CS 542 – Introduction to Software Security
Exercise on Directory traversal

Binhao Chen (bchen276@wisc.edu), Steven Yang (yang558@wisc.edu)

Due: October 27 at 2:30pm

1 Screenshots or printouts showing the inputs used for the attack, and the outputs you got from the system

```
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ ls
Main.class  Main.java  Makefile  safe_programs  unsafe_program
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ java Main ../unsafe_program
UNSAFE PROGRAM OUTPUT

Program Exit Code: 0
```

2 Your commented code for the mitigation. Highlight the code you added/modified.

```
1
2  import java.io.IOException;
3  import java.lang.ProcessBuilder.Redirect;
4  import java.nio.file.Path;
5  import java.nio.file.Paths;
6
7  public class Main {
8
9      /**
10      * Execute the safe program named by the first argument to this program.
11      *
12      * @param args
13      *             must be at least one non-empty string, namely the executable
            name
14      *             for execSafeProgram
15      */
16     public static void main(String[] args) {
17         if (args.length < 1) {
18             // must have an argument
19             System.err.println("Must have at least one argument.");
20             System.err.println("Usage: java Main \"executable_name\"");
21             System.exit(-1);
22         }
23         if (args[0].length() < 1) {
24             // first argument must be non-empty executable name
25             System.err.println("Executable name must not be empty.");
26             System.err.println("Usage: java Main \"executable_name\"");
27             System.exit(-1);
28         }
29
30         // execute the program and print the exit code
31         int retVal = execSafeProgram(args[0]);
32         System.out.println();
33         System.out.println("Program Exit Code: " + retVal);
```

```
34      }
35
36      /**
37       * Execute a program found within the directory "safe_programs" in the
             current
38       * working directory. Use as helper function to execute any of a pre-
             specified
39       * "safe" programs found in the safe directory.
40       *
41       * @param programName
42       *                     name of executable found in safe_programs
43       * @return exit value of program or -1 if unable to start, wait for, and
             join
44       *         the child process
45       * @throws SecurityException
46       *                             if the program name tries to escape the safe
47       *                             directory
48       */
49      private static int execSafeProgram(String programName) {
50          // find the program to execute
51          Path safeDir = Paths.get("safe_programs");
52          Path exePath = safeDir.resolve(programName);
53
54
55          // We first get the canonical paths (by using .toRealPath()) for both
                the safe directory and the final exePath by using Paths API. We
                compare the canonical path of the parent of this final exePath with
                 the canonical path of safeDir to determine that the specified file
                 is within the correct directory.
56
57          // If we get an input that does not satisfy this condition, print an
                error message and return a negative value (-1).
58          try {
59
60              Path real_safeDir = safeDir.toRealPath();
61              Path real_exePath = exePath.toRealPath();
62
63              Path real_exePath_parent = real_exePath.getParent();
64
65              if (!real_exePath_parent.equals(real_safeDir)) {
66                  System.out.println("Error: You can only access the designated
                        directory.");
67                  System.out.println("The permitted directory: " + real_safeDir.
                        toString());
68                  System.out.println("The directory you are attempting to access:
                         " + real_exePath_parent.toString());
69                  return -1;
70              }
71              ;
72
73          } catch (Exception e) {
74              System.out.println("Error extracting the Real Path. You can only
                    access the designated directory.");
75              return -1;
76          }
77
78          // configure program runtime to execute ./safe_programs/programName
                executable
79          ProcessBuilder procBuild = new ProcessBuilder(exePath.toString());
80
81          // capture output and print to current shell
82          procBuild.redirectErrorStream(true);
83          procBuild.redirectOutput(Redirect.INHERIT);
```

```
84
85          try {
86              // execute the program
87              Process p = procBuild.start();
88              // wait for program to return and exit
89              return p.waitFor();
90          } catch (IOException ex) {
91              // error starting process
92              System.out.println("Error running program: " + ex.getMessage());
93              ex.printStackTrace();
94              return -1;
95          } catch (InterruptedException ex) {
96              // error waiting for process
97              System.out.println("Error running program: " + ex.getMessage());
98              ex.printStackTrace();
99              return -1;
100         }
101     }
102 }
```

3 Screenshots or printouts showing the inputs and outputs after fixing the vulnerability, both both good and malicious inputs

```
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ nano Main.java
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ make
javac Main.java
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ java Main ../unsafe_program
Error: You can only access the designated directory.
The permitted directory: /home/user/Desktop/EXERCISES/3.6.1_directory_traversal/safe_programs
The directory you are attempting to access: /home/user/Desktop/EXERCISES/3.6.1_directory_traversal

Program Exit Code: -1
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ java Main pwd
/home/user/Desktop/EXERCISES/3.6.1_directory_traversal

Program Exit Code: 0
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ java Main whoami
user

Program Exit Code: 0
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$
```

4 Run a variety of tests to validate your mitigation. Can you find a test case that breaks your solution? If so describe the problem in your solution. If not, explain why your solution is robust

```
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ nano Main.java
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ make
javac Main.java
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ java Main ../unsafe_program
Error: You can only access the designated directory.
The permitted directory: /home/user/Desktop/EXERCISES/3.6.1_directory_traversal/safe_programs
The directory you are attempting to access: /home/user/Desktop/EXERCISES/3.6.1_directory_traversal

Program Exit Code: -1
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ java Main pwd
/home/user/Desktop/EXERCISES/3.6.1_directory_traversal

Program Exit Code: 0
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ java Main whoami
user

Program Exit Code: 0
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ nano Main.java
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ java Main ../....//unsafe_program
Error extracting the Real Path. You can only access the designated directory.

Program Exit Code: -1
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ java Main /home/user/Desktop/EXERCISES/3.6.1_directory_traversal
/safe_programs/....//unsafe_program
Error extracting the Real Path. You can only access the designated directory.

Program Exit Code: -1
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ java Main ....//unsafe_program
Error extracting the Real Path. You can only access the designated directory.

Program Exit Code: -1
user@software-security22:~/Desktop/EXERCISES/3.6.1_directory_traversal$ java Main ../unsafe_program/ /
Error: You can only access the designated directory.
The permitted directory: /home/user/Desktop/EXERCISES/3.6.1_directory_traversal/safe_programs
The directory you are attempting to access: /home/user/Desktop/EXERCISES/3.6.1_directory_traversal

Program Exit Code: -1
```

No, our solution is robust for this task. First, the resolve() method considers the path of safeDir to be a directory and resolves the given path (input programName) against this path. The resolved path (the exePath) is always starting with "safe_programs/". The toRealPath() method will remove any starting dots and slashes nested in the input programName. Thus we can always ensure only when the parent directory of the exePath is exactly the permitted directory (safeDir), the specified file (programName) can be accessed.

5 An explanation on your attack and your mitigations

Attack: We attack by passing "../unsafe_program" so that when the path is resolved, the ".." will jump to the parent directory of safe_programs (the assignment directory: /Desktop/EXERCISES/3.6.1_directory_traversal), then execute the unsafe_program.

Mitigation: We first canonicalize both paths with toRealPath() and then check if the parent of the executable is equal to the safeDir, which is "safe_program". If not, then print out error message and return -1;