recollective

Log4Shell Vulnerability Statement

Updated December 20, 2021

Overview

Apache Log4j is an open source Java logging library developed by the Apache Foundation. Log4j is widely used in many applications and is present, as a dependency, in many services. These include enterprise applications as well as numerous cloud services.

The Apache Log4j remote code execution (RCE) vulnerability, also known as "Log4Shell", is very easy to exploit. A simple one-line payload can exploit the bug and the attacker achieve the command execution on the vulnerable system.

This vulnerability, which was first reported on December 9, 2021 by Chen Zhaojun of Alibaba Cloud Security Team on December 9th.

This and other important vulnerabilities exist in Log4j versions 2.0-alpha1 through to 2.16.0.

Vulnerability Details

Apache Log4j version 2, up to and include 2.15.1 have Java Naming and Directory Interface (JNDI) features used in configuration, log messages, and parameters that do not protect against attacker controlled Lightweight Directory Access Protocol (LDAP) and other JNDI related endpoints.

An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled. From log4j 2.15.0, this behaviour has been disabled by default but can be circumvented.

It was also found that the initial recommendations to address this issue in Apache Log4j 2.15.0 was incomplete in certain non-default configurations. This could allows attackers with control over Thread Context Map (MDC) input data when the logging configuration uses a non-default Pattern Layout with either a Context Lookup (for example, $${ctx:loginId}) or a Thread Context Map pattern (%X, %mdc, or %MDC) to craft malicious input data using a JNDI Lookup pattern resulting in a denial of service (DOS) attack. Log4j 2.15.0 restricts JNDI LDAP lookups to localhost by default. Note that previous mitigations involving configuration such as to set the system property log4j2.noFormatMsgLookup to true do NOT mitigate this specific vulnerability.

Additionally, Apache Log4j versions 2.0-alpha1 through 2.16.0 did not protect from uncontrolled recursion from self-referential lookups. When the logging configuration uses a non-default Pattern Layout with a Context Lookup (for example, $${ctx:loginId}), attackers with control over

Thread Context Map (MDC) input data can craft malicious input data that contains a recursive lookup, resulting in a StackOverflowError that will terminate the process. This is also known as a DOS (Denial of Service) attack.

Impact Assessment

On December 10, 2021, the Recollective team conducted an immediate assessment of its codebases for instances of Log4j. No instances of the Log4j core implementation were found or in active use within Recollective. **Recollective was not exposed to the vulnerability.**

Recollective makes use of the Spring.io default logging system. Spring has confirmed that its default logging system cannot be exploited on its own:
https://spring.io/blog/2021/12/10/log4j2-vulnerability-and-spring-boot

Actions Taken

- All Recollective code and its build dependencies were carefully reviewed for references to Log4j. None were identified.
- A complete assessment of production deployments was also conducted to ensure that all environments and Java versions are up to date.
- Proof-of-concept tests were also executed against the Recollective environment to ensure the exploit was not present in any other capacity.
- Recollective is currently undergoing a pre-scheduled third-party penetration test assessment. A third-party Log4j vulnerability assessment will now also be conducted during the assessment.

Additional Resources

https://www.cve.org/CVERecord?id=CVE-2021-44228

https://www.cve.org/CVERecord?id=CVE-2021-45046

https://www.cve.org/CVERecord?id=CVE-2021-45105

https://logging-apache-org.translate.goog/log4j/2.x/security.html

https://www.randori.com/blog/cve-2021-44228/

https://github.com/NCSC-NL/log4shell/tree/main/software

https://www.fastly.com/blog/digging-deeper-into-log4shell-0day-rce-exploit-found-in-log4j