How Does CVE-2019-17571 Impact ESAPI?

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

Summary

Category:	Java deserialization issue, leading to potential code injection.
Module:	Log4J 1 – a dependency used by ESAPI (specifically Log4J 1.2.17 in the latest ESAPI version) to support "safe logging".
Announced:	2020-01-08 via ESAPI User Google Group (https://groups.google.com/a/owasp.org/forum/#!topic/esapi-project-users/XxKBjj3HuSw)
Credits:	• Dennis Bakker for first bringing this to our attention. (He beat the GitHub security bulletin by 4 days!) • Eddy Vos for suggesting a practical workaround for ESAPI users.
Affects:	All versions of ESAPI 2.x and all versions of ESAPI 1.x (no longer supported)
Details:	**Not exploitable as used by ESAPI. See discussion below.**
GitHub Issue #:	534 (https://github.com/ESAPI/esapi-java-legacy/issues/534)
CWE:	CWE-502 (Deserialization of Untrusted Data)
CVE Identifier:	CVE-2019-17571
CVSS Severity (version 3.1)	CVSS v3.1 Base Score: 9.8 (critical) Impact Subscore: 5.9 Exploitability Subscore: 3.9 CVSS Vector CVSS:3.1/AV:N/AC:L/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-2019-17571 states:

> "Included in Log4J 1.2 is a SocketServer class that is vulnerable to deserialization of untrusted data which can be exploited to remotely execute arbitrary code when combined with a deserialization gadget when listening to untrusted network traffic for log data. This affects Log4J versions up to 1.2 up to 1.2.17. "(*sic*)

So the real question that everyone is asking is will using ESAPI leave my application code exposed to CVE-2019-17571 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, the ESAPI development team believes that ESAPI's use of Log4J 1 does would not leave your application code using ESAPI exposed to CVE-2019-17571 in a manner that would make it exploitable".

Apache Log4J's SocketServer is a class that is intended for listening for (presumably) Log4J log events and centrally collecting them. ESAPI does not use this Apache Log4J class at all nor any other server-side Log4J class. By default, ESAPI internally only uses the following Log4J classes:

- org.apache.log4j.Level
- org.apache.log4j.Logger
- org.apache.log4j.Priority
- org.apache.log4j.spi.LoggerFactory
- org.apache.log4j.xml.DOMConfigurator

plus Log4J’s org.apache.log4j.ConsoleAppender class which is configured as the default Log4J appender class in “configuration/log4j.xml”, which is provided as an example Log4j configuration and which most likely would be replaced by some type of FileAppender root logger in the application actually using ESAPI.

None of these Log4J 1 classes use the vulnerable org.apache.log4j.net.SocketServer class, nor do they use any Log4J classes that use the SocketServer class. A manual code inspection of the log4j 1.2.17 code from GitHub reveals that the only non-test / non-documentation files from the Apache log4j 1.2.17 release (which is the latest Log4J 1 release available in Maven Central) that references the vulnerable SocketServer class are these files:

- src/main/java/org/apache/log4j/net/SimpleSocketServer.java
- src/main/java/org/apache/log4j/net/SocketServer.java
- contribs/MarkDouglas/SocketServer2.java
- examples/lf5/UsingSocketAppenders/UsingSocketAppenders.java
- examples/sort3.properties

Thus, our conclusion is, if you refrain from using Apache Log4J’s SocketServer or any of the above classes, your application should not be exploitable based on how ESAPI uses Apache Log4J 1. (Of course, how your *own application* code uses Log4J directly or via some other library is another matter entirely and outside the scope of this ESAPI security bulletin.)

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.

Impact

So, if ESAPI does not expose an exploitable path to CVE-2019-17571, 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.

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. However, if you are able to use JUL or SLF4J for ESAPI logging, there is a workaround 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, there is absolutely nothing you can do short of perhaps showing 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 [not-yet-released] ESAPI 2.2.1.0 release) or SLF4J (through which you could support Log4J 2), you can use the following (or similar) workaround when building your project, first suggested by Eddy Vos here.

First, in your application's **`ESAPI.properties`** file, change the value for the **`ESAPI.Logger`** property. To change ESAPI logging to use JUL, change that property to

    org.owasp.esapi.reference.JavaLogFactory

if you are using a version of ESAPI version of 2.2.0.0 or earlier, or to

    org.owasp.esapi.logging.java.JavaLogFactory

if you are using a version of ESAPI later than 2.2.0.0. To configure it to use SLF4J, change that property value to

    org.owasp.esapi.logging.slf4j.Slf4JLogFactory

Note that starting with 2.2.1.0, the ESAPI sample configuration file under

```
configuration/esapi/ESAPI.properties
```

is configured to use JUL as the default ESAPI logger.

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>2.2.0.0</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 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. In release 2.2.1.0 it has been deprecated so we cannot remove it immediately and continue to honor our deprecation policy until either two years from the (to-be-determined) 2.2.1.0 release date 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-2019-17571 exploitable to your application because of the restricted way that ESAPI uses Log4J 1 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 Log4J 1 dependencies in your application’s class path.

Acknowledgments

Kudos to Dennis Bakker for following the steps in the ESAPI README.md in GitHub about reporting potential vulnerabilities and sending the ESAPI project leaders a direct email rather than posting to a public forum. A special hat tip to him for notifying us 4 days before I received a notification from GitHub itself.

Also special recognition to Eddy Vos for suggesting a rather simple workaround that excludes ESAPI’s use Apache Log4J 1’s logger as mentioned in GitHub Issue 534.

References

https://nvd.nist.gov/vuln/detail/CVE-2019-17571

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

GitHub Issue #538 (https://github.com/ESAPI/esapi-java-legacy/issues/538)