# Project 4

# Log4J2 Development and Exploitation

# Overview

Task 1: In this project, you will learn a bit about Log4J2 and write a program that uses it. By starting with a deliberately vulnerable project, you’ll also learn how to mitigate CVE-2021-44228.

Task 2: Reproduce the Log4J2 exploitation demonstration I gave in class and write up a report that describes the exploit. You do not have to go into greater detail than discussed in class.

# Assignment

## Part I: Load initial set of code from GitHub

To successfully complete this assignment, you will need to start with baseline code. We will use Java JDK 8 and IntelliJ Ultimate edition to provide this support.

1. Open IntelliJ
2. Click the Get from VCS button
3. Specify your directory, if needed
4. Specify URL:
   1. https://github.com/douglundin/CSC245-Project4-Log4J2
5. Click Clone
6. You may get an error that the project JDK is not defined. In this case, choose your favorite JDK. This project was developed with Oracle JDK 8.
7. When you load the project into IntelliJ, be sure to load the Maven scripts and trust the project.

A screenshot of a computer

Description automatically generated

1. The program should run successfully.
2. If you have questions about the various files in the project, you can consult:

https://mkyong.com/logging/apache-log4j-2-tutorials/

## Part II: Develop short program that implements logging

1. Modify the program to include functionality as specified in Exercise 12.3 from your CSC160 & CSC161 class textbook. Solution is given in next step.

Graphical user interface, text, application, Word, email

Description automatically generated

1. Here’s the solution:

import java.util.\*;

public class Exercise12\_03 {

public static void main(String[] args) {

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);

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

int index = input.nextInt();

try {

System.out.println("The element is " + data[index]);

}

catch (Exception ex) {

System.out.println("Out of Bounds");

}

}

}

1. Your task is to add logging to this program. Read about logging Best Practices here but do not violate any Secure Software Best Practices:
   1. https://www.papertrail.com/solution/tips/logging-in-java-best-practices-and-tips/
2. Maven is used to “build” this project. When you loaded the project into IntelliJ, recall you had to tell Maven to run/load the log4j2 build script. As the build proceeded, Maven downloaded all the software libraries (e.g. Log4J) and installed them.
3. Maven should be installed but may need to be activated in IntelliJ.
4. Open the pom.xml file in the root of the project folder and locate this code

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<java.version>1.8</java.version>

<log4j.version>2.11.2</log4j.version>

<disruptor.version>3.4.2</disruptor.version>

<jackson.version>2.9.8</jackson.version>

</properties>

1. As you can see, Maven installed version 2.11.2 of Log4J2. Is this version safe or not? Check out the authoritative answer here:

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

1. If you upgraded Log4J2 by changing the pom.xml file (which you should have done – and provided comments), you now have to tell Maven to clean up.
   1. Click View | Tool Windows | Maven
   2. In the Maven window, expand log4j2 and Lifecycle
   3. Click on clean
   4. Click the green start button in the Maven window
   5. Maven will indicate log4j2 is now clean
2. Run the project and note in the run window that Apache Log4J Core has been upgraded to \_\_\_\_\_\_\_\_
3. Congratulations, your project is no longer vulnerable to CVE-2021-44228

# Deliverables

Task 1: Create a short report that describes what you learned about Log4J2 functionality and how you envision using it in your career as a software developer. Include the program’s Java code in an appendix as well as the pom.xml code in an appendix.

Your code should be well-documented with comments, include header comments, use proper variable and naming conventions and properly formatted. (Use the Google Java style guidelines found here: https://google.github.io/styleguide/javaguide.html). Be sure your report is neat, well-organized and is well-written with minimal spelling and grammar errors. All references used should be included in your document.

Task 2: Create a report that describes the steps that can be taken to demonstrate a Log4J exploit.

# Grading rubric

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Meets** | **Does not meet** |
| Demonstrates baseline project runs correctly | **30 points**  Imports baseline Java project into IntelliJ Ultimate edition. (10 pts)  Includes the date and time stamp when the application was run. (10 pts)  Demonstrates the code runs properly within IntelliJ environment. (10 pts) | **0 points**  Does not demonstrate successful import.  Does not include the date and time stamp when the application was run.  Does not demonstrate your code runs properly within the IntelliJ environment. |
| Demonstrates upgraded project runs correctly | **30 points**  Add Exercise 12.3 code to project with clear comments. (10 pts)  Includes the date and time stamp when the application was run. (10 pts)  Demonstrates the code runs properly within IntelliJ environment. (10 pts) | **0 points**  Does not demonstrate successful addition of code.  Does not include the date and time stamp when the application was run.  Does not demonstrate your code runs properly within the IntelliJ environment. |
| Fixes security issues | **60 points**  Successfully upgraded Log4J to a safe version. (20 pts)  Provided evidence that Log4J has been upgraded. (40 pts) | **0 points**  Does not upgrade Log4J to a safe version.  Does not provide evidence Log4J was upgraded. |
| Documentation and Submission | **40 points**  Code is well-documented with comments, including header comments, use of proper variable and naming conventions and properly formatted. (10 pts)  Document is neat, well-organized and is well-written with minimal spelling and grammar errors. (20 pts)  All references used should be included in your document. (10 pts) | **0 points**  Code is not well-documented with comments, including header comments, use of proper variable and naming conventions and properly formatted.  Document is not neat, well-organized or well-written with minimal spelling and grammar errors.  References were not included. |
| Demonstrates Log4J2 Exploit | **40 points**  Document is neat, well-organized and is well-written with minimal spelling and grammar errors. (30 pts)  All references used should be included in your document. (10 pts) | **0 points**  Document is not neat, well-organized or well-written with minimal spelling and grammar errors.  References were not included. |