How Does CVE-2021-29425 Impact ESAPI?

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

Summary

Category:	In Apache Commons IO before 2.7, When invoking the method FileNameUtils.normalize with an improper input string, like "//../foo", or "\\..\foo", the result would be the same value, thus possibly providing access to files in the parent directory, but not further above (thus "limited" path traversal), if the calling code would use the result to construct a path value.
Module:	Apache Commons IO – a transitive dependency used by ESAPI via AntiSamy to support general HTML sanitization. See the dependency tree below for details.
Announced:	2020-03-21 via ESAPI User Google Group (https://groups.google.com/a/owasp.org/g/esapi-project-users/c/_c_sb-SIFYk)
Credits:	• GitHub Dependabot • SnykBot
Affects:	All versions of ESAPI 2.x and all versions of ESAPI 1.x (no longer supported) if you are using AntiSamy via ESAPI's Validator.
Details:	**Not exploitable as used by ESAPI or AntiSamy. See discussion below.**
GitHub Issue #:	N/A
Related:	None.
CWE:	CWE-22 (NIST: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')) CWE-20 (Apache Software Foundation: Improper Input Validation)
CVE Identifier:	CVE-2021-29425
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:L/I:N/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.

Apache Commons IO is a FOSS library of utilities to assist with developing I/O functionality. The class org.apache.commons.io.FileNameUtils is a class of static methods providing several file name and file path manipulation utilities. The normalize() method provides file path name cannonicalization.

The Apache Commons IO library is not a direct dependency of ESAPI. The way that it gets pulled into ESAPI is via a transitive dependency. This is the portion of the dependency tree from ESAPI 2.2.3.0 (the prior release) that shows how it gets pulled in:

```
kww@feynman:~/Code/GitHub/esapi-2.2.3.0-official$ mvn -B dependency:tree
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< org.owasp.esapi:esapi >-------------------------
[INFO] Building ESAPI 2.2.3.0
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-dependency-plugin:3.1.2:tree (default-cli) @ esapi ---
[INFO] org.owasp.esapi:esapi:jar:2.2.3.0
[INFO] <... irrelevant dependencies deleted ...>
[INFO] +- org.owasp.antisamy:antisamy:jar:1.6.2:compile
[INFO] |  +- net.sourceforge.nekohtml:nekohtml:jar:1.9.22:compile
[INFO] |  +- org.apache.httpcomponents:httpclient:jar:4.5.13:compile
[INFO] |  +- org.apache.httpcomponents:httpcore:jar:4.4.14:compile
[INFO] |  +- org.apache.xmlgraphics:batik-css:jar:1.14:compile
[INFO] |  |  +- org.apache.xmlgraphics:batik-shared-resources:jar:1.14:compile
[INFO] |  |  +- org.apache.xmlgraphics:batik-util:jar:1.14:compile
[INFO] |  |  |  +- org.apache.xmlgraphics:batik-constants:jar:1.14:compile
[INFO] |  |  |  \- org.apache.xmlgraphics:batik-i18n:jar:1.14:compile
[INFO] |  |  \- org.apache.xmlgraphics:xmlgraphics-commons:jar:2.6:compile
[INFO] |  |     \- commons-io:commons-io:jar:1.3.1:compile
... deleted ...
```

So Commons IO is being pulled in via AntiSamy, which pulls in Apache Batik-CSS. Batik-CSS is part of a larger Apache Xmlgraphics Batik family which is mirrored on GitHub at https://github.com/apache/xmlgraphics-batik.

While it may be obvious, we will also note that AntiSamy also does **NOT** directly use Apache Commons IO.

Problem Description

According to the description in NIST's National Vulnerability Database (NVD), the current description for CVE-2021-29425 states:

> "In Apache Commons IO before 2.7, When invoking the method FileNameUtils.normalize with an improper input string, like "//../foo", or "\\..\ foo", the result would be the same value, thus possibly providing access to files in the parent directory, but not further above (thus "limited" path traversal), if the calling code would use the result to construct a path value."

So the real question that everyone is asking is will using ESAPI leave my application code exposed to CVE-2021-29425 n a manner that makes this CVE exploitable? That is the question this analysis attempts to answer, but the TL;DR answer for those of you not interested in the details is,

> "No, it does not, because nothing in the Batik family of libraries uses org.apache.commons.io.FileNameUtils and thus is not affected by this CVE."

Specifically, I cloned https://github.com/apache/xmlgraphics-batik and searched the source code for the use of Apache Commons IO and the only reference to any class was to org.apache.commons.io.IOUtils and even then it was *only* using the static 'copy()' method to copy files. And the copy method only uses code from the JRE (e.g., rt.jar classes and maybe some other implementation specific classes), but no Apache Commons IO classes.

Therefore there is no path that would allow the vulnerable FileNameUtils.normalize() method to be invoked via ESAPI or AntiSamy or even Batik-CSS for that matter. (Obviously any insecure reflection vulnerability or insecure deserialization vulnerability in your applications execution path would allow it to be called, but if that's the case you have much bigger problems than a limited path traversal attack, so end of discussion on that.)

Impact

So, if ESAPI does not expose an exploitable path to CVE-2021-29425, what then *is* the concern? The problem as we see it, and likely how many in the ESAPI users community view it, is that Software Composition Analysis (SCA) tools and/or services like OWASP Dependency Check, BlackDuck, Snyk, Veracode's SourceClear, GitHub, etc. will continue to give you warnings that you may be required to explain to your management in order to justify continue using ESAPI in your application.

Upgrading to Commons IO 2.8.0 to completely remediate this potential vulnerability from ESAPI would require that ESAPI be updated to require all ESAPI clients be using at least Java 8. Currently the minimal JRE baseline for ESAPI is Java 7. Changing this without

warning would be in conflict with ESAPI's deprecation policy, which is now officially described in ESAPI's README.md file, but which has long been our unofficial policy dating back to ESAPI 2.0.0.0. If you use ESAPI with Java 8 or later, you are encouraged to the workaround which is described in the next section.

If as an ESAPI user, you absolutely must continue to use ESAPI's with Java 7, they you will just have to accept the risk. But things like this are only going to get worse and ESAPI will soon be requiring Java 8 as a minimal JRE baseline. You can also show your management this security bulletin if you think that will help.

Workaround

If your application is using Java 8 or later, you can use this workaround and it should make any SCA scans stop complaining about CVE-2021-29425. If you application is using Java 7, you probably have bigger problems than worrying about noise in your SCA scans.

The specific example for this workaround assumes your application is using Maven, but most other modern build tools allow you to do similar things.

For Maven projects, edit your application's pom.xml, and update your reference of your application's dependency on the ESAPI jar in this manner:

```
    <dependency>
     <groupId>org.owasp.esapi</groupId>
     <artifactId>esapi</artifactId>
     <version>2.2.3.1</version>    <!-- Or whatever version you are using. -->
     <exclusions>
       <exclusion>
         <groupId>commons-io</groupId>
         <artifactId>commons-io</artifactId>
       </exclusion>
     </exclusions>
   </dependency>
   <!-- Add commons-io 2.8.0 (or later) as an explicit dependency to
         work around CVE-2021-29425.
    -->
   <dependency>
      <groupId>commons-io</groupId>
       <artifactId>commons-io</artifactId>
           <!-- Note: commons-io:2.7 and later requires Java 8 -->
           <version>2.8.0</version>
    </dependency>
```

The lines highlighted in yellow are lines you will add.

When you then build your project, this should ***exclude*** the vulnerable version of commons-io jar that ESAPI would normally pull in and replace it with the one you specified in your updated pom.xml file. (Of course, this is assuming you don't have the commons-io jar elsewhere as a direct or transitive dependency) Note that you do not have to specify a 'version' when you "exclude" it.

You can also exclude specific transitive dependencies using Gradle. If you use Grade, follow these general instructions. For other build tools, you're on your own.

Additional Precautions

Run OWASP Dependency Check or a similar SCA tool or service on your final project configuration after configuring the workaround to ensure that CVE-2021-29425 no longer shows up as a vulnerability in your application's class path.

Solution

The only "real" solution to this is to have OWASP ESAPI update to require Java 8 as a minimal JRE baseline. That is planned later on in the year, but we can not do so without sufficient advance warning. Stay tuned to the ESAPI-Dev and ESAPI-Users Google groups mailing lists.

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 CVE-2020-9488 exploitable to your application because of the restricted way that ESAPI uses Apache Commons IO classes.

References

https://nvd.nist.gov/vuln/detail/CVE-2021-29425