India Future Foundation

India Future Foundation's Analysis on Log4Shell

Contents

Executive Summary

Log4shell is a critical vulnerability within the widely-used logging tool Log4j, which millions of computers around the world run online services. Although fixes have been issued, many individuals, including organizations, governments, and individuals, are likely to be affected.

The log4j utility is well known and utilized by many applications and companies, counting the famous game Minecraft. It is additionally used in different Apache frameworks like Struts2, Kafka, Druid, Flink, and numerous commercial items.

In December 2021, several vulnerabilities were reported In Log4j:

- **CVE-2021-44228** - The "Log4shell" vulnerability affects Apache Log4j versions 2.0-beta9 to 2.14.1. It allows remote code execution and Information disclosure if exploited.
- **CVE-2021-45046** - Affects all versions from 2.0-beta9 to 2.15.0, excluding 2.12.2 and was initially reported as a Denial of Service when organizations run a vulnerable non-standard configuration. Later research found that the same vulnerable format allowed a bypass of the mitigations to Log4shell, allowing remote code execution and information disclosure.
- **CVE-2021-45105** - Affects Log4j versions from 2.0-beta9 to 2.16.0 – A similar denial of service issue to CVE-2021-45046 when organizations run a vulnerable non-standard configuration.
- **CVE-2021-44832** - This vulnerability may allow arbitrary code execution in compromised systems when the attacker has permission to modify the logging configuration file.

Who is affected by this?

- For individuals, Log4j is nearly certainly part of the gadgets and services you utilize online every day. The best thing you can do to secure yourself is to make sure your devices and apps are updated and updated frequently.
- For organizations, it may not be possible to immediately clear all the web servers, network devices and other software that utilize Log4j. That makes the organization make necessary mitigations.
- Cloud Services like Steam, Apple iCloud and Minecraft applications have already been vulnerable.

Where was Log4Shell Discovered?

Minecraft players accidentally discovered a bug called Log4j that gave players "remote code access" -- i.e., the ability to take complete control of a console/computer remotely.

The Log4j vulnerability was simple to execute and, for the few hours it was known among Minecrafter players, basically a super-easy way to wreck other players' Minecraft servers. All you had to do was sort a particular string of letters into chat, and you'd get the access remotely.

What makes Log4j so dangerous?

The Log4j vulnerability, which received the highest **CVSS (Common Vulnerability Scoring System) score possible -10.0**, is extremely dangerous due to many factors.

1. Log4j is one of the most popular Java logging frameworks. There are currently almost 7,000 Maven artefacts that depend on log4j and countless others Java projects that use it.

2. The vulnerability exploitation is trivial and persistent, with tons of weaponized exploits available on GitHub and other open sources.

3. Many programs rely on Java's programming language, and these programs contain something *like* an "activity log" that registers everything that happens -- these are how you can Ctrl-z your way out of types and mis clicks.

4. We can trigger this vulnerability with only one line of code; unpatched Java software has become a prime chasing ground for cybercriminals missing in coding abilities.

5. The potential for harm is up to the creative ability of the adversary—ransomware, crypto-jacking mining computer program, botnet software - anything the attacker needs to transfer to the system.

Altogether, these components have turned Log4Shell into one of the riskiest vulnerabilities that security experts have seen in a long-time.

Understanding CVE-2021-44228 Exploit

The exploit allows bad actors to gain control of a computer with a single line of text. In layman's terms, a log file recovers a new entry but happens to be reading and executing upon information inside the log record. With exceptionally crafted input, a victim computer would reach out to and connect to a remote malicious device to download and execute any harmful activities that the enemy has prepared.

How does it work?

- Data from the Client gets sent to the server (via any protocol).
- The server logs the information within the request, Containing the malicious payload.
- This payload triggers the log4j vulnerability, and the server requests attacker.com.
- This reaction contains a way to an inaccessible Java class file injected into the server process.
- This injected payload triggers a second stage and allows an attacker to execute arbitrary code (i.e., the ability to run any commands or code of the attacker's choice on a target machine or in a target).

CVE-2021-44228: Mitigation

- "Ask admin/system group to run a search/grep command on all servers to spot any record with the title "log4j", Then check if it is a vulnerable version or not".
- "Version 2.16.0 of log4j has been released without the vulnerability. log4j-core.jar is accessible on Apache Log4j, and You can download it and upgrade it on your system".
- Add "log4j.format.msg.nolookups = true" to the worldwide setup of your server/web applications".
- You must update to Log4j 2.16.0 for complete protection, as Log4j 2.16.0 fixes this issue by removing support for message lookup patterns and disabling JNDI functionality by default.
- Vendors rules and channels give Restricted security against the behaviour around the exploit but isn't a complete substitution for the actual "LOG4J" vendor patch.

Understanding CVE-2021-45046 Exploit

CVE-2021-45046 was initially reported as a denial-of-service vulnerability in Apache Log4j 2.0 through 2.15.0 and has since been updated to an RCE. Under specific non-default setups where a Context Lookup (e.g., $${ctx: loginId}) is used, an attacker that makes a JNDI lookup using malicious input information would be able to cause a DoS condition or accomplish RCE on a vulnerable server.

Does the mitigation for Log4Shell apply to CVE-2021-45046?

No., the previous mitigation for CVE-2021-44228 — setting formatMsgNoLookups to genuine — is inadequately mitigation altogether. That direction failed to account for other code paths where message lookups might happen. Because of this, Apache presently prescribes updating to a secure form of Log4j, beginning with 2.16.0. If that's not conceivable, Apache suggests evacuating the JndiLookup class path.

CVE-2021-45046 Mitigation

- All the clients should upgrade to the most recent Log4j 2.16.0 version since past mitigations in Apache Log4j 2.15.0 showed up to be incomplete.
- Both vulnerabilities (CVE-2021-44228 & CVE-2021-45046) can also be mitigated by removing the JndiLookup class from the class path.
- The message lookups feature can be disabled by setting system property “log4j2.formatMsgNoLookups” to “true”.
- The most immediate need for defenders is to decrease exposure by fixing and mitigating all corners of their infrastructure and examining exposed and possibly compromised systems. This helplessness can be everywhere.

Understanding CVE-2021-45105 Exploit

The Lookup pattern accepts the following format:

${lookupName:key: -default Value}

- **lookup Name** is the name or type of lookup to perform (examples are ctx, env, etc.).
- **key** is the variable's name to look for in the corresponding map object (in the ctx case, it is the Thread Context map).
- **default Value** is an optional value that tells the substitutor what to put in place of this lookup instance if a key is not in the map.

How does it work?

- The substitutor finds the value of the client within the context map and replaces the client with it. The value is ${ctx:user1: -${ctx:user}}.
- It then tries to assess the resulting lookup pattern that was the value of the client. Typically, possible because of the recursive functionality of lookup patterns.
- user1 does not exist in Thread Context, so the substitutor goes forward to the default value as a last resort.
- The default value is a string that in itself is a lookup design, ${ctx: client}, referencing back to the client variable. Note that it's the same as the initial lookup within the configuration format pattern. This implies that the substitutor will forever attempt to resolve the variables, more than once calling its substitute method until the stack-buffer overflows.

CVE-2021-45105: Mitigation

- Apache has since released a fix to relieve this vulnerability — 2.17.0, which fixed the StrSubstitutor logic and avoided cases in which there's a recursion in lookup inside the string itself.
- 2.17.0 moreover limited recursive lookups that originate from the lookup value itself.
- It is exceedingly suggested for clients of Log4j to upgrade to the most recent 2.17.0 version. If it isn't conceivable at the moment, make sure your Log4j version is at least upgraded to 2.16.0.

Understanding CVE-2021-44832

This bug may allow arbitrary code execution through a feature that loads remote configuration for JDBC (Java Database Network) inside an XML file. Suppose the attacker has to write permissions to this logging configuration file. In that case, the JDBC Appender can be adjusted with a data source that focuses on a URL that hosts a payload, driving to code execution.

The impact of CVE-2021-44832

If you're able to quickly upgrade to the most recent fixed version of Log4j at that point, you should follow that way. Vulnerability CVE-2021-44832 has been assigned a medium 6.6 CVSS score and requires impressively raised pre-conditions for an assailant to exploit effectively.

This vulnerability considers Log4j 2.17.0 (and older versions) vulnerable to code execution if an attacker can control and modify the contents of the logging configuration file to then point to a remote URI data source to load arbitrary Java code.

Fix in 2.17.1 and backported to older JVM-compatible versions of the library, mitigated that vulnerability by limiting the JNDI data source in the configuration file to only allow the use of the Java protocol and refuse any remote network calls to be made.

CVE-2021-44832: Mitigation

- Apache has released a new Log4j version, 2.17.1. This issue is fixed by restricting JNDI data source names to the java protocol in Log4j2 forms 2.17.1, 2.12.4, and 2.3.2.
- If you're on Java 8 and later, you should **upgrade to Log4j 2.17.1.**
- If you're on the 2.12.x branch for Java 7, **upgrade to Log4j 2.12.4.**
- If you're on the 2.3.x branch for Java 6, **upgrade to Log4j 2.3.2.**
- Configure your JDBC Appender not to use any protocol other than Java.

Why are there so many Log4j vulnerabilities all of a sudden?

Why there are so many quick disclosures and corresponding releases in this library, it's critical to understand the full context of what is happening and why it makes sense that new updates have been, and might still be, released.

Increased community exposure – The startling disclosure of the critical Remote Code Execution in Log4j has pulled within the community at massive onto this project. As such, numerous bugs that might have been ignored already are presently coming due to the efforts of the Java open-source community.

Rushed initial security fix – The actual events leading up to the disclosure of CVE-2021-44228 are still being caught on; it would appear that the usually maintained responsible disclosure window was not entirely followed to, as details of this vulnerability were leaked to the public before a complete fix was prepared. The Apache team had to fix the vulnerability as soon as possible. This led to different vectors and issues being recognized, resulting in the subsequent releases.

In reverse compatibility – Log4j is a core component of the Java environment. There's a need for any new version to maintain reverse compatibility with previous versions to avoid unnecessary breakage, possibly leaving users incapable of updating to the latest and secure version. Tragically, this critical and cautious approach means the core functionality can't be modified whole-scale. Therefore, additional attack vectors or workarounds undiscovered at the time of fix publication may exist and might be exposed as the community continues to review the threat landscape.

Log4shell CVEs Highlights

CVE-2021-44228

Base Score: 10 (Critical)
Impact: Remote Control Excess
Mitigation: Update to Log4j “2.16.0”

CVE- 2021-45046

Base Score:9(Critical)
Impact: DOS Attack and Remote Control Excess
Mitigation: Update to Log4j ”2.16.0”

CVE- 2021-45105

Base Score:7.5(High)
Impact: DOS Attack
Mitigation: Update to Log4j ”2.17.0”

CVE- 2021-44832

Base Score:6.6(Moderate)
Impact: Arbitrary code Execution
Mitigation: Update to Log4j ”2.17.1”

Reference

- https://hackernoon.com/log4j-vulnerability-a-legacy-of-cybersecurity-from-java-to-blockchain-and-minecraft
- https://gaming.ebaumsworld.com/articles/minecraft-players-discover-log4j-the-most-severe-security-vulnerability-in-internet-history/87055048/
- https://bgr.com/tech/internet-is-scrambling-to-fix-log4shell-the-worst-hack-in-history/
- https://arstechnica.com/information-technology/2021/12/minecraft-and-other-apps-face-serious-threat-from-new-code-execution-bug/
- https://www.sportskeeda.com/minecraft/minecraft-log4j-exploit-everything-known-far
- https://snyk.io/blog/log4j-2-16-vulnerability-cve-2021-45105-discovered/
- https://www.netskope.com/blog/cve-2021-45046-new-log4j-vulnerability-discover

Contact us

India Future Foundation

Phone: +91-1244045954, +91-9312580816
Email: helpline@indiafuturefoundation.com
BSMT, Building no. 2731 EP,
Sector 57,
Golf Course Extension Road,
Gurugram,
Haryana, India – 122003