WhiteSource

WhiteSource Prioritization Solution

DATASHEET

THE CHALLENGE

The number of reported open source security vulnerabilities continues to rise, and it is no longer practical to address all open source vulnerabilities and stay on schedule.

Therefore, prioritization has become critical for application security, especially in the case of open source vulnerabilities, where applications usually make calls to a small number of methods or functionalities.

THE SOLUTION

Effective Usage Analysis is a new technology that assesses the security impact of each open source vulnerability. This is done by analyzing whether the vulnerability is being reached or not. An effective vulnerability means that the proprietary code can reach the vulnerable functionality. Therefore, these vulnerabilities should be remediated with high priority.

Our research shows that only 15% to 30% of vulnerabilities are indeed effective, so teams can focus on a smaller number of alerts while remaining agile.

EFFECTIVE VS INEFFECTIVE

VULNERABILITIES IN A COMPONENT

EFFECTIVE VULNERABILITY
If the proprietary code is making calls to the vulnerable functionality

INEFFECTIVE VULNERABILITY
If the proprietary code is NOT making calls to the vulnerable functionality

TOP BENEFITS

1 **REDUCE SECURITY ALERTS UP TO 85%**

Prioritize detected vulnerabilities by filtering out the ones that you're not effectively using. Our data shows that 70% to 85% of the open source vulnerabilities in your code are not impacting the security of your applications, because your proprietary code never makes calls to these vulnerabilities. So why spend resources remediating them?

2 **SPEED UP THE REMEDIATION PROCESSES**

Expedite the resolution of issues with detailed call graphs, which pinpoint the path to the vulnerable method from the proprietary code. The call graph provides the location of the reference in each library along with the path: filename, class name, method name and line in the code, to help developers decide on the best course of remediation.

3 **STREAMLINE COLLABORATION BETWEEN TEAMS**

Use effectiveness as an objective indicator that determines the impact of a security vulnerability, to reduce friction between security and development teams. In addition, prioritizing vulnerabilities based on effectiveness and offering actionable remediation insights will lead to significant savings in development and analysis time, accurate risk assessment and increased remediation velocity.

PRODUCT SPECIFICATIONS

Languages	Supports Java, Scala and Kotlin projects (Maven or Gradle dependency management); POJO projects; JavaScript node.js (npm dependency manager); C#; Python.
Agents	The WhiteSource Unified Agent
Deployment options	Supports both cloud-based and on-premises WhiteSource deployments
Analysis scope	Analysis results can be displayed for different scopes: Organization, Product, Project, Library, as requested by the user. For Organization and Product scopes, the UI aggregates the findings to present a 'weighted' display, factoring the analysis results for the scope's constituent entities
Project types	Single projects (Java, JavaScript, Scala, Kotlin) or multi-projects (Java) featuring sub-projects
Project targets	Java/Scala/Kotlin: .jar, .war, .ear; JavaScript node.js: package.json
Reporting	Analysis results can be reported via a dedicated Alert report as well as exporting options to XML and Microsoft Excel (XLS) formats
API	Support for comprehensive, Organization-scope data (all Products and Projects associated with an Organization)

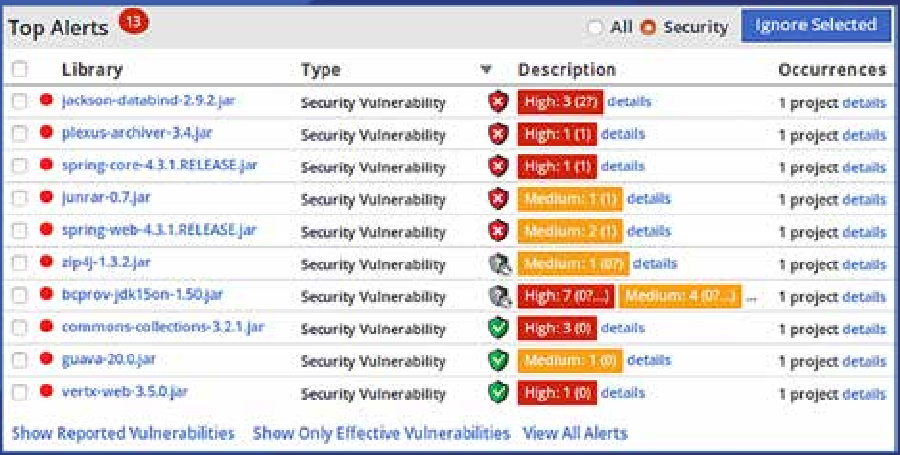
TOP ALERTS

The ‘shield’ icons indicate the effectiveness level of each alert.

This is an effective vulnerability. Your proprietary code is making calls to vulnerable functionality.

This is NOT an effective vulnerability.

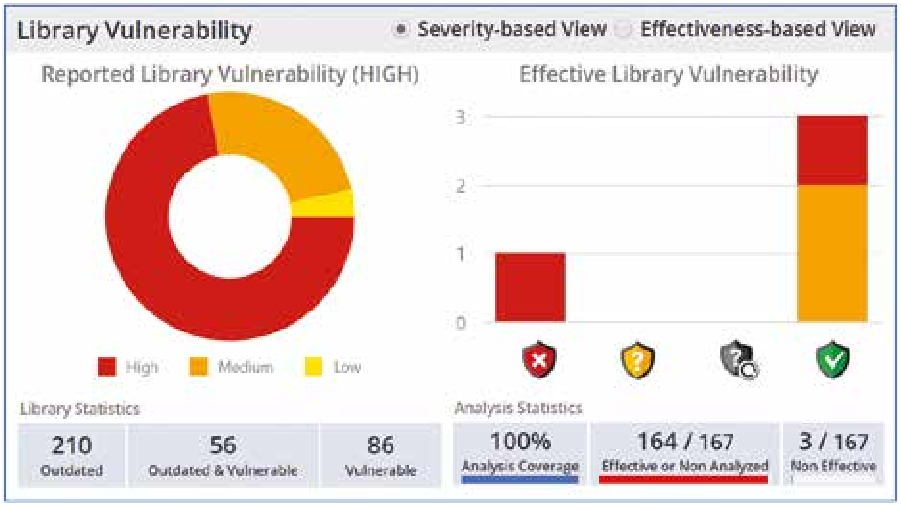
A new scan is recommended due to updated vulnerability information.



Top Alerts 13 — All / Security — Ignore Selected

	Library	Type		Description	Occurrences
☐ ●	jackson-databind-2.9.2.jar	Security Vulnerability	✗	High: 3 (2?) details	1 project details
☐ ●	plexus-archiver-3.4.jar	Security Vulnerability	✗	High: 1 (1) details	1 project details
☐ ●	spring-core-4.3.1.RELEASE.jar	Security Vulnerability	✗	High: 1 (1) details	1 project details
☐ ●	junrar-0.7.jar	Security Vulnerability	✗	Medium: 1 (1) details	1 project details
☐ ●	spring-web-4.3.1.RELEASE.jar	Security Vulnerability	✗	Medium: 2 (1) details	1 project details
☐ ●	zip4j-1.3.2.jar	Security Vulnerability	?	Medium: 1 (0?) details	1 project details
☐ ●	bcprov-jdk15on-1.50.jar	Security Vulnerability	?	High: 7 (0?...) Medium: 4 (0?...) ...	1 project details
☐ ●	commons-collections-3.2.1.jar	Security Vulnerability	✓	High: 3 (0) details	1 project details
☐ ●	guava-20.0.jar	Security Vulnerability	✓	Medium: 1 (0) details	1 project details
☐ ●	vertx-web-3.5.0.jar	Security Vulnerability	✓	High: 1 (0) details	1 project details

Show Reported Vulnerabilities   Show Only Effective Vulnerabilities   View All Alerts



Library Statistics

210	56	86
Outdated	Outdated & Vulnerable	Vulnerable

Analysis Statistics

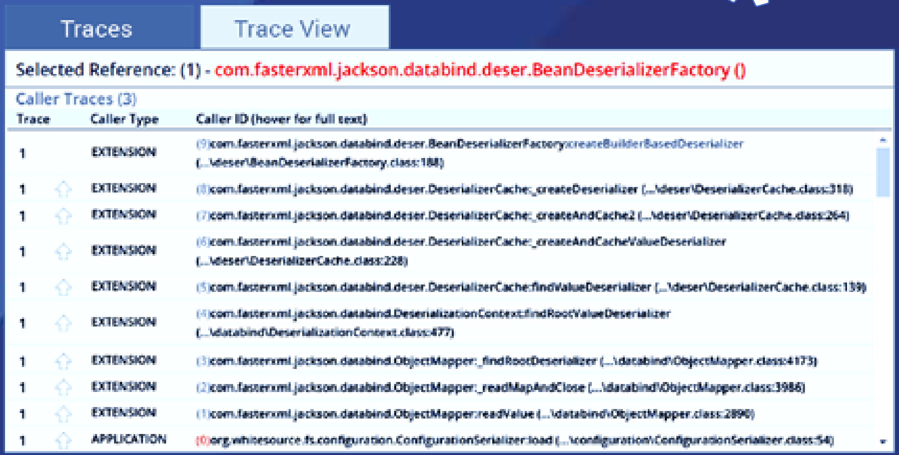
100%	164 / 167	3 / 167
Analysis Coverage	Effective or Non Analyzed	Non Effective

LIBRARY VULNERABILITY PANE

Review all reported vulnerabilities, per severity level (left side) and per effectiveness (right side). The Analysis Statistics section at the bottom displays the % of libraries analyzed with Effective Usage Analysis, and the portion of alerts with or without a green shield.

TRACE ANALYSIS VIEW

Follow the path from the proprietary code (at the very bottom) to the vulnerable entity (at the very top). Trace details include file name, class name, method (function) name and the line number in the code where it is referenced.



Traces | Trace View

Selected Reference: (1) - com.fasterxml.jackson.databind.deser.BeanDeserializerFactory ()

Caller Traces (3)

Trace	Caller Type	Caller ID (hover for full text)
1	EXTENSION	(9)com.fasterxml.jackson.databind.deser.BeanDeserializerFactory:createBuilderBasedDeserializer (...\deser\BeanDeserializerFactory.class:188)
1	EXTENSION	(8)com.fasterxml.jackson.databind.deser.DeserializerCache:_createDeserializer (...\deser\DeserializerCache.class:318)
1	EXTENSION	(7)com.fasterxml.jackson.databind.deser.DeserializerCache:_createAndCache2 (...\deser\DeserializerCache.class:264)
1	EXTENSION	(6)com.fasterxml.jackson.databind.deser.DeserializerCache:_createAndCacheValueDeserializer (...\deser\DeserializerCache.class:228)
1	EXTENSION	(5)com.fasterxml.jackson.databind.deser.DeserializerCache:findValueDeserializer (...\deser\DeserializerCache.class:139)
1	EXTENSION	(4)com.fasterxml.jackson.databind.DeserializationContext:findRootValueDeserializer (...\databind\DeserializationContext.class:477)
1	EXTENSION	(3)com.fasterxml.jackson.databind.ObjectMapper:_findRootDeserializer (...\databind\ObjectMapper.class:4173)
1	EXTENSION	(2)com.fasterxml.jackson.databind.ObjectMapper:_readMapAndClose (...\databind\ObjectMapper.class:3986)
1	EXTENSION	(1)com.fasterxml.jackson.databind.ObjectMapper:readValue (...\databind\ObjectMapper.class:2890)
1	APPLICATION	(0)org.whitesource.fs.configuration.ConfigurationSerializer:load (...\configuration\ConfigurationSerializer.class:54)