

# Assignment 4 - CT255 - Cybersecurity

Daniel Hannon (19484286)

December 2020

## 1 Terminal Output

```
Lintor Terminal
[daniel@Void3 Assignment 3]$ javac *.java
[daniel@Void3 Assignment 3]$ java Stegano1 A wby1.txt poop.txt 1101101
[daniel@Void3 Assignment 3]$ java Stegano1 E poop.txt
1101101
[daniel@Void3 Assignment 3]$ java Stegano2 A wby1.txt poop.txt 0110100001101111011101110111100101100001
[daniel@Void3 Assignment 3]$ java Stegano2 E poop.txt
0110100001101111011101110111100101100001
[daniel@Void3 Assignment 3]$
```

## 2 Java Code

### 2.1 Stegano1

```
1  /**
2   * Skeleton code for Steganography assignment.
3   *
4   * @author Michael Schukat
5   * @version 1.0
6   */
7
8  import java.io.BufferedReader;
9  import java.io.BufferedWriter;
10 import java.io.FileReader;
11 import java.io.FileWriter;
12 import java.io.IOException;
13
14 public class Stegano1
15 {
16     /**
17      * Constructor for objects of class Stegano1
18      */
19     public Stegano1()
20     {
21     }
22
23     public static void main(String[] args) {
24         String arg1, arg2, arg3, arg4;
25         Boolean err = false;
26
27         if (args != null && args.length > 1) { // Check for minimum number of arguments
28             arg1 = args[0];
29             arg2 = args[1];
30             if (arg2 == "") {
31                 err = true;
32             }
33             else if ((arg1.charAt(0) == 65) && (args.length > 3)) {
```

```

34     // Get other arguments
35     arg3 = args[2];
36     arg4 = args[3];
37     if (arg3 == "" || arg4 == "") {
38         err = true;
39     }
40     else {
41         // Hide bitstring
42         hide(arg2, arg3, arg4);
43     }
44 }
45 else if (arg1.charAt(0) == 69){
46     // Extract bitstring from text
47     retrieve(arg2);
48 }
49 else {
50     err = true;
51 }
52 }
53 else {
54     err = true;
55 }
56
57 if (err == true) {
58     System.out.println();
59     System.out.println("Use: Stegano1 <A:E><Input File><OutputFile><Bitstring>");
60     System.out.println("Example: Stegano1 A inp.txt out.txt 0010101");
61     System.out.println("Example: Stegano1 E inp.txt");
62 }
63 }
64 }
65
66 static void hide(String inpFile, String outFile, String binString) {
67     //
68     BufferedReader reader;
69     BufferedWriter writer;
70     int strLength = binString.length();
71     int stringPos = 0;
72     try {
73         reader = new BufferedReader(new FileReader(inpFile));
74         writer = new BufferedWriter(new FileWriter(outFile));
75         String line = reader.readLine();
76         while (line != null) {
77             // Your code starts here
78             if(stringPos < strLength) {
79                 // *Make Sure it does not try to access an illegal place in memory */
80                 if(binString.charAt(stringPos) == 48) { /*0*/
81                     line += " ";
82                 }
83                 if(binString.charAt(stringPos) == 49) { /*1*/
84                     line += " ";
85                 }
86             }
87             stringPos++;
88             // Store amended line in output file
89             writer.write(line);
90             writer.newLine();
91             // read next line
92             line = reader.readLine();
93         }
94     }

```

```

93     reader.close();
94     writer.close();
95 } catch (IOException e) {
96     e.printStackTrace();
97 }
98
99 }
100
101 static void retrieve(String inpFile) {
102     BufferedReader reader;
103     String output = "";
104     try {
105         reader = new BufferedReader(new FileReader(inpFile));
106         String line = reader.readLine();
107         while (line != null) {
108             if(line.length() != 0) { /*Edge Case */
109                 if(line.charAt((line.length() - 1)) == 32) {/*Check for whitespace*/
110                     if(line.length() >= 2 && line.charAt((line.length() - 2)) == 32) {
111                         //Another Edge Case and check for more whitespace
112                         output += "1";
113                     } else {
114                         output += "0";
115                     }
116                 }
117             } // read next line
118             line = reader.readLine();
119         }
120         System.out.println(output);
121         reader.close();
122     } catch (IOException e) {
123         e.printStackTrace();
124     }
125 }
126 }

```

## 2.2 Stegano2

```

1  /**
2   * Skeleton code for Steganography assignment.
3   *
4   * @author Michael Schukat
5   * @version 1.0
6   */
7
8  import java.io.BufferedReader;
9  import java.io.BufferedWriter;
10 import java.io.FileReader;
11 import java.io.FileWriter;
12 import java.io.IOException;
13
14 public class Stegano2
15 {
16     /**
17      * Constructor for objects of class Stegano2
18      */
19     public Stegano2()

```

```

20 {
21 }
22
23 public static void main(String[] args) {
24     String arg1, arg2, arg3, arg4;
25     Boolean err = false;
26
27     if (args != null && args.length > 1) { // Check for minimum number of arguments
28         arg1 = args[0];
29         arg2 = args[1];
30         if (arg2 == "") {
31             err = true;
32         }
33         else if ((arg1.charAt(0) == 65) && (args.length > 3)) {
34             // Get other arguments
35             arg3 = args[2];
36             arg4 = args[3];
37             if (arg3 == "" || arg4 == "") {
38                 err = true;
39             }
40             else {
41                 // Hide bitstring
42                 hide(arg2, arg3, arg4);
43             }
44         }
45         else if (arg1.charAt(0) == 69){
46             // Extract bitstring from text
47             retrieve(arg2);
48         }
49         else {
50             err = true;
51         }
52     }
53     else {
54         err = true;
55     }
56
57     if (err == true) {
58         System.out.println();
59         System.out.println("Use: Stegano1 <A:E><Input File><OutputFile><Bitstring>");
60         System.out.println("Example: Stegano1 A inp.txt out.txt 0010101");
61         System.out.println("Example: Stegano1 E inp.txt");
62     }
63 }
64
65
66 static void hide(String inFile, String outFile, String binString) {
67     //Perform this check immediately
68     if(binString.length() % 2 != 0) {
69         /*Check if the binary string has an odd length, appends an extra digit */
70         binString+="0";
71     }
72     BufferedReader reader;
73     BufferedWriter writer;
74     int strLength = binString.length();
75     int stringPos = 0;
76     int i = 0;
77     try {
78         reader = new BufferedReader(new FileReader(inFile));
79         writer = new BufferedWriter(new FileWriter(outFile));

```

```

79     String line = reader.readLine();
80     while (line != null) {
81         for(i = 0; i < 2; i++) {
82             if(stringPos < strLength) {
83                 if(binString.charAt(stringPos) == 48) { /*0*/
84                     line += " ";
85                 }
86                 if(binString.charAt(stringPos) == 49) { /*1*/
87                     line += "\t";
88                 }
89             }
90             stringPos++;
91         }
92         // Store amended line in output file
93         writer.write(line);
94         writer.newLine();
95         // read next line
96         line = reader.readLine();
97     }
98     reader.close();
99     writer.close();
100 } catch (IOException e) {
101     e.printStackTrace();
102 }
103
104 }
105
106 static void retrieve(String inFile) {
107     BufferedReader reader;
108     String output = "";
109     int i = 2;
110     try {
111         reader = new BufferedReader(new FileReader(inFile));
112         String line = reader.readLine();
113         while (line != null) {
114             if(line.length() >= 2) { /*Make sure there's a point in reading the line*/
115                 for(i = 2; i > 0; i--) {
116                     if(line.charAt(line.length() - i) == 32) { /*Space*/
117                         output += "0";
118                     }
119                     if(line.charAt(line.length() - i) == 9) { /*Tab*/
120                         output += "1";
121                     }
122                 }
123             }
124             // read next line
125             line = reader.readLine();
126         }
127         System.out.println(output);
128         reader.close();
129     } catch (IOException e) {
130         e.printStackTrace();
131     }
132 }
133 }

```