

How Does CVE-2022-23307 Impact ESAPI?

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

Summary

Category:	There is an RCE flaw caused by an insecure deserialization vulnerability in Apache Chainsaw , a Java-based GUI log viewer. CVE-2020-9493 identified a deserialization issue that was present in Apache Chainsaw 2.x prior to 2.1.0. However, prior to Chainsaw V2.0, Chainsaw was a component of Apache Log4j 1.2.x where the same issue exists and remains unfixed.		
Module:	Log4J 1 – a compile dependency used by ESAPI (specifically Log4J 1.2.17 in the latest ESAPI version) to support “safe logging”.		
Announced:	In this security bulletin.		
Credits:	Clueless, but well meaning Software Composition Analysis tools everywhere.		
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 2020-07-13 as part of the 2.2.1.0 release. Apache Chainsaw is a separate Java application that ESAPI has never used or has any dependency upon.		
GitHub Issue #:	None.		
Related:	CVE-2020-9493		
CWE:	CWE-502 (Deserialization of Untrusted Data)		
CVE Identifier:	CVE-2022-23307		
CVSS Severity (version 3.1)	CVSS v3.1 Base Score:	8.8 (High)	
	Impact Subscore:	5.9	
	Exploitability Subscore:	2.8	
	CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:L/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-2022-23307](#) states:

“CVE-2020-9493 identified a deserialization issue that was present in Apache Chainsaw. Prior to Chainsaw V2.0 Chainsaw was a component of Apache Log4j 1.2.x where the same issue exists.”

So the real question that everyone is asking is will using ESAPI leave my application code exposed to CVE-2022-23307 in a manner that makes this CVE exploitable?

The answer is

“No, absolutely NOT!”

While Apache Chainsaw may have been part of the log4j-1.2.17.jar, ESAPI does not use Chainsaw and has no dependencies on it. (The actual vulnerable class with the deserialization vulnerability is “org.apache.log4j.chainsaw.LoggingReceiver” by the way. ESAPI uses nothing in that org.apache.log4j.chainsaw package.) Now sure, if you as an ESAPI client are running Apache Chainsaw from the log4j-1.2.17.jar file instead of running a patched version of Apache Chainsaw (2.1.0 or later), that’s on you. This is clearly a vulnerability associated with Chainsaw and since it’s more or less a standalone Java-based GUI log server, you will almost assuredly be running it in a separate process different from your application that is using ESAPI.

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 since the 2.2.1.0 version, released on 2020-07-13. 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-2022-23307, what then *is* the concern (assuming you don’t use zChainsaw from the log4j-1.2.17.jar 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. (But we do plan it for removal on or shortly after 2022-07-13.) You are encouraged to use JUL or SLF4J for ESAPI logging, as a workaround which is described in the next section.

You can avoid CVE-2022-23307 (if you are not using Log4J 1 for the ESAPI Logger) by excluding it as noted in the Workaround section below. 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>2.4.0.0</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 Gradle, 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 look 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 13, 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-2022-23307 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 13, 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 ESAPI 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

None. But if someone gave me donuts, I'm sure I could credit someone! :)

References

<https://nvd.nist.gov/vuln/detail/CVE-2022-23307>

<https://nvd.nist.gov/vuln/detail/CVE-2020-9493> - The original insecure deserialization issue in Apache Chainsaw 2.x.