Logging Services

Security

The Logging Services Security Team takes security seriously. This allows our users to place their trust in Log4j for protecting their mission-critical data. On this page, we will help you find guidance on security-related issues and access to known vulnerabilities.

WARNING

Log4j 1 has reached End of Life in 2015, and is no longer supported. Vulnerabilities reported after August 2015 against Log4j 1 are not checked and will not be fixed. Users should upgrade to Log4j 2 to obtain security fixes.

Getting support

If you need help on building or configuring Logging Services projects or other help on following the instructions to mitigate the known vulnerabilities listed here, please use our user support channels.

TIP

If you need to apply a source code patch, use the building instructions for the project version that you are using. These instructions can be found in `BUILDING.adoc`, `BUILDING.md`, etc. files distributed with the sources.

Reporting vulnerabilities

If you have encountered an unlisted security vulnerability or other unexpected behaviour that has a security impact, or if the descriptions here are incomplete, please report them **privately** to the Logging Services Security Team.

IMPORTANT

We urge you to **carefully read the threat model** detailed in following sections before submitting a report. It guides users on certain safety instructions while using Logging Services software and elaborates on what counts as an unexpected behaviour that has a security impact.

Before reporting a vulnerability, please make sure to check:

- The FAQ for frequently reported issues that are not considered vulnerabilities.
- The list of known vulnerabilities to check if the issue has already been reported.

Common threat model

All the logging frameworks maintained by Apache Logging Services (Log4cxx, Log4j and Log4net) face similar challenges from malicious actors. The following sections outline the most common threats to logging frameworks and clarify the assumptions regarding the origin and trustworthiness of various data sources and the capabilities assumed of potential adversaries. Vulnerability reports that do not adhere to these assumptions will not be accepted and are **not** eligible for the YesWeHack Bug Bounty Program.

User types

Apache Logging Services distinguishes two kinds of users:

Trusted Users

Application developers and administrators are considered **trusted** users. They have unrestricted access to all the features of the logging framework and the environment it is deployed to.

Untrusted Users

All the other users are considered untrusted.

Data sources

Logging systems read data from multiple sources that are controlled by both trusted and untrusted users:

Trusted Sources

- Log4cxx, Log4j, and Log4net **trust** environment variables, configuration properties, and configuration files. To maintain security, the following responsibilities fall on the deployer:
  - Ensure that untrusted parties do not have write access to these resources.
  - Ensure these resources are transmitted only over **confidential** channels (e.g., HTTPS, secure file systems).
  - Be aware that **non-confidential** channels such as HTTP or JMX are **disabled by default** to prevent accidental exposure.
  - If configuration files use interpolation features (e.g., (Log4j Lookups)), ensure that only trusted data sources are used.
  - Pay special attention to values stored in the context map (see Thread Context in Log4j). Although the context map is only accessible by developers, it has been known to include user-provided data, such as HTTP headers, which can introduce risks.
- The logging frameworks **trust** that the objects passed to the log statements can be safely converted to strings:

    - These frameworks should not be used to log deserialized data from untrusted sources. See the related OWASP guide for details.
- If parameterized logging is used, the format string is **trusted**:
    - Programmers **should** use compile-time constants as format strings to prevent attackers from tampering messages. See Don't use string concatenation for an example.

Untrusted Sources

- Log4cxx, Log4j and Log4net **do not** trust log messages. No particular input validation for log messages is necessary.
- They **do not** trust the string representation of log parameters.
- The logging frameworks do not trust neither the keys nor the values in the thread context.

Adversary capabilities

The threats listed below are evaluated against an adversary with a well-defined and limited set of capabilities. Defining these capabilities clarifies which reports are in scope: a report that requires a capability not listed here is **not** considered a vulnerability.

In-scope adversary

An in-scope adversary is any party whose data reaches the logging framework **exclusively** through the untrusted sources described above. Such an adversary is assumed to be able to:

- Submit arbitrary byte sequences, including malformed text encodings and control characters (such as `CR`, `LF` and `NUL`), through log messages, the string representation of log parameters, and the keys and values of the thread context.
- Submit excessively long inputs, within whatever limits the calling application enforces.
- Submit input that resembles the framework's own interpolation or lookup syntax, including input that triggers recursive interpolation.

Out-of-scope adversary

The following adversaries are explicitly **out of scope**; a report relying on any of these capabilities will not be accepted:

- An adversary able to modify environment variables, configuration properties, or configuration files: these are trusted sources (see Data sources).
- An adversary able to execute arbitrary code in the same process as the logging framework. Code running in the same process shares the same trust level as the logging framework itself; there is no boundary to enforce. This includes code introduced through plugins, custom appenders, or other application extensions.

- An adversary able to cause a self-referential or otherwise non-terminating object structure to be passed to a log statement. The logging frameworks trust that logged objects can be safely converted to a string; converting such a structure is the responsibility of the calling code.
- An adversary observing side channels, such as the timing or memory behavior of the logging framework.
- A malicious destination of an appender (e.g. a hostile database, message broker, or mail server). Appender destinations are configured by trusted users and are treated as an extension of the deployer.

Threats

These are the most commonly encountered threats for users of Log4cxx, Log4j and Log4net:

Log Injection (CWE-117)

Log injection is a common attack vector to hide malicious activity in an application. Regarding this threat:

- **Unstructured layouts** such as Pattern Layout in Log4j do **not** protect users from log injection. These layouts are meant for **human** and not computer consumption.
- Log4cxx, Log4j and Log4net **must** prevent log injection in **structured** layouts, such as XML, JSON and RFC 5424.

Supply chain attacks (CWE-1357)

- Apache Logging Services projects **do** check the quality of our dependencies.
- Deprecated components such as the Cassandra, Kafka and CouchDB appenders are provided for backward compatibility purposes only. While we actively check for vulnerabilities in those components, they are *de facto* unmaintained, and we discourage their usage in production.
- All Apache Logging Services are signed with one of the keys in the Logging Services PMC KEYS file. We do **not** support artifacts that do not have a valid signature, and we encourage users to always check the integrity of the downloaded components. Additional information on how to verify releases signatures is available on the Download page

Information disclosure (CWE-200)

Since logging frameworks implement information disclosure by design:

- It is up to the deployer to prevent unauthorized access to log files and to ensure that the appropriate log levels are configured.
- It is up to the programmer to document which log levels and markers *might* contain sensitive data. Attention should be brought to the fact that libraries on which an application depends might have a different log level and marker convention.

- **Log masking** techniques are out-of-scope for Log4cxx, Log4j, and Log4net. It is up to the developer to ensure that sensitive data is properly masked **before** it is passed to the logging implementation. For this purpose, **third-party** frameworks like Safe-Logging should be used.

Log reliability (e.g. CVE-778)

Log4j is designed with **reliability** in mind:

- By **default**, Log4j **should** deliver log events to the appropriate resource even during a reconfiguration event or will log an error.
- While log events will be delivered to a resource, not all resources provide a confirmation mechanism. To ensure reliability along the entire logging pipeline, it is up to the deployer to use reliable transmission components: files, loopback network sockets or message-queue-based systems for example.
- Log4j provides configuration options that discard log events if the load on the application is high. Using these options invalidates the reliability guarantees.

Denial of service (CVE-779)

Since our logging frameworks are designed with reliability in mind:

- Our frameworks go to great lengths to minimize performance overhead, minimize latency and maximizing throughput. Since a universal solution does not exist, many configuration options exist to adapt the performance characteristics to a specific application. See Performance for more information.
- It is up to the deployer to ensure that the appenders can keep up with the logs written by using the appropriate appenders and configuring the appropriate level of logs.
- It is up to the developer to ensure that log statements, which are not enabled, generate minimal overhead. See the Log4j API Best Practices, for example.

Improper neutralization of Special Elements (CWE-138)

- Log4cxx, Log4j, and Log4net **do** allow users to pass untrusted strings to log statements and thread context, except in the format string of parameterized logging, as mentioned above.

Deserialization of untrusted data (CWE-502)

Log4cxx, Log4j, and Log4net **do not** deserialize data from any source as part of their normal operation. For backward compatibility, several classes in Log4j 2 and Log4net 2 still implement `Serializable` (in Java) or carry the `[Serializable]` attribute (in .NET); Log4j's `log4j-api` also ships an allowlist-based `FilteredObjectInputStream` utility to assist applications that nonetheless deserialize log event streams.

Regarding this threat:

- We provide **no guarantee** that deserializing a stream containing classes from these projects is safe, regardless of the source of the stream.
- Filtering such a stream by the `org.apache.logging` Java package, the `log4net` .NET namespace, or any allowlist derived from project-owned types is **not** sufficient to make deserialization safe.
- The hardening utilities we ship are **partial** and **not exhaustive**; bypasses are treated as opportunities for further hardening, not as vulnerabilities in the project.
- The application performing the deserialization is responsible for ensuring that the byte stream originates from a **trusted source**.

See the FAQ entry on CWE-502 for the recommended alternatives.

Revising this threat model

This threat model reflects the current design of Log4cxx, Log4j, and Log4net. It is **not** immutable: a revision is required whenever a change to one of the frameworks invalidates an assumption stated above. In particular, this document must be revisited when any of the following becomes true:

- A new public API is added that accepts a kind of input not yet covered by Data sources.
- An existing entry point begins to accept input from a new source, changing whether that input is trusted or untrusted.
- A security-relevant default changes, or a configuration option that affects the security posture is added or removed.
- A framework gains a network listener or any other inbound surface of its own.
- A vulnerability report cannot be cleanly classified as either in scope or out of scope using the assumptions above.

Internal refactors that do not change any of the above do **not** require a revision. Proposed changes to this document are reviewed by the Apache Logging Services PMC.

Vulnerability handling policy

The Logging Services Security Team follows the ASF Project Security guide for handling security vulnerabilities.

Reported security vulnerabilities are subject to voting (by means of *lazy approval*, preferably) in the private security mailing list before creating a CVE and populating its associated content. This procedure involves only the creation of CVEs and blocks neither (vulnerability) fixes, nor releases.

Vulnerability Disclosure Report (VDR)

Many Logging Services projects distribute CycloneDX Software Bill of Materials (SBOM) along with each deployed artifact. This is streamlined by Logging Parent for Maven-based projects.

Produced SBOMs contain BOM-links referring to a CycloneDX Vulnerability Disclosure Report (VDR) that Apache Logging Services uses for all projects it maintains. This VDR is accessible through the following URL: https://logging.apache.org/cyclonedx/vdr.xml

Known vulnerabilities

The Logging Services Security Team believes that accuracy, completeness and availability of security information is essential for our users. We choose to pool all information on this one page, allowing easy searching for security vulnerabilities over a range of criteria.

NOTE

Version ranges follow the VERS specification:

- Log4cxx: `semver` scheme
- Log4j: `maven` scheme
- Log4net: `nuget` scheme

For brevity, mathematical interval notation is used, with the union operator (`∪`) to represent multiple ranges.

CVE-2026-40023

Summary	Silent log event loss in `XMLLayout` due to unescaped XML 1.0 forbidden characters
CVSS 4.x Score & Vector	6.3 MEDIUM (CVSS:4.0/AV:N/AC:H/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:L/SA:N)
Components affected	Log4cxx
Versions affected	`[0, 1.7.0)`
Versions fixed	`1.7.0`

Description

Apache Log4cxx's `XMLLayout`, in versions before 1.7.0, fails to sanitize characters forbidden by the XML 1.0 specification in log messages, NDC, and MDC property keys and values, producing invalid XML out-

put. Conforming XML parsers must reject such documents with a fatal error, which may cause downstream log processing systems to drop or fail to index affected records.

An attacker who can influence logged data can exploit this to suppress individual log records, impairing audit trails and detection of malicious activity.

Remediation

Users are advised to upgrade to Apache Log4cxx version `1.7.0`, which fixes this issue.

Credits

This issue was discovered by Olawale Titiloye.

References

- CVE-2026-40023
- Pull request that fixes the issue

CVE-2026-40021

Summary	Silent log event loss in `XmlLayout` and `XmlLayoutSchemaLog4J` due to unescaped XML 1.0 forbidden characters
CVSS 4.x Score & Vector	6.3 MEDIUM (CVSS:4.0/AV:N/AC:H/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:L/SA:N)
Components affected	Log4net
Versions affected	`[0, 3.3.0)`
Versions fixed	`3.3.0`

Description

Apache Log4net's `XmlLayout` and `XmlLayoutSchemaLog4J`, in versions before 3.3.0, fail to sanitize characters forbidden by the XML 1.0 specification in MDC property keys and values, as well as the identity field that may carry attacker-influenced data. This causes an exception during serialization and the silent loss of the affected log event.

An attacker who can influence any of these fields can exploit this to suppress individual log records, impairing audit trails and detection of malicious activity.

Remediation

Users are advised to upgrade to Apache Log4net version `3.3.0`, which fixes this issue.

Credits

This issue was discovered by f00dat.

References

- CVE-2026-40021
- Pull request that fixes the issue

CVE-2026-34481

Summary	Improper serialization of non-finite floating-point values in `JsonTemplateLayout`
CVSS 4.x Score & Vector	6.3 MEDIUM (CVSS:4.0/AV:N/AC:H/AT:P/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:L/SA:N)
Components affected	`log4j-layout-template-json`
Versions affected	`[2.14.0, 2.25.4) ∪ [3.0.0-alpha1, 3.0.0-beta3]`
Versions fixed	`2.25.4`

Description

Apache Log4j's `JsonTemplateLayout`, in versions up to and including 2.25.3, produces invalid JSON output when log events contain non-finite floating-point values (`NaN`, `Infinity`, or `-Infinity`), which are prohibited by RFC 8259. This may cause downstream log processing systems to reject or fail to index affected records.

An attacker can exploit this issue only if both of the following conditions are met:

- The application uses `JsonTemplateLayout`.
- The application logs a `MapMessage` containing an attacker-controlled floating-point value.

Remediation

Users are advised to upgrade to Apache Log4j JSON Template Layout version `2.25.4`, which corrects this issue.

Credits

This issue was discovered by Ap4sh (Samy Medjahed) and Ethicxz (Eliott Laurie).

References

- CVE-2026-34481
- Pull request that fixes the issue

CVE-2026-34480

Summary	Silent log event loss in `XmlLayout` due to unescaped XML 1.0 forbidden characters
CVSS 4.x Score & Vector	6.9 MEDIUM (CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:L/SA:N)
Components affected	Log4j Core
Versions affected	`[2.0-alpha1, 2.25.4) ∪ [3.0.0-alpha1, 3.0.0-beta3]`
Versions fixed	`2.25.4`

Description

Apache Log4j Core's `XmlLayout`, in versions up to and including 2.25.3, fails to sanitize characters forbidden by the XML 1.0 specification producing invalid XML output whenever a log message or MDC value contains such characters.

The impact depends on the StAX implementation in use:

- **JRE built-in StAX:** Forbidden characters are silently written to the output, producing malformed XML. Conforming parsers must reject such documents with a fatal error, which may cause downstream log-processing systems to drop the affected records.
- **Alternative StAX implementations** (e.g., Woodstox, a transitive dependency of the Jackson XML Dataformat module): An exception is thrown during the logging call, and the log event is never delivered to its intended appender, only to Log4j's internal status logger.

Remediation

Users are advised to upgrade to Apache Log4j Core version `2.25.4`, which corrects this issue by sanitizing forbidden characters before XML output.

Credits

This issue was originally reported by Ap4sh (Samy Medjahed) and Ethicxz (Eliott Laurie), and independently reported by jabaltarik1.

References

- CVE-2026-34480
- Pull request that fixes the issue

CVE-2026-34479

Summary	Silent log event loss in `Log4j1XmlLayout` due to unescaped XML 1.0 forbidden characters
CVSS 4.x Score & Vector	6.9 MEDIUM (CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:L/SA:N)
Components affected	`log4j-1.2-api`
Versions affected	`[2.7, 2.25.4) ∪ [3.0.0-alpha1, 3.0.0-beta2]`
Versions fixed	`2.25.4`

Description

The `Log4j1XmlLayout` from the Apache Log4j 1-to-Log4j 2 bridge fails to escape characters forbidden by the XML 1.0 standard, producing malformed XML output. Conforming XML parsers are required to reject documents containing such characters with a fatal error, which may cause downstream log processing systems to drop or fail to index affected records.

Two groups of users are affected:

- Those using `Log4j1XmlLayout` directly in a Log4j Core 2 configuration file.
- Those using the Log4j 1 configuration compatibility layer with `org.apache.log4j.xml.XMLLayout` specified as the layout class.

Remediation

Users are advised to upgrade to Apache Log4j 1-to-Log4j 2 bridge version `2.25.4`, which corrects this issue.

NOTE

The Apache Log4j 1-to-Log4j 2 bridge is deprecated and will not be present in Log4j 3. Users are encouraged to consult the Log4j 1 to Log4j 2 migration guide, and specifically the section on eliminating reliance on the bridge.

Credits

This issue was originally reported by Ap4sh (Samy Medjahed) and Ethicxz (Eliott Laurie), and independently reported by jabaltarik1.

References

- CVE-2026-34479
- Pull request that fixes the issue

CVE-2026-34478

Summary	Log injection in `Rfc5424Layout` due to silent configuration incompatibility
CVSS 4.x Score & Vector	6.9 MEDIUM (CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:L/SA:N)
Components affected	Log4j Core
Versions affected	`[2.21.0, 2.25.4) ∪ [3.0.0-beta1, 3.0.0-beta3]`
Versions fixed	`2.25.4`

Description

Apache Log4j Core's `Rfc5424Layout`, in versions 2.21.0 through 2.25.3, is vulnerable to log injection via CRLF sequences due to undocumented renames of security-relevant configuration attributes.

Two distinct issues affect users of stream-based syslog services who configure `Rfc5424Layout` directly:

- The `newLineEscape` attribute was silently renamed, causing newline escaping to stop working for users of TCP framing (RFC 6587), exposing them to CRLF injection in log output.
- The `useTlsMessageFormat` attribute was silently renamed, causing users of TLS framing (RFC 5425) to be silently downgraded to unframed TCP (RFC 6587), without newline escaping.

Users of the `SyslogAppender` are not affected, as its configuration attributes were not modified.

Remediation

Users are advised to upgrade to Apache Log4j Core version `2.25.4`, which corrects this issue.

Credits

This issue was discovered by Samuli Leinonen.

References

- CVE-2026-34478
- Pull request that fixes the issue

CVE-2026-34477

Summary	`verifyHostName` attribute silently ignored in TLS configuration
CVSS 4.x Score & Vector	6.3 MEDIUM (CVSS:4.0/AV:N/AC:H/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:L/SA:N)
Components affected	Log4j Core
Versions affected	`[2.12.0, 2.25.4) ∪ [3.0.0-alpha1, 3.0.0-beta3]`
Versions fixed	`2.25.4`

Description

The fix for CVE-2025-68161 was incomplete: it addressed hostname verification only when enabled via the `log4j2.sslVerifyHostName` system property, but not when configured through the `verifyHostName` attribute of the `<Ssl>` element.

Although the `verifyHostName` configuration attribute was introduced in Log4j Core 2.12.0, it was silently ignored in all versions through 2.25.3, leaving TLS connections vulnerable to interception regardless of the configured value.

A network-based attacker may be able to perform a man-in-the-middle attack when **all** of the following conditions are met:

- An SMTP, Socket, or Syslog appender is in use.
- TLS is configured via a nested `<Ssl>` element.
- The attacker can present a certificate issued by a CA trusted by the appender's configured trust store, or by the default Java trust store if none is configured.

This issue does not affect users of the HTTP appender, which uses a separate `verifyHostname` attribute that was not subject to this bug and verifies host names by default.

Remediation

Users are advised to upgrade to Apache Log4j Core version `2.25.4`, which corrects this issue.

Credits

This issue was originally reported by Samuli Leinonen and independently reported by Naresh Kandula, Vitaly Simonovich, Raijuna, Danish Siddiqui (djvirus), Markus Magnuson, and Haruki Oyama (Waseda University).

References

- CVE-2026-34477
- Pull request that fixes the issue

CVE-2025-68161

Summary	Missing TLS hostname verification in Socket appender
CVSS 4.x Score & Vector	6.3 MEDIUM (CVSS:4.0/AV:N/AC:H/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:L/SA:N)
Components affected	Log4j Core
Versions affected	`[2.0-beta9, 2.25.3) ∪ [3.0.0-alpha1, 3.0.0-beta3]`
Versions fixed	`2.25.3`

Description

The Socket Appender in affected Log4j Core versions does not perform TLS hostname verification of the peer certificate, even when the `verifyHostName` configuration attribute or the `log4j2.sslVerifyHostName` system property is set to `true`.

This issue may allow a man-in-the-middle attacker to intercept or redirect log traffic under the following conditions:

- The attacker is able to intercept or redirect network traffic between the client and the log receiver.
- The attacker can present a server certificate issued by a certification authority trusted by the Socket Appender's configured trust store (or by the default Java trust store if no custom trust store is configured).

Remediation

Users are advised to upgrade to Log4j Core version `2.25.3`, which fully addresses this issue.

For earlier versions, the risk can be reduced by carefully restricting the trust store used by the Socket Appender.

NOTE

NOTE

When configuring a trust store for Log4j Core, we recommend following established best practices. For example, NIST SP 800-52 Rev. 2 (§4.5.2) recommends using a trust store that contains only the CA certificates required for the intended communication scope, such as a private or enterprise CA.

Credits

This issue was discovered by Samuli Leinonen.

It was reported through the Log4j Bug Bounty Program on YesWeHack funded by the Sovereign Tech Agency.

References

- CVE-2025-68161
- Pull request that fixes the issue

CVE-2025-54813

Summary	Improper escaping with JSONLayout
CVSS 4.x Score & Vector	6.3 MEDIUM (CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:N/VI:L/VA:N/SC:N/SI:L/SA:N)
Components affected	Log4cxx
Versions affected	`[0.11.0, 1.5.0)`
Versions fixed	`1.5.0`

Description

When using `JSONLayout`, not all payload bytes are properly escaped. If an attacker-supplied message contains certain non-printable characters, these will be passed along in the message and written out as part of the JSON message. This may prevent applications that consume these logs from correctly interpreting the information within them.

Remediation

Users are recommended to upgrade to version `1.5.0`, which fixes the issue.

Credits

This issue was discovered and remediated with support from the Sovereign Tech Agency, through the Log4j Bug Bounty Program on YesWeHack.

References

- CVE-2025-54813
- Pull request that fixes the issue

CVE-2025-54812

Summary	Improper HTML escaping in HTMLLayout
CVSS 4.x Score & Vector	2.1 LOW (CVSS:4.0/AV:N/AC:H/AT:N/PR:N/UI:A/VC:L/VI:L/VA:N/SC:L/SI:L/SA:N)
Components affected	Log4cxx
Versions affected	`[0, 1.5.0)`
Versions fixed	`1.5.0`

Description

When using `HTMLLayout`, logger names are not properly escaped when writing out to the HTML file. If untrusted data is used to retrieve the name of a logger, an attacker could theoretically inject HTML or JavaScript in order to hide information from logs or steal data from the user. In order to activate this, the following sequence must occur:

- Log4cxx is configured to use `HTMLLayout`.
- Logger name comes from an untrusted string.
- Logger with compromised name logs a message.
- User opens the generated HTML log file in their browser, leading to potential XSS.

Because logger names are generally constant strings, we assess the impact to users as LOW.

Remediation

Users are recommended to upgrade to version `1.5.0`, which fixes the issue.

Credits

This issue was discovered and remediated with support from the Sovereign Tech Agency, through the Log4j Bug Bounty Program on YesWeHack.

References

- CVE-2025-54812
- Pull request #509
- Pull request #514

CVE-2021-44832

Summary	JDBC appender is vulnerable to remote code execution in certain configurations
CVSS 3.x Score & Vector	6.6 MEDIUM (CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H)
Components affected	`log4j-core`
Versions affected	`[2.0-beta7, 2.3.1) ∪ [2.4, 2.12.3) ∪ [2.13.0, 2.17.0)`
Versions fixed	`2.3.1` (for Java 6), `2.12.3` (for Java 7), or `2.17.0` (for Java 8 and later)

Description

An attacker with write access to the logging configuration can construct a malicious configuration using a JDBC Appender with a data source referencing a JNDI URI which can execute remote code. This issue is fixed by limiting JNDI data source names to the `java` protocol.

Mitigation

Upgrade to `2.3.1` (for Java 6), `2.12.3` (for Java 7), or `2.17.0` (for Java 8 and later).

In prior releases confirm that if the JDBC Appender is being used it is not configured to use any protocol other than `java`.

References

- CVE-2021-44832
- LOG4J2-3242

CVE-2021-45105

Summary	Infinite recursion in lookup evaluation
CVSS 3.x Score & Vector	5.9 MEDIUM (CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H)
Components affected	`log4j-core`
Versions affected	`[2.0-alpha1, 2.3.1) ∪ [2.4, 2.12.3) ∪ [2.13.0, 2.17.0)`
Versions fixed	`2.3.1` (for Java 6), `2.12.3` (for Java 7), and `2.17.0` (for Java 8 and later)

Description

Log4j versions `2.0-alpha1` through `2.16.0` (excluding `2.3.1` and `2.12.3`), did not protect from uncontrolled recursion that can be implemented using self-referential lookups. 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 that contains a recursive lookup, resulting in a `StackOverflowError` that will terminate the process. This is also known as a *DoS (Denial-of-Service)* attack.

Mitigation

Upgrade to `2.3.1` (for Java 6), `2.12.3` (for Java 7), or `2.17.0` (for Java 8 and later).

Alternatively, this infinite recursion issue can be mitigated in configuration:

- In PatternLayout in the logging configuration, replace Context Lookups like `${ctx:loginId}` or `$${ctx:loginId}` with Thread Context Map patterns (`%X` , `%mdc` , or `%MDC`).
- Otherwise, in the configuration, remove references to Context Lookups like `${ctx:loginId}` or `$${ctx:loginId}` where they originate from sources external to the application such as HTTP headers or user input. Note that this mitigation is insufficient in releases older than `2.12.2` (for Java 7), and `2.16.0` (for Java 8 and later) as the issues fixed in those releases will still be present.

Note that only the `log4j-core` JAR file is impacted by this vulnerability. Applications using only the `log4j-api` JAR file without the `log4j-core` JAR file are not impacted by this vulnerability.

Credits

Independently discovered by Hideki Okamoto of Akamai Technologies, Guy Lederfein of Trend Micro Research working with Trend Micro's Zero Day Initiative, and another anonymous vulnerability researcher.

References

- CVE-2021-45105
- LOG4J2-3230

CVE-2021-45046

Summary	Thread Context Lookup is vulnerable to remote code execution in certain configurations
CVSS 3.x Score & Vector	9.0 CRITICAL (CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H)
Components affected	`log4j-core`
Versions affected	`[2.0-beta9, 2.3.1) ∪ [2.4, 2.12.3) ∪ [2.13.0, 2.16.0)`
Versions fixed	`2.3.1` (for Java 6), `2.12.3` (for Java 7), and `2.16.0` (for Java 8 and later)

Description

It was found that the fix to address CVE-2021-44228 in Log4j `2.15.0` was incomplete in certain non-default configurations. When the logging configuration uses a non-default Pattern Layout with a Thread Context Lookup (for example, `$${ctx:loginId}`), attackers with control over Thread Context Map (MDC) 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. Remote code execution has been demonstrated on macOS, Fedora, Arch Linux, and Alpine Linux.

Note that this vulnerability is not limited to just the JNDI lookup. Any other Lookup could also be included in a Thread Context Map variable and possibly have private details exposed to anyone with access to the logs.

Note that only the `log4j-core` JAR file is impacted by this vulnerability. Applications using only the `log4j-api` JAR file without the `log4j-core` JAR file are not impacted by this vulnerability.

Mitigation

Upgrade to Log4j `2.3.1` (for Java 6), `2.12.3` (for Java 7), or `2.16.0` (for Java 8 and later).

Credits

This issue was discovered by Kai Mindermann of iC Consult and separately by 4ra1n.

Additional vulnerability details discovered independently by Ash Fox of Google, Alvaro Muñoz and Tony Torralba from GitHub, Anthony Weems of Praetorian, and RyotaK (@ryotkak).

References

- CVE-2021-45046
- LOG4J2-3221

CVE-2021-44228

Summary	JNDI lookup can be exploited to execute arbitrary code loaded from an LDAP server
CVSS 3.x Score & Vector	10.0 CRITICAL (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H)
Components affected	`log4j-core`
Versions affected	`[2.0-beta9, 2.3.1) ∪ [2.4, 2.12.2) ∪ [2.13.0, 2.15.0)`
Versions fixed	`2.3.1` (for Java 6), `2.12.2` (for Java 7), and `2.15.0` (for Java 8 and later)

Description

In Log4j, the JNDI features used in configurations, 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.

Note that only the `log4j-core` JAR file is impacted by this vulnerability. Applications using only the `log4j-api` JAR file without the `log4j-core` JAR file are not impacted by this vulnerability.

Mitigation

Log4j 1 mitigation

WARNING

> Log4j 1 has reached End of Life in 2015, and is no longer supported. Vulnerabilities reported after August 2015 against Log4j 1 are not checked and will not be fixed. Users should upgrade to Log4j 2 to obtain security fixes.

Log4j 1 does not have Lookups, so the risk is lower. Applications using Log4j 1 are only vulnerable to this attack when they use JNDI in their configuration. A separate CVE (CVE-2021-4104) has been filed for this vulnerability. To mitigate, audit your logging configuration to ensure it has no `JMSAppender` configured. Log4j 1 configurations without `JMSAppender` are not impacted by this vulnerability.

Log4j 2 mitigation

Upgrade to Log4j `2.3.1` (for Java 6), `2.12.2` (for Java 7), or `2.15.0` (for Java 8 and later).

Credits

This issue was discovered by Chen Zhaojun of Alibaba Cloud Security Team.

References

- CVE-2021-44228
- LOG4J2-3198
- LOG4J2-3201
- LOG4J2-3242

CVE-2020-9488

Summary	Improper validation of certificate with host mismatch in SMTP appender
CVSS 3.x Score & Vector	3.7 LOW (CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N)
Components affected	`log4j-core`
Versions affected	`[2.0-beta1, 2.3.2) ∪ [2.4, 2.12.3) ∪ [2.13.0, 2.13.2)`
Versions fixed	`2.3.2` (for Java 6), `2.12.3` (for Java 7) and `2.13.2` (for Java 8 and later)

Description

Improper validation of certificate with host mismatch in SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender.

The reported issue was caused by an error in `SslConfiguration`. Any element using `SslConfiguration` in the Log4j `Configuration` is also affected by this issue. This includes `HttpAppender`, `SocketAppender`, and `SyslogAppender`. Usages of `SslConfiguration` that are configured via system properties are not affected.

Mitigation

Upgrade to `2.3.2` (Java 6), `2.12.3` (Java 7) or `2.13.2` (Java 8 and later).

Alternatively, users can set the `mail.smtp.ssl.checkserveridentity` system property to `true` to enable SMTPS hostname verification for all SMTPS mail sessions.

Credits

This issue was discovered by Peter Stöckli.

References

- CVE-2020-9488
- LOG4J2-2819

CVE-2018-1285

Summary	XXE via attacker-controlled log4net config files
CVSS 3.x Score & Vector	9.8 HIGH (AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H)
Components affected	`log4net`
Versions affected	`[0,2.0.10)`
Versions fixed	`2.0.10`

Description

Apache log4net versions before 2.0.10 do not disable XML external entities when parsing log4net configuration files. This allows for XXE-based attacks in applications that accept attacker-controlled log4net configuration files.

Threat Model

According to the current threat model, this is no longer considered a vulnerability. The attack requires an attacker-controlled log4net configuration file, which is outside the scope of the threat model.

Mitigation

Users are advised to upgrade to Apache Log4net version `2.0.10`, which fixes this issue.

Credits

This issue was discovered by Karthik Kumar Balasundaram.

References

- CVE-2018-1285
- LOG4NET-575

- Security fix commit
- Pull request that fixes the issue

CVE-2017-5645

Summary	TCP/UDP socket servers can be exploited to execute arbitrary code
CVSS 2.0 Score & Vector	7.5 HIGH (AV:N/AC:L/Au:N/C:P/I:P/A:P)
Components affected	`log4j-core`
Versions affected	`[2.0-alpha1, 2.8.2)`
Versions fixed	`2.8.2` (for Java 7 and later)

Description

When using the TCP socket server or UDP socket server to receive serialized log events from another application, a specially crafted binary payload can be sent that, when deserialized, can execute arbitrary code.

Mitigation

Java 7 and above users should migrate to version `2.8.2` or avoid using the socket server classes. Java 6 users should avoid using the TCP or UDP socket server classes, or they can manually backport the security fix commit from `2.8.2`.

Credits

This issue was discovered by Marcio Almeida de Macedo of Red Team at Telstra.

References

- CVE-2017-5645
- LOG4J2-1863
- Security fix commit

owners.