How Does the Apache Xerces Vulnerability (SNYK-JAVA-XERCES-608891) Impact ESAPI?

Kevin W. Wall <kevin.w.wall@gmail.com>

Summary

Category:	Improper input validation.
Module:	Apache Xerces – a transitive dependency used by ESAPI (specifically in xercesImpl-2.12.0.jar or earlier versions in ESAPI 2.x). There is no official patch for Apache Xerces available from the standard Maven Central repository. (See “Workarounds” section for further information.)
Announced:	ESAPI 2.2.2.0 release notes, 2020-November-27.
Credits:	Chess Hazlett. Brought to our attention via Snyk-bot.
Affects:	All versions of ESAPI 2.x
Details:	**Not exploitable as indirectly used by ESAPI. See discussion below.**
GitHub Issue #:	N/A; see https://app.snyk.io/vuln/SNYK-JAVA-XERCES-608891 and the referenced https://bugzilla.redhat.com/show_bug.cgi?id=1860054 for details.
CWE:	There is no known specific CVE for this Red Hat bug id, but as per aforementioned bugzilla.redhat.com URL, it is related to CVE-2020-14621.
CVE Identifier:	N/A (however, related to CVE-2020-14621)
For the related CVE, CVE-2020-14621	
CVSS Severity (version 3.1)	CVSS v3.1 Base Score: 5.3 (Medium) Impact Subscore: 1.4 Exploitability Subscore: 3.9 CVSS Vector CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N

Background

OWASP ESAPI (the OWASP Enterprise Security API) is a free, open source, web application security control library that makes it easier for programmers to write lower-risk applications. The ESAPI for Java library is designed to make it easier for programmers to retrofit security into existing applications. ESAPI for Java also serves as a solid foundation for new development.

ESAPI does not *directly* use the xercesImpl jar, but it is used *indirectly* via the HTML sanitation functionality provided via OWASP AntiSamy, as well as via ESAPI's built-in WAF functionality via the xom jar.

AntiSamy is used by the ESAPI Validator component; specifically, it is called by any of the several getValidSafeHTML() methods in the Validator interface as implemented by the DefaultValidator class. It is also used directly by the HTMLValidationRule class if your application uses that ESAPI class directly. AntiSamy uses classes from the xercesImpl jar via its AntiSamyDOMScanner and AntiSamySAXScanner classes as well as its MagicSAXFilter class (which is not used via ESAPI).

XOM is used by the ESAPI WAF component, specifically by the ESAPI ConfigurationParser class, which your application likely is not using. In fact, if you are not using any of the classes in the org.owasp.esapi.waf package, then the XOM dependency does not expose you to any of the classes from the xercesImpl jar *at all*.

More analysis details are provided in the sections that follow.

Problem Description

According to the description in Snyk, the current description for this vulnerability states:

> "Affected versions of this package are vulnerable to Improper Input Validation due to the way the XMLSchemaValidator class in the JAXP component of Wildfly[1] enforced the "use-grammar-pool-only" feature. This flaw allows a specially-crafted XML file to manipulate the validation process in certain cases. This issue is the same flaw as CVE-2020-14621, which affected OpenJDK, and uses a similar code."

Given this, some might be concerned that using ESAPI leaves applications exposed to this vulnerability in a manner that makes it exploitable. The high-level answer is, "The ESAPI development team does not believe that ESAPI's indirect use of xercesImpl leaves applications using ESAPI exposed to this Apache Xerces vulnerability". The rationale behind our position is below.

Analysis of Potential Exposure to ESAPI

To determine this, one of the ESAPI project co-leads (Kevin Wall) used the Snyk reference to track down the commit that was identified to fix this problem. In examining that commit, we see that is a one-line fix to the Apache Xerces' XMLSchemaValidator class.

Neither AntiSamy nor XOM directly use this XMLSchemaValidator class. Also, neither use any Xerces classes that sets the "use-grammar-pool-only" internal feature or sets the internal Xerces' property "https://www.apache.org/xml/properties/internal/grammar-pool" while configuring any of the Xerces' parsers. Finally, neither of them call the Xerces'

---

[1] Note: Wildfly is the new name for what was previously known as Red Hat JBoss.

setFeature() method set via Xerces’ Constants.USE_GRAMMAR_POOL_ONLY_FEATURE (i.e., “internal/validation/schema/use-grammar-pool-only”).

Therefore, our conclusion is that neither AntiSamy nor XOM use Apache Xerces xercesImpl jar in a manner that expose them to this vulnerability as described in the Snyk reference. And because AntiSamy nor XOM are exposed to this vulnerability, neither is ESAPI.

Impact

If ESAPI does not expose an exploitable path to this vulnerability, what then *is* the concern? The problem as we see it, and likely how many in the ESAPI user community view it, is that Software Composition Analysis (SCA) tools and/or services like OWASP Dependency Check, BlackDuck, Snyk, Veracode's SourceClear, GitHub Dependabot, etc. will continue to give you warnings that you may be required to explain to your management in order to justify continuing to use ESAPI in your application.

Removing dependencies that reference xercesImpl (specifically, AntiSamy and XOM) would require either:

1. completely dropping this currently provided functionality in ESAPI,
2. replacing the current ESAPI code with equivalent functionality from alternate FOSS libraries that do not use Apache Xerces (e.g., replacing AntiSamy with the OWASP Java HTML Sanitizer), or
3. rebuilding this current functionality directly from scratch, without dependencies on any FOSS libraries.

Since ESAPI 2.x is in maintenance mode, and we do not believe that ESAPI exposes this vulnerability to applications using ESAPI 2.x, and a workaround (see below) is available, we not intend to do any of those things.

Workaround

Note that there ***is*** a patch available for this Apache Xerces vulnerability, but it is not available from the standard Maven Central repository so it does no good to client applications using ESAPI if were to include it in our builds because to grab that same patch, applications would need to do the same. The available patch is in the JBoss repository and is patched as xercesImpl 2.12.0.SP03. If you wish to use it, this section describes how to apply it.

This workaround around assumes that you still wish to continue using ESAPI in your application. If you can consider alternatives to ESAPI and you are only using ESAPI’s Encoders, then you could consider replacing that functionality with the very similar OWASP

Java Encoder Project. Replacing other ESAPI components with alternatives is less straight forward. See “Should I use ESAPI?” for some possible suggestions.

If you want to use the JBoss repository patch, do the following:

Step 1: In your application’s pom.xml, update your reference of your dependency on the ESAPI jar, like so:

```
<dependency>
  <groupId>org.owasp.esapi</groupId>
  <artifactId>esapi</artifactId>
  <version>2.2.2.0</version>
  <exclusions>
    <exclusion>
      <groupId>xerces</groupId>
      <artifactId>xercesImpl</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

(Or replace version “2.2.2.0” with whatever version of ESAPI you are using; hopefully the latest version.) When you then build your project, this should exclude the xercesImpl jar from being imported as a dependency of ESAPI or anything it depends on (e.g., XOM and AntiSamy). (Note that you do not have to specify a ‘version’.) Note that this of course will only work if you have no other direct or transitive dependencies on xercesImpl.

[You can also exclude specific transitive dependencies using Gradle. If you use Grade, follow these general instructions and figure out the rest of the Gradle configuration steps based on the following steps for Maven.]

Step 2: You then need to explicitly add the patched dependency to your project’s pom.xml file:

```
<dependency>
  <groupId>xerces</groupId>
  <artifactId>xercesImpl</artifactId>
  <version>2.12.0.SP03</version>
</dependency>
```

Step 3: However, because this patch is not available in Maven Central, you need configure Maven for your project so that it also uses the JBoss Maven repository:

```
<repositories>
   <!-- This repo is temporarily added so we can get the patched version of Xerces -->
   <repository>
     <id>jboss-repo</id>
     <url>http://repository.jboss.org/nexus/content/groups/public/</url>
   </repository>
</repositories>
```

Solution

The only “real” solution to this is for the ESAPI team to wait for an official patch from the Apache Xerces folks that gets uploaded to standard Maven Central and then have OWASP ESAPI create a new patch release with an officially patched version of xercesImpl.

Until then you will either have to a) live with the workaround outlined above,b) accept the warnings from various SCA scanners, c) explicitly suppress them, or d) import the patched version of Xerces issued directly by Apache if/when that becomes available. If you decide to live with the SCA scanner warnings, perhaps you can show your management this ESAPI security bulletin to convince them that using ESAPI does not make this vulnerability exploitable to your application because of the restricted way that ESAPI indirectly uses the xercesImpl classes.

Additional Precautions

Run OWASP Dependency Check or a similar SCA tool or service on your final project configuration to ensure that you have no unpatched dependencies in your application’s class path.

Acknowledgments

Kudos to the Snyk-bot for calling this to our attention.

References

https://app.snyk.io/vuln/SNYK-JAVA-XERCES-608891

https://nvd.nist.gov/vuln/detail/CVE-2020-14621