Study of the Most Critical Security Vulnerability of the Decade: Log4Shell

Santosh Pai
Research Scholar
College of Computer Science and Information Science
Srinivas University
Mangalore, India
ORCID ID: 0000000250531673
E-Mail: g.santoshpai@gmail.com

Dr. Srinivasa Rao Kunte R
Research Professor
College of Computer Science and Information Science
Srinivas University
Mangalore, India
ORCID ID: 0000000250621505
E-Mail: kuntesrk@gmail.com

***Abstract*—Traceability is one of the crucial features of a software application. Logging is considered one of the essential features supported by any software. Logging helps debug the faults in the software by executing the erroneous flows in the software. Log4j2 is a prevalent Opensource Logging Framework created and maintained by Apache Foundation. Several popular Java-based software applications depend on the Log4j2 library for effective logging. Logging being a non-critical and non-functional software requirement, the industry was not expecting a Critical security vulnerability. Different security researchers discovered five security vulnerabilities. Most of them are dangerous and actively exploited vulnerabilities. Attackers actively exploit some of these vulnerabilities. This paper discusses the security vulnerabilities found in Log4j2 in 2021, popularly known as Log4Shell. The article also suggests techniques to avoid such security risks.**

Keywords— Log4j2, Log4Shell, Logging, Java.

I. INTRODUCTION

Modern Software applications are Cloud deployed. Cloud adoption has several advantages such as Availability, Reliability, Resilience, and Reduced Cost compared to hosting a Data Center by the organization itself. However, a significant challenge is debugging the application, as the infrastructure is often not controlled by the application creator. Logs are tools to identify software issues and understand events leading to software issues. Traditional logging involved creating text strings and storing them in the file. There was a need to develop a standard logging library to be re-used by other software applications. Log4j was born to solve this problem for Java applications. Java being a popular Object-Oriented programming language, Log4j was adopted quickly after its general availability. Most of the famous and widely used Java applications use Log4j for logging. To further enhance the logging functionalities provided, Apache Foundation launched Log4j2 in 2012. After its initial proof of concept, the public release happened in 2014. It has revolutionized logging by providing several features over the years. Log4j2 has done 48 releases since its initial release in 2012.

II. LOG4SHELL

Chen Zhaojun, an employee of Alibaba Cloud, discovered a security issue in the Log4j2 library [1]. They reported the problem to Apache Foundation with the details of the possible exploits. The Log4j2 team created tickets [2] [3] to track the issue, and they immediately started to work on it. Apache Foundation assigned a CVSS score of 10.0 to the problem and publicly disclosed the case on Dec. 9, 2021 [4]. The vulnerability disclosure came with a workaround to disable the faulty implementation and a fix to upgrade the library to the latest version.

Some scholarly articles discuss Log4Shell. The papers discuss the impacts and how one specific organization overcomes the effect [5] [6] [7] [8]. We discuss Log4Shell vulnerability impacts on the organization and the software development lifecycle, specifically code commits and releases.

III. WORKING PRINCIPLE

A threat exploits a security vulnerability. A threat actor is performing actions to exploit a vulnerability. For an attacker to safely execute the attack, many factors shall be in favor. An attacker shall be able to trigger an attack with the least cost and shall remain undetected by the security system installed. Log4Shell does not require an attacker to authenticate to the application, and hence it was both easy to trigger the attack and stay undetected.

Triggering the attack requires the following steps [9].

1. Attacker sets up an LDAP Server.
   Complexity: Simple steps taking less than an hour to install and configure.
   Cost: Opensource LDAP Servers are available without any software license cost [10] [11]. System perquisites of 384 MB RAM and Java 8 [12] are basic configurations in a modern computer. The attack does not need a specially designed computer.
2. The attacker sends a crafted message to the vulnerable application.
   Complexity: Message creation takes less than an hour.
   Cost: Attack is triggered using free tools such as curl [13]. There is no associated software license cost.
3. The vulnerable application connects to the attacker's LDAP server to download the malicious payload.
4. The vulnerable application executes the malicious payload [14].

At this stage, the application is in the control of the attacker. Figure 1 shows the attack steps.

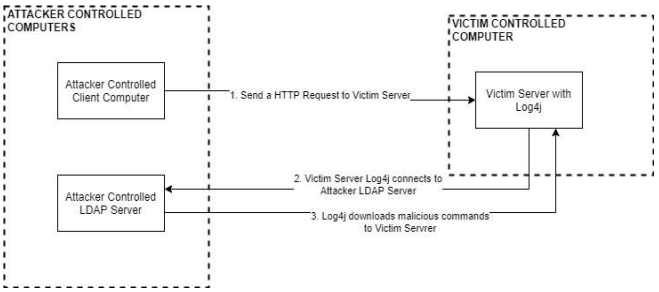

FIGURE 1: LOG4SHELL ATTACK

Using this technique, an attacker could trigger different types of attacks described below.

Remote Code Execution: The attacker can make the software application download the commands from the attacker server. The victim's computer executes the downloaded commands, allowing a remote code execution opportunity [15].

Privilege Escalation: Applications on the victim server execute as a privileged operation. Privileged operations can perform administrative actions on the server, including switching off other functions and rebooting the servers. An attacker could execute privileged commands on the server via the attacked application.

Denial of Service: Availability is one of the crucial requirements for an application. Using the Log4Shell attack, an attacker could slow down the victim server by overloading the server with large computation commands. The server starts to deny the requests from legitimate clients creating an availability problem for the business.

IV. IMPACT OF LOG4SHELL

We used National Cyber Security Centrum, Netherlands data to understand the impacted organizations. The data is available as a text file in a publicly accessible GitHub repository [16]. We converted the text file data into tables and sorted them by the number of products impacted. The list has 98 vendors and 583 products. Table 1 indicates the vendor with more than ten products affected due to Log4Shell.

TABLE 1. IMPACTED ORGANIZATIONS AND PRODUCTS

Vendor Name	*Count of impacted Products*
VMware	56
HPE	40
Siemens Healthineers	34
Lenovo	33
Cisco	30
Cloudera	30
Avaya	28
Philips	18
Schneider Electric	18
Fortinet	14
Splunk	13
Thales	13
OxygenXML	12
Atlassian	11
Apache	10
IBM	10
Intel	10

There are many other vendors and products not listed in this. The public cloud service provider, Amazon Web Service, has reported more than fifty impacted products and services in their Log4j security advisory [17]. Google Cloud has documented more than six services impacted [18]. McAfee disclosed several products affected in their advisory [19], and so did RedHat [20].

The vulnerability is impacting tech giants and their products. The organizations create advisories and update them regularly to provide the latest information to their customers.

V. ROOT CAUSE ANALYSIS

The technical community highly adopts Log4J. In 2013, a feature titled "JNDI Lookup plugin support" was added to the library [21]. A typical logging implementation would format the user input and store it in a file or a database. The newly added feature allowed the execution of the commands at the server instead of logging it. The client can craft and submit the malicious request to the server using simple HTTP requests [22].

Log4Shell is the feature that allows the execution of untrusted commands from external clients. The feature is enabled by default, making all implementations vulnerable.

VI. LOG4SHELL MITIGATION

Contributors of the Log4j2 project and Apache Foundation quickly published workarounds [23] and fixes for the exploits. Log4j2 code is available as open source on Github [24]. Using the known code and related code change history, we collected data to analyze the impact of Log4Shell on implementation changes in the code repository. Following sub-sections detail each factor.

A. *Time to fix*

We collected data from the vulnerability disclosures and corresponding code changes in the Log4j2 code base [25]. Table 2 details the vulnerability fix time. Time to Fix is the days between public disclosure of the vulnerability by Apache Foundation and the release of correction for the vulnerability.

TABLE 2. VULNERABILITY TIME TO FIX

CVE Number	*Qualitative Score*	*Time to Fix*
CVE-2021-44228	Critical	9

Observations: It took nine days to fix the issue causing the Critical vulnerability. The Apache Foundation had published a workaround used by the organizations to protect their applications from Log4Shell. Nine days look high considering the critically of the issue. However, several

factors that impact fixing a problem, including knowledge of the codebase, re-producing the issues, testing all regression cases, etc., are time-consuming. These are extremely important to avoid breaking some other part of the software. Overall, the Time to Fix for Log4Shell was handled well by the contributors from Apache Foundation.

B. Size of Code Change

Another data collected on the vulnerabilities is the size of change done in the code to remove the security issue from the Log4j2 library. Table 3 indicates the number of files in the code base modified by adding and removing new code blocks. The changes are measured using the output obtained from the code change history from the GitHub repository [25].

TABLE 3. CODE CHANGE SIZE

CVE Number	***Qualitative Score***	***Size of code change***		
		Number of Files Modified	***Number of Lines Added***	***Number of Lines Removed***
CVE-2021-44228	Critical	18	187	85

Observations: A security issue can cause harm at multiple places in the implementation. The single Critical vulnerability required modification to 18 files with more than 200 code line changes. The community does not track the effort spent on such activities, but we believe it was significant.

C. Commit History

We collected the commit history by cloning the repo [24] to a Ubuntu Linux. The 12108 commits since the project's inception were screened by Year and Month to obtain the chart in Figure 2.

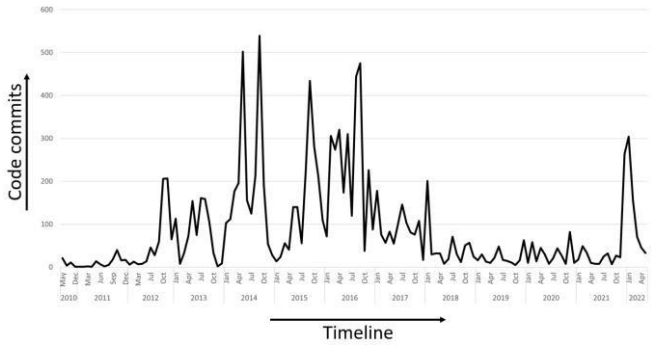

FIGURE 2: COMMIT HISTORY

Observations: There have been few commits since the project's inception from 2010 to 2013. We observe a spike in commits between 2014 and 2017. In 2014, Log4j was made available for general use, and hence there is a spike in code commits. Between 2014 and 2017, many features were added to the library, making it rich in logging. The commits are reduced and stable between 2018 to 2021. Around Nov 2021, we observed a spike in commits. The spike indicates the Log4Shell commits. Most of the commits were related to Log4Shell Critical vulnerability. An increased number of commits means increased test execution, code review, and gating checks needed to ensure the quality of the code.

D. Release History

We took data from Apache Log4j2 releases from inception to May 2022 as shown in Figure 3.

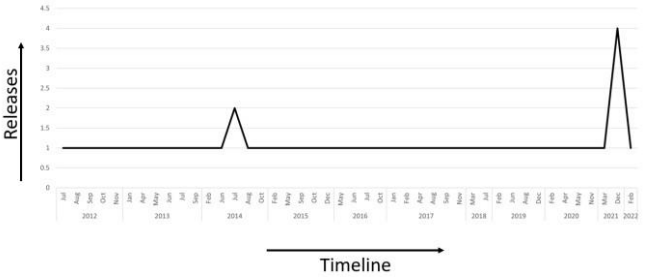

FIGURE 3: RELEASE HISTORY

It is evident that After 2014, the spike of four Releases happened from Dec 2021 to Feb 2022 to remediate the Log4Shell security issue. Table 4 maps the CVEs to releases.

TABLE 4. LOG4J2 RELEASE HISTORY

CVE Number	***Release Number***	***Release Date***
CVE-2021-44228	2.15.0	6-Dec-2021
CVE-2021-45046	2.16.0	13-Dec-2021
CVE-2021-45105	2.17.0	17-Dec-2021
CVE-2021-44832	2.17.1	27-Dec-2021

Observations: It took the Apache Foundation four releases to fix the vulnerabilities in the Log4js library. For the first time since its inception, Apache Foundation made four frequent releases in a single calendar month. The release cycle indicates how a security issue in the software can trigger unplanned releases. Releases require efforts from different teams such as the Development team, Test team, Integration team, Performance test team, Release team, and Documentation team. All the teams work hand to hand to create a successful release.

VII. AVOIDING LOG4SHELLS

Based on the root cause analysis and mitigations provided by Apache Foundation, the following are advisable to avoid security vulnerabilities like Log4Shell. We have provided the Pros and Cons of each approach.

1. *Feature Control*: The open-source libraries and software creators can provide optional features instead of making everything available by default. Optional features can be enabled and disabled based on the consumer requirement [26]. Feature flag control avoids security vulnerabilities from the optional features. In the case of events such as Log4Shell, the consumer can disable the functionality till the fix is applied.

   Pros: Helps control the optional feature. Feature flags can control the application or the library functionalities and reduce the application's attack surface.

Cons: Requires administrative efforts to manage the features across different applications. Granular feature control adds the complexity of enabling and disabling the features. More implementation to control features could also create a more complex application. The more complex application tends to have security vulnerabilities.

2. *Disable External Connections*: At the server level, the Administrator can set firewall rules to allow only the applications expected to communicate with an external network. Firewalls help avoid sending data outside the network boundary. Some of the issues have used this as a solution in the past [27].

   Pros: Firewall rules are easy to configure. It does not require a change in the application.

   Cons: This would not be possible in some cases, as the application might have use cases such as DNS resolution or using third-party services. These cases require external communication enabled at the server level.

3. *Whitelisting External Servers*: Applications communicate with configured servers. Any other attempts to connect to an unknown server would fail.

   Pros: Avoids communicating with the unknown peer. Easy to create a list and add a new server to the list.

   Cons: Administration of the whitelist is an overhead. A server added once will be trusted even if it changes to an attacker-controlled server at a later point in time.

4. *Zero Trust Approach*: In this approach, an external server is always verified cryptographically before starting the communication. Verifying the server is possible using TLS [28] and is used by the browsers to communicate with unknown servers.

   Pros: Server verification is a widely used and standard method and less prone to attacks.

   Cons: Run time encryption and decryption signature verification could lead to performance impacts on the application.

VIII. Observations

Based on the analysis of the vulnerability and the fixes provided, following are the observations.

1. Log4j is a popular logging application with rich features and provides a better logging experience to the users.
2. Log4Shell security vulnerability impacts most popular software applications that use specific versions of the vulnerable software library.
3. A novice attacker can exploit Log4Shell using simple tools. There is no license cost involved in using any of the tools.
4. Log4Shell attack had a significant impact on different industry verticals such as Healthcare, Telecom, and E-commerce.
5. Apache Foundation quickly provided the necessary workaround and documentation to overcome the attacks.
6. Log4Shell required unexpected code changes in the community contributed repository. The changes were published using new software releases.

IX. Recommendations

Open-Source Security is a broad topic, and Log4Shell is one of the most critical security issues of the decade. Research is required in open-source security to avoid introducing security vulnerabilities in the software application.

Acknowledgment

We would like to thank Apache Foundation for detailed documentation on the vulnerability. It has helped us gather data to analyze the impact and identify the root cause of the vulnerability.

References

[1] "Log4j – Apache Log4j Security Vulnerabilities," logging.apache.org. https://logging.apache.org/log4j/2.x/security.html (accessed May 30, 2022).

[2] "[LOG4J2-3201] Limit the protocols JNDI can use and restrict LDAP. - ASF JIRA," issues.apache.org. https://issues.apache.org/jira/browse/LOG4J2-3201 (accessed May 30, 2022).

[3] "[LOG4J2-3198] Message lookups should be disabled by default - ASF JIRA," issues.apache.org. https://issues.apache.org/jira/browse/LOG4J2-3198 (accessed May 30, 2022).

[4] "Apache Releases Log4j Version 2.15.0 to Address Critical RCE Vulnerability Under Exploitation | CISA," www.cisa.gov. https://www.cisa.gov/uscert/ncas/current-activity/2021/12/10/apache-releases-log4j-version-2150-address-critical-rce (accessed May 30, 2022).

[5] R. Hiesgen, M. Nawrocki, T. C. Schmidt, and M. Wählisch, "The Race to the Vulnerable: Measuring the Log4j Shell Incident," arXiv:2205.02544 [cs], May 2022, [Online]. Available: https://arxiv.org/abs/2205.02544

[6] D. Everson, L. Cheng, and Z. Zhang, "Log4shell: Redefining the Web Attack Surface," doi: 10.14722/madweb.2022.23010.

[7] A. Jones, "Security Posture: A Systematic Review of Cyber Threats and Proactive Security," Senior Honors Theses, Apr. 2022, Accessed: Apr. 27, 2022. [Online]. Available: https://digitalcommons.liberty.edu/honors/1147/

[8] A. Rudolph, "What is Log4j and Why Did the Government of Canada Turn Everything Off?," policycommons.net, Jan. 2022, Accessed: May 30, 2022. [Online]. Available: https://policycommons.net/artifacts/2327268/what-is-log4j-and-why-did-the-government-of-canada-turn-everything-off/3087909/

[9] koz, "log4j-shell-poc," GitHub, Jan. 28, 2022. https://github.com/kozmer/log4j-shell-poc (accessed May 30, 2022).

[10] "OpenLDAP, Main Page," www.openldap.org. https://www.openldap.org/ (accessed May 30, 2022).

[11] "Welcome to Apache Directory — Apache Directory," directory.apache.org. https://directory.apache.org/ (accessed May 30, 2022).

[12] "1.3 - Installing and starting the server — Apache Directory," directory.apache.org. https://directory.apache.org/apacheds/basic-ug/1.3-installing-and-starting.html (accessed May 30, 2022).

[13] "curl - Tool Documentation," curl.se. https://curl.se/docs/tooldocs.html (accessed May 30, 2022).

[14] "cyber struggle/L4sh," GitHub, May 26, 2022. https://github.com/cyberstruggle/L4sh (accessed May 30, 2022).

[15] "Remote code execution zero-day exploit in Java logging library (log4j2) | Synopsys," Application Security Blog, Dec. 10, 2021. https://www.synopsys.com/blogs/software-security/zero-day-exploit-log4j-analysis/ (accessed May 30, 2022).

[16] "Log4shell vulnerabilities (CVE-2021-44228, CVE-2021-45046, CVE-2021-4104, CVE-2021-45105)," GitHub, May 28, 2022. https://github.com/NCSC-NL/log4shell (accessed May 30, 2022).

[17] "Update for Apache Log4j2 Security Bulletin (CVE-2021-44228)," Amazon Web Services, Inc. https://aws.amazon.com/security/security-bulletins/AWS-2021-006/ (accessed May 30, 2022).

[18] "Apache Log4j 2 Vulnerability Security Advisory," Google Cloud. https://cloud.google.com/log4j2-security-advisory (accessed May 30, 2022).

[19] "McAfee Enterprise coverage for Apache Log4j CVE-2021-44228 Remote Code Execution," kc.mcafee.com. https://kc.mcafee.com/corporate/index?page=content&id=KB95091 (accessed May 30, 2022).

[20] "Red Hat Customer Portal - Access to 24x7 support and knowledge," access.redhat.com. https://access.redhat.com/security/cve/cve-2021-44228 (accessed May 30, 2022).

[21] "[LOG4J2-313] JNDI Lookup plugin support - ASF JIRA," issues.apache.org. https://issues.apache.org/jira/browse/LOG4J2-313 (accessed May 30, 2022).

[22] A. Morag, "Threat Alert: Tracking Real-World Apache Log4j Attacks," blog.aquasec.com. https://blog.aquasec.com/real-world-log4j-attacks-analysis (accessed May 30, 2022).

[23] "ED 22-02: Apache Log4j Recommended Mitigation Measures | CISA," www.cisa.gov. https://www.cisa.gov/uscert/ed-22-02-apache-log4j-recommended-mitigation-measures (accessed May 30, 2022).

[24] "Apache Log4j 2," GitHub, Mar. 22, 2022. https://github.com/apache/logging-log4j2 (accessed May 30, 2022).

[25] "Commits · apache/logging-log4j2," GitHub. https://github.com/apache/logging-log4j2/commits/release-2.x (accessed May 30, 2022).

[26] "FeatureHub Documentation :: FeatureHub Docs," docs.featurehub.io. https://docs.featurehub.io/featurehub/latest/index.html (accessed May 30, 2022).

[27] "Preventing SMB traffic from lateral connections and entering or leaving the network," support.microsoft.com. https://support.microsoft.com/en-us/topic/preventing-smb-traffic-from-lateral-connections-and-entering-or-leaving-the-network-c0541db7-2244-0dce-18fd-14a3ddeb282a (accessed May 30, 2022).

[28] "RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2," Ietf.org, 2022. https://datatracker.ietf.org/doc/html/rfc5246#section-7.4.2 (accessed May 30, 2022).