

Using Google gson vs FasterXML Jackson

Contributors

Keong Lim ,Tian Lee

References

2018-10-03 AAI Meeting Notes

2018-09-26 AAI Meeting Notes

PTL 2018-11-05

Project Recommendations for Package Upgrades

AAI R3 Security/Vulnerability Threat Analysis

Faster XML Jackson usage in Portal Code and replacing it with Gson

☒ **AAI-628** - Review security issues: com.fasterxml.jackson.core (CVE-2017-7525) **CLOSED**

☐ **AAI-908** - Security: CVE-2017-7525 jackson-mapper-asl:1.9.13 **CLOSED**

☐ **AAI-940** - Security: CVE-2017-7525 jackson-databind:2.2.3 **CLOSED**

☐ **AAI-928** - Security: CVE-2017-17485 jackson-databind 2.8.10 **CLOSED**

☐ **AAI-1218** - [Champ] Fix JSON serialization of properties in update-notification-raw payload **CLOSED**

Seccom Recommendations



15th July 2019: "Out of Scope" as per [Fixing Vulnerabilities in the ONAP Code Base](#)

Jackson Replacement

Security subcommittee has recommended teams move away from jackson, and will be presenting alternatives and asking for an assessment from each project. Our team will need to do an analysis - this would not be trivial, especially given how many of our repos are impacted. As of now, this would be a very high LOE for the team, we need to understand what the recommendation from the SECCOM is before we can provide better details on what the LOE would be.

Vulnerable Packages per ONAP Project

All versions of all packages listed below are associated to a critical or severe CVE.				
Packages that Affect Most ONAP Projects	Version Count	Projects Affected	Latest Version	Notes
com.fasterxml.jackson.core : jackson-databind :	24	21	2.9.6	Use Jackson XML handler instead of the default XStream handler as described here
com.google.guava : guava :	17	21	25.1-jre	
org.eclipse.jetty : jetty-http :	13	5	9.4.11.v20180605	CVE-2018-12525, CVE-2018-12526
org.springframework : spring-core :	13	13	5.0.8	
org.springframework : spring-expression :	18	13	5.0.8	
org.springframework : spring-web :	18	13	5.0.8	
commons-beanutils : commons-beanutils :	5	11	1.9.3	
org.apache.tomcat.embed : tomcat-embed-core :	5	11	9.0.10	
org.eclipse.jetty : jetty-servlet :	11	11	9.4.12.RC0	
org.springframework : spring-webmvc :	13	11	5.0.8	
org.codehaus.jackson : jackson-mapper-asl :	5	10	1.9.13.3	
org.webjars : bootstrap :	6	9	4.1.2	
com.fasterxml.jackson.core : jackson-core :	11	9	9.9.6	
org.webjars : jquery :	10	9	3.3.1-1	
ch.qos.logback : logback-classic :	4	8	1.2.3	
angular :	7	7	7	1.9 are all alpha releases
org.apache.httpcomponents : httpclient :	5	7	4.5.6	
org.eclipse.jetty : jetty-server :	7	7	9.4.12RC0	
xerces : xercesimpl :	4	7	12.2.0	



Vulnerable Packages per ONAP Project

All versions of all packages listed below are associated to a critical or severe CVE.		
Packages that Affect Most ONAP Projects	Version Count	List of Versions
com.fasterxml.jackson.core : jackson-databind :	24	2.2.2, 2.4.4, 2.4.5, 2.5.1, 2.5.2, 2.5.3, 2.5.4, 2.5.5, 2.5.6, 2.5.7, 2.5.8, 2.5.9, 2.5.10, 2.5.11, 2.5.12, 2.5.13, 2.5.14, 2.5.15, 2.5.16, 2.5.17, 2.5.18, 2.5.19, 2.5.20, 2.5.21, 2.5.22, 2.5.23, 2.5.24, 2.5.25, 2.5.26
com.google.guava : guava :	17	14.0, 14.1, 14.2, 14.3, 14.4, 14.5, 14.6, 14.7, 14.8, 14.9, 14.10, 14.11, 14.12, 14.13, 14.14, 14.15, 14.16, 14.17, 14.18, 14.19, 14.20, 14.21, 14.22, 14.23, 14.24, 14.25, 14.26, 14.27, 14.28, 14.29, 14.30, 14.31, 14.32, 14.33, 14.34, 14.35, 14.36, 14.37, 14.38, 14.39, 14.40, 14.41, 14.42, 14.43, 14.44, 14.45, 14.46, 14.47, 14.48, 14.49, 14.50, 14.51, 14.52, 14.53, 14.54, 14.55, 14.56, 14.57, 14.58, 14.59, 14.60, 14.61, 14.62, 14.63, 14.64, 14.65, 14.66, 14.67, 14.68, 14.69, 14.70, 14.71, 14.72, 14.73, 14.74, 14.75, 14.76, 14.77, 14.78, 14.79, 14.80, 14.81, 14.82, 14.83, 14.84, 14.85, 14.86, 14.87, 14.88, 14.89, 14.90, 14.91, 14.92, 14.93, 14.94, 14.95, 14.96, 14.97, 14.98, 14.99, 15.0, 15.1, 15.2, 15.3, 15.4, 15.5, 15.6, 15.7, 15.8, 15.9, 16.0, 16.1, 16.2, 16.3, 16.4, 16.5, 16.6, 16.7, 16.8, 16.9, 17.0, 17.1, 17.2, 17.3, 17.4, 17.5, 17.6, 17.7, 17.8, 17.9, 18.0, 18.1, 18.2, 18.3, 18.4, 18.5, 18.6, 18.7, 18.8, 18.9, 19.0, 19.1, 19.2, 19.3, 19.4, 19.5, 19.6, 19.7, 19.8, 19.9, 20.0, 20.1, 20.2, 20.3, 20.4, 20.5, 20.6, 20.7, 20.8, 20.9, 21.0, 21.1, 21.2, 21.3, 21.4, 21.5, 21.6, 21.7, 21.8, 21.9, 22.0, 22.1, 22.2, 22.3, 22.4, 22.5, 22.6, 22.7, 22.8, 22.9, 23.0, 23.1, 23.2, 23.3, 23.4, 23.5, 23.6, 23.7, 23.8, 23.9, 24.0, 24.1, 24.2, 24.3, 24.4, 24.5, 24.6, 24.7, 24.8, 24.9, 25.0, 25.1, 25.2, 25.3, 25.4, 25.5, 25.6, 25.7, 25.8, 25.9, 26.0, 26.1, 26.2, 26.3, 26.4, 26.5, 26.6, 26.7, 26.8, 26.9, 27.0, 27.1, 27.2, 27.3, 27.4, 27.5, 27.6, 27.7, 27.8, 27.9, 28.0, 28.1, 28.2, 28.3, 28.4, 28.5, 28.6, 28.7, 28.8, 28.9, 29.0, 29.1, 29.2, 29.3, 29.4, 29.5, 29.6, 29.7, 29.8, 29.9, 30.0, 30.1, 30.2, 30.3, 30.4, 30.5, 30.6, 30.7, 30.8, 30.9, 31.0, 31.1, 31.2, 31.3, 31.4, 31.5, 31.6, 31.7, 31.8, 31.9, 32.0, 32.1, 32.2, 32.3, 32.4, 32.5, 32.6, 32.7, 32.8, 32.9, 33.0, 33.1, 33.2, 33.3, 33.4, 33.5, 33.6, 33.7, 33.8, 33.9, 34.0, 34.1, 34.2, 34.3, 34.4, 34.5, 34.6, 34.7, 34.8, 34.9, 35.0, 35.1, 35.2, 35.3, 35.4, 35.5, 35.6, 35.7, 35.8, 35.9, 36.0, 36.1, 36.2, 36.3, 36.4, 36.5, 36.6, 36.7, 36.8, 36.9, 37.0, 37.1, 37.2, 37.3, 37.4, 37.5, 37.6, 37.7, 37.8, 37.9, 38.0, 38.1, 38.2, 38.3, 38.4, 38.5, 38.6, 38.7, 38.8, 38.9, 39.0, 39.1, 39.2, 39.3, 39.4, 39.5, 39.6, 39.7, 39.8, 39.9, 40.0, 40.1, 40.2, 40.3, 40.4, 40.5, 40.6, 40.7, 40.8, 40.9, 41.0, 41.1, 41.2, 41.3, 41.4, 41.5, 41.6, 41.7, 41.8, 41.9, 42.0, 42.1, 42.2, 42.3, 42.4, 42.5, 42.6, 42.7, 42.8, 42.9, 43.0, 43.1, 43.2, 43.3, 43.4, 43.5, 43.6, 43.7, 43.8, 43.9, 44.0, 44.1, 44.2, 44.3, 44.4, 44.5, 44.6, 44.7, 44.8, 44.9, 45.0, 45.1, 45.2, 45.3, 45.4, 45.5, 45.6, 45.7, 45.8, 45.9, 46.0, 46.1, 46.2, 46.3, 46.4, 46.5, 46.6, 46.7, 46.8, 46.9, 47.0, 47.1, 47.2, 47.3, 47.4, 47.5, 47.6, 47.7, 47.8, 47.9, 48.0, 48.1, 48.2, 48.3, 48.4, 48.5, 48.6, 48.7, 48.8, 48.9, 49.0, 49.1, 49.2, 49.3, 49.4, 49.5, 49.6, 49.7, 49.8, 49.9, 50.0, 50.1, 50.2, 50.3, 50.4, 50.5, 50.6, 50.7, 50.8, 50.9, 51.0, 51.1, 51.2, 51.3, 51.4, 51.5, 51.6, 51.7, 51.8, 51.9, 52.0, 52.1, 52.2, 52.3, 52.4, 52.5, 52.6, 52.7, 52.8, 52.9, 53.0, 53.1, 53.2, 53.3, 53.4, 53.5, 53.6, 53.7, 53.8, 53.9, 54.0, 54.1, 54.2, 54.3, 54.4, 54.5, 54.6, 54.7, 54.8, 54.9, 55.0, 55.1, 55.2, 55.3, 55.4, 55.5, 55.6, 55.7, 55.8, 55.9, 56.0, 56.1, 56.2, 56.3, 56.4, 56.5, 56.6, 56.7, 56.8, 56.9, 57.0, 57.1, 57.2, 57.3, 57.4, 57.5, 57.6, 57.7, 57.8, 57.9, 58.0, 58.1, 58.2, 58.3, 58.4, 58.5, 58.6, 58.7, 58.8, 58.9, 59.0, 59.1, 59.2, 59.3, 59.4, 59.5, 59.6, 59.7, 59.8, 59.9, 60.0, 60.1, 60.2, 60.3, 60.4, 60.5, 60.6, 60.7, 60.8, 60.9, 61.0, 61.1, 61.2, 61.3, 61.4, 61.5, 61.6, 61.7, 61.8, 61.9, 62.0, 62.1, 62.2, 62.3, 62.4, 62.5, 62.6, 62.7, 62.8, 62.9, 63.0, 63.1, 63.2, 63.3, 63.4, 63.5, 63.6, 63.7, 63.8, 63.9, 64.0, 64.1, 64.2, 64.3, 64.4, 64.5, 64.6, 64.7, 64.8, 64.9, 65.0, 65.1, 65.2, 65.3, 65.4, 65.5, 65.6, 65.7, 65.8, 65.9, 66.0, 66.1, 66.2, 66.3, 66.4, 66.5, 66.6, 66.7, 66.8, 66.9, 67.0, 67.1, 67.2, 67.3, 67.4, 67.5, 67.6, 67.7, 67.8, 67.9, 68.0, 68.1, 68.2, 68.3, 68.4, 68.5, 68.6, 68.7, 68.8, 68.9, 69.0, 69.1, 69.2, 69.3, 69.4, 69.5, 69.6, 69.7, 69.8, 69.9, 70.0, 70.1, 70.2, 70.3, 70.4, 70.5, 70.6, 70.7, 70.8, 70.9, 71.0, 71.1, 71.2, 71.3, 71.4, 71.5, 71.6, 71.7, 71.8, 71.9, 72.0, 72.1, 72.2, 72.3, 72.4, 72.5, 72.6, 72.7, 72.8, 72.9, 73.0, 73.1, 73.2, 73.3, 73.4, 73.5, 73.6, 73.7, 73.8, 73.9, 74.0, 74.1, 74.2, 74.3, 74.4, 74.5, 74.6, 74.7, 74.8, 74.9, 75.0, 75.1, 75.2, 75.3, 75.4, 75.5, 75.6, 75.7, 75.8, 75.9, 76.0, 76.1, 76.2, 76.3, 76.4, 76.5, 76.6, 76.7, 76.8, 76.9, 77.0, 77.1, 77.2, 77.3, 77.4, 77.5, 77.6, 77.7, 77.8, 77.9, 78.0, 78.1, 78.2, 78.3, 78.4, 78.5, 78.6, 78.7, 78.8, 78.9, 79.0, 79.1, 79.2, 79.3, 79.4, 79.5, 79.6, 79.7, 79.8, 79.9, 80.0, 80.1, 80.2, 80.3, 80.4, 80.5, 80.6, 80.7, 80.8, 80.9, 81.0, 81.1, 81.2, 81.3, 81.4, 81.5, 81.6, 81.7, 81.8, 81.9, 82.0, 82.1, 82.2, 82.3, 82.4, 82.5, 82.6, 82.7, 82.8, 82.9, 83.0, 83.1, 83.2, 83.3, 83.4, 83.5, 83.6, 83.7, 83.8, 83.9, 84.0, 84.1, 84.2, 84.3, 84.4, 84.5, 84.6, 84.7, 84.8, 84.9, 85.0, 85.1, 85.2, 85.3, 85.4, 85.5, 85.6, 85.7, 85.8, 85.9, 86.0, 86.1, 86.2, 86.3, 86.4, 86.5, 86.6, 86.7, 86.8, 86.9, 87.0, 87.1, 87.2, 87.3, 87.4, 87.5, 87.6, 87.7, 87.8, 87.9, 88.0, 88.1, 88.2, 88.3, 88.4, 88.5, 88.6, 88.7, 88.8, 88.9, 89.0, 89.1, 89.2, 89.3, 89.4, 89.5, 89.6, 89.7, 89.8, 89.9, 90.0, 90.1, 90.2, 90.3, 90.4, 90.5, 90.6, 90.7, 90.8, 90.9, 91.0, 91.1, 91.2, 91.3, 91.4, 91.5, 91.6, 91.7, 91.8, 91.9, 92.0, 92.1, 92.2, 92.3, 92.4, 92.5, 92.6, 92.7, 92.8, 92.9, 93.0, 93.1, 93.2, 93.3, 93.4, 93.5, 93.6, 93.7, 93.8, 93.9, 94.0, 94.1, 94.2, 94.3, 94.4, 94.5, 94.6, 94.7, 94.8, 94.9, 95.0, 95.1, 95.2, 95.3, 95.4, 95.5, 95.6, 95.7, 95.8, 95.9, 96.0, 96.1, 96.2, 96.3, 96.4, 96.5, 96.6, 96.7, 96.8, 96.9, 97.0, 97.1, 97.2, 97.3, 97.4, 97.5, 97.6, 97.7, 97.8, 97.9, 98.0, 98.1, 98.2, 98.3, 98.4, 98.5, 98.6, 98.7, 98.8, 98.9, 99.0, 99.1, 99.2, 99.3, 99.4, 99.5, 99.6, 99.7, 99.8, 99.9, 100.0, 100.1, 100.2, 100.3, 100.4, 100.5, 100.6, 100.7, 100.8, 100.9, 101.0, 101.1, 101.2, 101.3, 101.4, 101.5, 101.6, 101.7, 101.8, 101.9, 102.0, 102.1, 102.2, 102.3, 102.4, 102.5, 102.6, 102.7, 102.8, 102.9, 103.0, 103.1, 103.2, 103.3, 103.4, 103.5, 103.6, 103.7, 103.8, 103.9, 104.0, 104.1, 104.2, 104.3, 104.4, 104.5, 104.6, 104.7, 104.8, 104.9, 105.0, 105.1, 105.2, 105.3, 105.4, 105.5, 105.6, 105.7, 105.8, 105.9, 106.0, 106.1, 106.2, 106.3, 106.4, 106.5, 106.6, 106.7, 106.8, 106.9, 107.0, 107.1, 107.2, 107.3, 107.4, 107.5, 107.6, 107.7, 107.8, 107.9, 108.0, 108.1, 108.2, 108.3, 108.4, 108.5, 108.6, 108.7, 108.8, 108.9, 109.0, 109.1, 109.2, 109.3, 109.4, 109.5, 109.6, 109.7, 109.8, 109.9, 110.0, 110.1, 110.2, 110.3, 110.4, 110.5, 110.6, 110.7, 110.8, 110.9, 111.0, 111.1, 111.2, 111.3, 111.4, 111.5, 111.6, 111.7, 111.8, 111.9, 112.0, 112.1, 112.2, 112.3, 112.4, 112.5, 112.6, 112.7, 112.8, 112.9, 113.0, 113.1, 113.2, 113.3, 113.4, 113.5, 113.6, 113.7, 113.8, 113.9, 114.0, 114.1, 114.2, 114.3, 114.4, 114.5, 114.6, 114.7, 114.8, 114.9, 115.0, 115.1, 115.2, 115.3, 115.4, 115.5, 115.6, 115.7, 115.8, 115.9, 116.0, 116.1, 116.2, 116.3, 116.4, 116.5, 116.6, 116.7, 116.8, 116.9, 117.0, 117.1, 117.2, 117.3, 117.4, 117.5, 117.6, 117.7, 117.8, 117.9, 118.0, 118.1, 118.2, 118.3, 118.4, 118.5, 118.6, 118.7, 118.8, 118.9, 119.0, 119.1, 119.2, 119.3, 119.4, 119.5, 119.6, 119.7, 119.8, 119.9, 120.0, 120.1, 120.2, 120.3, 120.4, 120.5, 120.6, 120.7, 120.8, 120.9, 121.0, 121.1, 121.2, 121.3, 121.4, 121.5, 121.6, 121.7, 121.8, 121.9, 122.0, 122.1, 122.2, 122.3, 122.4, 122.5, 122.6, 122.7, 122.8, 122.9, 123.0, 123.1, 123.2, 123.3, 123.4, 123.5, 123.6, 123.7, 123.8, 123.9, 124.0, 124.1, 124.2, 124.3, 124.4, 124.5, 124.6, 124.7, 124.8, 124.9, 125.0, 125.1, 125.2, 125.3, 125.4, 125.5, 125.6, 125.7, 125.8, 125.9, 126.0, 126.1, 126.2, 126.3, 126.4, 126.5, 126.6, 126.7, 126.8, 126.9, 127.0, 127.1, 127.2, 127.3, 127.4, 127.5, 127.6, 127.7, 127.8, 127.9, 128.0, 128.1, 128.2, 128.3, 128.4, 128.5, 128.6, 128.7, 128.8, 128.9, 129.0, 129.1, 129.2, 129.3, 129.4, 129.5, 129.6, 129.7, 129.8, 129.9, 130.0, 130.1, 130.2, 130.3, 130.4, 130.5, 130.6, 130.7, 130.8, 130.9, 131.0, 131.1, 131.2, 131.3, 131.4, 131.5, 131.6, 131.7, 131.8, 131.9, 132.0, 132.1, 132.2, 132.3, 132.4, 132.5, 132.6, 132.7, 132.8, 132.9, 133.0, 133.1, 133.2, 133.3, 133.4, 133.5, 133.6, 133.7, 133.8, 133.9, 134.0, 134.1, 134.2, 134.3, 134.4, 134.5, 134.6, 134.7, 134.8, 134.9, 135.0, 135.1, 135.2, 135.3, 135.4, 135.5, 135.6, 135.7, 135.8, 135.9, 136.0, 136.1, 136.2, 136.3, 136.4, 136.5, 136.6, 136.7, 136.8, 136.9, 137.0, 137.1, 137.2, 137.3, 137.4, 137.5, 137.6, 137.7, 137.8, 137.9, 138.0, 138.1, 138.2, 138.3, 138.4, 138.5, 138.6, 138.7, 138.8, 138.9, 139.0, 139.1, 139.2, 139.3, 139.4, 139.5, 139.6, 139.7, 139.8, 139.9, 140.0, 140.1, 140.2, 140.3, 140.4, 140.5, 140.6, 140.7, 140.8, 140.9, 141.0, 141.1, 141.2, 141.3, 141.4, 141.5, 141.6, 141.7, 141.8, 141.9, 142.0, 142.1, 142.2, 142.3, 142.4, 142.5, 142.6, 142.7, 142.8, 142.9, 143.0, 143.1, 143.2, 143.3, 143.4, 143.5, 143.6, 143.7, 143.8, 143.9, 144.0, 144.1, 144.2, 144.3, 144.4, 144.5, 144.6, 144.7, 144.8, 144.9, 145.0, 145.1, 145.2, 145.3, 145.4, 145.5, 145.6, 145.7, 145.8, 145.9, 146.0, 146.1, 146.2, 146.3, 146.4, 146.5, 146.6, 146.7, 146.8, 146.9, 147.0, 147.1, 147.2, 147.3, 147.4, 147.5, 147.6, 147.7, 147.8, 147.9, 148.0, 148.1, 148.2, 148.3, 148.4, 148.5, 148.6, 148.7, 148.8, 148.9, 149.0, 149.1, 149.2, 149.3, 149.4, 149.5, 149.6, 149.7, 149.8, 149.9, 150.0, 150.1, 150.2, 150.3, 150.4, 150.5, 150.6, 150.7, 150.8, 150.9, 151.0, 151.1, 151.2, 151.3, 151.4, 151.5, 151.6, 151.7, 151.8, 151.9, 152.0, 152.1, 152.2, 152.3, 152.4, 152.5, 152.6, 152.7, 152.8, 152.9, 153.0, 153.1, 153.2, 153.3, 153.4, 153.5, 153.6, 153.7, 153.8, 153.9, 154.0, 154.1, 154.2, 154.3, 154.4, 154.5, 154.6, 154.7, 154.8, 154.9, 155.0, 155.1, 155.2, 155.3, 155.4, 155.5, 155.6, 155.7, 155.8, 155.9, 156.0, 156.1, 156.2, 156.3, 156.4, 156.5, 156.6, 156.7, 156.8, 156.9, 157.0, 157.1, 157.2, 157.3, 157.4, 157.5, 157.6, 157.7, 157.8, 157.9, 158.0, 158.1, 158.2, 158.3, 158.4, 158.5, 158.6, 158.7, 158.8, 158.9, 159.0, 159.1, 159.2, 159.3, 159.4, 159.5, 159.6, 159.7, 159.8, 159.9, 160.0, 160.1, 160.2, 160.3, 160.4, 160.5, 160.6, 160.7, 160.8, 160.9, 161.0, 161.1, 161.2, 161.3, 161.4, 161.5, 161.6, 161.7, 161.8, 161.9, 162.0, 162.1, 162.2, 162.3, 162.4, 162.5, 162.6, 162.7, 162.8, 162.9, 163.0, 163.1, 163.2, 163.3, 163.4, 163.5, 163.6, 163.7, 163.8, 163.9, 164.0, 164.1, 164.2, 164.3, 164.4, 164.5, 164.6, 164.7, 164.8, 164.9, 165.0, 165.1, 165.2, 165.3, 165.4, 165.5, 165.6, 165.7, 165.8, 165.9, 166.0, 166.1, 166.2, 166.3, 166.4, 166.5, 166.6, 166.7, 166.8, 166.9, 167.0, 167.1, 167.2, 167.3, 167.4, 167.5, 167.6, 167.7, 167.8, 167.9, 168.0, 168.1, 168.2, 168.3, 168.4, 168.5, 168.6, 168.7, 168.8, 168.9, 169.0, 169.1, 169.2, 169.3, 169.4, 169.5, 169.6, 169.7, 169.8, 169.9, 170.0, 170.1, 170.2, 170.3, 170.4, 170.5, 170.6, 170.7, 170.8, 170.9, 171.0, 171.1, 171.2, 171.3, 171.4, 171.5, 171.6, 171.7, 171.

Survey of Replacement Options

Articles with comparisons and benchmarks:

- <https://blog.takipi.com/the-ultimate-json-library-json-simple-vs-gson-vs-jackson-vs-json/>
- <https://github.com/fabienrenaud/java-json-benchmark>

Rationale for eliminating some options from the articles above (about 20 libraries in total):

- Related to or derived from Jackson code
- Requires change to compilers and compile-time processes
- Counter-productive to CII Badging criteria, see also <https://github.com/coreinfrastructure/best-practices-badge>
 - Unmaintained in recent years
 - Vulnerabilities not addressed
 - "Bus factor" too low
 - Number of contributors and reviewers too low

Short-list of libraries as reasonable options to be explored, including:

- <https://github.com/alibaba/fastjson>
- <https://github.com/google/gson>
- <https://github.com/square/moshi>
- <https://github.com/owlike/genson>

Quick CVE comparison:

- <https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=fastjson+or+gson+or+moshi+or+genson>
- <https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=fasterxml+or+jackson>

Code Analysis

Search on AAI source code shows:

- approx 611 hits in 227 files for "fasterxml", which includes pom.xml and Java imports
- approx 978 hits in 215 files for "gson", which includes pom.xml and Java imports and initialising Java object
- zero hits for "fastjson"
- zero hits for "moshi"
- zero hits for "genson"

File	Modified
Text File gson-usage-aai.txt	Oct 01, 2018 by Keong Lim
Text File fasterxml-usage-aai.txt	Oct 01, 2018 by Keong Lim

[Download All](#)

Code Examples

- aai\aa-common\aa-auth\src\main\java\org\onap\aaiauth\auth\AuthCore.java
- aai\aa-common\aa-core\src\main\java\org\onap\aaiauth\AAIAuthCore.java

FasterXML Jackson example

```
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;

public synchronized void loadUsers(String authFilename) throws Exception {
    users = new HashMap<>();

    mapper = new ObjectMapper(); // can reuse, share globally
    JsonNode rootNode = mapper.readTree(new File(authFilename));
    JsonNode rolesNode = rootNode.path(AuthConstants.ROLES_NODE_PATH);

    for (JsonNode roleNode : rolesNode) {
        String roleName = roleNode.path(AuthConstants.ROLE_NAME_PATH).asText();

        AuthRole role = new AuthRole();
        JsonNode usersNode = roleNode.path(AuthConstants.USERS_NODE_PATH);
        JsonNode functionsNode = roleNode.path(AuthConstants.FUNCTIONS_NODE_PATH);
        for (JsonNode functionNode : functionsNode) {
            String function = functionNode.path(AuthConstants.FUNCTION_NAME_PATH).asText();
            JsonNode methodsNode = functionNode.path(AuthConstants.METHODS_NODE_PATH);
            boolean hasMethods = handleMethodNode(methodsNode, role, function);

            if (!hasMethods) {
                // iterate the list from HTTP_METHODS
                for (HTTP_METHODS meth : HTTP_METHODS.values()) {
                    String thisFunction = meth.toString() + ":" + function;

                    role.addAllowedFunction(thisFunction);
                }
            }
        }

        handleUserNode(usersNode, roleName, role);
    }

    usersInitialized = true;
}
```

gson example

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.google.gson.JsonArray;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

private synchronized void reloadUsers() {

    Map<String, AAIUser> tempUsers = new HashMap<>();

    try {
        LOGGER.debug("Reading from " + globalAuthFileName);
        String authFile = new String(Files.readAllBytes(Paths.get(globalAuthFileName)));

        JsonParser parser = new JsonParser();
        JsonObject authObject = parser.parse(authFile).getAsJsonObject();
        if (authObject.has("roles")) {
            JsonArray roles = authObject.getAsJsonArray("roles");
            for (JsonElement role : roles) {
                if (role.isJsonObject()) {
                    JsonObject roleObject = role.getAsJsonObject();
                    String roleName = roleObject.get("name").getAsString();
                    Map<String, Boolean> usrs = this.getUsernamesFromRole
(roleObject);

                    List<String> aaiFunctions = this.getAAIFunctions(roleObject);

                    usrs.forEach((key, value) -> {
                        final AAIUser au = tempUsers.getOrDefault(key, new
AAIUser(key, value));

                        au.addRole(roleName);
                        aaiFunctions.forEach(f -> {
                            List<String> httpMethods = this.
getRoleHttpMethods(f, roleObject);

                            httpMethods.forEach(hm -> au.setUserAccess(f,
hm));

                            this.validFunctions.add(f);
                        });

                        tempUsers.put(key, au);
                    });
                }
            }
            if (!tempUsers.isEmpty()) {
                users = tempUsers;
            }
        }
    } catch (FileNotFoundException e) {
        ErrorLogHelper.logError("AAI_4001", globalAuthFileName + ". Exception: " + e);
    } catch (JsonProcessingException e) {
        ErrorLogHelper.logError("AAI_4001", globalAuthFileName + ". Not valid JSON: " + e);
    } catch (Exception e) {
        ErrorLogHelper.logError("AAI_4001", globalAuthFileName + ". Exception caught: " + e);
    }
}
```

Side-by-side comparison

FasterXML Jackson version	Google gson version	Comments
---------------------------	---------------------	----------

<pre>mapper = new ObjectMapper();</pre>	<pre>JsonParser parser = new JsonParser();</pre>	
<pre>JsonNode rootNode = mapper.readTree(new File (authFilename)); JsonNode rolesNode = rootNode.path (AuthConstants. ROLES_NODE_PATH);</pre>	<pre>JsonObject authObject = parser.parse (authFile).getAsJsonObject(); JsonArray roles = authObject. getAsJsonArray("roles");</pre>	Jackson's JsonNode is a more abstract data structure, compared with Gson's more concrete data structures JsonObject and JsonArray.
<pre>String function = functionNode.path (AuthConstants. FUNCTION_NAME_PATH). asText();</pre>	<pre>String roleName = roleObject.get ("name").getString();</pre>	Code structure differs at this point (function name vs role name) but the general intent of the code is equivalent (get the element name as a string).
<pre>public synchronized void loadUsers(String authFilename) throws Exception (no exception handling in this method)</pre>	<pre>} catch (JsonProcessingException e) { ErrorLogHelper.logError("AAI_4001", globalAuthFileName + ". Not valid JSON: " + e);</pre>	For some reason, this version still catches com.fasterxml.jackson.core.JsonProcessingException even though it uses Google gson for parsing. Not a good idea to defer exception handling to the caller since the caller has no idea why/how/when/where the parsing failed and might be left with an invalid data structure as well.
<pre>boolean hasMethods = handleMethodNode (methodsNode, role, function);</pre>	<pre>usrs.forEach((key, value) -> { ... });</pre>	Method call vs Java lambda call is not really relevant to the Jackson replacement, but consistency of style could be an overall goal if the code is being re-factored anyway.

POC: Replacing default Spring Boot Jackson dependencies with Gson - [Tian Lee](#)

I have investigated the feasibility of replacing all Jackson Spring Boot dependencies with Gson, by converting the two AAF security microservices, aaf-fproxy and aaf-rproxy to use Gson only.

The basic method I followed is detailed here: <https://www.callicoder.com/configuring-spring-boot-to-use-gson-instead-of-jackson/>

What I did:

- Exclude all transitive Jackson dependencies being pulled in from the Spring Boot dependencies.
 - Exclude spring-boot-starter-json from spring-boot-starter-web
 - Remove any spring-boot-starter-actuator dependencies, as they can only work with Jackson.
- Add a dependency to Gson (2.8.5), if one does not exist already.

At this point, if the application is only using Jackson for automatically serializing and deserializing request and response objects in its REST APIs, the conversion should be complete. Your Spring Boot application should now be switched to using the Gson implementation, and function as before.

Notes:

- I did not need to add the "preferred-json-mapper" property to my application.properties as stated in the link above. Spring Boot 2.0.3 seems to be capable of detecting and using the Gson dependencies on its classpath automatically.
- Additional complications and code changes may arise if you are explicitly using any of the Jackson library classes in your code. These will need to be manually converted to use the equivalent Gson classes instead.
- Jackson dependencies may be pulled in transitively from other AAI modules (such as aai-common). Excluding these manually in your own pom may be risky, so ideally they need to be fixed at the source.

Example an example of the changes I made to the pom.xml can be found in this changelist for the AAF rProxy project: <https://gerrit.onap.org/r/gitweb?p=aaf/cadi.git;a=commitdiff;h=0d9b3896ad594816b1eb7048949114e6a18c4bd4>

(Note that this changelist contains other code changes but only the pom.xml changes are required for switching to Gson)

Discussion from Seccom meeting 10th Oct

- https://lists.onap.org/g/onap-seccom/topic/jackson_replacement_agenda/26718744

Actions:

- Maybe approach Cassandra team about also replacing Jackson?
- Share this with PTLs and aim for implementation in Dublin release?

Cassandra Usage of Jackson

- Mailing list archives for Cassandra: <http://cassandra.apache.org/community/#mailing>
- Suggested upgrade of Jackson major version (1.4 to 2+): <https://issues.apache.org/jira/browse/CASSANDRA-4102>
- Key feature of JSON support: <https://issues.apache.org/jira/browse/CASSANDRA-7970>
 - Comment asking for JSON abstraction: <https://issues.apache.org/jira/browse/CASSANDRA-7970?focusedCommentId=14333620&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-14333620>
 - "Maybe we could abstract slightly our use of jackson (put the helpers we need in Json.java maybe?), so that 1) we have only one place to change if we upgrade jackson and the API change (or we want to change of library) and 2) we save creating multiple ObjectMapper or JsonStringEncoder objects."
- Suggested move from json-simple to Jackson: <https://issues.apache.org/jira/browse/CASSANDRA-8785>
- Upgrade of Jackson minor version due to vulnerabilities (1.9.x): <https://issues.apache.org/jira/browse/CASSANDRA-8974>
- Upgrade of Jackson major version due to vulnerabilities (1.9.x to 2.9.5): <https://issues.apache.org/jira/browse/CASSANDRA-14427>
 - Similar analysis of the databind problems and gave themselves a similar waiver: <https://issues.apache.org/jira/browse/CASSANDRA-14427?focusedCommentId=16479994&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-16479994>
 - "Assuming that condition is met, ObjectMapper also needs to be able to be able to handle polymorphic types. This can be done in 2 ways:
 - `ObjectMapper.enableDefaultTyping()`. We don't do this in the code base.
 - Explicitly defining polymorphic types using annotations `JsonSubType` or `JsonTypeInfo`. We don't do this either."
- Request sent to 'user@cassandra.apache.org' mailing list:
 - <https://lists.apache.org/thread.html/02880c1b361eda0b1a2602be0fdc217d316b0b8ac85b175d73aa3cdd@<user.cassandra.apache.org>>
- Response from Greg Matza, who is "Enterprise Account Executive" at ScyllaDB: <https://www.scylladb.com/company/#team>

▪ **From:** Greg Matza [mailto:greg@scylladb.com]
Sent: Friday, 12 October 2018 09:56
To: Keong Lim <Keong.Lim@huawei.com>
Subject: ScyllaDB

Keong,

I saw your post on the Cassandra mailing list.

ScyllaDB is an open source drop-in replacement for Apache Cassandra. You can easily substitute Scylla for Cassandra in JanusGraph - in fact IBM has publicly spoken about why they chose Scylla instead of Cassandra for their JanusGraph as a Service. <https://www.scylladb.com/tech-talk/performance-evaluation-scylla-database-backend-janusgraph-scylla-summit-2017/>

And, because ScyllaDB is written in C++, there are no Jackson dependencies. So you'll be able to satisfy your security concerns, and take advantage of the fact that Scylla is faster and easier to manage.

We believe that there are already users of ScyllaDB within Huawei. For example, in March of 2018, our CEO and CTO met with Huawei Cloud, who are interested in offering Scylla as a Service. That potential partnership is still under discussion.

If you are interested in learning more, I would recommend downloading the software at <https://www.scylladb.com/download/>, and joining our ScyllaDB Users Slack, where there are many Scylla users and engineers who can answer questions. <http://slack.scylladb.com>. (There is even a Chinese-language channel - #general-cn - on the Slack, if you prefer that to the mostly-English #general channel)

Good luck, and I hope to see you on the Slack!!!

Greg

Greg Matza

650-400-9648

greg@scylladb.com

■ **From:** Greg Matza [mailto:greg@scylladb.com]
Sent: Friday, 12 October 2018 12:02
To: Keong Lim <Keong.Lim@huawei.com>
Subject: Re: ScyllaDB

Sounds great!

I know that time is very short, but we are holding the Scylla Summit in 4 weeks - 6th November and 7th November. We will be near San Francisco. <http://www.scylladb.com/scylla-summit-2018/> We welcome you and your team to attend the Summit, in order to better understand the technology and integrate with the user community.

There is also a full day of training on 5th November that would likely be valuable.

If it is possible for you and your team to come, I will be happy to waive the \$500 fee for Training and Summit. Let me know, and I can arrange for the free registration. If necessary, I can also get you a letter of invitation, which may help with obtaining a visa.

Greg

■ **From:** Greg Matza [mailto:greg@scylladb.com]
Sent: Wednesday, 24 October 2018 03:58
To: Keong Lim <Keong.Lim@huawei.com>
Subject: Re: ScyllaDB

Keong,

Google Alert notified me of your posting on the ONAP site. From that notification, I saw your notes on our conversation and learned a bit more about your project.

We are excited by the prospect of inclusion in the ONAP AAI project. Our co-founders - Dor Laor and Avi Kivity are well experienced with Linux Foundation projects, as they are the creator and early manager of the KVM Hypervisor. So working with OSS Foundations is in our company's DNA. In addition, telecom are among the early adopters of Scylla - AT&T, Verizon, Huawei, Comcast, T-mobile are all either running Scylla in Production or are in POC.

As you continue to research the viability of Scylla for AAI, I'd like to make our technical and executive resources available to you. From a technical perspective, we are available to help with any POC or evaluation. (Installation, hardware recommendations, monitoring setup, or other practical questions). From an executive perspective, we'd be happy to discuss any potential questions with roadmap, licensing or other 'vision'-type questions.

Let me know if/when you are ready to engage with our technical or executive resources.

Greg

- **From:** Greg Matza [<mailto:greg@scylladb.com>]
Sent: Tuesday, 11 December 2018 13:01
To: Keong Lim <Keong.Lim@huawei.com>
Cc: stephen.terrell@ericsson.com; jf2512@att.com; Maheedhar Gunturu <maheedhar@scylladb.com>
Subject: Re: ScyllaDB

Keong,

Unfortunately, we don't have a public demo lab that you can point your application at. The closest we have is the Scylla Test Drive, which spins up a cluster on AWS and gives you SSH access to the nodes, but the cluster only runs for an hour, and you are mostly stuck with cassandra-stress. (<https://www.scylladb.com/test-drive/>) Better than nothing, but not perfect.

For your testing, we are happy to share all the necessary instructions on how to get started using Scylla with Kubernetes and give you access to our Helm Charts and stateful sets. We are also happy to make ourselves available over Slack and /or web conferences to help set everything up.

Hardware requirements are <https://docs.scylladb.com/getting-started/system-requirements/>. Scylla will probably run on your laptop, but the official testing and Production recommendations are:

Installation	Cores	Memory	Disk	Network
Test, minimal	4	2 GB	Single plain SSD	1 Gbps
Production	20 cores - 2 socket, 10 cores each	128 GB	RAID-0, 4 SSDs, 1-5TBs	10 Gbps

Ultimately, if you can share workload specs with us, we can make hardware sizing recommendations.

Slack invitations are available at <http://slack.scylladb.com>. Binaries are at <https://www.scylladb.com/download/#binary> (or <https://www.scylladb.com/download/#docker>). Let us know when you have the cycles to do some testing.

Greg

[blocked URL](#)

On Mon, Dec 10, 2018 at 4:35 PM Keong Lim <Keong.Lim@huawei.com> wrote:

Hi Greg,

To that end, do you have some public demo lab available that runs Scylla in a Kubernetes cluster, where we could push some data and queries through to observe the behavior and collect stats?

What would be the min/max/default CPU/RAM resources to configure in the pods?

ONAP project has its own Cassandra clusters running in various test labs, so a virtual-side-by-side comparison might provide an interesting data point to consider.

Keong

From: Greg Matza [mailto:greg@scylladb.com]
Sent: Tuesday, 11 December 2018 11:02
To: Keong Lim <Keong.Lim@huawei.com>
Cc: stephen.terrell@ericsson.com; jf2512@att.com
Subject: Re: ScyllaDB

Sounds good.

Of course, Jackson vulnerabilities may be the bleeding wound that needs to be patched up most urgently. But there are also other benefits of Scylla vs. Cassandra. Namely, easier maintenance (no JVM tuning, no cache tuning, etc.), and typically cheaper/better performance on fewer nodes.

I just mention this because the idea of Scylla as a wholesale replacement for all the C throughout AAI (and even ONAP) came up on your last discussion. (I listened to the recording) Obviously, we'd encourage that! We think our software is awesome! But, beyond our boosterism, there might be real value to users, beyond the Security aspects.*

Look forward to hearing from you,

Greg

On Mon, Dec 10, 2018 at 3:56 PM Keong Lim <Keong.Lim@huawei.com> wrote:

Hi Greg,

The issue of replacing Jackson libraries came up again from a security standpoint, so there may be some appetite to look at a Cassandra replacement as part of that work.

The discussion right now is about which use cases will be included in the next release, but priorities have not been agreed yet.

Will keep Scylla in mind when the topic does come up again!

Keong

From: Greg Matza [mailto:greg@scylladb.com]
Sent: Tuesday, 11 December 2018 10:28
To: Keong Lim <Keong.Lim@huawei.com>
Subject: Re: ScyllaDB

Keong,

Congrats on the Casablanca release.

We're here to assist with any testing or discussion, once the Cassandra/Scylla question comes up again, as part of Dublin.
Any thoughts on when that might be?

Greg

ScyllaDB as replacement for Cassandra

- Intended as a drop-in replacement for Cassandra (works with existing Cassandra driver in JanusGraph)
- Written in C++ rather than Java
- Benchmarks faster than Cassandra: <https://www.scylladb.com/product/benchmarks/>
- Available under AGPL: <https://github.com/scylladb/scylla>
- AWS test cluster: <https://www.scylladb.com/test-drive/>
- Hardware requirement: <https://docs.scylladb.com/getting-started/system-requirements/>
- Possibility of doing virtual-side-by-side comparison test?

Quick CVE comparison:

- <https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=scylla+or+scylladb>
- <https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=cassandra>

Discussion from PTL meeting 15th Oct

- https://lists.onap.org/g/onap-discuss/topic/jackson_replacement_agenda/27240860
- [PTL 2018-10-15](#)

Actions:

- Should ONAP pay for Jackson project to fix the vulnerabilities?
- Could ScyllaDB be used to replace Cassandra?
- 2019-09-17: point Greg Matza towards [Mike Elliott](#) of OOM team as maintainer of shared/common Cassandra (and other databases).

VF2F December 2018

- Mentioned again for Dublin release S3P requirements at [ONAP Project Developers Event, Dec 10 - 12, 2018, \(Virtual Webinars\)](#)

Conclusion So Far

For AAI project:

- code already uses Google gson, so
 - gson has already been scanned for vulnerabilities
 - gson does not appear on Seccom lists for package upgrade or replacement
- currently no usage for the alternative Json libraries, so
 - introducing the new libraries may also bring in new vulnerabilities and problems

- there is already at least one worked example for translating from Jackson usage to gson usage, facilitating further conversions to gson
- the POC shows that transitive dependencies on Jackson could also be eliminated in some cases
- there are nearly 30 AAI repositories and over 200 files that need to be updated
- fully eliminating Jackson may not be possible due to other tools, such as Cassandra
 - could Cassandra be replaced by using ScyllaDB?