



# From Log4Shell to Log4SBOM

Piotr Karwasz, Apache Software Foundation





# Who are we?



## Apache Software Foundation:

- American 501(c)(3) nonprofit: (3)—Charitable organization
- More than 800 members
- More than 8000 contributors (committers)
- 200 Top-level projects
- Each TLP lead by a Project Management Committee
- Apache Logging Services PMC, since December 2003

<https://apache.org/>

<https://logging.apache.org/>

## Piotr Karwasz:

- 2000: OSS *aficionado*.
- 2009: Ph.D. in Mathematics (UHP, Nancy).
- Father of three daughters: Mimi, Lili and Nati.
- 2017: I started my own IT company.
- 2022, January: start contributing to Log4j.
- 2022, July: Logging Services PMC member.
- 2024, March: ASF member.
- 2024, June: Logging Services PMC chair.

<https://oss.copernik.eu/>

<https://linkedin.com/in/ppkarwasz/>



# What is Log4j?

- One of the logging libraries of Apache Logging Services, together with Log4cxx, Log4Net, Log4j Kotlin, Log4j Scala.
- 2001: Ceki Gülcü creates Log4j 1
- 2003: Ceki Gülcü brings Log4j to ASF
- 2005-2011: Ceki Gülcü starts working on SLF4J/Logback successor
- 2012: Last Log4j 1 release
- 2014: Log4j 2 API/Core is published by: G. Gregory, R. Goers, R. Popma, M. Sicker and others
- 2015: end-of-life of Log4j 1

```
logger.debug("Opening file {}!", file);  
logger.info("Hello {}!", user);  
logger.error("Failed to foo!", e);
```

Remember,  
remember,  
the 9th of December!  
(2021)



Source: [Devianart](#)

@papa osmubal



# CVE-2021-44228 (Log4Shell)

“An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers...”—NVD Database

## Ingredients:

1. Log4j Core did expand placeholders like `${env:HOME}` in configuration files, but also in **log messages**.
2. JNDI is a Java technology to retrieve configuration values, services or (Easter egg)... download code.
3. Log4j Core supported `${jndi:...}`

## Characteristics:

- Affects a lot of people:  
Log4j Core had already more than 10 M downloads monthly.
- Easily exploitable:  
For example Minecraft from Mojang AB passed all chat messages in the game to Log4j Core.
- Limited exploitability on up-to-date JDK versions.

# Timeline of 2.15.0 release

November 24th, 7:51 UTC:

Chen Zhaojun reports the vulnerability

November 24th, 17:30 UTC:

Team discusses the report. It is bad.

November 25th: Thanksgiving!

November 26th, 4:00 UTC:

CVE number requested.

November 30th:

Patch supplied (public PR).

December 5th:

Patch amended, reviewed and merged.

December 7th:

Release vote for **2.15.0** RC1 (72 hours)

December 9th:

Users notice the PR solves a security issue.

Problem with RC1, RC2 vote (7 hours)

Version **2.15.0** released with 7 votes.

**Note:** Release **2.15.0** was the first of 4 releases that patched a total of 4 CVEs and ended on December 28th with the **2.17.1** release.

# Reactions

## Is Log4JS npm package vulnerable to CVE-2021-44228 Log4J vulnerability

Asked 2 years, 9 months ago Modified 2 years, 9 months ago Viewed 8k times

As the title says. Looked online for a clear answer but can't find an answer anywhere as most of them just link to Log4J.

13

log4j log4js-node Edit tags



Share Edit Follow Close Flag



Add a comment

Start a bounty

asked Dec 13, 2021 at 14:18



1 Answer

Sorted by:  [Reset to default](#)



25



The answer is simple: **Log4JS** and **Log4J** share only a similar name and API. The codebases are entirely different (and written in different languages). The vulnerability of Log4J does not apply obviously to Log4JS.

This kind of vulnerability could not even be easily implemented in JavaScript. Java's vulnerability is based on [JNDI](#) lookups, which usually are used to retrieve simple configuration data. However they also allow to retrieve serialized Java objects and new classes (cf. [Oracle's documentation](#)).



The JavaScript equivalent of this vulnerability would be a formatter that replaces:

### The Overflow Blog

Looking under the hood that powers multimod

### Featured on Meta

Join Stack Overflow: first Stack IRL Comm

User activation: Learn opportunities

What does a new us homepage experience Overflow?

Announcing the new Reviewer Stats Widg

### Hot Meta Posts

35 How should I handle the Staging Ground?

21 Is archiving the best duplicates from the S

Linked

- Log4j questions on StackOverflow increased tenfold to 2%.
- Do they use Log4j?
- There is a considerable increase in upgrades from Log4j 1 (not affected, EOL 2015) to Log4j 2.
- Some companies upgraded Log4j Core multiple times during December 2021.
- Others didn't...

## log4j Latest Statistics

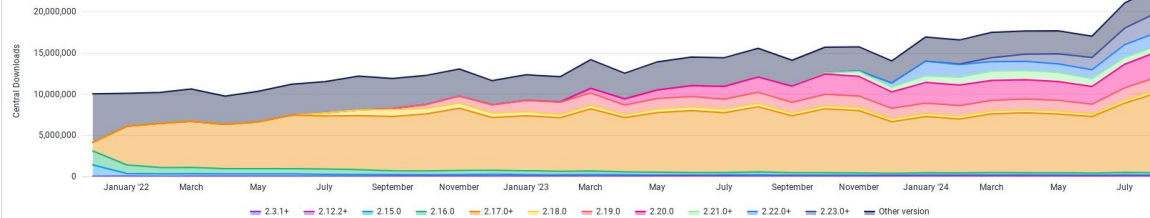
471,179,072

Total Downloads Since Dec 10, 2021  
24 % vulnerable

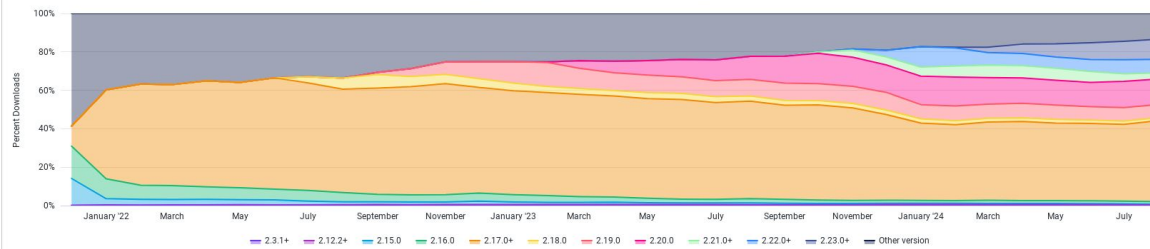
16 %

Vulnerable Downloads Last 7 Days  
5,239,729 total downloads

log4j Monthly Central Downloads



log4j Percent Monthly Central Downloads



Source: Sonatype [Log4j Updates and Vulnerabilities](#)





# Apache Log4j Reactions

# Lessons learned

## Too many bundled features:

learn to say NO (intelligently).

## Supply chain problems:

- Tests are flaky (slow down release),
- Site generation is slow,
- Release procedure is complex,
- Keep dependencies up-to-date (and tell about it).

## Documentation problems:

- Is hard to find,
- Is not complete, some obscure features are not documented,
- Does not contain best practices.

## Helped solving the problems:

- [Tidelift](#) supports Log4j since January 2023,
- German [Sovereign Tech Fund](#) with a grant to Christian Grobmeier, Volkan Yazıcı and me, since September 2023.



# Handling features

# Securing optional features

Handling features is hard:

- Features bring users,
- Features bring security exposure,
- OSS is a meritocracy:

Maintainers have the right to their features in exchange for their work.

- Log4j created a `3.x` branch in 2018 to split each optional dependency, including JNDI into its own artifact.  
Completed: IX 2024
- Removal of seldom downloaded artifacts.
- Ramp-up program:

We accept new modules with a proven user base and a maintainer. These modules start as third-party.



# Supply chain

# Preparing a release

We need an expert Release Manager to:

- Select the changes for a new release,
- Run all the test suites,
- Build the website,
- Sign the release,
- Prepare the release notes,
- Handle the voting procedure,
- Release the new version.

Now, as one of the first ASF projects, we (almost) fully automate:

- Running tests,
- Upgrading dependencies,
- Deploying snapshots,
- Staging the new website,
- Staging and signing the artifacts and source archives,
- Staging the voting procedure.

# Key supply chain elements

- Can we trust automation?  
ASF policy requires the RM to create the binaries.  
[Reproducible Builds Project](#):  
All our Java builds are reproducible!
- [Dependabot](#): upgrades dependencies since 2017.  
We accept those upgrades automatically if tests pass.
- [GitHub Actions](#) is the CI/CD engine we use.
- Lots of Maven plugins and test libraries  
that don't get credit enough!

# Software Bill of Materials

An SBOM is:

- An inter-ecosystem format to list dependencies.
- A useful tool to manage dependencies and their versions.
- A worldwide network of machine-readable and interconnected security documents.

Present:

- Publishing of SBOM for all Log4j artifacts. The dependency versions are not enforced, but merely suggested.
- Maven specificity: dependents don't profit from all version suggestions.
- Usage of SBOM links to point to a machine-readable VDR.
- Features contributed back to [CycloneDX Maven Plugin 2.8.0](#)



# SBOMs future (?)

- Integration of SBOMs into ecosystem-specific dependency management systems.

## Transparency Exchange API for:

- Automatically import VDR/VEX entries from dependencies to stage VEX entries. “Vulnerability Bot”
- Push our VDR, VEX and version suggestions to consumers/dependents.





# Documentation

# Security through education

Logging is not always safe:

- Unstructured logging:  
[CWE-93 CRLF Injection](#)
- Presence of sensitive information in logs:  
[CWE-215: SI in Debug Code](#)
- Injection of `{ }` Log4j formatting patterns:

```
String user = "root { }";  
String what = "login";  
log.(user + "failed to { }", what);
```

- Reliable and secure transport.

Solutions:

- Rewrite of documentation website.  
Learn from the source, not ChatGPT.
- Generation of reference from code:  
Living documentation,  
Developers can not forget.
- Provide best practices and tips:  
The maintainers knowledge base was  
mainly unwritten.

**Tip:** there will be an in-depth book by Christian Grobmeier published by [Manning](#).

APACHE LOG4J™ 2

- About
- Download
- Javadocs
- Maven, Ivy, Gradle Artifacts
- Runtime Dependencies
- Release Notes
- FAQ
- Performance
- Articles and Tutorials
- Security
- Support
- Thanks

FOR CONTRIBUTORS

- Guidelines
- Style Guide

MANUAL

- Introduction
- Architecture
- API Separation
- Log4j 1.x Migration
- Java API
- Scala API

Configuration

Configuration Architecture

- Arbiters
- Automatic Configuration
- Additivity
- Automatic Reconfiguration
- Chainsaw Support
- Configuration Syntax
- XML Syntax
- JSON Syntax
- YAML Syntax
- Properties Syntax
- Configuring Loggers
- Configuring Appenders
- Configuring Filters
- Property Substitution

## Configuration

Inserting log requests into the application code requires a fair amount of planning and effort. Observation shows that approximately 4 percent of code is dedicated to logging. Consequently, even moderately sized applications will have thousands of logging statements embedded within their code. Given their number, it becomes imperative to manage these log statements without the need to modify them manually.

Configuration of Log4j 2 can be accomplished in 1 of 4 ways:

1. Through a configuration file written in XML, JSON, YAML, or properties format.
2. Programmatically, by creating a ConfigurationFactory and Configuration implementation.
3. Programmatically, by calling the APIs exposed in the Configuration interface to add components to the default configuration.
4. Programmatically, by calling methods on the internal Logger class.

This page focuses primarily on configuring Log4j through a configuration file. Information on programmatically configuring Log4j can be found at [Extending Log4j 2](#) and [Programmatically Configuring Log4j](#).

All available formats are functionally equivalent. For example, a configuration file in XML can be rewritten using the properties format (and the opposite) without any loss of functionality. However, the hierarchical nature of a Log4j configuration can be captured better in formats which naturally support nesting so XML, JSON, and YAML files, are usually easier to use.

Note that unlike Log4j 1.x, the public Log4j 2 API does not expose methods to add, modify or remove appenders and filters or manipulate the configuration in any way.

## Configuration Architecture

In part because support for XML was added first, Log4j's configuration is reflected as a tree structure. In fact every configuration dialect, including the ConfigurationBuilder, generates a Node for every configuration element. A node is a fairly simple structure that contains a set of attributes, a set of child nodes and a PluginType. It is important to note that every Node must have a corresponding plugin, as the plugin is the component that actually performs the work represented by the node.

Every document type supported by Log4j has a ConfigurationFactory. The factory itself is a Log4j plugin that declares what file extensions it supports and what its priority is. Properties have the highest precedence with a value of 8, followed by yaml, json and xml. When autoconfiguration is performed Log4j will call each of these factories in order to determine which, if any, support the specified configuration file format. If one is found that factory will create the corresponding Configuration object and pass the reference to the configuration data to it.

Every configuration implementation, such as XMLConfiguration, YamlConfiguration, JsonConfiguration, etc. has the primary task of converting the configuration text into the Node tree, typically by parsing the text with whatever tool is available for that document type. It should be noted that while most of the supported document types are inherently tree structured, the Java properties syntax is not. Because of the need to convert the syntax into a Node tree the Java properties syntax used by Log4j required all properties follow a naming pattern that made the tree structure clear. As a consequence, the Java Properties format tends to be more verbose than using a different document type.

Once the Node tree is created control is delegated to AbstractConfiguration, which converts the Nodes into their respective Java objects using Log4j's Plugin system and provides all the common functionality.

## Arbiters

In some situations it is desirable to have a single logging configuration that can be used in any deployment environment. For example, it may be necessary to have a different default logging level in production than in development. Another case might be where one type of appender is used when running natively but another is used when deployed to a docker container. One way to handle that is to use a tool such as Spring Cloud Config Server that can be environment aware and serve a different file for each environment. Another option is to include Arbiters in the configuration.

An Arbiter is a Log4j plugin that has the job of determining whether other configured elements should be included in the generated configuration. While all other "Core" plugins are designed to execute as part of Log4j's runtime logic Arbiters execute after the Node tree has been constructed but before the tree is converted to a configuration. An Arbiter is a Node itself which is always removed from the Node tree before it the tree is processed. All an arbiter really does is provide a method that returns a boolean result that determines whether the child nodes of the arbiter should remain in the configuration or be pruned.

Arbiters may occur anywhere an element is allowed in the configuration. So an Arbiter could encapsulate something as simple as a single property declaration or a whole set of Appenders or Loggers. Arbiters may also be nested although Arbiters that are the descendant of another arbiter will only be evaluated if the ancestor returned true. The child elements of an Arbiter must be valid elements for whatever element is the parent of the Arbiter.

This example shows two Arbiters configured that will include either a Console Appender or a List Appender depending on whether the value of the env System Property is "dev" or "prod".

```
1. <Configuration name="ConfigTest" status="ERROR" monitorInterval="5">
2.   <Appenders>
3.
4.     <SystemPropertyArbiter propertyName="env" propertyValue="dev">
5.       <Console name="Out">
6.         <PatternLayout pattern="%m%n"/>
```

## Configuration file

Using a configuration file is the most popular and recommended approach for configuring Log4j Core. In this page we will examine the composition of a configuration file and how Log4j Core uses it.

**TIP**

If you are looking for a quick start on using Log4j in your application or library, please refer to [Getting started](#) instead.

## Configuration file location

Upon initialization of a new [logger context](#), [the anchor of the logging implementation](#), Log4j Core assigns it a context name and scans the following classpath locations for a configuration file in following order:

1. Files named `log4j2-test<contextName>.<extension>`
2. Files named `log4j2-test.<extension>`
3. Files named `log4j2<contextName>.<extension>`
4. Files named `log4j2.<extension>`

The `<contextName>` and `<extension>` placeholders above have the following meaning

**<contextName>**

A name derived from the runtime environment:

- For standalone Java SE applications, it is a random identifier.
- For web applications, it is an identifier derived from the application descriptor. See [Log4j Web application configuration](#) for details.

**<extension>**

A file extension supported by a `ConfigurationFactory`. The order in which an extension will be searched for first depends on the order of the associated `ConfigurationFactory`. See [Predefined ConfigurationFactory plugins](#) for details.

If no configuration file is found, Log4j Core uses the [DefaultConfiguration](#) and the `status logger` prints a warning. The default configuration prints all messages less severe than [log4j2.level](#) to the console.

**Contents**

- Configuration file location
  - Predefined ConfigurationFactory plugins
- Syntax
  - Main configuration elements
  - Additional configuration elements
- Global configuration attributes
  - monitorInterval
  - status
- Loggers
  - name
  - additivity
  - level
  - includeLocation
  - Appender references
  - Additional context properties
  - Filters
- Appender references
  - ref
  - level
  - Filters
- Property substitution
  - Runtime property substitution
- Arbiters
- Composite configuration
- Format specific notes
  - XML format
    - Global configuration attributes
      - schema
      - strict
    - XInclude
  - Java properties format
- Extending
  - Plugin preliminaries
  - Extending ConfigurationFactory plugins
  - Plugins represented in a configuration file

## Configuration

### Class

`org.apache.logging.log4j.core.config.Configuration`

### Provider

`org.apache.logging.log4j:log4j-core`

A Log4j configuration contains many components of which two are required: `Appenders` and `Loggers`.

### XML snippet

```
<Configuration dest="err"
  monitorInterval="0"
  name=""
  schema=""
  shutdownHook=""
  shutdownTimeout=""
  status="ERROR"
  strict="false">
  <Properties/>
  <Appenders/>
  <CustomLevels/>
  <Loggers/>
</Configuration>
```

### Attributes

Optional attributes are denoted by `?`-suffixed types.

Name	Type	Default	Description
<code>dest</code>	<code>String?</code>	<code>err</code>	<p>Specifies the destination for status logger events. The possible values are:</p> <ul style="list-style-type: none"><li>• <code>out</code> for using standard out (default)</li><li>• <code>err</code> for using standard error</li><li>• a string that is interpreted in order as URI, URL or the path to a local file</li></ul> <p>If the provided value is invalid, then the default destination of standard out will be used.</p>

### Contents

- [CronTriggeringPolicy](#)
- [DefaultRolloverStrategy](#)
- [DirectFileRolloverStrategy](#)
- [DirectWriteRolloverStrategy](#)
- [NoOpTriggeringPolicy](#)
- [OnStartupTriggeringPolicy](#)
- [RolloverStrategy](#)
- [SizeBasedTriggeringPolicy](#)
- [TimeBasedTriggeringPolicy](#)
- [TriggeringPolicy](#)
- [AbstractAction](#)
- [AbstractPathAction](#)
- [Action](#)
- [Delete](#)
- [Duration](#)
- [IfAccumulatedFileCount](#)
- [IfAccumulatedFileSize](#)
- [IFail](#)
- [IfAny](#)
- [IfFileName](#)
- [IfLastModified](#)
- [IfNot](#)
- [PathCondition](#)
- [SortByModificationTime](#)
- [PathSorter](#)
- [PosixViewAttribute](#)
- [ScriptCondition](#)
- [IdlePurgePolicy](#)
- [PurgePolicy](#)
- [Route](#)
- [Routes](#)
- [Routing](#)
- [ArrayBlockingQueue](#)
- [AsyncLogger](#)
- [AsyncRoot](#)
- [AsyncWaitStrategyFactory](#)
- [BlockingQueueFactory](#)



Testing quality

# Test quality requirements

All software component should have tests that:

- Provide 73,9% of coverage,
- Are supplied with each PR,
- Test the required **behaviour**,
- Do not test implementation details,
- Do not have false negatives,
- Do not have false positives (flakiness).

September 2023:

- Sequential tests,
- 30-40% of test runs failed for no reason,
- Build times up to 60 minutes.

September 2024:

- Parallel tests,
- Dynamic tests (fuzzing),
- 8% of test runs fails (21% flaky),
- Build + deploy around 30 minutes.
- Searchable build failure database:  
[Gradle Develocity](#)





# Q & A

<https://logging.apache.org/>



# Thanks

Agnieszka Karwasz and my angels:  
Milena, Liliana, Natalia

[Apache Logging Services team:](#)

C. Kozak, D. McColl, D. Psenner, G. Gregory,  
J. Friedrich, M. Sicker, R. Goers, R. Gupta,  
R. Popma, R. Middleton, R. Grabowski, S. Deboy,  
S. Webb and Th. Schöning.

See also <https://logging.apache.org>

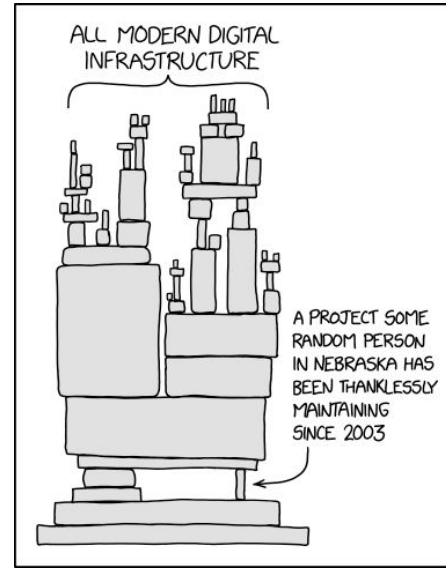
Partners in crime (STF project):

Christian Grobmeier and Volkan Yazıcı

Financial supporters:

[Tidelift](#) and [Sovereign Tech Fund](#)

Remember about:



Source: [XKCD](#)