CS 483/5583 Software Security

Assignment #5

Instructions:

- If you are enrolled in CS 483, you may collaborate with another CS 483 student and submit a single copy of your solutions with both names included. This option is not available for CS 5583 students.
- Turn in your answers and **source code** as a single zip file.
- This assignment is based on the SimpleWebServer program discussed in class. Should you opt for a different programming language, you must accurately translate the Java code.
- If you decide to utilize large language models (e.g., ChatGPT and Gemini), you must supply the prompts alongside the responses.
- Grading policy

Problem 1	**Problem 2**	**Problem 3**	**Problem 4**	**Problem 5**	**Total**
10 points	30 points	20 points	20 points	20 points	100 points

1. What if a client connects to SimpleWebServer, but never sends any data or disconnects? What type of attack would such a client be able to conduct?

2. Update SimpleWebServer.java and SimpleWebClient.java to enable the client to upload a file. When the user inputs **PUT <fileToUpload> <DestinationPath>** in the client program, the client should read the content of **<fileToUpload>** on the client side and send the command **PUT <DestinationPath>** along with the file content to the server. The server program will then save the file content to **<DestinationPath>** on the server and log all client requests into a log file. Below are sample methods for text file storage and logging for your convenience. It's your responsibility to address any potential bugs. You may adjust the storeFile method to handle binary files if necessary. Please provide screenshots of your program execution.

```
public void storeFile(BufferedReader br, OutputStreamWriter osw, String pathname) throws Exception {
        FileWriter fw = null;
        try {
```

```
                        fw = new FileWriter(pathname);
                        Scanner sc = new Scanner(br);
                        while (sc.hasNext()) {
                                String line = sc.nextLine();
                                fw.write(line+"\n");
                                System.out.println(line);
                        }
                        fw.close();
                        sc.close();
                        System.out.println(pathname+" is saved!");
                } catch(Exception e) {
                }
        }

        public void logEntry(String filename, String record){
                FileWriter fw = new FileWriter(filename, true);
                fw.write((new Date()).toString()+" "+record);
                fw.close();
        }
```

3. Rewrite the serveFile method to incorporate a maximum file size limit. If a user tries to download a file larger than the maximum allowed size, log an entry to a file named error_log.txt and return a "403 Forbidden" HTTP response code. Please provide screenshots of your program execution.
4. Describe and implement: (a) an attack that defaces (overwrites) the index.html homepage and (b) an attack that removes the log data. Please provide screenshots of your attacks.
5. Suppose you are an attacker who has obtained the compiled SimpleWebServer.class. Describe an attack where, after SimpleWebServer is restarted (e.g., due to an exception caused by another attack), the functionality described in (3) is disabled.