Identifying and Remediating the Critical Apache Log4j Cybersecurity Vulnerability

December 2021

On December 9th, 2021, a critical zero-day vulnerability, which has the potential of providing threat actors access to millions of computers worldwide, was discovered. Due to the critical nature of this vulnerability, and the risk that it poses to our clients, Polsinelli has partnered with Tracepoint to develop an overview of the issue, and provide you with a set of steps that your organization can take to identify if you are vulnerable and patch the vulnerability if you are.

Note: this vulnerability is still being investigated and the recommended remediation steps may change after the writing of this alert.

We recommend that your organization's IT and IT Security personnel work with the legal team to assess your risk and address the vulnerability wherever it is found in your network. Be sure to take specific care to review systems that third-party vendors control or maintain, as these systems may also be vulnerable.

Technical Summary

The zero-day vulnerability lies in Apache's Log4j2 Java-based logging library. Apache Log4j is a logging framework that is embedded in applications that allow developers to log various types of user activity. It is specifically written in Java and is leveraged in a variety of enterprise systems and web applications. The recently discovered vulnerability is tracked and has been labeled as CVE-2021-44228 (dubbed Log4Shell or LogJam). The CVE-2021-44228 vulnerability allows a threat actor to perform remote code execution on an application server, regardless of whether it is housed on premises at your organization, or in the cloud. As a result, a threat actor can download a malicious payload – aka a virus – to a server running Log4j.

Based on how Log4j is used within most organizations, widely used enterprise software such as Salesforce, Microsoft Azure, VMware, NetApp, Qnap, and many others are reportedly vulnerable. A full list is being tracked **here**.

Additionally, the vulnerability can be exploited reliability and without the threat actor needing to authenticate to a network. Finally, if the vulnerability exists on systems used or managed by your company's vendors, it could provide a threat actor access to your network, if your vendor does not take remediation steps.

As a result, if your organization uses Apache Log4j library, and you're running anything older then **log4j-2.15.0**, you need to upgrade to at least **log4j-2.15.0.rc2**, or even better

Author

Pavel (Pasha) A. Sternberg
Principal
310.229.1335
psternberg@polsinelli.com

For more information on how to respond to this vulnerability, please contact Polsinelli's and/or Tracepoint's Incident Response teams at: **incidentresponse@polsinelli.com** and **incident@tracepoint.com** or contact the author of this article.

Identifying and Remediating the Critical Apache Log4j Cybersecurity Vulnerability

log4j 2.16.0. **log4j 2.16.0** is the latest version and by default completely disables JNDI, which is the culprit for this security vulnerability.

Security Recommendations

- Patch / Update log4j to the latest version; preferably **log4j 2.16.0**, but if that is not possible, then at least **log4j-2.15.0.rc2**.
- Follow up with vendors to inquire about their use of log4j.
- For public facing websites, if you're running CloudFlare WAF look into the mitigation efforts hosted by Cloudflare. (**https://blog.cloudflare.com/cve-2021-44228-log4j-rce-0-day-mitigation/**)
- Block ALL JNDI Strings at the WAF
  - Block ${jndi:
  - This may inhibit certain logging functions while deeper mitigations are underway
- Review your local security logs for the following syntax/ strings; identification of these strings can identify if you've been potentially exploited.
  - "${jndi:ldap:/"
  - "${jndi:rmi:/"
  - "${jndi:ldaps:/"
  - "${jndi:dns:/"
  - This is likely to be bypassed by more advanced attackers using strings such as: "${jndi:${lower:l}${lower:d}a${lower:p}:\"
- Update JNDI configuration
  - Applicable to Log4j releases >= 2.10
    - Set the system property **log4j2.formatMsgNoLookups** or the environment variable **LOG4J_FORMAT_MSG_NO_LOOKUPS** to **true**.
  - Applicable to Log4j releases >= 2.7 and <= 2.14.1
    - Modify all **PatternLayout** patterns to specify the message converter as **%m{nolookups}** instead of just **%m**
  - Applicable to Log4j releases >=2.0-beta9 and <=2.10.0
    - Remove the JndiLookup class from the classpath: zip -q -d log4j-core-*.jar org/apache/logging/log4j/core/lookup/JndiLookup.class
    - Source: **https://logging.apache.org/log4j/2.x/index.html**
- Another avenue to verify if you're vulnerable is to use "Huntress" Labs Shell Tester. (**https://log4shell.huntress.com/**)
- Grey Noise has taken on additional IPs and have tagged the log4j vulnerability in there repository to help assist people in blocking malice domains and IPs in there firewall. (**https://www.greynoise.io/**)

POLSINELLI | TRACEPOINT