Project 4a

LOG4J

Luke Allevato | CSC245380: Secure Software Development | 3/21/2022

Contents

log4J Logging Levels

This program primarily dealt with the java.org.apache.log4J.Logger library. The log4J logger contains multiple different levels of log messages, each used for a specific purpose. The different types of logs included in this library are as follows:

TABLE 1

Level	Description
ALL	All levels including custom levels
DEBUG	Designates fine-grained informational events that are most useful to debug an application.
INFO	Designates informational messages that highlight the progress of the application at coarse-grained level.
WARN	Designates potentially harmful situations.
ERROR	Designates error events that might still allow the application to continue running.
FATAL	Designates very severe error events that will presumably lead the application to abort.
OFF	The highest possible rank and is intended to turn off logging.
TRACE	Designates finer-grained informational events than the DEBUG.

All information from https://www.tutorialspoint.com/log4j/log4j_logging_levels.htm.

These levels form a hierarchy of severity. Depending on the level set by the program configuration, it will only let through certain levels. For example, if the logging level is set to INFO, the program would let through FATAL, ERROR, WARN, and INFO levels, but not DEBUG or

TRACE. The log request level must be greater than or equal to the configured program log level. Different log levels can also be configured to write to different locations, allowing for easy parsing of logs.

ArrayLogger Program

BEFORE

The ArrayLogger program used in this report populates an array of size 100 with randomly generated numbers. It then asks the user for an integer and prints out the value at the index number the user input. If the user inputs an out of bounds number, an exception is thrown.

The program ran on Jog4J version 2.11.2. This version contained the major exploit CVE-2021-44228. CISA recommends upgrading all programs using log4J to version 2.17.1, 2.12.4, or 2.3.2. The program was run on 3/21/2022 13:44:24 with the following output:

```
Enter an index: 2

The element is 2612
```

AFTER

The program now logs the user's input and the results, whether that be a value or an out of bounds exception. The program in this report uses the INFO and ERROR levels and sets the program log level to ALL so all requests will be logged. The program was run again on 3/21/2022 with these lines as output:

```
Enter an index: 2

The element is 5670
```

There is no difference in performance after logging capabilities were added (Note: The difference in value is due to the random nature of the program itself). However, the `pom.xml` file was also updated to use log4J version 2.17.1. This version fixes the vulnerability from previous versions. The log entries from the program are shown below.

```
2022-03-21 13:54:07,715 INFO e.a.c.ArrayLogger [main] User input
= 2

2022-03-21 13:54:07,718 INFO e.a.c.ArrayLogger [main] The element
is 5670
```

Additionally, the program contains a line of output confirming the log4J version has been upgraded:

```
2022-03-21 14:52:33,262 main DEBUG Apache Log4j Core 2.17.1
initializing configuration XmlConfiguration[location=C:\Users…]
```

Application

While putting together this project, I learned about the different levels of logs used in log4J (See above). While adding logging capabilities, I learned how to format logs to include relevant information and make them readable for humans and AI alike. This was also the first program that I had made that uses Maven, a powerful application building software which builds some properties of a program automatically.

This program was interesting for me to run because as a software developer, I anticipate adding logging to many of the programs I create, and as a *Java* programmer, I anticipate doing logging with some form of log4J. While the fix is known for the severe vulnerabilities found in some versions of log4J, it will take years (or maybe decades) to find all devices that use log4J and update them. As I embark on my software developing career, I would not be surprised if I spend much of my time mitigating log4J vulnerabilities.

References

https://www.tutorialspoint.com/log4j/log4j_logging_levels.htm

https://www.cisa.gov/uscert/apache-log4j-vulnerability-guidance

https://www.papertrail.com/solution/tips/logging-in-java-best-practices-and-tips

https://google.github.io/styleguide/javaguide.html

Appendix A: ArrayLogger

```
import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import java.util.*;


public class ArrayLogger {


    private static final Logger logger =
LogManager.getLogger(ArrayLogger.class);


        public static void main(String[] args) {


            // Creates array

            int[] data = new int[100];


            // Initialize array

            for (int i = 0; i < 100; i++)

                data[i] = (int)(Math.random() * 10000);


            Scanner input = new Scanner(System.in);


            // Accept input and log it

            System.out.print("Enter an index: ");

            int index = input.nextInt();

            logger.info("User input = " + index);
```

```
            //Print messages to both console and log
            try {
                System.out.println("The element is " + data[index]);
                logger.info("The element is " + data[index]);
            }
            catch (Exception ex) {
                System.out.println("Out of Bounds");
                logger.error("Out of Bounds");
            }
        }
    }
```

Appendix B: pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>edu.arapahoe.csc245</groupId>
    <artifactId>log4j2</artifactId>
    <version>1.0</version>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java.version>1.8</java.version>
        <!--20220321 Upgraded to log4J 2.17.1 in accordance with CVE-2021-
44228-->
        <log4j.version>2.17.1</log4j.version>
        <disruptor.version>3.4.2</disruptor.version>
```

```
    <jackson.version>2.9.8</jackson.version>
</properties>

<dependencies>

    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-api</artifactId>
        <version>${log4j.version}</version>
    </dependency>

    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <version>${log4j.version}</version>
    </dependency>

    <!-- https://logging.apache.org/log4j/2.x/manual/async.html -->
    <dependency>
        <groupId>com.lmax</groupId>
        <artifactId>disruptor</artifactId>
        <version>${disruptor.version}</version>
    </dependency>

    <!-- https://logging.apache.org/log4j/2.x/runtime-dependencies.html -->
    <!-- log4j.yml need this -->
    <dependency>
        <groupId>com.fasterxml.jackson.dataformat</groupId>
        <artifactId>jackson-dataformat-yaml</artifactId>
        <version>${jackson.version}</version>
    </dependency>
```

```
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>${jackson.version}</version>
    </dependency>

</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.0</version>
            <configuration>
                <source>${java.version}</source>
                <target>${java.version}</target>
            </configuration>
        </plugin>

        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>3.2.0</version>
            <executions>
                <!-- Attach the shade into the package phase -->
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>shade</goal>
```

```
                        </goals>
                        <configuration>
                            <transformers>
                                <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransfo
rmer">
<mainClass>edu.arapahoe.csc245.HelloWorld</mainClass>
                                </transformer>
                            </transformers>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```