1. Develop set of security requirements

Specify what the code could do in every possible error condition to identify possible vulnerabilities.

Sanitize input from the user such carriage returns as currently it would crash the server, which would be a huge financial burden this is an import web server.

The web server should immediately disconnect from any web client that sends a malformed HTTP request to the server.

Sharing Requirements with Quality Assurance - QA should develop test case and pentest the code to vulnerabilities that you can then fix

Once you find the error, ensure they are handled securely to protect against end-users from exploiting the system to potentially access confidential data. Do not just send back internal errors for any error that you could run across. Instead, work with the error to understand the attack vector and patch it accordingly.

Include validation and fraud checks such as MOD 10 checksum. This checksum can be used to check for simple error cases if a customer for example mistypes their credit card number on a site. To help against fraud, always require a CVC code, as criminals usually do not have those 3 digits. Failure to put these simple checks could cause massive fraud to take place on your site.

Implement access control- only admins should be able to access the web server files. Otherwise, customers on the side should get the plain vanilla site. Use the idea of "least privilege", only give someone access if they absolutely need to. Otherwise they may intentionally or unintentionally use their power maliciously.

Think security or bust- if you fail to adhere to security, your service will most likely fail as your hard work will be stolen by others. If security has not yet been implemented, delay the service to ensure security guidelines are in place first.

6a.
The file could be a virus. It could also be a ZipBomb to DOS the server.

6b.
Make sure you enforce a max file size and run the file against VirusTotal, also run the file in a sandboxed environment and do not store it on corporate file networks.

6c.

```
103     }
104     public void storeFile(BufferedReader br, OutputStreamWriter osw, String pathname) throw
105         if (pathname.charAt(0) == '/')
106             pathname = '.' + pathname;
107 //          pathname = pathname.substring(1);
108         System.out.println(pathname);
109         FileWriter fw = null;
110         try {
111             fw = new FileWriter(pathname);
112             String s = br.readLine();
113             while (!s.equals("null")) {
114                 s = br.readLine();
115                 fw.write(s + "\n");
116             }
117             fw.close();
118             osw.write("HTTP/1.0 201 Created");
119         } catch (Exception e) {
120             System.out.println("exception");
121             osw.write("HTTP/1.0 500 Internal Server Error");
122         }
123     }
124     public void serveFile(OutputStreamWriter osw, String pathname) throws Exception {
167
168
169
170     public void logEntry(String pathname, String record) throws IOException {
171         if (pathname.charAt(0) == '/')
172             pathname = '.' + pathname;
173         FileWriter fw = new FileWriter(pathname, true);
174         fw.write(getTimestamp() + " " + record);
175         fw.close();
176     }
177
178     public String getTimestamp() {
179         return (new Date()).toString();
180     }
181
```

Here is the code added to the file and then ran in the following section.

6d. Attack 1

```
pc@gg:~$ curl http://localhost:8080/index.html
My Home Page
pc@gg:~$ curl -T 'index.html' 'http://localhost:8080'
pc@gg:~$ curl http://localhost:8080/index.html
User-Agent: curl/7.81.0
Accept: */*
Content-Length: 39
Expect: 100-continue

i hacked the home page of ur site
```

6e. Attack 2

```
pc@gg:~$ curl --path-as-is http://localhost:8080/../MyMusic.txt
this is my music file
```

Accessed a file outside the webserver directory

6 (1):
The threat actors could be various actors including script kiddies, cybercriminals, nation states, or hacktivists. Basically anyone that wants access to your web server, whether it's a political, financial, or personal goal. Hacktivists would most likely be involved with defacing the index.html, while a cybercriminal would likely be the one going after your personal files.

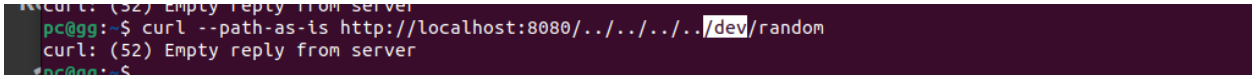
6 (2):
For attack 1, the attacker had to use the same file name as the index.html when using the PUT request to upload a file. Because the code did not check against this, the attacker could modify the home page of the site.

For attack 2, the attacker was able to access personal files on the system. This was due to not escaping ../ in the request pathname when retrieving files.

7a.

```
    }
    public void serveFile(OutputStreamWriter osw, String pathname) throws Exception {
        if (pathname.charAt(0) == '/')
            pathname=pathname.substring(1);
        File file = new File(pathname);
        System.out.println(file.length());
        if((file.length() / (1024*1024*1024)) > 1) {
            System.out.println("File too large");
            osw.write("HTTP/1.0 404 File too lard\n\n");
            return;
        }
```

/dev/random bypasses this restriction as it is a 0 byte file on my system.



It gets an empty reply as the java code is run in eclipse in user mode, not the su to access system files.

7b.

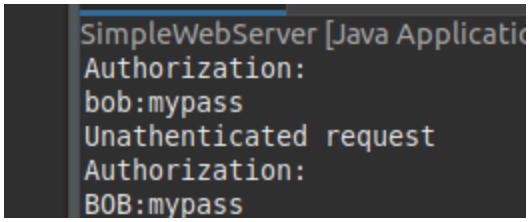
One could implement the file limit by also pre approving the list of files that the user can access, so that the user is not able to access large system files in the first place.

9a.

```
76          /* parse the HTTP request */
77          StringTokenizer st = new StringTokenizer(header, " ");
78          StringTokenizer st2 = new StringTokenizer(authorization, " ");
79
80          command = st.nextToken();
81          pathname = st.nextToken();
82
83          String auth = st2.nextToken();
84          String basic = st2.nextToken();
85          String base64 = st2.nextToken();
86
87          System.out.println(auth);
88          if (!auth.contains("Authorization:")) {
89              osw.write ("HTTP/1.0 401 Unauthorized");
90              osw.write ("WWW-Authenticate: Basic realm=BasicAuthWebServer");
91              System.out.println("Unauthorized request");
92              return;
93          }
94          byte[] decodedBytes = Base64.getDecoder().decode(base64);
95          String decodedString = new String(decodedBytes);
96          System.out.println(decodedString);
97
98          StringTokenizer st3 = new StringTokenizer(decodedString, ":");
99          String username = st3.nextToken();
100         String pass = st3.nextToken();
101
102         if (!(username.equals(USERNAME) && pass.equals(PASSWORD))) {
103             osw.write ("HTTP/1.0 401 Unathenticated");
104             System.out.println("Unathenticated request");
105             return;
106         }
107
```

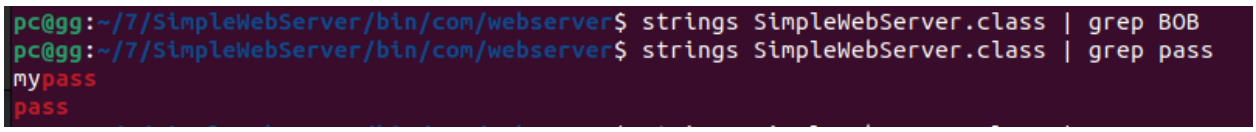
```
curl: (52) Empty reply from server
pc@gg:~$ curl --path-as-is --user bob:mypass http://localhost:8080/index.html
^C
pc@gg:~$ curl --path-as-is --user BOB:mypass http://localhost:8080/index.html
User-Agent: curl/7.81.0
Accept: */*
Content-Length: 39
Expect: 100-continue

i hacked the home page of ur site
null
Tue Jan 31 12:00:01 CST 2023 TESTpc@gg:~$
```



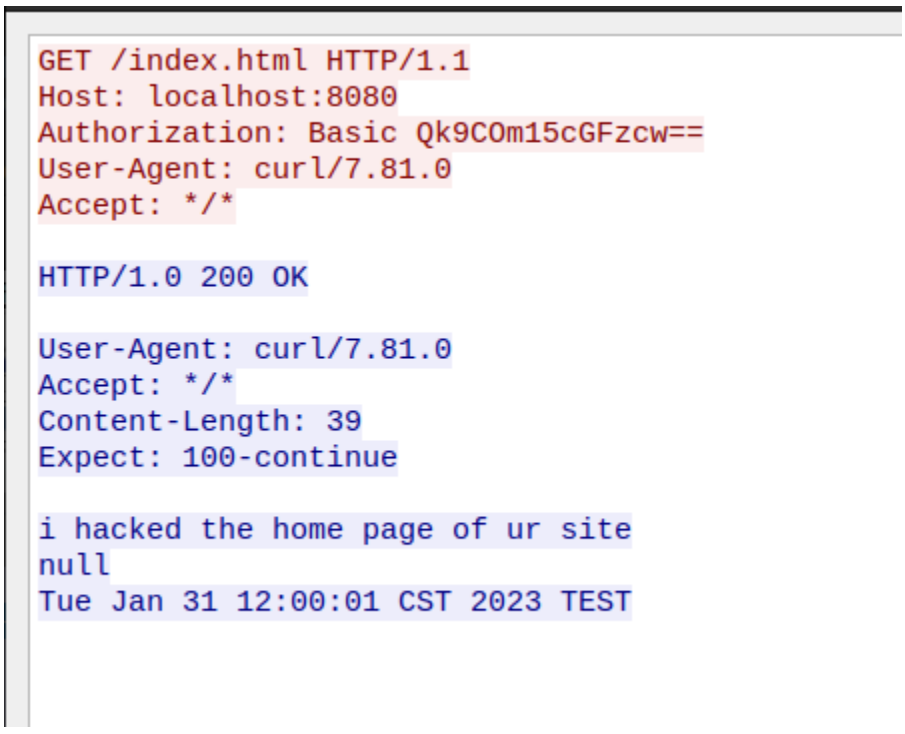
First request failed due to incorrect password, and second went through and we were able to read index.html off the web server.

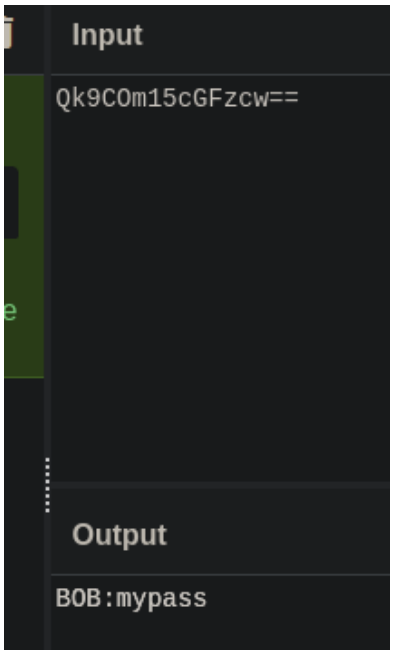
9b.



The strings utility was able to find the password (mypass) but not the username (BOB)

9c.




Input
Qk9COm15cGFzcw==
Output
BOB:mypass