sonatype

Open Source Security Intelligence is NOT Created Equal

A SIDE-BY-SIDE
COMPARISON OF
COMPETITORS
VS. NEXUS

The Challenge

The application architecture team within a large publicly traded utility company was investigating tools to help them automatically manage security risk associated with open source software components and third-party libraries. Specifically, the company wanted a solution that could help them rapidly identify and remediate application security risks across their entire SDLC whenever new open source vulnerabilities are publicly disclosed.

To evaluate potential partners, the company invited Sonatype and Black Duck to participate in a technical proof of concept by scanning one of their production applications. The company then compared the results of the scans side-by-side in order to determine which technology provided the most accurate results.

This document provides a detailed summary of what the company found when comparing the accuracy of Nexus Intelligence versus Black Duck.

Summary Report Card:

Nexus Intelligence vs Black Duck – December 2018

	Nexus Intelligence	Black Duck
Total Components Identified	154	159
Correctly Identified (True Positives)	148	148
Vulnerabilities Falsely Identified (False Positives)	0	7
Vulnerabilities Not Recognized (False Negatives)	0	20
Final Grade	A+	D

The Result

In an initial comparison of Sonatype's scan to the Black Duck scan, the number of components identified were nearly equal. However, upon a closer look, the utility company discovered material differences in the vulnerabilities reported and, most importantly, the overall quality of data.

The category of the differences fell into two categories:

1. False Positives – most related to the same vulnerability.

False positives are common in many security tools, and the bane of software engineers and security. False positives are reports of vulnerabilities where none exist. In open source security, this occurs most often when the vendor poorly matches components and misidentifies a secure component as a similar (vulnerable) component or incorrectly identifies which versions of a component were subject to a vulnerability.

The Black Duck scan did find seven vulnerabilities that were not highlighted in the Sonatype scan. Initially, the utility company felt that by not identifying the same list of potential vulnerabilities Black Duck did, Sonatype had somehow failed to report something meaningful.

However, digging past the surface, the case wasn't that Sonatype failed to identify those potential vulnerabilities, they came up in the scan. However, they were noise at the version level the utility company was using and therefore, were left out of the software Bill of Materials. Further to the point, Sonatype noted that **five of the seven** vulnerabilities reported on the Black Duck scan were duplicate copies of vulnerability **CVE-2017-7525** and were listed in an attempt to cover all use cases. While this is useful for research, it is important to note that with more precise data, it's possible to report **only** the correct version range and classes, creating more actionable and focused scan results. The bottom line? Sonatype found the same potential vulnerabilities, but precise data applied to the utility company's specific code base eliminated noise from the Nexus scan.

Unfortunately, with duplicate false positives that are presented as entirely new vulnerabilities, the utility company's developers would likely spend their valuable time researching and attempting to upgrade components that did not actually pose a threat, in this case, five times over. More inconceivably, the development team may be asked to chase down duplicate false positives when real threats (false negatives) exist in the code base but were not detected by the Black Duck scan.

In contrast, as a result of Sonatype's curated, analyzed, and fingerprinted data, duplicate vulnerabilities were matched to one "root" vulnerability and omitted from the scan results so the development team could move past the noise and focus their energy on the vulnerabilities that pose a serious threat.

False Positives Identified by Black Duck

Vulnerability Namew	Type	Findings
1. CVE-2013-7285	False positive. Wrong version.	Black Duck and Sonatype both identified. Black Duck implicated and Sonatype did not.
2. CVE-2017-15095	Duplicate of **CVE-2017-7525**	Black Duck False Positive, Sonatype intentionally left out
3. CVE-2017-17485	Duplicate of **CVE-2017-7525**	Black Duck False Positive, Sonatype intentionally left out
4. CVE-2017- 4995	Duplicate of **CVE-2017-7525**	Black Duck False Positive, Sonatype intentionally left out
5. CVE-2018-7489	Duplicate of **CVE-2017-7525**	Black Duck False Positive, Sonatype intentionally left out
6. CVE-2018-5968	Duplicate of **CVE-2017-7525**	Black Duck False Positive, Sonatype intentionally left out
7. CVE-2018-1000613	False positive. Wrong version.	Black Duck and Sonatype both identified. Black Duck implicated and Sonatype did not

Broad Scope and False Recognition

As highlighted above, of the seven false positives detected, the first one was a version error, two through six were duplicates of a "root" vulnerability that were correctly identified, and the seventh was a separate vulnerability, but still the wrong version.

- **Vulnerability 1:** This vulnerability was associated with xstream-1.4.9 which both the Nexus and the Black Duck scan identified in the Bill of Materials. Sonatype's research shows, though, that this issue doesn't exist in versions past 1.4.6, making this identification an error. Another nuance to note is that when this vulnerability was posted to NVD, only **CVSS 2** existed and as such, it has a score of 5.5. With **CVSS 3**, this issue is in the 9's.

- **Vulnerabilities 2-6:** These were duplicates of correctly identified vuln CVE 2017-7525. However, even though the vulnerability was a true positive, the cascading components that were flagged afterwards as being vulnerable were simply duplicates of the original and therefore, "noise".

- **Vulnerability 7:** According to the Black Duck scan, the potential client was using version **1.55** of Bouncy Castle. However, Sonatype's deeper research shows that this problem was introduced when the XMSS/XMSS^MF functionality was added. That functionality was added in version **1.57** of this package, therefore, Sonatype did not flag version 1.55 as vulnerable.

2. False Negatives-- missed vulnerabilities, ticking time bombs.

False negatives are vulnerabilities that exist in an application but were not detected in testing. A tool that misses vulnerabilities leaves a company exposed to attacks but with an unwarranted sense of security. In open source security, false negatives can occur because: a) the vendor could not detect the presence of the component against which a vulnerability was reported; b) the vendor could not associate secondary (transitive) dependencies used by the components it recognized; c) the vendor incorrectly identified which versions of a component were subject to a vulnerability, or d) the vendor used an incomplete vulnerability database that did not include the vulnerabilities.

In a comparison of the two scans, Sonatype's Nexus Lifecycle scan identified and **implicated twenty vulnerable components** the Black Duck scan did not. Of those twenty:

- **7** vulnerabilities were valid CVEs against components the Black Duck scan did identify. However, while the Black Duck scan recognized these components in the utility company's application, the Black Duck scan did not implicate those components as being vulnerable. Effectively, misidentifying them as "safe".
- **10** vulnerabilities were associated with components the **Black Duck scan did not identify at all.** Since they did not show up in the scan at all, developers wouldn't know those components were vulnerable until they were compromised.
- **3** of them were Sonatype proprietary vulnerabilities against components the Black Duck scan identified, but again did not implicate as being vulnerable – misidentified as "safe".
- **8** of the vulnerabilities had CVSS scores of **7** or greater
- **14 of them were against tomcat and Spring Framework components, which were the backbone of this potential customer's application.**

The stats made it clear: If the utility company chose Black Duck in this case, future scans might not be able to comprehensively detect security vulnerabilities within their open source components.

One very good example of a missed false negative from the comparative scans is **CVE-2018-11040**, the CPE implicated the *spring_framework* group of packages. The Black Duck scan, being name-based, did not take into consideration *springframework* (no underscore). In this case, a Central (Maven) Search indicated that spring-web belongs to the springframework (no underscore) group and the vulnerability went completely undetected by the Black Duck scan. Something as simple as a character oversight or misplacement can trigger an entire series of false negatives. This "miss" proves that name-based or entirely automated component matching may put application security at risk. **Precision matters**.

The Decision

When it comes to using open source components to manufacture modern software, the bottom line is this – precise intelligence is critical. After making a side-by-side comparison of the two scans, the utility company recognized that inaccurate or incomplete data would leave their developers struggling with vulnerabilities and other quality issues that would lead directly to higher costs, reduced innovation, and reputation risks.

In addition, Black Duck failed to offer in-depth remediation guidance which would have left the developers sourcing that guidance on their own. Only Sonatype includes in-depth remediation guidance written in a language easily understood by developers, so they can reduce MTTR (mean time to resolution) and innovate faster.

This large, publicly traded utility company chose Sonatype's Nexus Lifecycle for end-to-end security automation across their DevOps pipeline. Sonatype's Nexus Intelligence with Advanced Binary Fingerprinting powers Nexus Lifecycle to help organizations:

- Precisely identify the highest quality open source components.
- Scale fast and automate open source governance at every phase of the software development lifecycle.
- Control component usage with flexible policy enforcement across varying teams, languages, and application profiles.

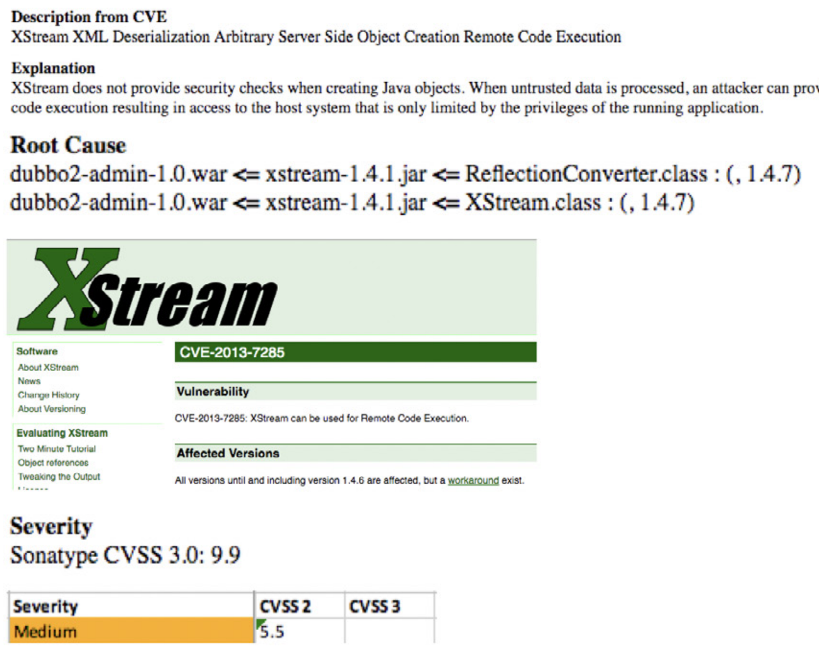
With precise identification, organizations have the power to error-proof the software supply chain. This means eliminating-- with certainty-- the risks and inefficiencies that diminish innovation, while unlocking the full potential of talented developers to build better.

We welcome any questions that you might have about this case study and encourage you to sample the incredible value of Nexus Intelligence. Test your own app with the **Nexus Vulnerability Scanner** today to see how our precise data can help your team stay ahead of the threat.

Appendix

CVE-2013-7285

It was found that XStream could deserialize arbitrary user-supplied XML content, representing objects of any type. A remote attacker able to pass XML to XStream could use this flaw to perform a variety of attacks, including remote code execution in the context of the server running the XStream application.



Description from CVE
XStream XML Deserialization Arbitrary Server Side Object Creation Remote Code Execution

Explanation
XStream does not provide security checks when creating Java objects. When untrusted data is processed, an attacker can provide crafted data that allows arbitrary code execution resulting in access to the host system that is only limited by the privileges of the running application.

Root Cause
dubbo2-admin-1.0.war <= xstream-1.4.1.jar <= ReflectionConverter.class : (, 1.4.7)
dubbo2-admin-1.0.war <= xstream-1.4.1.jar <= XStream.class : (, 1.4.7)

XStream

Software
About XStream
News
Change History
About Versioning

Evaluating XStream
Two Minute Tutorial
Object references
Tweaking the Output

CVE-2013-7285

Vulnerability

CVE-2013-7285: XStream can be used for Remote Code Execution.

Affected Versions

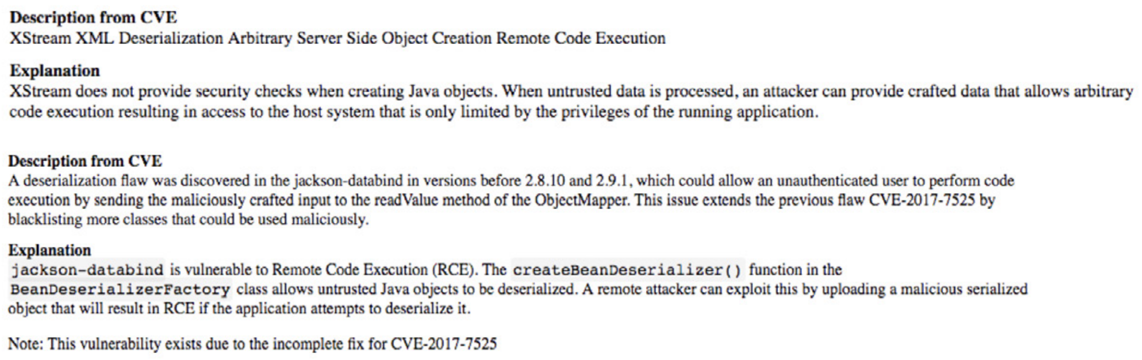
All versions until and including version 1.4.6 are affected, but a workaround exist.

Severity
Sonatype CVSS 3.0: 9.9

Severity	CVSS 2	CVSS 3
Medium	5.5	

https://access.redhat.com/security/cve/cve-2013-7285

CVE-2017-7525 a true positive and the cascading duplicates:



Description from CVE
XStream XML Deserialization Arbitrary Server Side Object Creation Remote Code Execution

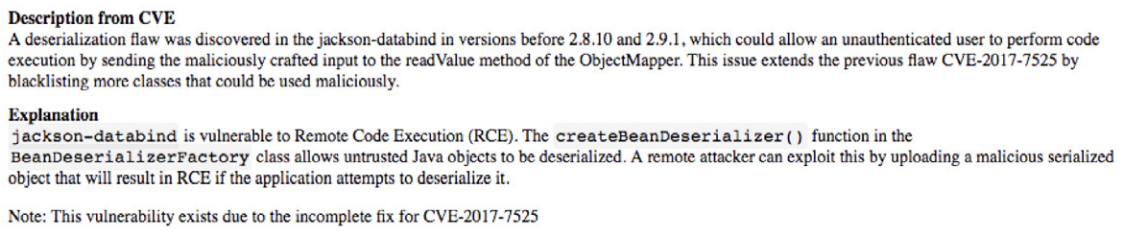
Explanation
XStream does not provide security checks when creating Java objects. When untrusted data is processed, an attacker can provide crafted data that allows arbitrary code execution resulting in access to the host system that is only limited by the privileges of the running application.

Description from CVE
A deserialization flaw was discovered in the jackson-databind in versions before 2.8.10 and 2.9.1, which could allow an unauthenticated user to perform code execution by sending the maliciously crafted input to the readValue method of the ObjectMapper. This issue extends the previous flaw CVE-2017-7525 by blacklisting more classes that could be used maliciously.

Explanation
`jackson-databind` is vulnerable to Remote Code Execution (RCE). The `createBeanDeserializer()` function in the `BeanDeserializerFactory` class allows untrusted Java objects to be deserialized. A remote attacker can exploit this by uploading a malicious serialized object that will result in RCE if the application attempts to deserialize it.

Note: This vulnerability exists due to the incomplete fix for CVE-2017-7525

CVE- 2017-15095

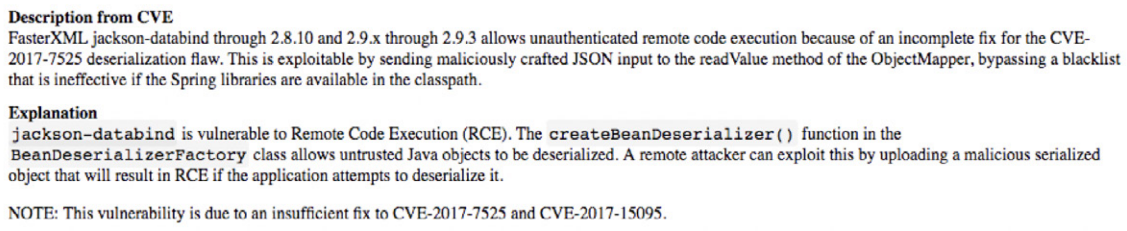


Description from CVE
A deserialization flaw was discovered in the jackson-databind in versions before 2.8.10 and 2.9.1, which could allow an unauthenticated user to perform code execution by sending the maliciously crafted input to the readValue method of the ObjectMapper. This issue extends the previous flaw CVE-2017-7525 by blacklisting more classes that could be used maliciously.

Explanation
`jackson-databind` is vulnerable to Remote Code Execution (RCE). The `createBeanDeserializer()` function in the `BeanDeserializerFactory` class allows untrusted Java objects to be deserialized. A remote attacker can exploit this by uploading a malicious serialized object that will result in RCE if the application attempts to deserialize it.

Note: This vulnerability exists due to the incomplete fix for CVE-2017-7525

CVE-2017-17485



Description from CVE
FasterXML jackson-databind through 2.8.10 and 2.9.x through 2.9.3 allows unauthenticated remote code execution because of an incomplete fix for the CVE-2017-7525 deserialization flaw. This is exploitable by sending maliciously crafted JSON input to the readValue method of the ObjectMapper, bypassing a blacklist that is ineffective if the Spring libraries are available in the classpath.

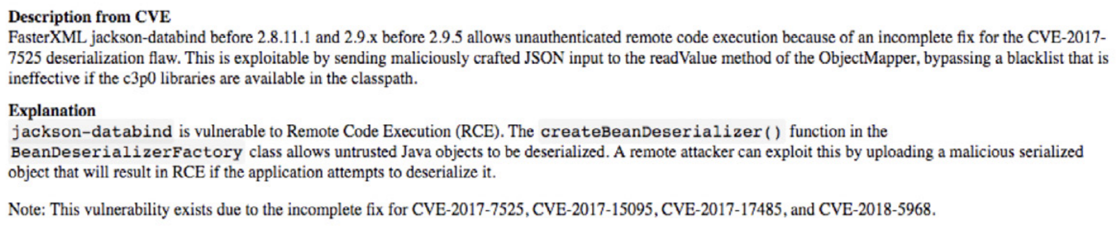
Explanation
`jackson-databind` is vulnerable to Remote Code Execution (RCE). The `createBeanDeserializer()` function in the `BeanDeserializerFactory` class allows untrusted Java objects to be deserialized. A remote attacker can exploit this by uploading a malicious serialized object that will result in RCE if the application attempts to deserialize it.

NOTE: This vulnerability is due to an insufficient fix to CVE-2017-7525 and CVE-2017-15095.

CVE-2017-4995 – Listed in GitHub as a duplicate of CVE-2017-7525

Reference: https://github.com/FasterXML/jackson-databind/issues/1599

CVE-2018-7489



Description from CVE
FasterXML jackson-databind before 2.8.11.1 and 2.9.x before 2.9.5 allows unauthenticated remote code execution because of an incomplete fix for the CVE-2017-7525 deserialization flaw. This is exploitable by sending maliciously crafted JSON input to the readValue method of the ObjectMapper, bypassing a blacklist that is ineffective if the c3p0 libraries are available in the classpath.

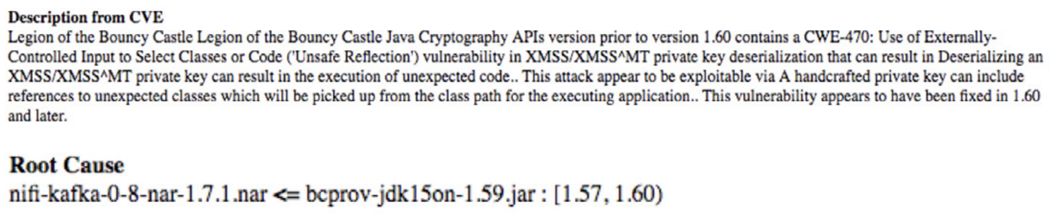
Explanation
`jackson-databind` is vulnerable to Remote Code Execution (RCE). The `createBeanDeserializer()` function in the `BeanDeserializerFactory` class allows untrusted Java objects to be deserialized. A remote attacker can exploit this by uploading a malicious serialized object that will result in RCE if the application attempts to deserialize it.

Note: This vulnerability exists due to the incomplete fix for CVE-2017-7525, CVE-2017-15095, CVE-2017-17485, and CVE-2018-5968.

CVE-2018-5968

https://nvd.nist.gov/vuln/detail/CVE-2018-7489

CVE-2018-1000613

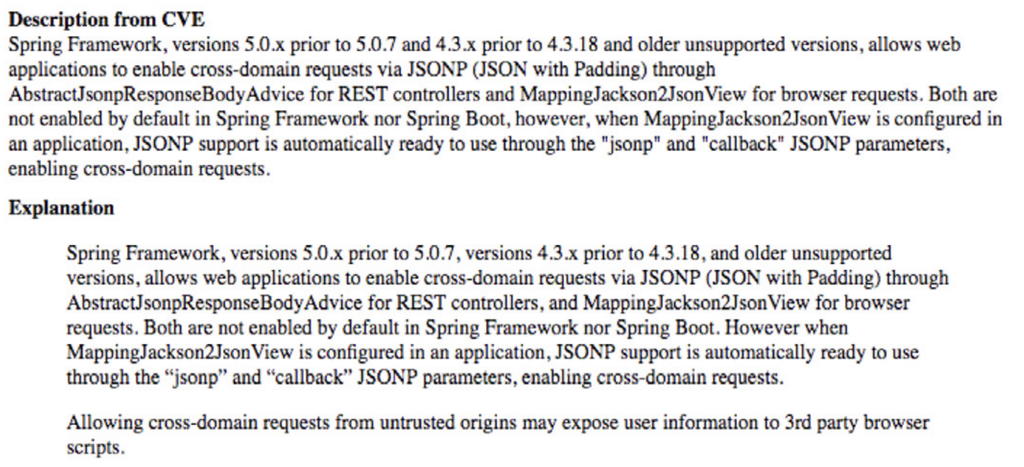


Description from CVE
Legion of the Bouncy Castle Legion of the Bouncy Castle Java Cryptography APIs version prior to version 1.60 contains a CWE-470: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection') vulnerability in XMSS/XMSS^MT private key deserialization that can result in Deserializing an XMSS/XMSS^MT private key can result in the execution of unexpected code.. This attack appear to be exploitable via A handcrafted private key can include references to unexpected classes which will be picked up from the class path for the executing application.. This vulnerability appears to have been fixed in 1.60 and later.

Root Cause
nifi-kafka-0-8-nar-1.7.1.nar <= bcprov-jdk15on-1.59.jar : [1.57, 1.60)

https://www.bouncycastle.org/releasenotes.html

CVE-2018-11040



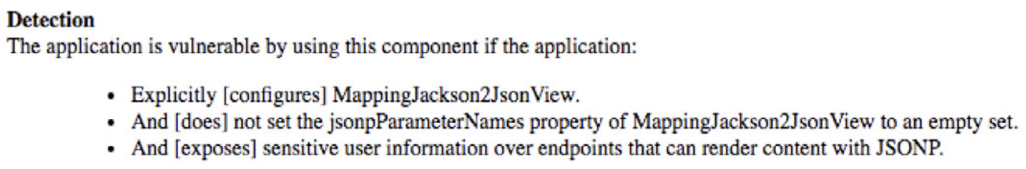
Description from CVE
Spring Framework, versions 5.0.x prior to 5.0.7 and 4.3.x prior to 4.3.18 and older unsupported versions, allows web applications to enable cross-domain requests via JSONP (JSON with Padding) through AbstractJsonpResponseBodyAdvice for REST controllers and MappingJackson2JsonView for browser requests. Both are not enabled by default in Spring Framework nor Spring Boot, however, when MappingJackson2JsonView is configured in an application, JSONP support is automatically ready to use through the "jsonp" and "callback" JSONP parameters, enabling cross-domain requests.

Explanation

Spring Framework, versions 5.0.x prior to 5.0.7, versions 4.3.x prior to 4.3.18, and older unsupported versions, allows web applications to enable cross-domain requests via JSONP (JSON with Padding) through AbstractJsonpResponseBodyAdvice for REST controllers, and MappingJackson2JsonView for browser requests. Both are not enabled by default in Spring Framework nor Spring Boot. However when MappingJackson2JsonView is configured in an application, JSONP support is automatically ready to use through the “jsonp” and “callback” JSONP parameters, enabling cross-domain requests.

Allowing cross-domain requests from untrusted origins may expose user information to 3rd party browser scripts.

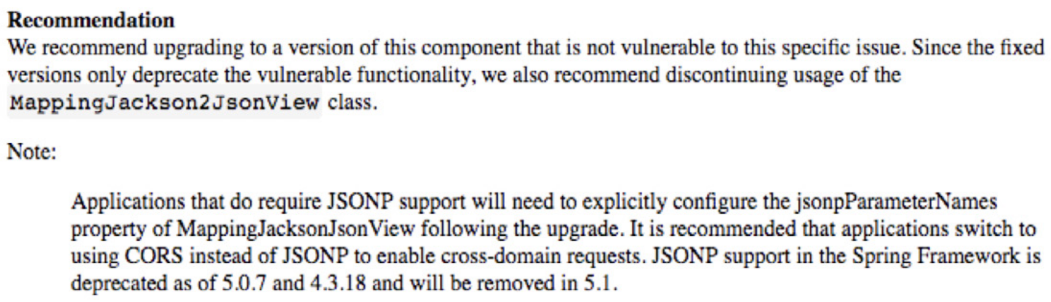
Reference: https://pivotal.io/security/cve-2018-11040



Detection
The application is vulnerable by using this component if the application:

- Explicitly [configures] MappingJackson2JsonView.
- And [does] not set the jsonpParameterNames property of MappingJackson2JsonView to an empty set.
- And [exposes] sensitive user information over endpoints that can render content with JSONP.

Reference: https://pivotal.io/security/cve-2018-11040

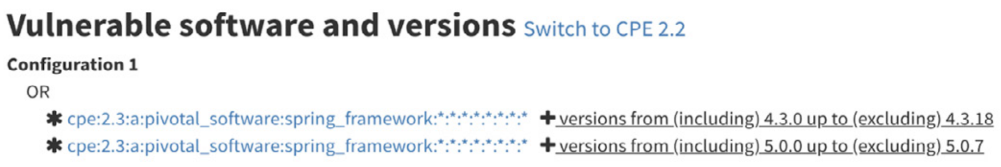


Recommendation
We recommend upgrading to a version of this component that is not vulnerable to this specific issue. Since the fixed versions only deprecate the vulnerable functionality, we also recommend discontinuing usage of the `MappingJackson2JsonView` class.

Note:

Applications that do require JSONP support will need to explicitly configure the jsonpParameterNames property of MappingJacksonJsonView following the upgrade. It is recommended that applications switch to using CORS instead of JSONP to enable cross-domain requests. JSONP support in the Spring Framework is deprecated as of 5.0.7 and 4.3.18 and will be removed in 5.1.

Reference: https://pivotal.io/security/cve-2018-11040

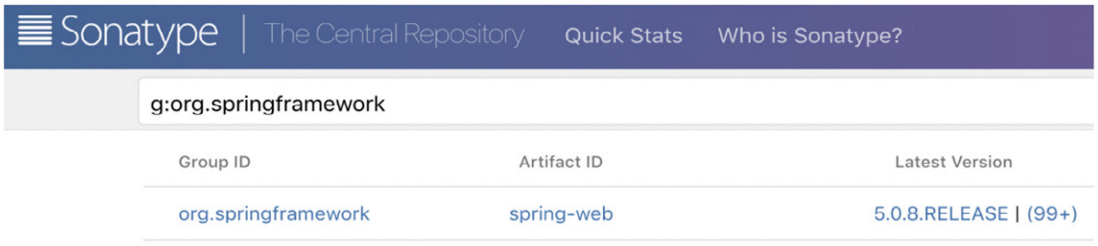


Vulnerable software and versions Switch to CPE 2.2

Configuration 1

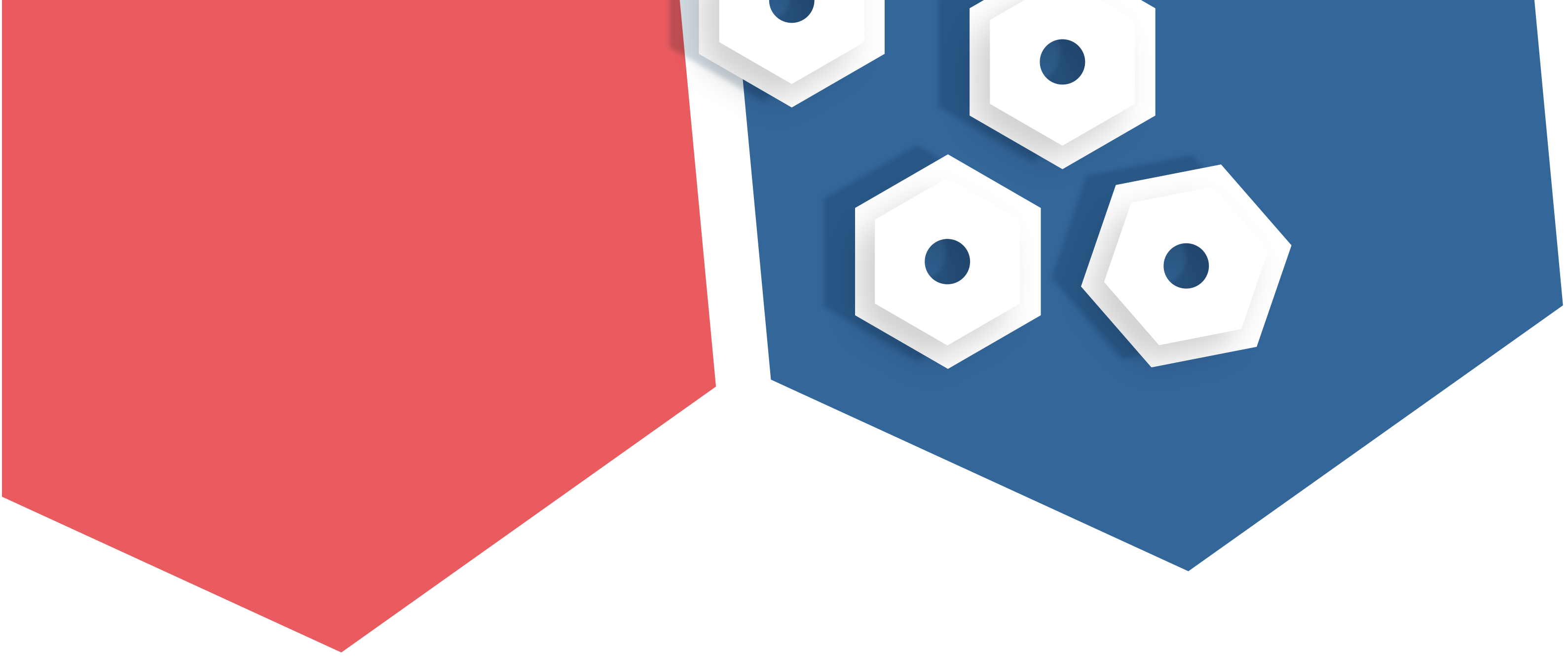
OR

✱ cpe:2.3:a:pivotal_software:spring_framework:*:*:*:*:*:*:*:* ✚ versions from (including) 4.3.0 up to (excluding) 4.3.18
✱ cpe:2.3:a:pivotal_software:spring_framework:*:*:*:*:*:*:*:* ✚ versions from (including) 5.0.0 up to (excluding) 5.0.7



g:org.springframework

Group ID	Artifact ID	Latest Version
org.springframework	spring-web	5.0.8.RELEASE | (99+)



sonatype

More than 10 million software developers rely on Sonatype to innovate faster while mitigating security risks inherent in open source. Sonatype's Nexus platform combines in-depth component intelligence with real-time remediation guidance to automate and scale open source governance across every stage of the modern DevOps pipeline. Sonatype is privately held with investments from TPG, Goldman Sachs, Accel Partners, and Hummer Winblad Venture Partners. Learn more at www.sonatype.com

Headquarters	**European Office**	**APAC Office**	**Sonatype Inc.**
8161 Maple Lawn Blvd, Suite 250	1 Primrose Street	5 Martin Place, Level 14	www.sonatype.com
Fulton, MD 20759	London EC2A 2EX	Sydney 2000, NSW	Sonatype Copyright 2019
United States – 1.877.866.2836	United Kingdom	Australia	All Rights Reserved.