developed and/or consumed without needing access to source code. By automating binary analysis, these products identify open source components in third-party software, detects N-Day and Zero-Day vulnerabilities and generates a detailed software bill of materials (SBOM) and vulnerability reports. Resulting application intelligence and vulnerability visibility mitigates risk, improves software security and strengthens enterprise security posture to defend against software supply chain attacks.

Sources and Resources

1. VDC Research Study: Finding Sources of Security in the Complex Software Supply Chains of Tomorrow
2. Osterman Research White Paper: Uncovering the Presence of Open-Source Components in Commercial Software
3. Gartner Enterprise Software Forecast, 2021
4. Forrester Research: The State of Application Security, 2021

Neustar International Security Council (NISC) Survey:
https://www.helpnetsecurity.com/2022/03/02/log4j-vulnerability-security-professionals/

National Telecommunications and Information Administration (NTIA)
SBOM Working Group
https://www.ntia.gov/SBOM

Cybersecurity & Infrastructure Security Agency (CISA)
Software Bill of Materials
https://www.cisa.gov/sbom

2021 Presidential Cybersecurity Executive Order
Section 4. Enhancing Software Supply Chain Security
https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/