LOG4J LOG4SHELL SURVIVAL GUIDE

Use this Log4j Log4Shell cheat sheet for information on the vulnerability and recommended actions depending on the versions you have.

LOG4J VERSION	< 2.14.1	2.15.0	2.16.0	2.17.0	2.17.1
RISK	CRITICAL	MEDIUM	VERY LOW	VERY LOW	NONE
	Vulnerable in default configuration; attacker controls any part of logged data FIXED LEGACY VERSIONS NONE v2.3.2, v2.12.4 - Backported fixes from 2.17.1 VERY LOW v2.3.1, v2.12.3 - Backported fixes from 2.17.0 v2.12.2 - Backported fixes from 2.16.0	Vulnerable in rare non-default configuration (CVE-2021-45046). Remote attack only known to work on macOS and FreeBSD due to hostname mitigation	Same as 2.15.0 + configuration must specify “log4j2.enableJndi” (new flag unlikely to be inherited from existing project). CVE-2021-45105 exists but has negligible impact	CVE-2021-44832 exists but has negligible impact	No known vulnerabilities
ACTION	**Upgrade immediately**	**Check configuration, consider upgrade**	No action needed	No action needed	No action needed

Possible mitigations - if upgrade is not feasible:

1. Remove vulnerable JNDI lookup class by deleting “JndiLookup” from log4j-core-*.jar files - best option if upgrade is unavailable. Watch out for missed .jar files. No known bypass.

2. Disable message lookup (in v2.10.0-2.15.0) by setting log4j2.noFormatMsgLookup / LOG4J_FORMAT_MSG_NO_LOOKUPS. Apply globally to protect the entire machine.
Bypass - in rare conditions (CVE-2021-45046).

3. Upgrade Java version to 6u211, 7u201, 8u191, or 11.0.1 - not recommended as only mitigation. Bypass - possible only if local gadget classes exist.

LEARN MORE

Log4j Detection with JFrog OSS Scanning Tools
All You Need to Know Blog
Technical Overview Webinar
Q&A

FIND, FIX AND FORTIFY FASTER WITH JFROG

Learn how to use the JFrog Platform to discover and control every usage of Log4j packages in your software supply chain.
Get OSS tools for identifying Log4j utilization and risk in your source code and binaries.