How Does CVE-2021-4104 Impact ESAPI?

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

Summary

Category:	Improper validation of certificate with host mismatch in Apache Log4j JMSAppender. This potentially could allow insecure deserialization of untrusted logged data eventually resulting in remote code execution.
Module:	Log4J 1 – a compile dependency used by ESAPI (specifically Log4J 1.2.17 in the latest ESAPI version) to support "safe logging".
Announced:	2021-12-18 via ESAPI User Google Group (https://groups.google.com/a/owasp.org/g/esapi-project-users/???TBD???)
Credits:	Dependency Check, Snyk, and https://slf4j.org/log4shell.html
Affects:	All versions of ESAPI 2.x and all versions of ESAPI 1.x (no longer supported) if you are using ESAPI's deprecated log4j 1 logging.
Details:	**Not exploitable as used by ESAPI. See discussion below.** ESAPI Log4j logging deprecated on 2020-07-23 as part of the 2.2.1.0 release.
GitHub Issue #:	None specifically, but see References section, below.
Related:	No other ESAPI security bulletin, unless you want to include the other 2 or 3 Log4J 1.x related security bulletins.
CWE:	CWE-502 (Deserialization of Untrusted Data)
CVE Identifier:	CVE-2021-4104

CVSS Severity (version 3.1)	CVSS v3.1 Base Score: 8.1 (High)
	    Impact Subscore: 5.9
	    Exploitability Subscore: 2.2
	CVSS Vector CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H

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.

One of the security controls provided by ESAPI for Java is its provision for "safe logging" which is designed as a defense against "CWE-117: Improper Output Neutralization for Logs". Traditionally, ESAPI 1.2 and 2.x, ESAPI has supported both Java's standard java.util.logging (henceforth referred to as JUL) and Apache's Log4J 1.x. In ESAPI 2.2.0.0, support for SLF4J was added. For versions of ESAPI, up through and including ESAPI 2.2.0.0, the default configuration for ESAPI has been to use Log4J 1.x and Log4J's ConsoleAppender. For release 2.2.1.0, the default ESAPI logger was changed to use JUL. It should be noted that this decision to use JUL was made prior to the ESAPI development team becoming aware of this CVE in question. We switched to making JUL ESAPI's default logger shortly after Jeremiah Stacey added some important missing functionality to ESAPI's JUL support to bring it more in line with ESAPI's Log4J logging format. Thus the reason for the decision was not because of this Log4J CVE though; rather it was made simply on the basis of Apache Log4J 1 being significantly past end-of-life and no longer being supported.

Also, as of ESAPI 2.2.1.0, ESAPI has annotated all ESAPI Log4J-related classes as '@deprecated'. However, ESAPI's deprecation policy is that the ESAPI development would not delete any classes, methods, or fields marked as '@deprecated' until either 2 years had passed since the first release when that annotation was added or in the next *major* release number (which, in this case, would be 3.0). This is ESAPI's promise to try to give development teams adequate prior warning before doing something that might break backward compatibility for ESAPI users in their application code.

Problem Description

According to the description in NIST's National Vulnerability Database (NVD), the current description for CVE-2021-4104 states:

> "JMSAppender in Log4j 1.2 is vulnerable to deserialization of untrusted data when the attacker has write access to the Log4j configuration. The attacker can provide TopicBindingName and TopicConnectionFactoryBindingName configurations causing JMSAppender to perform JNDI requests that result in remote code execution in a similar fashion to CVE-2021-44228. Note this issue only affects Log4j 1.2 when specifically configured to use JMSAppender, which is not the default. Apache Log4j 1.2 reached end of life in August 2015. Users should upgrade to Log4j 2 as it addresses numerous other issues from the previous versions."

So the real question that everyone is asking is will using ESAPI leave my application code exposed to CVE-2021-4104 in 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, not unless you have your application configured in the following manner:

1. You are using the deprecated ESAPI log4j logging

2. You have changed your default log4j.xml configuration (or log4j.properfies file, if you're still doing that) file to use JMSAppender for one of its Appenders.

3. And attacker is able to overwrite the contents of your Log4J 1 configuration file (typically, log4j.xml, but possibly log4j.properties)."

In other words, it's not likely, and with ESAPI's default configuration, not possible.

Apache Log4J's JMSAppender is a class that is intended for sending log events to JMS Topic. The events are serialized and transmitted as JMS message type `ObjectMessage`.

The problem is that if you are using JMSAppender in your Log4J configuration and the attacker can overwrite or otherwise alter your Log4J configuration file, then Log4J 1.2.17 is subject to an Insesure Deserialization attack via JNDI lookups, resulting in remote code execution. (Insecure Deserialization of untrusted data is essentially the root cause the recent Log4J 2 CVE, CVE-2021-44228 (aka, Log4Shell).

Even the default configuration log4j.xml file that is is provided under the GitHub releases as part of the various 'esapi-<version>-configuration.jar' files only uses ConsoleAppender and is otherwise intentionally rather brain-dead to force client applications to more or less create their own log4j configurations.

Thus, our conclusion is, if you refrain from using Apache Log4J's JMSAppender your application should not be exploitable from *this* **specific** *log4j CVE* even if you still insist using an unsupported Apache library that is way past its end-of-service date and that ESAPI has deprecated.

Despite that, the ESAPI development team **still** advises *against* configuring ESAPI.Logger to use Log4J 1. That is why now we have deprecated the use of Log4J 1 in ESAPI. The mere fact that Log4J 1.x is unsupported means that there will be no further security patches for it and the next one that does appear in Log4J 1 conceivably could be exploitable through ESAPI when using some more common appender such as FileAppender, ConsoleAppender, etc.

Impact

So, if ESAPI does not expose an exploitable path to CVE-2021-4104, what then *is* the concern (assuming you don't use JMSAppender,etc. as described above)? 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.

Removing support for Log4J 1 completely from ESAPI 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. You are encouraged to use JUL or SLF4J for ESAPI logging, as a workaround which is described in the next section.

If as an ESAPI user, you absolutely must continue to use ESAPI's Log4J 1's logger for compatibility with the rest of your application using Log4J 1.x, make sure that your logging is not configured to use Log4j's JMSAppender. You can also show your management this security bulletin. (Of course, if your application has Log4J 1 as a *direct* dependency which you can't eliminate, it matters little that it is a *transitive* dependency to your application through ESAPI.)

Workaround

If you are okay with configuring ESAPI logging to use either JUL (which is the new ESAPI default starting with the ESAPI 2.2.1.0 release) or SLF4J (through which you could support Log4J *2* through something like slf4j-log4j2), you can use the following (or similar) workaround when building your project:

First, in your application's **ESAPI.properties** file, change the value for the **ESAPI.Logger** property to either use JUL or SLF4J.

Second, in your application's pom.xml, reference your dependency on the ESAPI jar in this manner:

```
<dependency>
 <groupId>org.owasp.esapi</groupId>
 <artifactId>esapi</artifactId>
 <version>2.2.3.1</version>
 <exclusions>
   <exclusion>
     <groupId>log4j</groupId>
     <artifactId>log4j</artifactId>
   </exclusion>
 </exclusions>
</dependency>
```

(Or whatever version of ESAPI you are using; hopefully the latest version.) When you then build your project, this should exclude the log4j jar from your classpath. (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 Log4J 1.x.

You can also exclude specific transitive dependencies using Gradle. If you use Grade, follow these general instructions.

Note that using this workaround with SLF4J requires ESAPI 2.2.0.0 or later; however, you should be able to use this for any version of ESAPI 2.x if you are willing to use JUL for ESAPI

logging. (Note that if you switch to using JUL for ESAPI versions prior to 2.2.1.0, the logging output will like slightly different than what you get with it configured to use Log4J 1, but it will still do "safe" logging.)

Solution

The only "real" solution to this is to have OWASP ESAPI completely remove Log4J 1 as a dependency. (See GitHub issue #534.) In release 2.2.1.0 it has been deprecated but we cannot remove it immediately and continue to honor our deprecation policy (until either two years from the release where it was first announced, which was on the July 23, 2020 release date for the 2.2.1.0 release or in the next major release of ESAPI, which will be 3.0). ESAPI 3.0 is currently only in planning stages. Until then you will either have to live with the workaround or accept the warnings from various SCA scanners. 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-2021-4104 exploitable to your application because of the restricted way that ESAPI uses Log4J 1 classes.

Additional Possible Future Plans About Deprecating Log4j 1

The ESAPI development team is also kicking around some ideas such as:

- GitHub Issue #610 - Logging a WARNING for the first ESAPI log4j output that comes out that warns that ESAPI log4j has been deprecated in release 2.2.1.0 and will be removed in the first release after the 2 year period (i.e., after July 23, 2022).

- GitHub Issue #609 - Changing the llog4j 1 dependency in ESAPI's pom.xml to make the log4j 1.2.17 jar to have a scope of either 'provided' or 'optional' so that ESAPI will no longer include it as a dependency. That is likely to have it's own serious consequences for those who are still using iESAPI logging with log4j so we eventually decided it was not worth the effort and closed this.

Additional Precautions

Run OWASP Dependency Check or a similar SCA tool or service on your final project configuration to ensure that you have no Log4J 1 dependencies in your application's class path.

Acknowledgments

Log4Shell, without which, serendipity would not have been as likely.

References

https://nvd.nist.gov/vuln/detail/CVE-2021-4104

GitHub Issue #534 (https://github.com/ESAPI/esapi-java-legacy/issues/534)

GitHub Issue #609 (https://github.com/ESAPI/esapi-java-legacy/issues/609)

GitHub Issue #610 (https://github.com/ESAPI/esapi-java-legacy/issues/610)