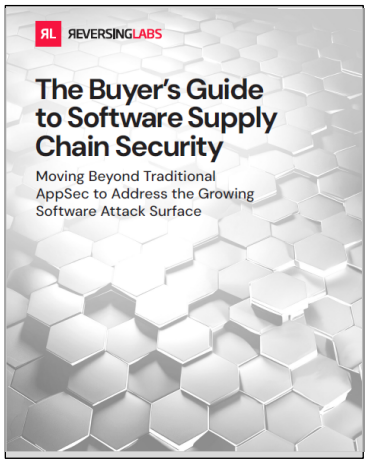

¢KŜ .dzèŜNJ4ă DdzλŘŜ ű2 {2Ŧűǿl·NJŜ {dzLJLJfè /KI·λƴ{ŜŎdzNJλűè

Thank you for downloading this ReversingLabs NJŜă2dzNŎŜ Carahsoft is the distributor for ReversingLabs Cybersecurity solutions available via TX DIR and other contract vehicles.

To learn how to take the next step toward acquiring ReversingLabs solutions, please check out the following resources and information:

For additional resources:
carah.io/ReversingLabsResources

For upcoming events:
carah.io/ReversingLabsEvents

For additional ReversingLabs solutions:
carah.io/ReversingLabsSolutions

For additional Cybersecurity solutions:
carah.io/Cybersecurity

To set up a meeting:
ReversingLabs@carahsoft.com
844-445-5688

To purchase, check out the contract vehicles available for procurement:
carah.io/ReversingLabsContracts

For more information, contact Carahsoft or our reseller partners:
ReversingLabs@carahsoft.com | 844-445-5689

REVERSINGLABS

The Buyer's Guide to Software Supply Chain Security

Moving Beyond Traditional AppSec to Address the Growing Software Attack Surface

According to research by ReversingLabs, software supply chain threats have risen 1300% from 2021 to 2023[1]. With concerns of software supply chain security (SSCS) on the rise, security-minded enterprises have turned to familiar application security testing (AST) tools like software composition analysis (SCA) to identify open-source vulnerabilities; static application security testing (SAST) for vulnerabilities in source code; and dynamic application security testing (DAST) to find vulnerabilities in running web applications. Those technologies are invaluable. And they're not enough. The truth of the matter is the biggest software-based attacks on enterprises in the last few years have employed a myriad of attack vectors, many of which don't rely on exploitable vulnerabilities in either open-source or proprietary software.

Many will point to the 2017 Equifax breach as the prime example of how important it is to ensure that open-source components are used and maintained securely by development teams. However, more recent software supply chain attacks, such as those affecting SolarWinds, 3CX, CircleCI, and Codecov, show that malicious actors are deploying malware, tampering with source code, and leveraging exposed development secrets to further their malicious campaigns, without needing to exploit known- or undiscovered software vulnerabilities.

For enterprises, the lesson of these attacks is clear: By omitting commercial and proprietary software assets from your software supply chain security strategy, you overlook large swaths of your attack surface and create gaps in threat detection that have already proven to be exploitable by cybercriminals and other bad actors.

Both producers and buyers of commercial software have a role to play in ensuring trusted applications are not a vehicle for the next software supply chain attack. In order to safeguard the applications your organization builds, buys, and ships, you need a solution that goes beyond traditional application security (AppSec) testing capabilities to test compiled software binaries and uncover threats like malware and tampering that traditional AST technologies are not designed to detect.

True Software Supply Chain Security Requires More Than Just Open-Source Testing

Modern enterprises rely on an enormous portfolio of proprietary, commercial, and open-source software and services to deliver customer value at scale. And, thanks to the forces of 'digital transformation,' that large software footprint only gets larger every day. The increasing reliance of organizations on software gives bad actors many avenues to access sensitive corporate assets and data: Whether that is by compromising open-source packages in a public repository or hijacking a commercial application from a trusted vendor, layers of uncertainty and visibility gaps present plenty of avenues for bad actors to access sensitive data through indirect means. Based on research from ReversingLabs, in 2023 alone, 11,000 malicious software packages were discovered across three major open-source repositories: npm, PyPI, and RubyGems, a 28% increase from 2022[2]. Many of these malicious components are obfuscated or encrypted, making detection using traditional AST tools nearly impossible.

[1] ReversingLabs, State of SSCS 2024 Report, January 2024
[2] ReversingLabs, State of SSCS 2024 Report, January 2024

SCA is not the panacea to address the entire software supply chain dilemma. It only addresses part of the issue – open source vulnerabilities.

SCA, in particular, has grown in recognition due to the increase in the usage of open source code. Unfortunately, SCA has been conflated to be a panacea addressing the entire software supply chain dilemma when, in fact, it only addresses part of the issue - open-source vulnerabilities. SCA, like most other AST tools, focuses on detecting vulnerabilities outlined in the OWASP Top 10 (SQL injection, cross-site scripting, buffer overflows, etc.), in pre-production environments; it is not designed to detect threat categories outside of known vulnerabilities such as malware.

Furthermore, SCA caters to a "shift-left" approach to vulnerability testing where scans typically take place before the build stage within the software development lifecycle (SDLC). While that helps with finding vulnerabilities early, security teams completely miss out on malicious code intentionally inserted in the multitude of components that are added later in the build process as part of the final package. It's at this stage where your software is susceptible to becoming a vehicle for malware - well outside the radar of SCA or any other traditional AST tool.

SOFTWARE SUPPLY CHAIN VISIBILITY: THE MISSING ARTIFACTS

Today's embedded malware, tampering, or exposed secrets go beyond open source. It includes proprietary and commercial code too. It includes proprietary code, commercial code, and, most importantly, the components and artifacts that are included in any build process. Those elements can increase the code by orders of magnitude.

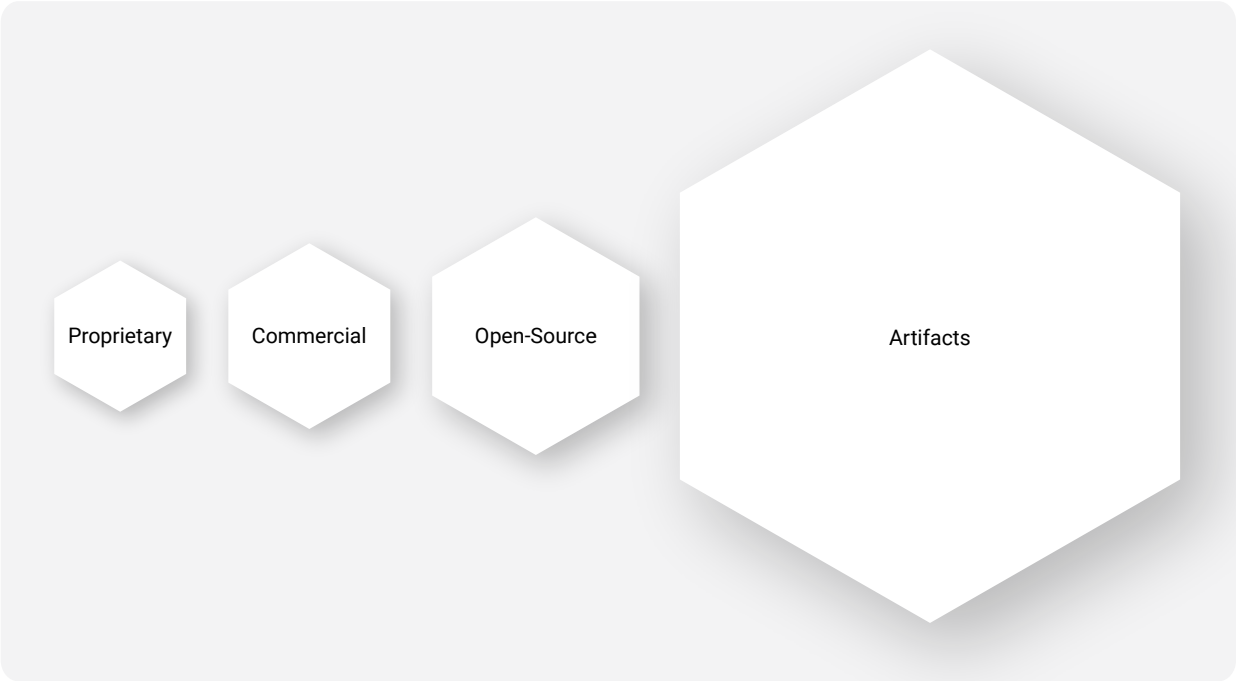

Figure 1: The components that make up a full software binary once compiled

Understanding the post-compilation, pre-deployment build is critical.

As outlined in Figure 1, it is important to consider all the parts that make up the final software package. While SCA and other pre-compilation tools are meant to prevent vulnerabilities from making it to the build, post-build artifacts can increase the size of the total software package from 10 to 1,000 times or more.

For example, in a particular piece of software analyzed by ReversingLabs, the proprietary, commercial, and open-source components accounted for 44,000 components, and the additional artifacts upon build accounted for an additional 1.26 million components. That's almost a 3000% increase in the total size of the software binary. This is why understanding the post-compilation, pre-deployment build is so critical.

Consider the automotive industry and how it goes about conducting crash safety tests on new vehicles. Vehicles undergo exhaustive crash testing from all angles on fully assembled vehicles. Testing individual components would not give a clear indication as to how the vehicle would fare in a real-world accident. The same principle applies to software and illustrates why looking at the complete package is so critical. For example, if malicious code is embedded in the ReadMe PDF of a software package, neither SCA, SAST, nor DAST would see it. Analyzing the entire software package in its final executable state, including every commercial, proprietary, and open-source component, is the only way to check for hidden threats that SCA and other AST tools are not trained to find.

COMMERCIAL SOFTWARE REPRESENTS AN ENTERPRISE'S LARGEST UNDER ADDRESSED ATTACK SURFACE

A full software supply chain security solution must encompass much more than open-source dependencies; vendor-provided commercial applications within your software supply chain are just as susceptible to exploitation as open-source software. Within the last few years, we've seen examples of high-profile software supply chain breaches that were not perpetrated by exploiting a known vulnerability, but by other advanced threat vectors, and would have been undetectable if relying solely on AST tools to find vulnerabilities:

- In 2021, CodeCov's Bash Uploader script was clandestinely compromised, exposing the CI environments of a portion of their 29,000 customers.[3]
- In 2023, the continuous integration/continuous deployment (CI/CD) platform, CircleCI, was compromised when an unauthorized third party gained access to the company's internal development systems through stolen credentials, impacting not only CircleCI but several customer environments.[4]
- In 2023, North Korean threat actor group Lazarus Group compromised the endpoint client of VoIP software vendor 3CX with a first-of-its-kind cascading software supply chain attack, delivering malware to over 60,000 victims.[5]
- In 2023, state-sponsored North Korean threat actors breached Taiwanese multimedia software company CyberLink by hijacking and modifying a legitimate installer to push malware that was detected on over 100 devices worldwide.[6]

[3] Bleeping Computer, Hundreds of networks reportedly hacked in Codecov supply-chain attack, April 2021
[4] Security Boulevard, Lessons learned from the CircleCI secrets breach, February 2023
[5] Wired, The Huge 3CX Breach Was Actually 2 Linked Supply Chain Attacks, April 2023
[6] Bleeping Computer, Microsoft: Lazarus hackers breach CyberLink in supply chain attack, November 2023

We see a common thread in the examples mentioned: the damage from these attacks was able to spread because the attackers took advantage of the good reputation of each of these organizations, hiding malicious code in trusted, commercially available software packages. These are the complex mechanisms that traditional AST tools are not designed to detect.

Going Beyond Just Vulnerability Detection

While vulnerabilities should be accounted for within SDLC workflows, SAST, DAST, and SCA are not designed to flag the other critical software supply chain threats that invite malicious code, tampering, or secrets leakage. SCA, specifically, is dependent on cross-referencing open-source libraries against publicly available vulnerability databases like the National Vulnerability Database (NVD). In other words, SCA is only as effective as what it is taught to detect.

SCA is only as effective as what it is taught to detect

To truly digest the scope of the software supply chain dilemma, it's important to understand how software supply chain attacks are carried out so that your team has the knowledge, skills, and tools to sniff them out preemptively. Having the mechanisms in place to flag advanced software supply chain threat categories within your application portfolio - both the software you build and buy - can help mitigate risk across the enterprise.

SOFTWARE SUPPLY CHAIN THREAT CATEGORIES TO MONITOR

Below are some notable examples of software supply chain attack vectors to look for:

Embedded Malware

Within the scope of software supply chain security, malware encompasses any software explicitly designed to harm, exploit, or compromise computer systems, networks, or devices. Malware manifests in various forms, and even the most trusted commercial software packages can act as a potential carrier.

Tampering

Software tampering is the unauthorized modification or alteration of computer software conducted without the knowledge or consent of its rightful owner or developer. This practice encompasses malicious activities ranging from reverse engineering to injecting malware, to falsifying digital signatures in order to distribute malicious code under the guise of a trusted, vendor-provided application.

Secrets Manipulation

Secrets manipulation is when sensitive credentials to internal software development environments or assets such as API tokens, passwords, or SSH keys leak their way into the hands of an attacker. This can happen through a variety of means such as plain text passwords, weak cryptography, flawed CI/CD or packaging automation, and compromised developer accounts or malicious insiders. Attackers will sometimes find these credentials buried within public code repositories, granting them access to internal software delivery pipelines which, in turn, provides an avenue to inject malicious code.

Suspicious Behaviors

Suspicious behavior in software packages typically entails actions being performed that are outside the typical function of the application. An example of this would be a virtual meeting software having access to remote desktop sessions even if it's not being run. Another example would be a software package that enumerates running processes or detects virtualized environments.

Weak Mitigations

Security mitigations are safeguards to help improve the overall software package security. These safeguards are triggered during software package analysis to check your compiled code and inform you if any of the built-in validation rules are violated. Weak mitigations affecting Windows and Linux binaries can include missing vulnerability protections, insecure coding practices, outdated toolchains, inadequate prevention methods, and missing fortified functions.

Vulnerabilities

Not to be ignored, a software vulnerability is a weakness or flaw in a software application, system, or component that malicious actors can exploit to compromise the confidentiality, integrity, or availability of the software or the data it handles. Software vulnerabilities range from coding errors and design flaws to misconfigurations and insecure user practices.

Now that we have a baseline understanding of the avenues threat actors take to carry out software supply chain attacks, your security, development, and procurement teams can develop the proper workflows and checks to close off the gates from threat actors attempting to access sensitive data or compromise critical systems through backdoor or indirect means.

Best Practices to Apply Software Supply Chain Security Across the Enterprise

Tackling the vast scope of securing your software supply chain is a daunting task, but having a baseline understanding of how software packages make their way into your environment helps to identify potential gaps and take the necessary steps to fill them. While many organizations may start by assessing their open-source software risk profile, it would be a mistake to stop there. Because SCA is purpose-built to ensure vulnerabilities embedded in open-source software don't make it to the build, it misses a lot of potential risks introduced downstream - those that don't involve vulnerabilities at all.

A holistic approach to software supply chain security must go beyond SCA and traditional AppSec approaches and take into account your entire software portfolio including open-source, commercial, and proprietary software packages. Having a way to break down software packages into their core components will allow you to identify, flag, and contain potential threat vectors that traditional application security tools miss.

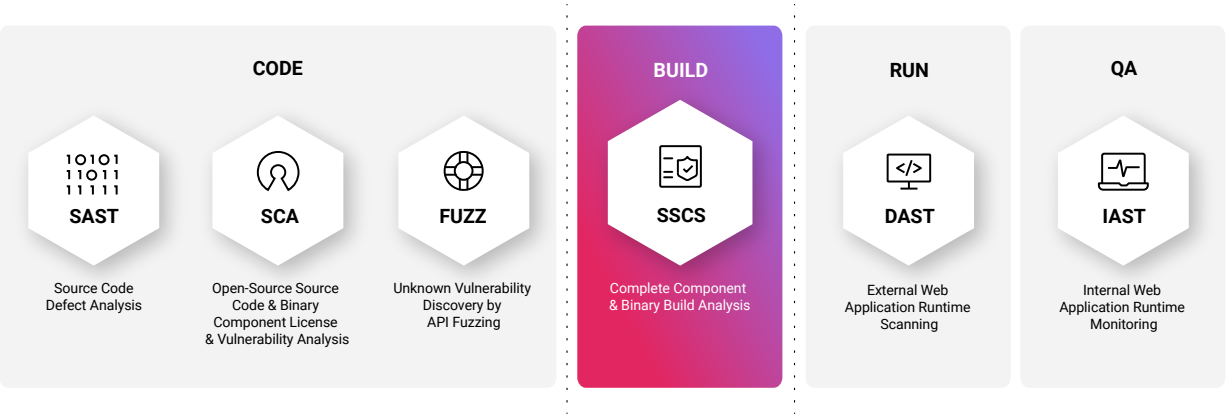

Figure 2: Software supply chain security delivers the critical release exam before releasing to customers by instituting a final build test. Post-compilation. Pre-deployment.

INSTITUTE A CRITICAL RELEASE EXAM FOR YOUR SOFTWARE PACKAGES

Independent software vendors (ISVs) already have a plethora of tools they use to assess the security of their applications pre-release. Everything from classic application security tools like SCA, SAST, and DAST, has a role to play within a specific stage of the software development lifecycle.

There are hundreds of ways that malware can potentially be introduced into a final software package.

The glaring pitfall with these tools is that they are hyper-focused within their individual swimlane - none of them are designed to detect malware or analyze proprietary, commercial, open source, and artifacts as a unit. Due to the complex nature and myriad moving parts of today's software construction processes, there are hundreds of ways that malware can potentially be introduced into a final software package.

How do you know if there have been significant changes made to the application without your team knowing? Only by testing the application as it is deployed in production do you get full visibility into additional components or changes that are red flags denoting advanced threat exposures like malware, tampering, suspicious behaviors, or falsified signatures.

ASSESS THIRD-PARTY SOFTWARE RISK INDEPENDENTLY

For third-party risk management (TPRM) teams, assessing the security standing of vendor-provided commercial software is a manual and laborious process. Too often, enterprise software buyers are forced to make buying decisions based on answers provided in point-in-time security surveys and pen-testing results. Despite these measures, 83% of TPRM leaders still find risks embedded in vendor applications according to Gartner.

In order to reduce material third-party risk, TPRM and procurement teams need to have the freedom to institute their own security testing regimen on commercial applications so that they are not beholden to the black box that is their vendors' security practices. Making this move would allow organizations to make more informed, risk-based decisions within their vendor evaluation process, along with having continuous visibility into new threats that may emerge from patches or updates.

INSTITUTE A COMPREHENSIVE SBOM THAT COVERS MORE THAN JUST OPEN SOURCE

An emerging practice that has gained a lot of traction in recent years is requiring a software bill of materials (SBOM) that details each component that makes up the final package of the application. Think of an SBOM as an ingredient label for your software. SCA tools have been delivering SBOMs for a while but they only detail the open-source components, leaving out any commercial or proprietary components which can pose just as much of a risk.

SBOMs are typically delivered in widely accepted formats such as OWASP's CycloneDX and Linux Foundation's SPDX templates. For SBOMs to be truly effective, they must consider the entire software package, providing visibility into individual binaries, artifacts, and packages. It's only by taking this holistic approach that you can conduct deeper investigations into more advanced attack methods.

Several federal mandates in recent years have required any software vendor to provide a comprehensive SBOM if they hope to do business with federal agencies. However, TPRM teams within private organizations have also recognized SBOMs as a way to verify secure development practices by vendors, helping them mitigate the threat of a software supply chain breach through commercial software.

A piecemeal approach to software supply chain security - one that tests custom and open-source components without context into the final software build - fails to address the needs of enterprises that build and buy software. ReversingLabs Spectra Assure goes beyond the limitations of traditional SCA and AST tools by providing software producers and buyers alike with key capabilities that provide visibility into the core components of software packages and flag advanced threat exposures beyond just vulnerabilities.

Spectra Assure: Software Supply Chain Visibility Beyond Just Open Source

Spectra Assure for software supply chain security is powered by the first AI-driven complex binary analysis engine that highlights advanced threats that SCA and other traditional AST tools miss. This proprietary and proven binary analysis technique is designed to analyze the entire software package including proprietary, commercial, and open-source components, plus all artifacts added during compilation, breaking down extremely large files to its core components with ease - all without requiring source code. By assessing the final package in its entirety, Spectra Assure is able to dissect every binary, artifact, container, and package for specific software supply chain threat vectors including malware, vulnerabilities, code tampering, secrets leakage, and suspicious behaviors just to name a few.

THE POWER OF COMPLEX BINARY ANALYSIS

SCA tools typically analyze package managers, manifest files, or source-code repositories to find vulnerabilities in open-source software. They are limited by the need for known signatures of open-source dependencies to enable cross-referencing those signatures against a vulnerability database like the NVD. Because they are used in a pre-build environment, they lack visibility into post-build artifacts and file structures that present openings for more novel, sophisticated attacks. In order to flag the most advanced software supply chain threats, enterprise security, and development teams need to adopt a more robust approach to analyzing complex files in a post-compilation, pre-deployment state.

Spectra Assure provides visibility into customer risk exposures by assessing the risk profile of your application in its final build state. Powered by a complex binary analysis engine, Spectra Assure digs deep into the internal contents of files, recursively unpacking and deobfuscating them at lightning speed to reveal the risks and threats inside. Analysis of a 1GB file can take as little as two minutes. Furthermore, Spectra Assure does not require source code or execution of the application, allowing software buyers and third-party risk managers to verify that internal components - whether they be proprietary, commercial, or open-source - come from trusted sources.

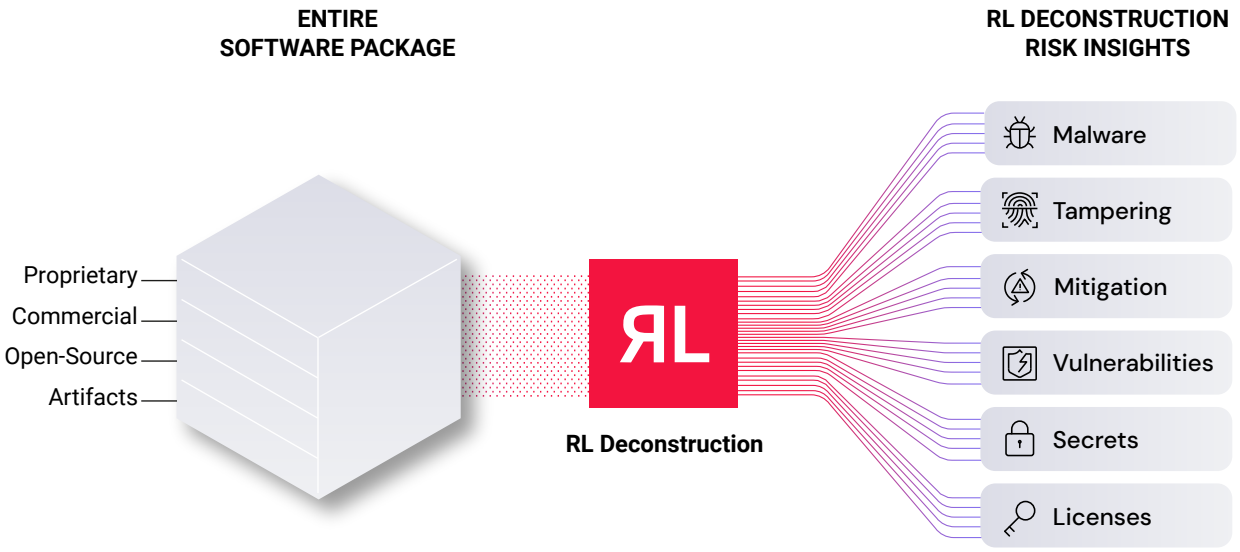

Figure 3: Complex binary analysis powering Spectra Assure dissects custom, commercial, and open-source components, along with any executables, files, containers, or other elements to find embedded software supply chain threats.

BENEFITS FOR SOFTWARE PRODUCERS AND PRODUCT SECURITY TEAMS

AppSec teams need to institute the proper checks within their software development process to ensure that updates do not inadvertently introduce new attack vectors into the wild for bad actors to exploit. To do this, the entire application needs to be analyzed in its final post-compilation, pre-deployment state to get an accurate assessment of the security of the package. Just like the automotive industry crash tests the fully assembled vehicle, software producers must test the fully assembled application for a comprehensive analysis of its risk profile.

Spectra Assure makes this simple by integrating with existing CI/CD tools, workflows, and ephemeral environments without requiring any proprietary source code to be uploaded for analysis. Spectra Assure automatically assesses each update and compares reproducible builds as a final check before releases or deployments. The final executable is analyzed for malware, vulnerabilities, and tampering, along with any exposed secrets stemming from packaging errors or insider threat actors.

For application and product security teams, false positives and alert fatigue is a productivity killer. Spectra Assure provides development, AppSec, and product security teams the context they need to act on embedded threats quickly and avoid unnecessary delays. Each alert, regardless of threat category, is coupled with simple-to-follow remediation guidance and risk ranking that can be customized so your team is alerted to the threats that matter most. Automated differential analysis provides continuous, release-to-release monitoring to ensure that no malicious or vulnerable components have been added and that functional behaviors are within normal parameters. Think of Spectra Assure as an early warning system for developers - flagging malicious code, vulnerabilities, unexpected or risky behaviors, and exposed secrets before they present your customers to unnecessary risk.

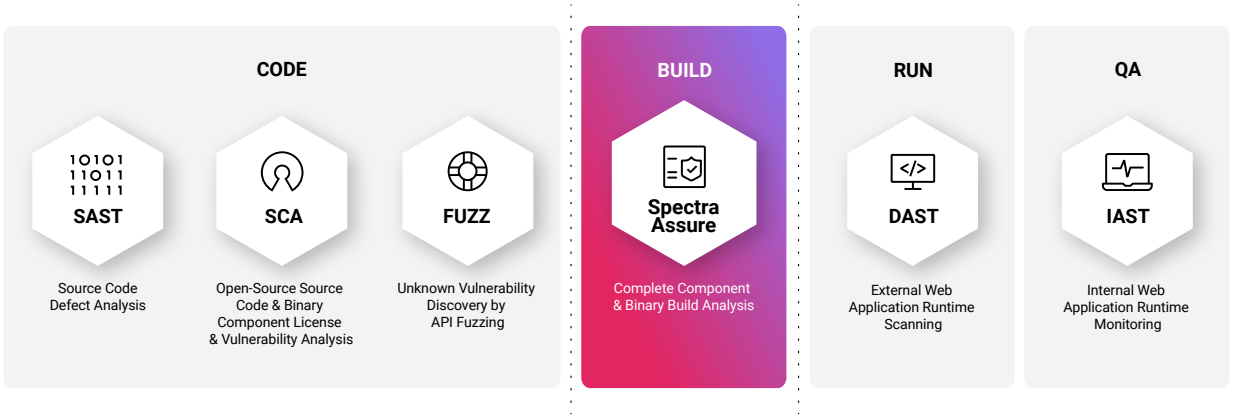

Figure 4: Spectra Assure addresses the gap in the software development lifecycle with complete analysis without the need for source code

BENEFITS FOR SOFTWARE BUYERS AND RISK-MANAGEMENT ORGANIZATIONS

TPRM teams commonly rely on security assessment surveys within their vendor risk-management program to verify third-party software for potential risks. However, without access to source code, enterprises are forced to take the vendor at their word regarding their secure software practices.

Spectra Assure enables TPRM teams with a means to do a thorough analysis of commercial third-party applications to ensure they are not exposing their environment to unnecessary risk.

By providing enterprise software consumers with a comprehensive SBOM, Spectra Assure showcases the core components of the application including open-source, commercial third-party, and proprietary components purpose-built by the vendor. With this visibility, TPRM organizations have the context they need to verify that their vendor uses components from reliable and trustworthy sources. SBOMs provided by SCA tools only provide a fraction of this information by concentrating solely on open-source components.

Figure 5: Spectra Assure provides a complete third-party software analysis before releasing to enterprise environments.

Beyond SBOMs, Spectra Assure gives TPRM organizations the means to analyze commercial software for advanced software supply chain threats without being beholden to whatever security tooling their vendor might use. ReversingLabs's AI-driven complex binary analysis engine can unpack over 400 types of packages and analyze over 4,800 unique file types to scan for malware, secrets, suspicious behaviors, and more without requiring source code. This deep analysis enables TPRM organizations to make much more educated decisions regarding their vendor onboarding process and how they work with Security Operations to continuously monitor illicit behavior and threats stemming from their applications.

Conclusion

Since its introduction, software composition analysis has been a boon for enterprise security teams to deliver secure software at speed and, for many, is a reasonable first step in securing their software supply chain. However, SCA, like most AST tools, are not designed to handle the vast majority of software supply chain threats that have been proven to be exploitable in several high-profile breaches over recent years. The software attack surface gap has outgrown what traditional AppSec approaches can cope with.

For modern enterprises, those that both produce and consume commercial applications, a more holistic approach is needed - one that does not rely on open-source software components in a vacuum, but rather provides visibility into the full software binary and assesses the spectrum of software supply chain threats by contextualizing the entire software package within its build environment.

Spectra Assure for Software Supply Chain Security is the only solution that provides the mechanisms for both third-party risk management and development teams to assess risk, enforce policies, and rapidly remediate software supply chain threats across proprietary and third-party applications of any sort.

Learn More about ReversingLabs

ReversingLabs is the trusted authority in file and application security, protecting software development and powering advanced security solutions for the most advanced cybersecurity and Fortune 500 companies. The ReversingLabs Titanium Platform® powers the software supply chain security and threat intelligence solutions essential to advancing enterprise cybersecurity maturity globally. Tracking over 35 billion files daily, and the ability to deconstruct full software binaries in seconds or minutes, only ReversingLabs provides that final exam to determine whether a single file or full software binary presents a risk.

To learn more about Spectra Assure capabilities and solutions, request a free trial here.

REQUEST A FREE TRIAL HERE

reversinglabs.com

REVERSINGLABS

Worldwide Sales: +1.617.250.7518
sales@reversinglabs.com