APACHE LOG4J VULNERABILITY

CVE-2021-44228
Severity 10/10 – Critical

Executive Summary

While Six Degrees continues to work on mitigations for the recently discovered Zero-Day, the CSOC has developed a more in-depth report to alleviate some concerns.

A vulnerability **(CVE-2021-44228)** in Log4J (Java library) was recently discovered that allows an unauthenticated attacker to perform remote code execution and gain complete access to a target system, via a vulnerable version of the Log4J library. **Any application** that uses a vulnerable Log4J version is affected, regardless of whether this is a server-side or a client-side application. If it reads some input from an attacker, and passes that to the Log4J library, there is an exploitation path.

The scope of impact has expanded to a plethora of products and devices. A few examples of vulnerable products are provided below.
A more detailed list can be found in the following link: **www.rumble.run/blog/finding-log4j/**.

Cisco	**VMware**	**LogRhythm**	**Checkpoint**
SolarWinds	**Splunk**	**ServiceNow**	**Sophos**
SecurityOnion	**RSA SecureID**	**F-Secure**	**Sentry**

Initial Remediations

1. Scan your applications to check whether you are using vulnerable Log4J versions 2.14.1 or under. If a vulnerable version is detected, update to fixed version 2.15.0 to avoid an exploit.
    a. The scans you will be performing are on JAR files, especially nested layers of JAR files and checking their current versions:
        i. https://github.com/mergebase/log4j-detector
        ii. https://github.com/Diverto/nse-log4shell
        iii. https://github.com/anchore/syft
2. Patch any vulnerable device that utilises Log4J versions.
3. Block any outgoing traffic that is not required. If you have a server running a Java application that only needs to accept incoming traffic, prevent everything outgoing.
4. If patching to the latest version is not possible, please refer to the 'Workarounds' section in Annex A.

Vulnerability Overview

Log4J is a Java library that specialises in logging which can be configured via a configuration file, meaning changes are relatively easy to deploy. The output from Log4J can go to the console, but it can also go to an email server, a database table, a log file, or various other destinations. This vulnerability effectively opens the door on numerous platforms to your internal environment or a portal into others.

A specially crafted Apache string, parsed and processed by the vulnerable **Log4J 2** component, will allow malicious actors to trigger the vulnerability through any user provided input.

Successful exploitation allows for arbitrary code execution in the targeted application. No prior access to the system is needed to log the string that can cause the logging event remotely, using commands like curl against a target system to log the malicious string in the application log.

Given the fact that logging code and functionalities in applications and services are typically designed to process a variety of external input data coming from upper layers and from many possible vectors, the biggest risk factor of this vulnerability is predicting whether an application has a viable attack vector path that will allow the malformed exploit string to reach the vulnerable **Log4J 2** code and trigger the attack.

Mitigation and Remediation

The vulnerable versions of Log4j 2 are versions **2.0** to version **2.14.1** inclusive. The first fixed version is **2.15.0**. We strongly encourage you to update to the latest version if you can. Some recommendations from various resources are stating to roll-back to older versions, which are easier to mitigate. However, this could leave your environment vulnerable to different exploits, which are otherwise mitigated by newer instances.

Six Degrees is working on deploying emergency patching on all our managed clients that may be affected. Please await communication from the Six Degrees Client Success Team or a Six Degrees Network Engineer to inform you of the work to be carried out and an allocated timeframe when this maintenance will occur.

If it is suspected that a breach has occurred, Six Degrees offers a Cyber Security Incident Response service to assist with log analytics, incident management and mitigation. This can be requested by contacting the CSOC Team on 03450945065 or emailing soc@cnsgroup.co.uk.

References

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=2021-44228
- https://www.docker.com/blog/apache-log4j-2-cve-2021-44228/
- https://msrc-blog.microsoft.com/2021/12/11/microsofts-response-to-cve-2021-44228-apache-log4j2/
- https://nvd.nist.gov/vuln/detail/CVE-2021-44228
- https://www.reddit.com/r/blueteamsec/comments/rd38z9/log4j_0day_being_exploited/

Annex A – Workarounds

To help mitigate the risk of this vulnerability until the more complete security update can be applied, clients should consider the following mitigations steps. A service restart will be required for these changes to take effect. These workarounds should not be considered a complete solution to resolve this vulnerability:

- In case the **Log4J 2** vulnerable component cannot be updated, **Log4J 2** versions 2.10 to 2.14.1 support the parameter **log4j2.formatMsgNoLookups** to be set to 'true', to disable the vulnerable feature. Ensure this parameter is configured in the startup scripts of the Java Virtual Machine: **-Dlog4j2.formatMsgNoLookups=true.**
- Alternatively, customers using **Log4J** 2.10 to 2.14.1 may set the **LOG4J_FORMAT_MSG_NO_LOOKUPS="true"** environment variable to force this change.
- Kubernetes administrators may use **"kubectl set env"** to set the **LOG4J_FORMAT_MSG_NO_LOOKUPS="true"** environment variable to apply the mitigation across Kubernetes clusters where the Java applications are running Log4j 2.10 to 2.14.1, effectively reflecting on all pods and containers automatically.
- For releases from 2.0-beta9 to 2.10.0, the mitigation is to remove the **JndiLookup** class from the classpath: **zip -q -d log4j-core-*.jar org/apache/logging/log4j/core/lookup/JndiLookup.class**