Project 3

CODE REVIEW

Luke Allevato | CSC245380: Secure Software Development | 3/9/2022

Table of Contents

Introduction

On 3/7/2022 head programmer Doug Lundin made a formal request to Student Software for a formal peer review of his code for PROJECT 3. This code is part of Arapahoe Community College Software's new Summit Weather service and is to be submitted to shareholders on 3/14/2022. The Student Software department had a deadline of 11:59PM MST on 3/9/2022 to submit the code review. Some minor changes were made, as outlined in this document.

BACKGROUND

Arapahoe Community College Software's Summit Weather service is a robust application for determining the local weather for each Arapahoe Community College (ACC) campus so that instructors, faculty, and students can respond appropriately to weather changes. The project began production on 10/1/2021 and is expected to be released in beta on 4/1/2022. The code for PROJECT 3 must be approved at a shareholders' meeting on 3/14/2022 in order to timely enter into further testing and deployment.

SUMMARY OF FINDINGS

The Student Software team 1 violation of the SEI CERT Oracle Coding Standards for Java and multiple violations of the guidelines found in *Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs*. These mostly impacted readability of the code, rather than being security vulnerabilities that could lead to exploits. The Student Software team made no changes to the output of the code, and there should be no noticeable changes in functionality.

Analysis

PRE-REVIEW PERFORMANCE

The flawed code works as intended, calling an Open Weather Map API that returns the weather conditions for the surrounding area.

```
{"coord":{"lon":-
104.8561,"lat":39.3722},"weather":[{"id":800,"main":"Clear","description":"clea
r
sky","icon":"01d"}],"base":"stations","main":{"temp":41.32,"feels_like":38.84,"
temp_min":36.21,"temp_max":47.25,"pressure":1013,"humidity":36},"visibility":10
000,"wind":{"speed":4,"deg":208,"gust":8.99},"clouds":{"all":0},"dt":1647705485
,"sys":{"type":2,"id":2001899,"country":"US","sunrise":1647695062,"sunset":1647
738596},"timezone":-21600,"id":5416329,"name":"Castle Rock","cod":200}

Current temperature in Castle Rock, US is: 41.32 degrees.


Process finished with exit code 1
```

PRE-REVIEW CODE

1. The pre-review code contained multiple potential readability issues. The first of these was a unbracketed `while` loop in line 68, which violates the Java coding guideline “Use braces for the body of an if, for, or while statement. This makes it unclear what lines are executing within the loop.

   ```
   while ((line = rd.readLine()) != null)

          result .append(line);

   System.out.println(result);
   ```

2. The variable name `owm` is an arbitrary string that does noy convey information about what is held by a variable. This violates the Java coding guideline “Use meaningful symbolic constants to represent literal values in program logic”.

```
String owm = "foo",        // Include the API key here
       LOCATION = "Castle Rock, US";
```

3. On line 90 there is a redundant `for` loop followed by a semicolon. This line does not run any code except for the loop itself and breaks the Java coding guideline "Do not place a semicolon immediately following an if, for, or while condition".

```
for (int i=0;i<10;i++);
       System.out.println("Current temperature in " + LOCATION +" is: "
              + getTempForCity(LOCATION,owm) + " degrees.");
```

4. In line 94 there was a line setting the `urlString` to "". The `urlString` variable is only used in a feature that is not currently functional, but this line is redundant and goes against the Java coding guideline "Do not attempt to help the garbage collector by setting local reference variables to null".

```
urlString = "";
```

5. The last item on this list is the only one that can really be classified as a "vulnerability". In the case that an `IOException` is caught in line 74, it is possible to encounter an indeterminate state, as there is no exit line. This is noncompliant with CERT Oracle Coding Standard FIO14-J "FIO14-J. Perform proper cleanup at program termination".

```
catch (IOException e){
       System.out.println(e.getMessage());
       return "Temp not available (API problem?)";
}
```

POST-REVIEW CODE RECOMMENDATIONS

1. Add braces to `while` statements to improve potential readability issues.

```
while ((line = rd.readLine()) != null) {
            result.append(line);
        }
        System.out.println(result);
```

2. Update ambiguous variable names to meaningful constants.

```
String APIKey = "foo", // Include the API key here
            LOCATION = "Castle Rock, US";
```

3. Remove redundant `for` loops.

```
System.out.println("Current temperature in " + LOCATION + " is: "
```

```
        + getTempForCity(LOCATION, APIKey) + " degrees.");
```

4. Remove garbage collection-helping statements.

5. Add `Runtime.getRuntime().exit(1)` statement at end of program.

```
Runtime.getRuntime().exit(1);
```

POST-REVIEW PERFORMANCE

The performance is unchanged after mitigation.

```
{"coord":{"lon":-
104.8561,"lat":39.3722},"weather":[{"id":800,"main":"Clear","description":"clea
r
sky","icon":"01d"}],"base":"stations","main":{"temp":41.32,"feels_like":38.84,"
temp_min":36.21,"temp_max":47.25,"pressure":1013,"humidity":36},"visibility":10
000,"wind":{"speed":4,"deg":208,"gust":8.99},"clouds":{"all":0},"dt":1647705485
,"sys":{"type":2,"id":2001899,"country":"US","sunrise":1647695062,"sunset":1647
738596},"timezone":-21600,"id":5416329,"name":"Castle Rock","cod":200}

Current temperature in Castle Rock, US is: 41.32 degrees.


Process finished with exit code 1
```

Recommendations

Most of the post-review code recommendations are simply to improve readability and are therefore low on the priority list. The Student Software team presents the following table to aid in prioritizing which vulnerabilities to mitigate first.

Table 1

Vulnerability	Priority	Difficulty of Mitigation
1	Low	Easy
2	Low	Medium
3	Low	Easy
4	Medium	Medium (dependent on application functionality)
5	High	Easy

Conclusion

The code for Summit Weather's PROJECT 3 works as intended, however, the author would do well to keep the Java Coding Guidelines and the CERT Oracle Coding Standard for Java in mind. While none of the flaws in this program lead to significantly exploitable vulnerabilities, it is crucial to keep best coding practices in mind for the future.

Appendix A: Pre-Review Code

```
package edu.arapahoe.csc245;


import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.URL;

import java.net.URLConnection;

import java.util.HashMap;

import java.util.Map;


import com.google.gson.*;

import com.google.gson.reflect.*;


////////////////////////////////////////////////////////////////////////////
/////

//

// This program was created for Arapahoe Community College's CSC-245 course and

// identifies the current temperature for a location using the Open Weather Map
API.

//

// The use of the API (openweathermap.org) was applied for and access granted
202010321

// The key comes with several technical constraints regarding its usage,
including:

//     Hourly forecast: unavailable

//     Daily forecast: unavailable
```

```
//     Calls per minute: 60
//     3 hour forecast: 5 days
//
// Details on the use of the API can be found here:
//     https://openweathermap.org/current
//
// The default location is Castle Rock, CO (encoded as Castle Rock, US) but can
be
// changed, as required. The GPS coordinates for Castle Rock, CO is
// latitude 39.3722        longitude -104.8561
//
// CSC 245 Secure Software Development
//
// Change log:
//      20210321 API access granted
//      20210322 Initially created (ddl)
//
// Dependencies:
//      gson-2.2.2.jar is needed for correct functioning
//           Oracle JDK 1.8
//
////////////////////////////////////////////////////////////////////////////
/////



public class CSC245_Project3_Insecure {


       // Java Maps are used with many API interactions. OpenWeatherMap also
uses Java Maps.
       public static Map<String, Object> jsonToMap(String str) {
              Map<String, Object> map = new Gson().fromJson(
                           str, new TypeToken<HashMap<String, Object>>()
{}.getType()
```

```
                        );
                return map;
        }

        public static String getTempForCity (String cityString, String api_key) {
                String urlString =
"http://api.openweathermap.org/data/2.5/weather?q=" +
                        cityString + "&appid=" + api_key + "&units=imperial";
                try {
                        StringBuilder result = new StringBuilder();
                        URL url = new URL(urlString);
                        URLConnection conn = url.openConnection();
                        BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
                        String line;
                        while ((line = rd.readLine()) != null)
                                result .append(line);
                        System.out.println(result);

                        Map<String, Object > respMap = jsonToMap
(result.toString());
                        Map<String, Object > mainMap = jsonToMap
(respMap.get("main").toString());

                        return mainMap.get("temp").toString();

                } catch (IOException e){
                        System.out.println(e.getMessage());
                        return "Temp not available (API problem?)";
                }
        }

        public static void main(String[] args) {
```

```
            String owm = "foo",        // Include the API key here
                        LOCATION = "Castle Rock, US";
            String urlString =
"http://api.openweathermap.org/data/2.5/weather?q=" + LOCATION +
                        "&appid=" + owm + "&units=imperial";


            // The following line is out of scope for mitigation and can be
ignored.
//          System.out.println("URL invoked to get temperature data=" +
urlString);


            for (int i=0;i<10;i++);
                  System.out.println("Current temperature in " + LOCATION +"
is: "
                              + getTempForCity(LOCATION,owm) + " degrees.");


            urlString = "";


      }


}
```

Appendix B: Post-Review Code

```
package edu.arapahoe.csc245;


import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;
import java.util.HashMap;
import java.util.Map;
```

```
import com.google.gson.*;

import com.google.gson.reflect.*;


////////////////////////////////////////////////////////////////////////////
/////

//

// This program was created for Arapahoe Community College's CSC-245 course and

// identifies the current temperature for a location using the Open Weather Map
API.

//

// The use of the API (openweathermap.org) was applied for and access granted
202010321

// The key comes with several technical constraints regarding its usage,
including:

//     Hourly forecast: unavailable

//     Daily forecast: unavailable

//     Calls per minute: 60

//     3 hour forecast: 5 days

//

// Details on the use of the API can be found here:

//     https://openweathermap.org/current

//

// The default location is Castle Rock, CO (encoded as Castle Rock, US) but can
be

// changed, as required. The GPS coordinates for Castle Rock, CO is

// latitude 39.3722        longitude -104.8561

//

// CSC 245 Secure Software Development

//

// Change log:

//      20210321 API access granted

//      20210322 Initially created (ddl)

//      20220309 Brackets added to while statement in lines 66-67 for clarity

//      20220309 Removed redundant for loop in line 90
```

```
//      20220309 Removed line 97 urlString = ""; to prevent helping garbage
collector

//      20220309 Renamed variable owm to APIKey for clarity

//      20220309 Added Runtime.getRuntime.exit(1) to end of code to prevent
exiting in an indeterminate state

//

// Dependencies:

//      gson-2.2.2.jar is needed for correct functioning

//           Oracle JDK 1.8

//

////////////////////////////////////////////////////////////////////////////
/////




public class CSC245_Project3_Secure {



    // Java Maps are used with many API interactions. OpenWeatherMap also uses
Java Maps.

    public static Map<String, Object> jsonToMap(String str) {

        Map<String, Object> map = new Gson().fromJson(

                str, new TypeToken<HashMap<String, Object>>() {}.getType()

        );

        return map;

    }



    public static String getTempForCity (String cityString, String api_key) {

        String urlString = "http://api.openweathermap.org/data/2.5/weather?q="
+

                cityString + "&appid=" + api_key + "&units=imperial";

        try {

            StringBuilder result = new StringBuilder();

            URL url = new URL(urlString);

            URLConnection conn = url.openConnection();
```

```
            BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));

            String line;

            while ((line = rd.readLine()) != null) {

                result.append(line);

            }

            System.out.println(result);


            Map<String, Object > respMap = jsonToMap (result.toString());

            Map<String, Object > mainMap = jsonToMap
(respMap.get("main").toString());


            return mainMap.get("temp").toString();


        } catch (IOException e){

            System.out.println(e.getMessage());

            return "Temp not available (API problem?)";

        }


    }


    public static void main(String[] args) {

        String APIKey = "foo",          // Include the API key here

                LOCATION = "Castle Rock, US";

        //String urlString =
"http://api.openweathermap.org/data/2.5/weather?q=" + LOCATION +

        //        "&appid=" + APIKey + "&units=imperial";


        // The following line is out of scope for mitigation and can be
ignored.

//           System.out.println("URL invoked to get temperature data=" +
urlString);


        System.out.println("Current temperature in " + LOCATION + " is: "
```

```
                + getTempForCity(LOCATION, APIKey) + " degrees.");


        Runtime.getRuntime().exit(1);
    }

}
```