Contrast
SECURITY

SOLUTION BRIEF

Contrast SCA: Automated Open-Source Security Without The Noise

Executive Summary

Open-source software (OSS) affords many freedoms to developers who need to build feature-rich applications at speed. However, OSS and other third-party software also adds challenges with visibility and governance across an organization's software supply chain. Risks related to open-source vulnerabilities and licensing are compounded when development and security teams clash over what constitutes real risk in high-velocity software development environments like Agile and DevOps. Modern software development requires a modern software composition analysis (SCA) solution that provides real-time visibility and continuous monitoring into third-party software risk without impeding development.

Contrast SCA offers a new approach to SCA by prioritizing the risk that matters most. Contrast SCA streamlines remediation efforts by analyzing which libraries are actually used during application runtimes—down to the specific class, file, or module. It automatically creates an inventory of an organization's OSS and commercial off-the-shelf (COTS) libraries while providing continuous observability into new vulnerabilities, with no manual scanning required. Integrated policy controls and dependency risk management within native continuous integration/continuous deployment (CI/CD) workflows enable scalable open-source visibility and governance without creating bottlenecks for developers.

Ubiquitous Open-Source Risks Require Unique Application Security Tools

OSS components do not always pass the same quality and standards checks as proprietary code. The number of disclosed open-source vulnerabilities has nearly tripled over the past five years, from 6,487 in 2015 to 18,358 in 2020.[2] And this does not include all of the vulnerabilities that have not been reported. Unless each open-source component is evaluated before implementation, it is easy to incorporate vulnerabilities that reduce the overall quality of code.[3] In order to simplify the management of OSS components and reduce application vulnerabilities across the software development life cycle (SDLC), teams need full awareness of their open-source risks.

> *33% of applications are never tested for security vulnerabilities, with nearly 80% of applications containing at least one serious (critical or high) vulnerability.*[1]

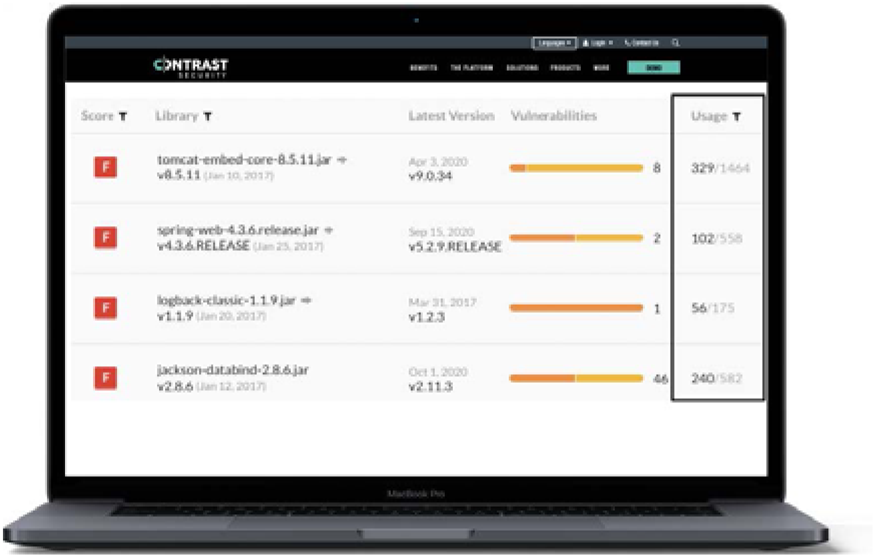
Contrast SCA offers a new, embedded approach to software composition analysis (SCA) that removes much of the overhead from application security and development teams. Unlike traditional SCA tools that only provide point-in-time assessments, Contrast SCA leverages a single-agent platform to continuously evaluate third-party libraries within the application runtime. This eliminates the need for a separate assessment with different tools. There are no scans to manage and no extra steps for developers—just continuous insight. Contrast SCA detects which open-source libraries are called in runtime, the dependencies associated with them, and if they are exposing the organization to unnecessary security risks or legal problems due to open-source licensing complications.

> *An estimated 84% of security breaches exploit vulnerabilities at the application layer. Consequently, in the time between release and patch, adversaries can leverage the weaknesses in the code to compromise vulnerable systems.*[4]

Runtime Analysis Delivers Fast and Accurate Remediation

Managing third-party software risks requires the quickest possible turnaround for resolving issues once they emerge. Open-source vulnerabilities that can be exploited are very valuable on the dark market. For example, information about undocumented vulnerabilities in popular software that can give root or equivalent access sells for as much as $1 million.[5] But even when these zero-day vulnerabilities are eventually discovered and disclosed, it can take teams months to apply all patches and push them to production.[6]

Traditional SCA tools deliver limited, point-in-time source-code scans that lack the context of what is actually being used by the application. To help accelerate application protection and shrink the window of exposure, Contrast SCA enables early detection of open-source vulnerabilities and licensing risks in the developer environment with continuous verification across continuous integration/continuous deployment (CI/CD) pipelines. Contrast SCA performs runtime analysis to accurately identify specific libraries that are exercised by the application down to their specific class, file, or module (depending on the language). This intelligent runtime analysis enables application security and development teams to prioritize and focus remediation efforts on vulnerabilities that matter.



Score	Library	Latest Version	Vulnerabilities	Usage
F	tomcat-embed-core-8.5.11.jar v8.5.11 (Jan 10, 2017)	Apr 3, 2020 v9.0.34	8	329/1464
F	spring-web-4.3.6.release.jar v4.3.6.RELEASE (Jan 25, 2017)	Sep 15, 2020 v5.2.9.RELEASE	2	102/558
F	logback-classic-1.1.9.jar v1.1.9 (Jan 20, 2017)	Mar 31, 2017 v1.2.3	1	56/175
F	jackson-databind-2.8.6.jar v2.8.6 (Jan 12, 2017)	Oct 1, 2020 v2.11.3	46	240/582

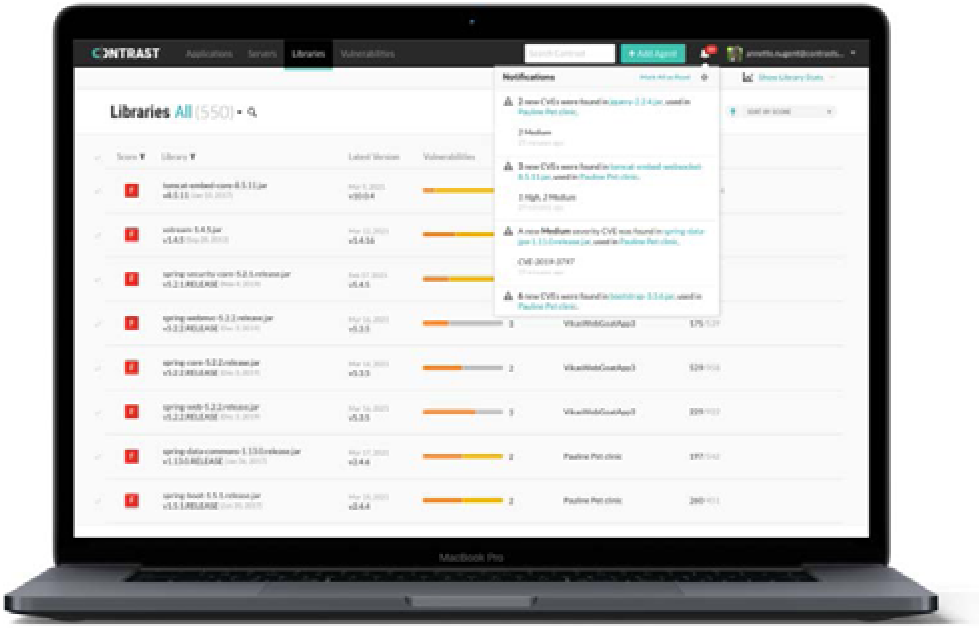
Continuous Visibility of all OSS Components

To maintain a healthy codebase, organizations should monitor all the OSS components they have introduced across the application portfolio. Visibility gives security experts the opportunity to respond to events in a timely manner.[8]

The nature of open source means that new vulnerabilities in popular libraries and frameworks are regularly discovered and reported. When new security vulnerabilities are published, they are often accompanied with an explanation of how to exploit it as a way to validate the finding. The caveat of this practice is that attackers have access to this same information, leaving unpatched libraries open to exploitation. Contrast SCA provides application security teams a way to rapidly respond to emerging threats by continuously monitoring for new vulnerabilities in deployed libraries and providing automated alerts. Contrast SCA automatically creates and maintains an organization wide inventory of open-source and commercial off-the-shelf (COTS) library code mapped to applications, servers, and environments—identifying what runs where and what needs to be secured.

> “
>
> *Some tools use static analysis to guess whether a component might be used by the application. This approach is plagued with both false positives and false negatives. These inaccuracies create noise that interferes with effective and automatic prioritization of vulnerabilities, while increasing an application’s risk exposure.*[7]

Having full visibility into applications' software bill of materials enables security and development teams to assign ownership and rapidly respond to emerging threats across the software supply chain.



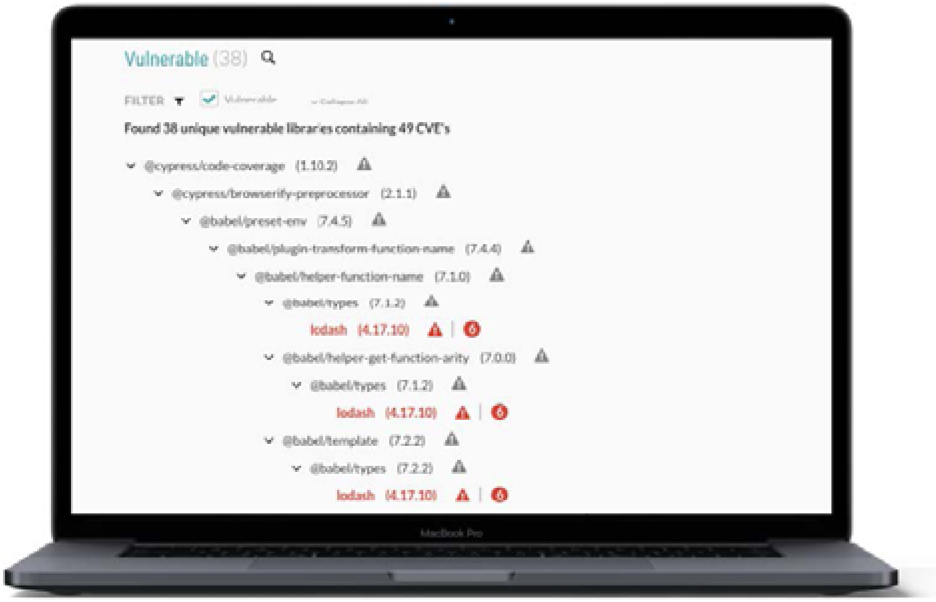
Benchmarking of Software Supply Chain Risk

The software supply chain has multiple layers—some less obvious than others. In order to carry out their function, OSS libraries pull dependencies from public and private repositories during the build process, adding layers of risk into the application that is unaccounted for at most organizations. Development and application security teams are often unaware of how these dependencies are pulled into their code.

Using the Contrast command-line interface (CLI) within Contrast SCA, developers can run quick tests for vulnerable top-level libraries prior to committing code. Additionally, the CLI highlights transitive dependencies introduced during the build process by populating a dependency tree within Contrast SCA. This dependency tree provides much-needed awareness and context into the deepening layers of dependency risk. Dependencies add compounding layers of complexity to the task of closing security gaps within the software supply chain. This is exacerbated by the fact that attackers have identified alternative means to create backdoors into critical business systems by way of piggybacking onto native developer tools and pulling in third-party code from public repositories.

Dependency confusion is a prominent example of how attackers leverage the software supply chain as an alternative attack vector to accessing sensitive data.[9] Similar to the way typosquatting attacks rely on erroneous spelling mistakes to trick users into installing malicious software, dependency confusion attacks provide an avenue for malicious code to be pulled into the application by tricking the package manager into pulling in code from a public repository that uses the same naming conventions as internally maintained software packages. Dependency confusion enables an attacker to plant malicious code within a public repository that exposes the victim to remote code execution, taking over the host, or exfiltration of sensitive data.

When run, the Contrast CLI warns developers if there are any internal dependencies that could be a dependency confusion risk. This includes highlighting libraries that are not scoped for a project and benchmarking libraries with suspicious versioning. Providing the proper context and enabling developers to flag dependency risk within their software supply chain allows them to reap the benefits of incorporating third-party software while removing bottlenecks that come from security audits and validating results.



```
Found 38 unique vulnerable libraries containing 49 CVE's

⌄ @cypress/code-coverage (1.10.2) ⚠
    ⌄ @cypress/browserify-preprocessor (2.1.1) ⚠
        ⌄ @babel/preset-env (7.4.5) ⚠
            ⌄ @babel/plugin-transform-function-name (7.4.4) ⚠
                ⌄ @babel/helper-function-name (7.1.0) ⚠
                    ⌄ @babel/types (7.1.2) ⚠
                        lodash (4.17.10) ⚠ | 6
                    ⌄ @babel/helper-get-function-arity (7.0.0) ⚠
                        ⌄ @babel/types (7.1.2) ⚠
                            lodash (4.17.10) ⚠ | 6
                    ⌄ @babel/template (7.2.2) ⚠
                        ⌄ @babel/types (7.2.2) ⚠
                            lodash (4.17.10) ⚠ | 6
```

Figure 1: The dependency tree contextualizes how vulnerable dependencies are pulled into the application.

Automatic Enforcement of Policy-Based Controls

The best way to manage OSS risks is to have clearly written and enforced policies.[10] Contrast SCA allows application security managers to set security and licensing policies that establish third-party governance across their application portfolio. Contrast SCA triggers alerts when security and licensing violations are detected within native CI/CD workflows and even automate policy enforcement by blocking a build that contains Common Vulnerabilities and Exposures (CVE) above a certain policy threshold. All this information is streamed to security and development teams in real time, enabling short feedback loops and quick action.

> *Automation tools can help create a vulnerability inventory, keep track of the vulnerabilities, and prioritize remediation.*[11]

A Platform for Superior OSS Application Security

Safeguarding the software supply chain requires a multipronged approach, and traditional SCA tools have too many limitations when it comes to accurately protecting third-party software assets. Contrast offers the only application security solution that can identify vulnerable open-source components, determine how they are actually used by the application, and prevent exploitation. Contrast's comprehensive single-agent application security platform—made up of Contrast Protect, Contrast Assess, and Contrast SCA—protects applications across the entire SDLC.

Contrast Protect provides real-time protection from targeted CVE attacks. This gives developers time to update libraries in existing applications while enabling rapid response to new, zero-day threats.

Contrast Assess helps developers deliver secure, quality code by identifying vulnerabilities in their entire code base—custom and open source—based on runtime analysis. Assess enables rapid vulnerability remediation by integrating into native development and testing workflows.

Contrast SCA delivers automated third-party software security testing without the noise. Contrast SCA supports key business requirements that include:

- Reducing noise for developers by eliminating the need for manual source-code analysis
- Full visibility and real-time monitoring of third-party software inventory
- Prioritizing remediation efforts by tracking the libraries that actually get used by applications during runtime operation
- Understanding the depth of risk that library dependencies can produce
- Building scalable policies and integrating checks into developer workflows

[1] Drew Spaniel and Rob Roy, "Software Security Is National Security," Institute for Critical Infrastructure Technology, April 2019.
[2] Sarah Coble, "2020 Saw 6% Rise in Number of CVEs Reported," Infosecurity, January 14, 2021.
[3] Ilai Bavati, "5 Best Practices for Managing Open-Source Components," DevOps.com, September 11, 2019.
[4] Drew Spaniel and Rob Roy, "Software Security is National Security," Institute for Critical Infrastructure Technology, April 2019.
[5] Mark Curphey and David A. Wheeler, "Improving Trust and Security in Open Source Projects," The Linux Foundation, February 2020.
[6] Abby Ross, "Why Fixing Security Vulnerabilities Is Not That Simple," Security Intelligence, October 1, 2019.
[7] Augusto Barros, "From my Gartner Blog—Considering Remediation Approaches For Vulnerability Prioritization," Security Boulevard, May 2, 2019.
[8] Gilad David Maayan, "How to Make Your CSO Happy with Your Open Source Components," CPO Magazine, August 28, 2019.
[9] Matt Austin, "Dependency Confusion: A New Third-party Risk for the Software Factory," Contrast Security, February 24, 2021.
[10] James G. Gatto, "Open Source Software Policies—Why You Need Them And What They Should Include," National Law Review, June 18, 2019.
[11] Gilad David Maayan, "How to Make Your CSO Happy with Your Open Source Components," CPO Magazine, August 28, 2019.

240 3rd Street
2nd Floor
Los Altos, CA 94022
Phone: 888.371.1333
Fax: 650.397.4133

Contrast
SECURITY

contrastsecurity.com