Rohith Rajagopalan

1001518031

Assignment-10

Part-1

Manual Analysis:

- We observe that we declare a new file reader fr in line 103 but we fail to close it. This leads to many vulnerabilities
- Another vulnerability is that in line 102 we have a try block followed by an except block but its missing a finally block which can also lead to problems

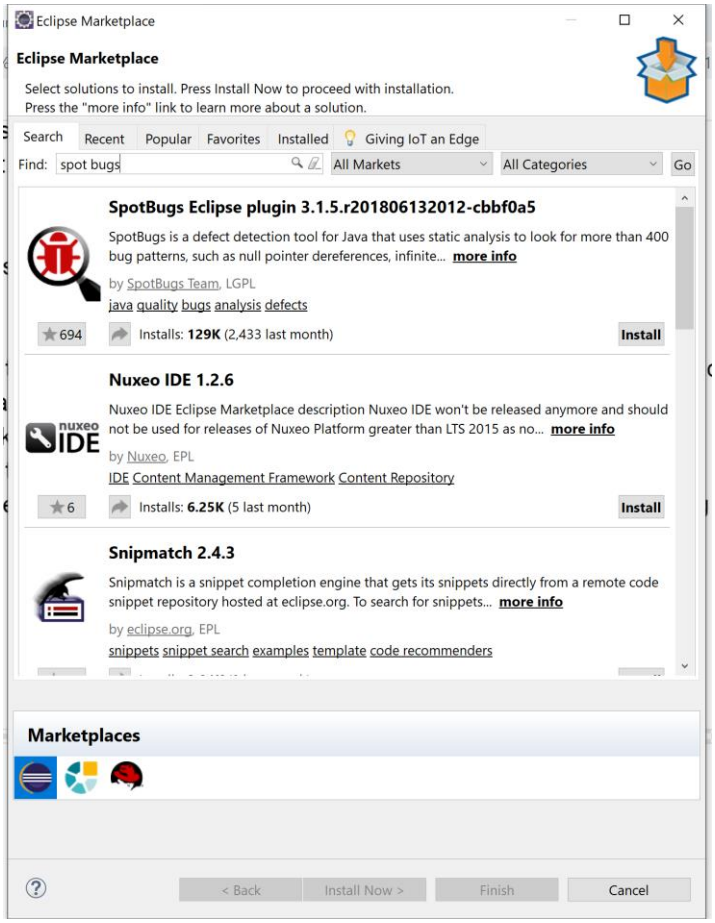
eclipse-workspace - C:\Users\Rohith\Downloads\SimpleWebServer.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

SimpleWebServer.java

```
91      /* remove the initial slash at the beginning
92         of the pathname in the request */
93      if (pathname.charAt(0)=='/')
94          pathname=pathname.substring(1);
95
96      /* if there was no filename specified by the
97         client, serve the "index.html" file */
98      if (pathname.equals(""))
99          pathname="index.html";
100
101     /* try to open file specified by pathname */
102     try {
103         fr = new FileReader (pathname);
104         c = fr.read();
105     }
106     catch (Exception e) {
107         /* if the file is not found,return the
108            appropriate HTTP response code  */
109         osw.write ("HTTP/1.0 404 Not Found\n\n");
110         return;
111     }
112
113     /* if the requested file can be successfully opened
114        and read, then return an OK response code and
115        send the contents of the file */
116     osw.write ("HTTP/1.0 200 OK\n\n");
117     while (c != -1) {
118         sb.append((char)c);
119         c = fr.read();
120     }
121     osw.write (sb.toString());
122     }
123
124     /* This method is called when the program is run from
125        the command line. */
126     public static void main (String argv[]) throws Exception {
127
128     /* Create a SimpleWebServer object, and run it */
129     SimpleWebServer sws = new SimpleWebServer();
130     sws.run();
131     }
132 }
133
```

For the next part of the experiment we use Spot Bugs(version-3.1.5) and SonarLint(Version-5.5.1)

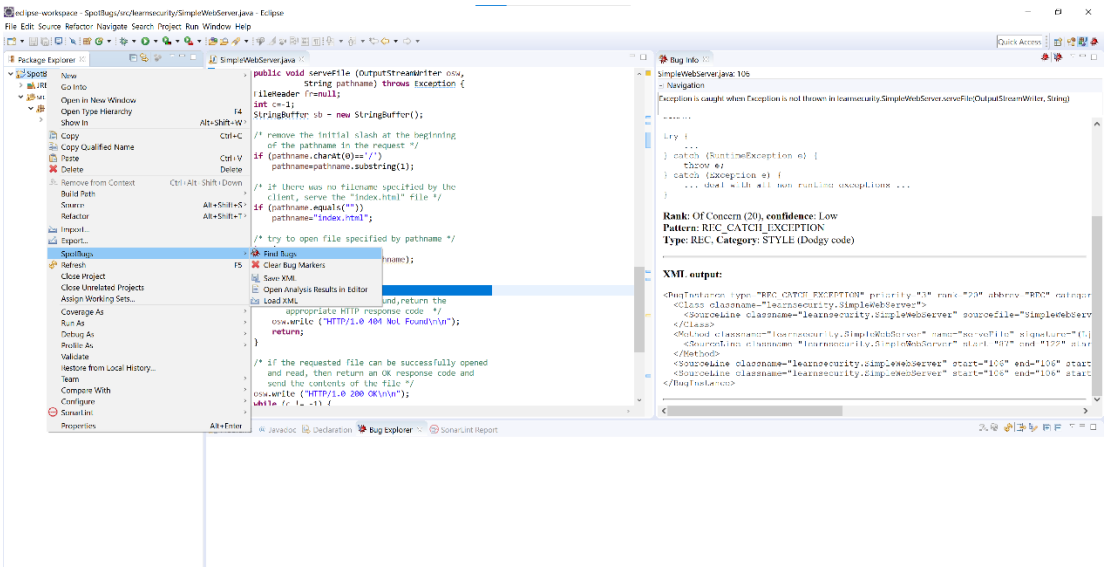


SpotBugs(3.1.5):

Procedure:

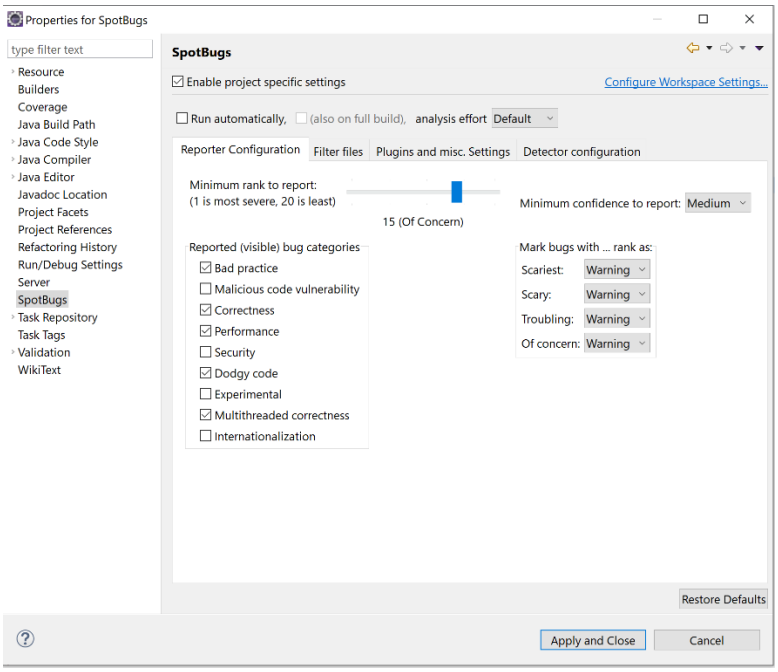
- First we download the plug in from eclipse marketplace.
- We then navigate to the properties tab of our project.
- Here we select the SpotBugs option and alter various configurations such as minimum rank to report , minimum confidence to report and reported bug categories.
- We then select(right click) the project and under the SpotBugs option select “find bugs”.
- We test with various configurations and record the respective outputs.

- The bugs found can be seen in the bugs explorer tab and its respective information is available in the bugs information tab.
- The following image shows the running of the plugin

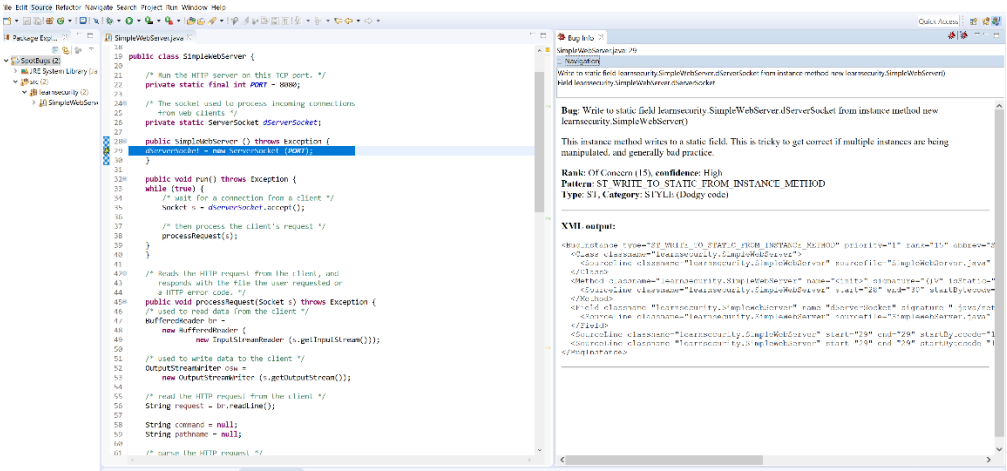


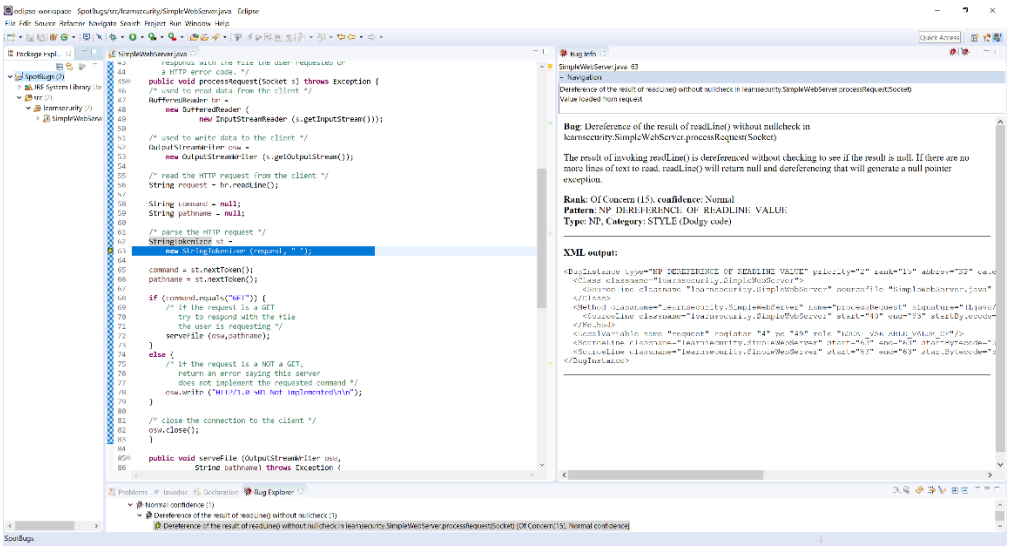
Test Congiguration-1:

We ran the SpotBugs plugin with the following configuration



Results:





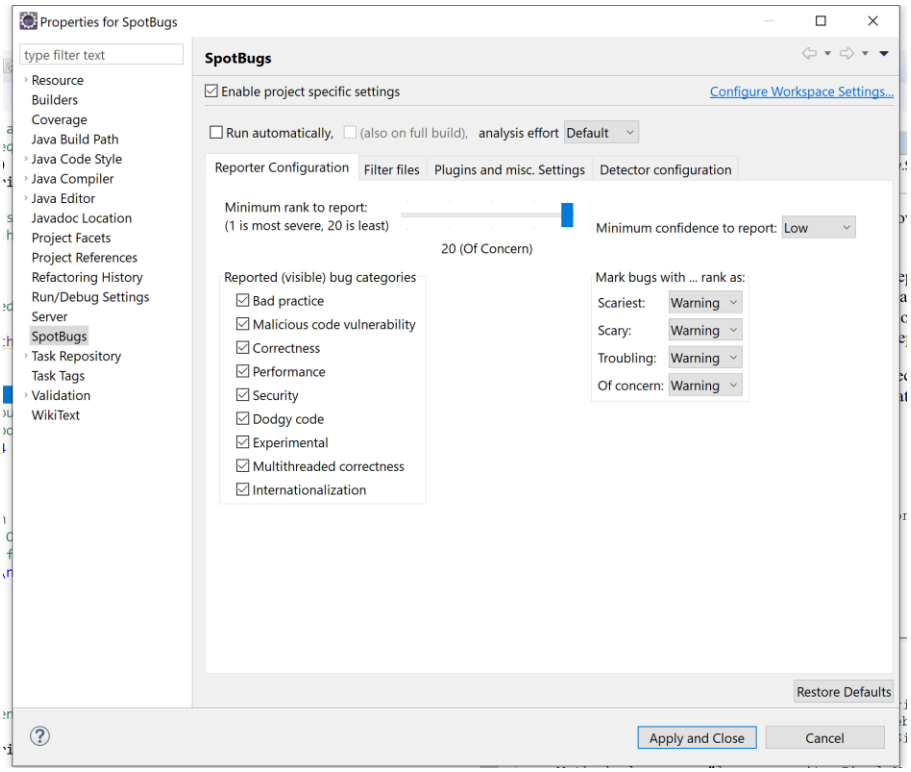
Number Of Bugs-2

High Confidence Bugs-1

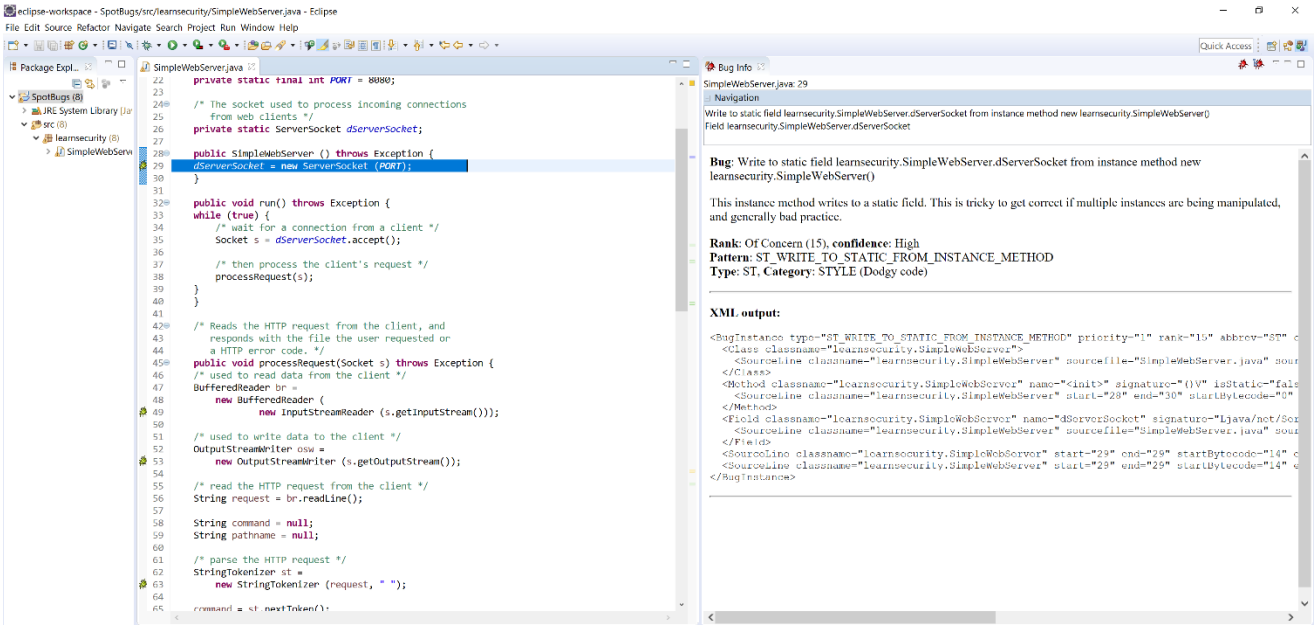
Normal Confidence Bugs-1

Test Configuration -2

We ran the SpotBugs plugin with the following configuration


Properties for SpotBugs
type filter text
Resource
Builders
Coverage
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project Facets
Project References
Refactoring History
Run/Debug Settings
Server
SpotBugs
Task Repository
Task Tags
Validation
WikiText
SpotBugs
Enable project specific settings
Configure Workspace Settings...
Run automatically,
(also on full build),
analysis effort
Default
Reporter Configuration
Filter files
Plugins and misc. Settings
Detector configuration
Minimum rank to report:
(1 is most severe, 20 is least)
20 (Of Concern)
Minimum confidence to report:
Low
Reported (visible) bug categories
Bad practice
Malicious code vulnerability
Correctness
Performance
Security
Dodgy code
Experimental
Multithreaded correctness
Internationalization
Mark bugs with ... rank as:
Scariest:
Warning
Scary:
Warning
Troubling:
Warning
Of concern:
Warning
Restore Defaults
Apply and Close
Cancel

Results:


eclipse-workspace - SpotBugs/src/learnsecurity/SimpleWebServer.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Expl...
SpotBugs (8)
JRE System Library
src (8)
learnsecurity (8)
SimpleWebServ
SimpleWebServer.java
private static final int PORT = 8080;
/* The socket used to process incoming connections
from web clients */
private static ServerSocket dServerSocket;
public SimpleWebServer () throws Exception {
dServerSocket = new ServerSocket (PORT);
}
public void run() throws Exception {
while (true) {
/* wait for a connection from a client */
Socket s = dServerSocket.accept();
/* then process the client's request */
processRequest(s);
}
}
/* Reads the HTTP request from the client, and
responds with the file the user requested or
a HTTP error code. */
public void processRequest(Socket s) throws Exception {
/* used to read data from the client */
BufferedReader br =
new BufferedReader (
new InputStreamReader (s.getInputStream()));
/* used to write data to the client */
OutputStreamWriter osw =
new OutputStreamWriter (s.getOutputStream());
/* read the HTTP request from the client */
String request = br.readLine();
String command = null;
String pathname = null;
/* parse the HTTP request */
StringTokenizer st =
new StringTokenizer (request, " ");
Bug Info
SimpleWebServer.java: 29
Navigation
Write to static field learnsecurity.SimpleWebServer.dServerSocket from instance method new learnsecurity.SimpleWebServer()
Field learnsecurity.SimpleWebServer.dServerSocket
Bug: Write to static field learnsecurity.SimpleWebServer.dServerSocket from instance method new learnsecurity.SimpleWebServer()
This instance method writes to a static field. This is tricky to get correct if multiple instances are being manipulated, and generally bad practice.
Rank: Of Concern (15), confidence: High
Pattern: ST_WRITE_TO_STATIC_FROM_INSTANCE_METHOD
Type: ST, Category: STYLE (Dodgy code)
XML output:
<BugInstance type="ST_WRITE_TO_STATIC_FROM_INSTANCE_METHOD" priority="1" rank="15" abbrev="ST"
<Class classname="learnsecurity.SimpleWebServer">
<SourceLine classname="learnsecurity.SimpleWebServer" sourcefile="SimpleWebServer.java"
</Class>
<Method classname="learnsecurity.SimpleWebServer" name="<init>" signature="()V" isStatic="false
<SourceLine classname="learnsecurity.SimpleWebServer" start="28" end="30" startBytecode="0"
</Method>
<Field classname="learnsecurity.SimpleWebServer" name="dServerSocket" signature="Ljava/net/Ser
<SourceLine classname="learnsecurity.SimpleWebServer" sourcefile="SimpleWebServer.java"
</Field>
<SourceLine classname="learnsecurity.SimpleWebServer" start="29" end="29" startBytecode="14"
<SourceLine classname="learnsecurity.SimpleWebServer" start="29" end="29" startBytecode="14"
</BugInstance>

eclipse-workspace - SpotBugs/src/learnsecurity/SimpleWebServer.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Package Expl...

- SpotBugs (8)
  - JRE System Library [Jav
  - src (8)
    - learnsecurity (8)
      - SimpleWebServe

SimpleWebServer.java

```
42    /* Reads the HTTP request from the client, and
43       responds with the file the user requested or
44       a HTTP error code. */
45    public void processRequest(Socket s) throws Exception {
46    /* used to read data from the client */
47    BufferedReader br =
48        new BufferedReader (
49                new InputStreamReader (s.getInputStream()));
50
51    /* used to write data to the client */
52    OutputStreamWriter osw =
53        new OutputStreamWriter (s.getOutputStream());
54
55    /* read the HTTP request from the client */
56    String request = br.readLine();
57
58    String command = null;
59    String pathname = null;
60
61    /* parse the HTTP request */
62    StringTokenizer st =
63        new StringTokenizer (request, " ");
64
65    command = st.nextToken();
66    pathname = st.nextToken();
67
68    if (command.equals("GET")) {
69        /* if the request is a GET
70           try to respond with the file
71           the user is requesting */
72        serveFile (osw,pathname);
73    }
74    else {
75        /* if the request is a NOT a GET,
76           return an error saying this server
77           does not implement the requested command */
78        osw.write ("HTTP/1.0 501 Not Implemented\n\n");
79    }
80
81    /* close the connection to the client */
82    osw.close();
83    }
84
```

Bug Info

SimpleWebServer.java: 49

Navigation

Found reliance on default encoding in learnsecurity.SimpleWebServer.processRequest(Socket): new java.io.InputStreamReader(InputStream)
Called method new java.io.InputStreamReader(InputStream)

Bug: Found reliance on default encoding in learnsecurity.SimpleWebServer.processRequest(Socket): new java.io.InputStreamReader(InputStream)

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

Rank: Of Concern (19), **confidence**: High
Pattern: DM_DEFAULT_ENCODING
Type: Dm, **Category**: I18N (Internationalization)

XML output:

```
<BugInstance type="DM_DEFAULT_ENCODING" priority="1" rank="19" abbrev="Dm" category="I18N" first
  <Class classname="learnsecurity.SimpleWebServer">
    <SourceLine classname="learnsecurity.SimpleWebServer" sourcefile="SimpleWebServer.java" sour
  </Class>
  <Method classname="learnsecurity.SimpleWebServer" name="processRequest" signature="(Ljava/net/
    <SourceLine classname="learnsecurity.SimpleWebServer" start="48" end="83" startBytecode="0"
  </Method>
  <Method classname="java.io.InputStreamReader" name="<init>" signature="(Ljava/io/InputStream;)
    <SourceLine classname="java.io.InputStreamReader"/>
  </Method>
  <SourceLine classname="learnsecurity.SimpleWebServer" start="49" end="49" startBytecode="12" e
  <SourceLine classname="learnsecurity.SimpleWebServer" start="49" end="49" startBytecode="12" e
</BugInstance>
```

eclipse-workspace - SpotBugs/src/learnsecurity/SimpleWebServer.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Package Expl...

- SpotBugs (8)
  - JRE System Library [Jav
  - src (8)
    - learnsecurity (8)
      - SimpleWebServe

SimpleWebServer.java

```
42    /* Reads the HTTP request from the client, and
43       responds with the file the user requested or
44       a HTTP error code. */
45    public void processRequest(Socket s) throws Exception {
46    /* used to read data from the client */
47    BufferedReader br =
48        new BufferedReader (
49                new InputStreamReader (s.getInputStream()));
50
51    /* used to write data to the client */
52    OutputStreamWriter osw =
53        new OutputStreamWriter (s.getOutputStream());
54
55    /* read the HTTP request from the client */
56    String request = br.readLine();
57
58    String command = null;
59    String pathname = null;
60
61    /* parse the HTTP request */
62    StringTokenizer st =
63        new StringTokenizer (request, " ");
64
65    command = st.nextToken();
66    pathname = st.nextToken();
67
68    if (command.equals("GET")) {
69        /* if the request is a GET
70           try to respond with the file
71           the user is requesting */
72        serveFile (osw,pathname);
73    }
74    else {
75        /* if the request is a NOT a GET,
76           return an error saying this server
77           does not implement the requested command */
78        osw.write ("HTTP/1.0 501 Not Implemented\n\n");
79    }
80
81    /* close the connection to the client */
82    osw.close();
83    }
84
```

Bug Info

SimpleWebServer.java: 53

Navigation

Found reliance on default encoding in learnsecurity.SimpleWebServer.processRequest(Socket): new java.io.OutputStreamWriter(OutputStream)
Called method new java.io.OutputStreamWriter(OutputStream)

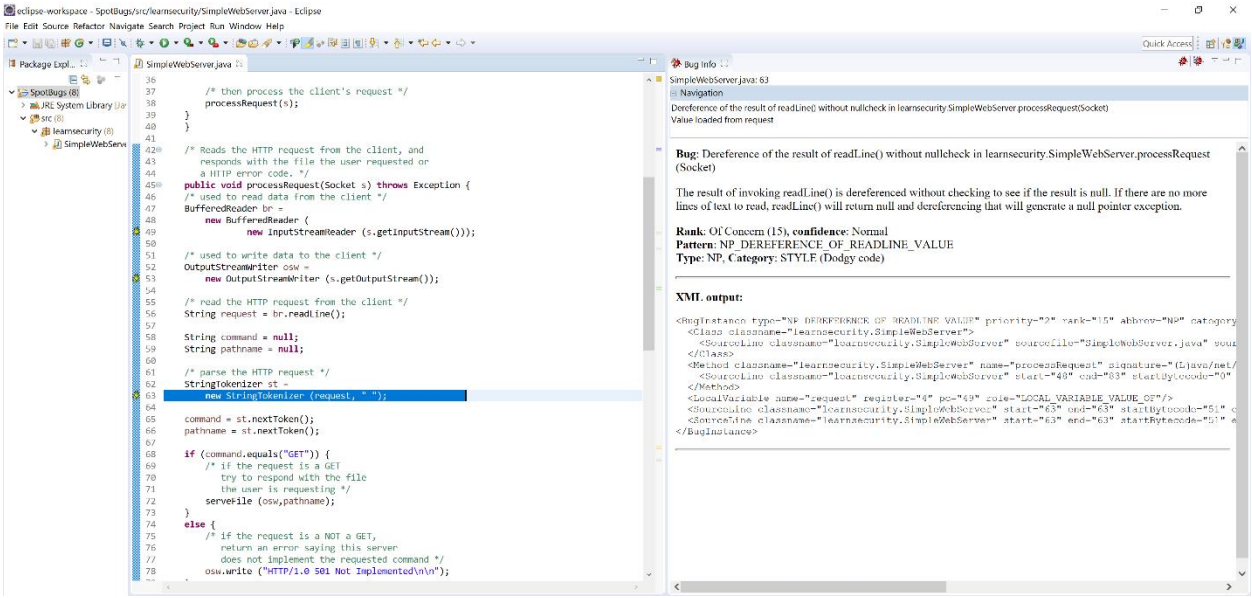
Bug: Found reliance on default encoding in learnsecurity.SimpleWebServer.processRequest(Socket): new java.io.OutputStreamWriter(OutputStream)

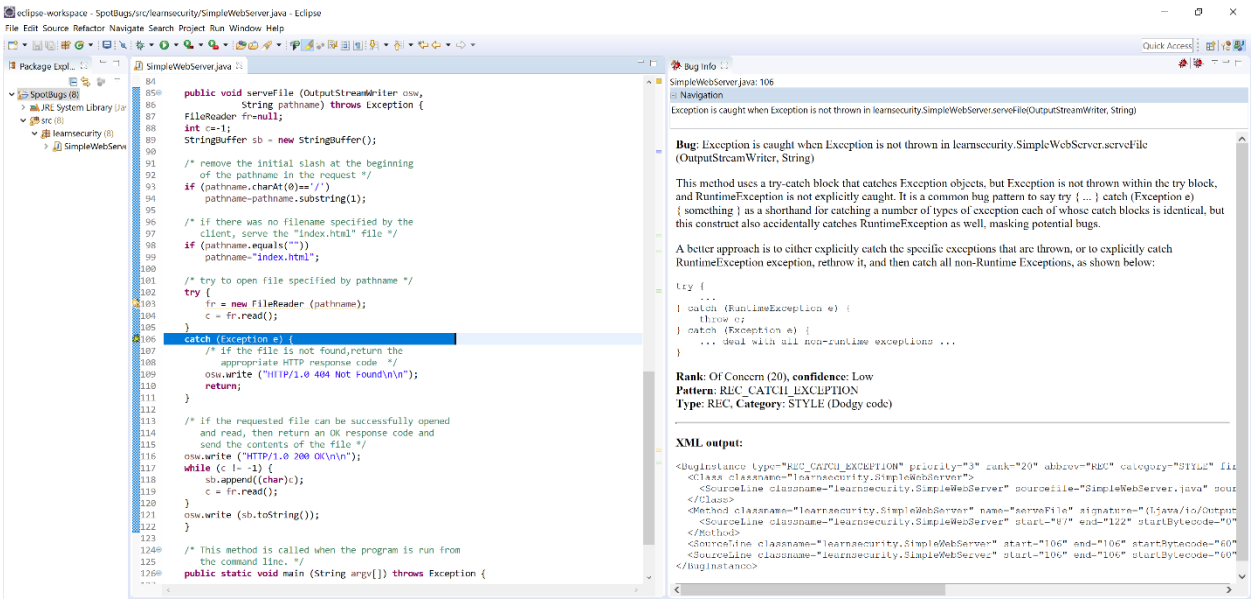
Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

Rank: Of Concern (19), **confidence**: High
Pattern: DM_DEFAULT_ENCODING
Type: Dm, **Category**: I18N (Internationalization)

XML output:

```
<BugInstance type="DM_DEFAULT_ENCODING" priority="1" rank="19" abbrev="Dm" category="I18N" first
  <Class classname="learnsecurity.SimpleWebServer">
    <SourceLine classname="learnsecurity.SimpleWebServer" sourcefile="SimpleWebServer.java" sour
  </Class>
  <Method classname="learnsecurity.SimpleWebServer" name="processRequest" signature="(Ljava/net/
    <SourceLine classname="learnsecurity.SimpleWebServer" start="48" end="83" startBytecode="0"
  </Method>
  <Method classname="java.io.OutputStreamWriter" name="<init>" signature="(Ljava/io/OutputStrea
    <SourceLine classname="java.io.OutputStreamWriter"/>
  </Method>
  <SourceLine classname="learnsecurity.SimpleWebServer" start="53" end="53" startBytecode="27" e
  <SourceLine classname="learnsecurity.SimpleWebServer" start="53" end="53" startBytecode="27" e
</BugInstance>
```





Number Of Bugs-5

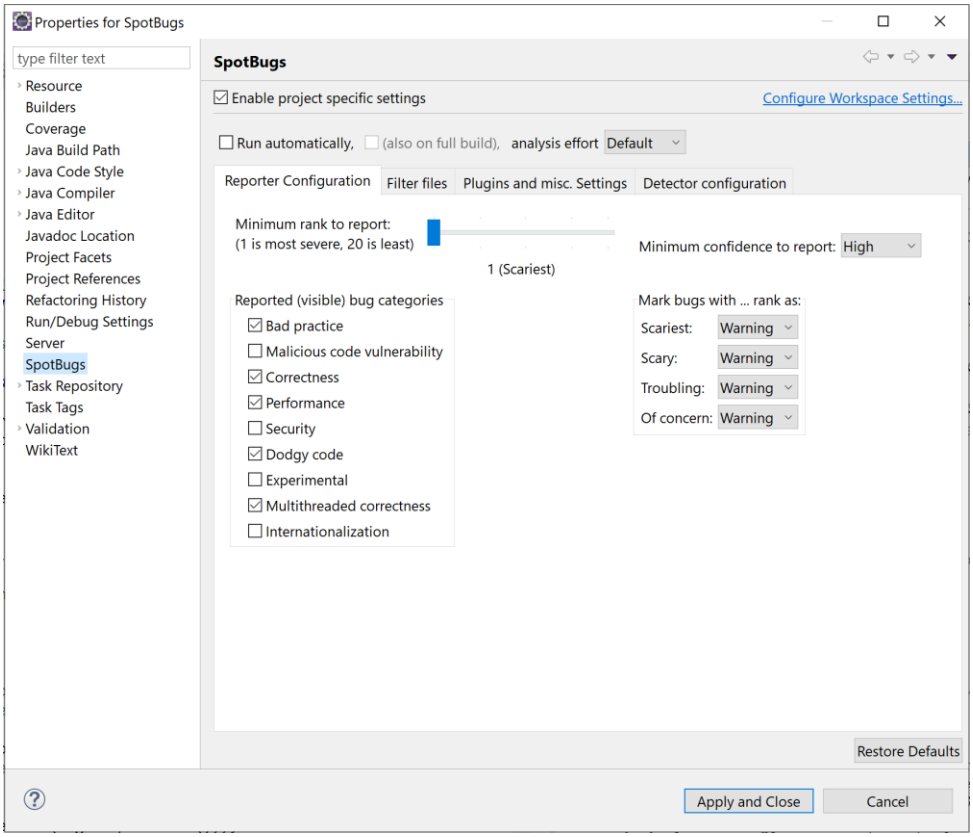
High Confidence Bugs-3

Low Confidence Bugs-1

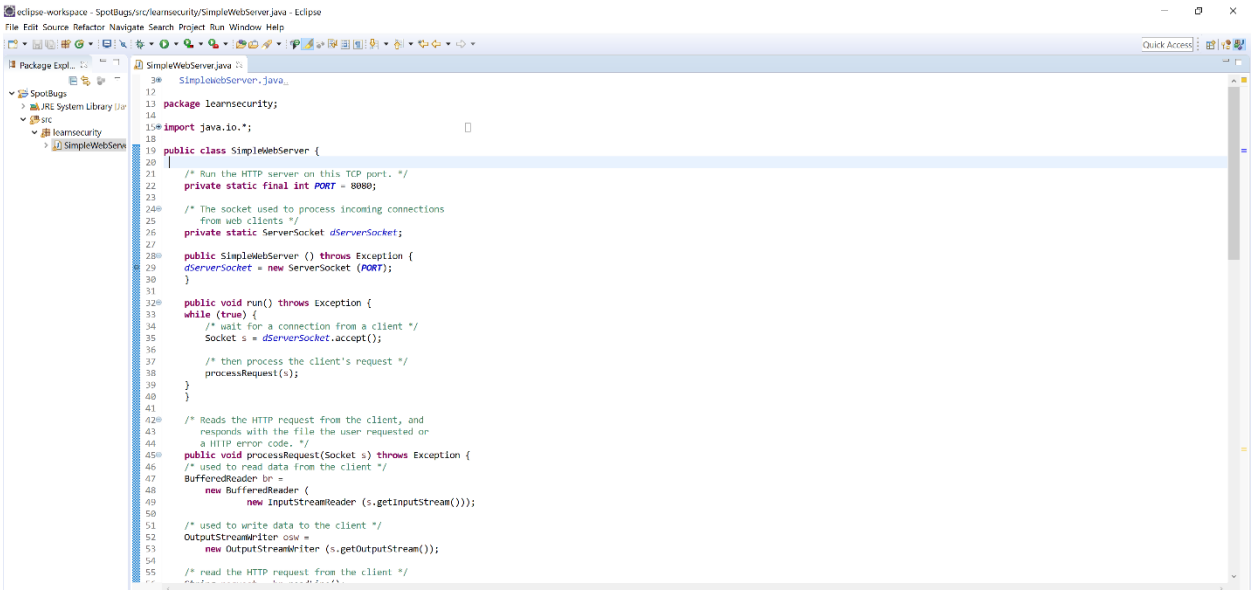
Normal Confidence Bugs-1

Test Configuration -3

We ran the SpotBugs plugin with the following configuration


Properties for SpotBugs
type filter text
Resource
Builders
Coverage
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project Facets
Project References
Refactoring History
Run/Debug Settings
Server
SpotBugs
Task Repository
Task Tags
Validation
WikiText
SpotBugs
Enable project specific settings
Configure Workspace Settings...
Run automatically, (also on full build), analysis effort Default
Reporter Configuration
Filter files
Plugins and misc. Settings
Detector configuration
Minimum rank to report:
(1 is most severe, 20 is least)
1 (Scariest)
Minimum confidence to report: High
Reported (visible) bug categories
Bad practice
Malicious code vulnerability
Correctness
Performance
Security
Dodgy code
Experimental
Multithreaded correctness
Internationalization
Mark bugs with ... rank as:
Scariest: Warning
Scary: Warning
Troubling: Warning
Of concern: Warning
Restore Defaults
Apply and Close
Cancel

Results:

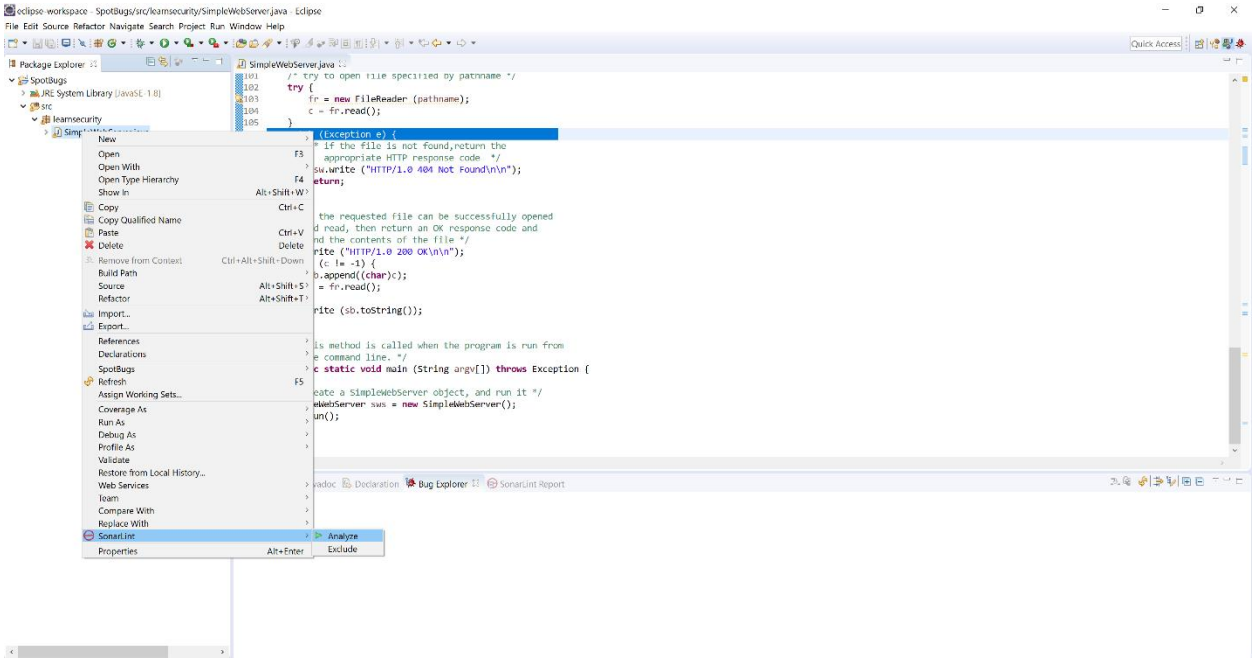

eclipse-workspace - SpotBugs/src/learnsecurity/SimpleWebServer.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Expl...
SpotBugs
JRE System Library
src
learnsecurity
SimpleWebServer.java
SimpleWebServer.java_
package learnsecurity;
import java.io.*;
public class SimpleWebServer {
/* Run the HTTP server on this TCP port. */
private static final int PORT = 8080;
/* The socket used to process incoming connections
from web clients */
private static ServerSocket dServerSocket;
public SimpleWebServer () throws Exception {
dServerSocket = new ServerSocket (PORT);
}
public void run() throws Exception {
while (true) {
/* wait for a connection from a client */
Socket s = dServerSocket.accept();
/* then process the client's request */
processRequest(s);
}
}
/* Reads the HTTP request from the client, and
responds with the file the user requested or
a HTTP error code. */
public void processRequest(Socket s) throws Exception {
/* used to read data from the client */
BufferedReader br =
new BufferedReader (
new InputStreamReader (s.getInputStream()));
/* used to write data to the client */
OutputStreamWriter osw =
new OutputStreamWriter (s.getOutputStream());
/* read the HTTP request from the client */

Number Of Bugs-0

SonarLint:

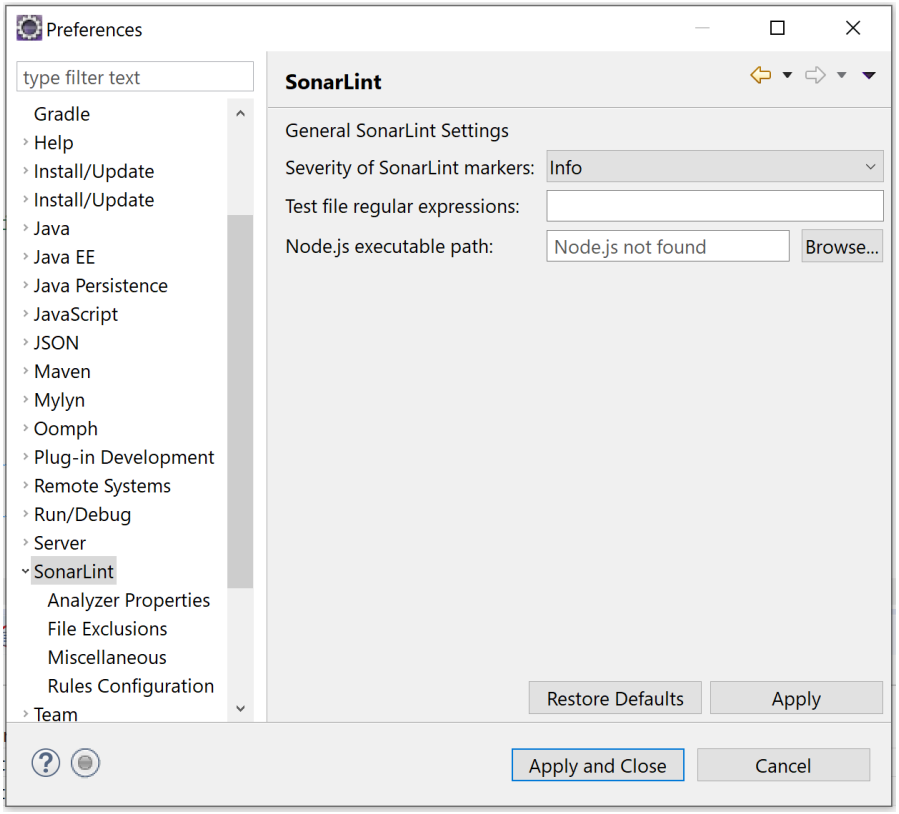
Procedure:

- We First download the tool from eclipse marketplace
- We then vary the configuration by modifying the severity of SonarLint markers(Info,Warning,Error) and record our observation
- We select the project(right click) and then select SonarLint and choose analyze as shown in the image below
- We see the output in terms of the SonarLint report

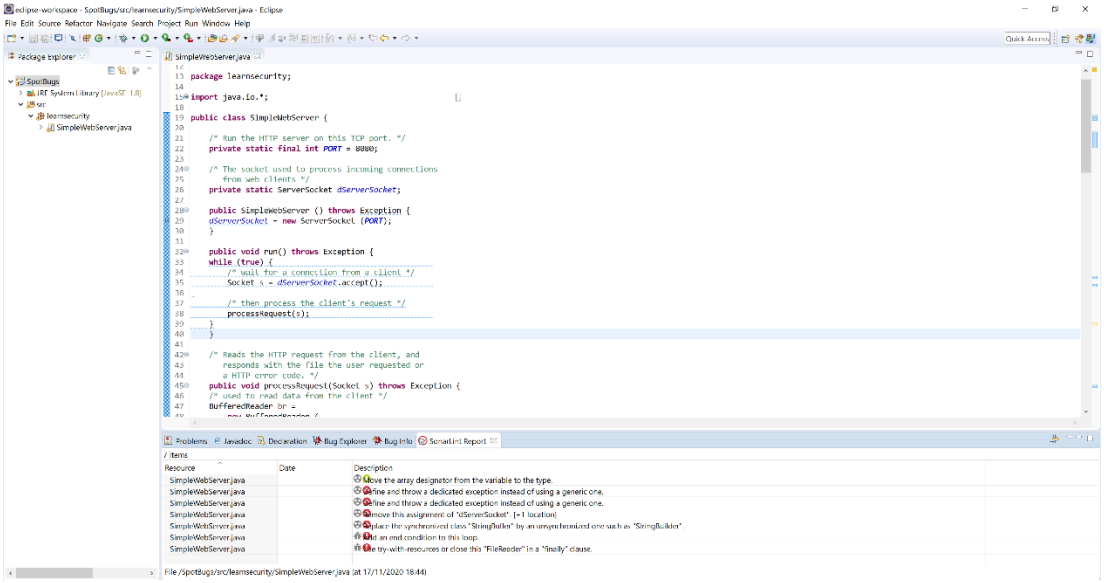


Test Configuration -1

We ran the SonarLint plugin with the Secerity of SonarLint set to info



Results:



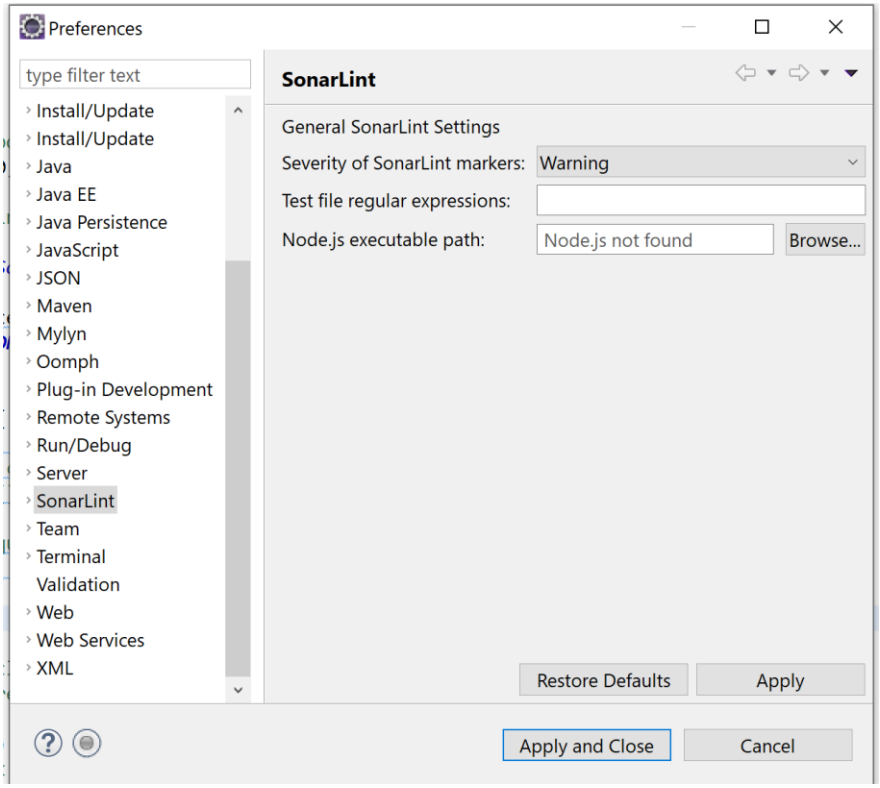
Number of Major code smells-4

Number of minor code smells-1

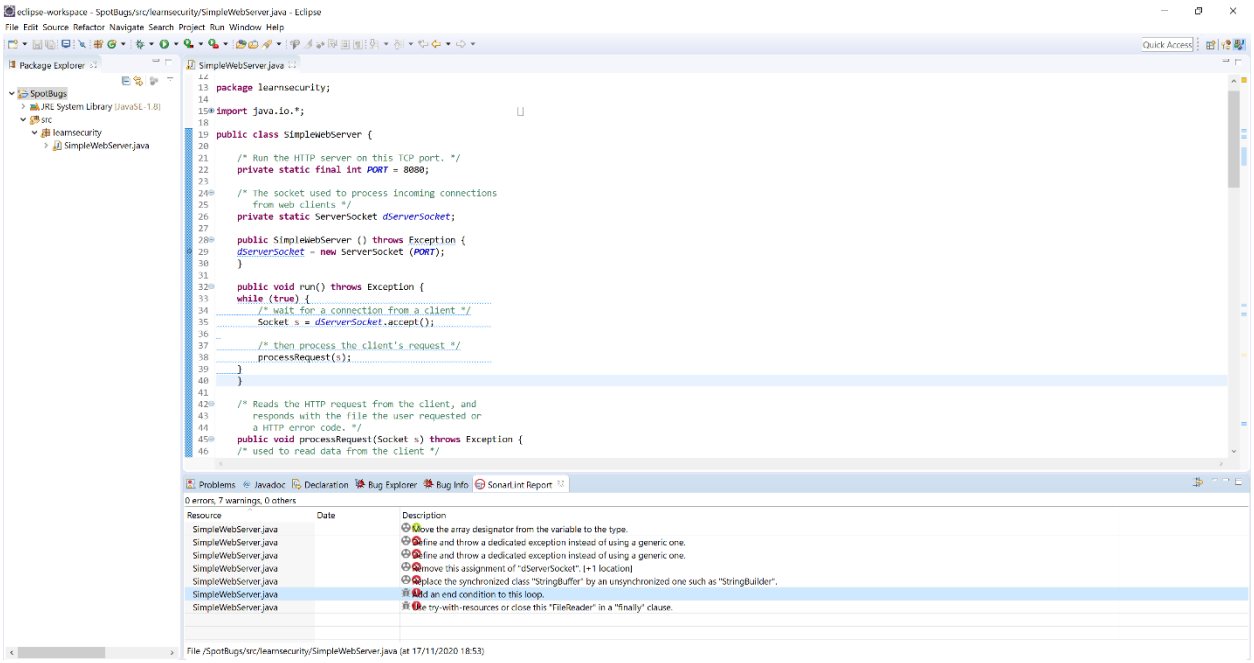
Number Of bugs-2

Test Configuration -2

We ran the SonarLint plugin with the Secerity of SonarLint set to warnings



Results:



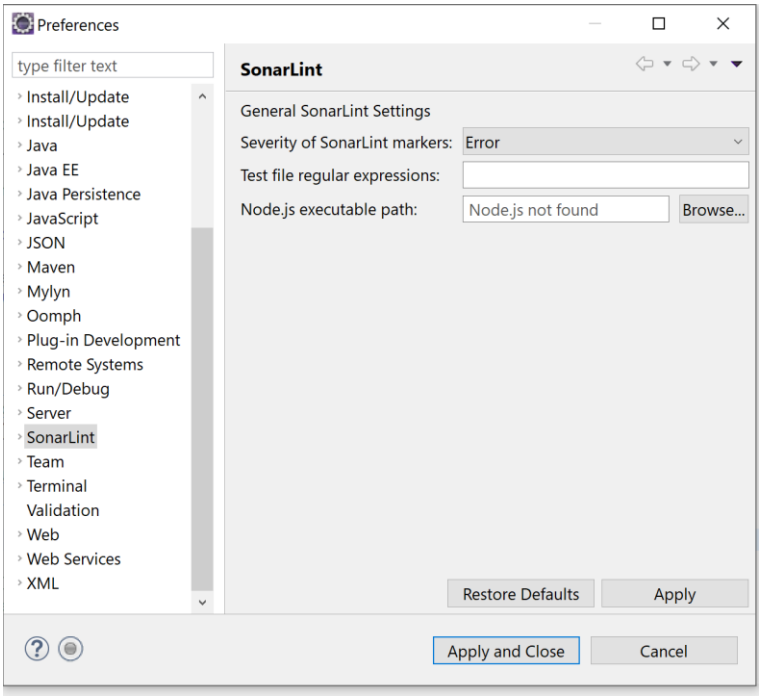
Number of Major code smells-4

Number of minor code smells-1

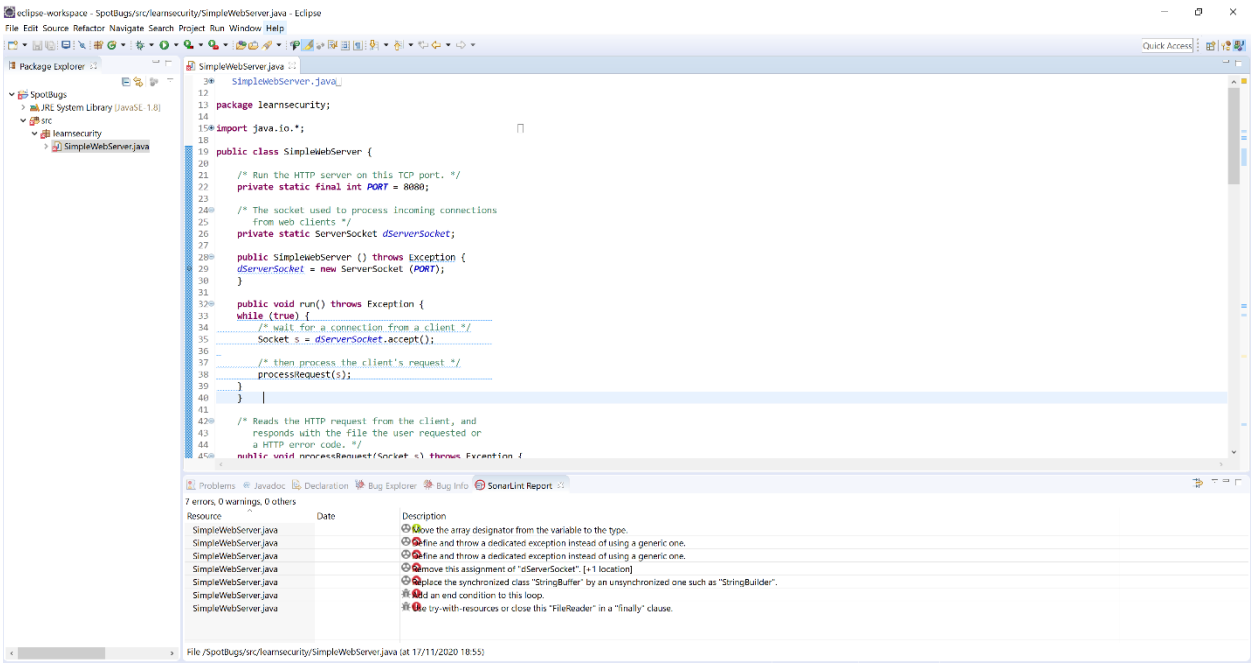
Number Of bugs-2

Test Configuration -3

We ran the SonarLint plugin with the Secerity of SonarLint set to Error



Results:



Number of Major code smells-4

Number of minor code smells-1

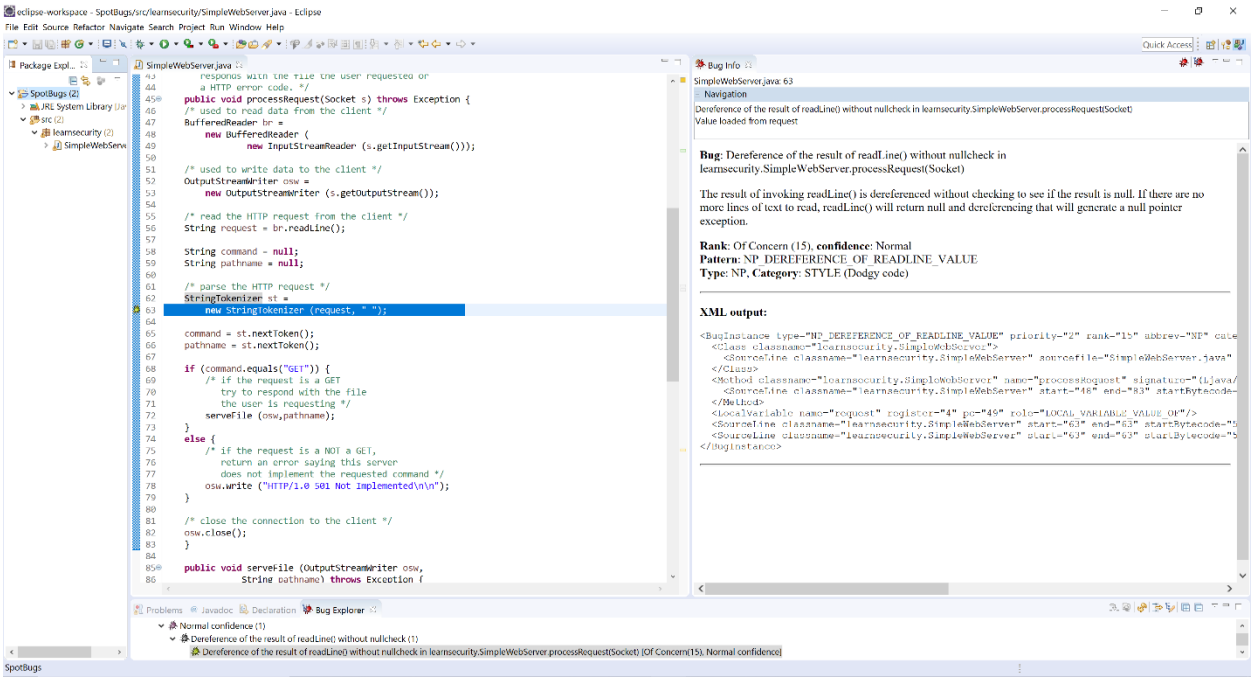
Number Of bugs-2

Comparison Of The Tools:

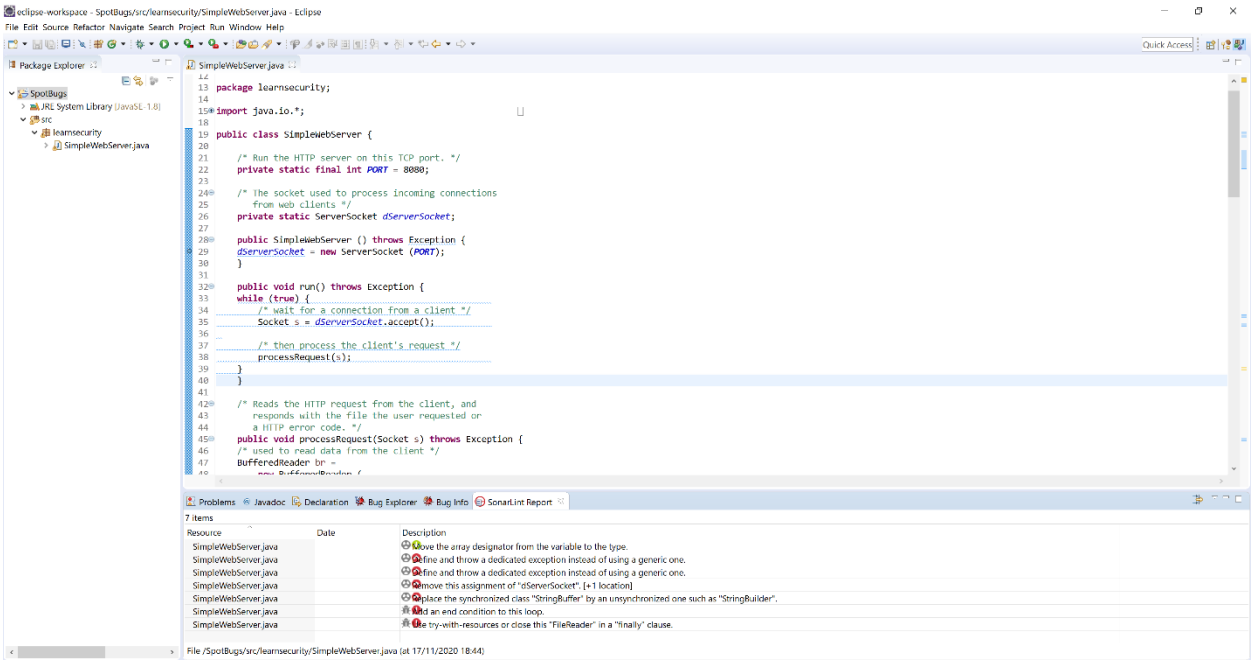
- We Notice that SonarLint analyzes the source code and SpotBugs analyzes the binary
- Both the tools SonarLine and SpotBugs are Bug finding tools
- In SpotBugs the output can be exported as an XML file but in SonarLint the output cannot be exported and can be seen only in the terminal

Show an example (if one exists) of a finding that is reported by one tool and not others.

SpotBugs:



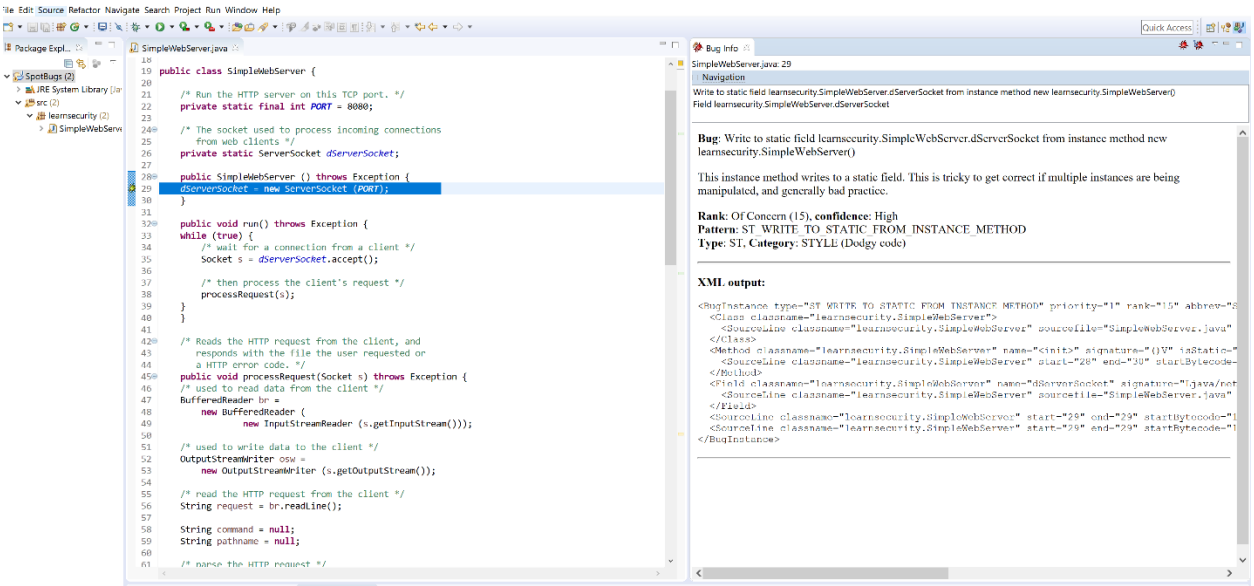
SonarLint:



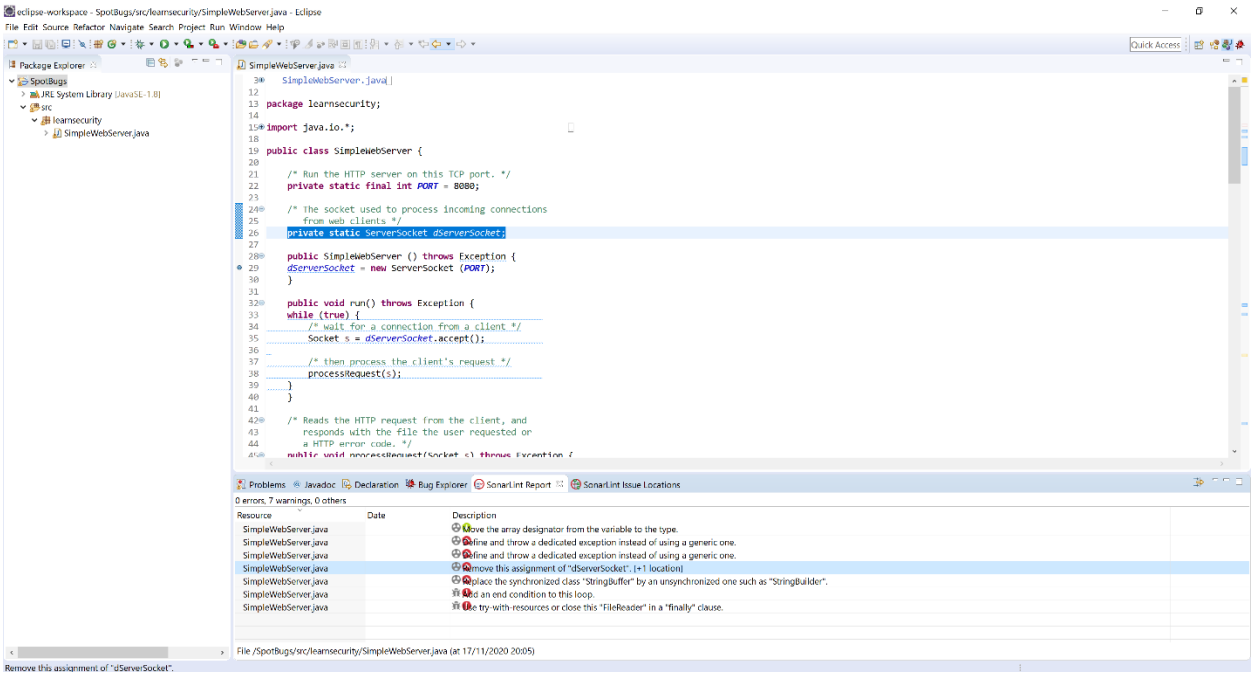
From the above we see that the Bug reported by SpotBugs(line 63) is not reported by SonarLint

Show an example (if one exists) of a finding reported by multiple tools.

SpotBugs:



SonarLint:



From the above it is clear that both the tools report the Bug on line 29 of the code

PART-2

Here I have used my code snippet I had written

Code:

```
package learnsecurity;

        public class LCM {

                public static void main(String[] args) {

                        int x1 = 72, x2 = 120;
                        int LCM;
                        if (x1>x2) {
                                LCM=x1;
                        }
                        else {
                                LCM=x2;
                        }

                        while(true) {
```

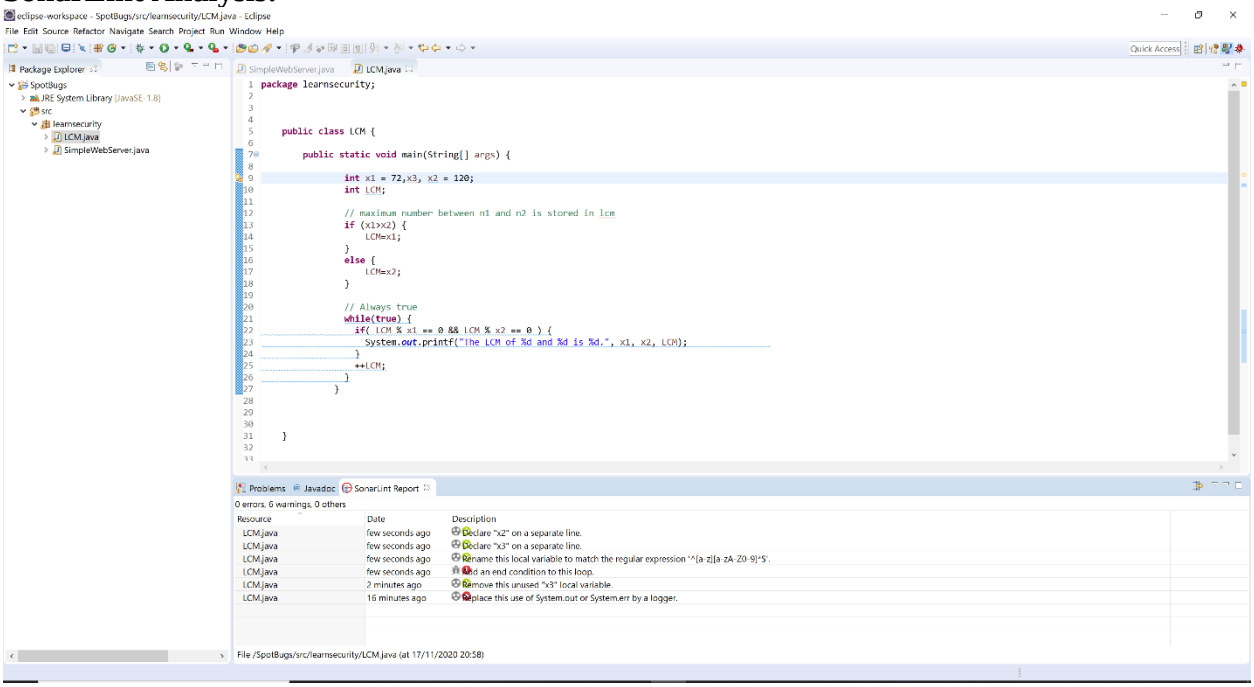
```
                    if( LCM % x1 == 0 && LCM % x2 == 0 ) {
                      System.out.printf("The LCM of %d and %d is %d.", x1, x2,LCM);

                    }
                    ++LCM;
                  }
                }

      }
```

```
public class LCM {

    public static void main(String[] args) {

            int x1 = 72, x2 = 120;
            int LCM;

            // maximum number between n1 and n2 is stored in lcm
            if (x1>x2) {
                LCM=x1;
            }
            else {
                LCM=x2;
            }

            // Always true
            while(true) {
              if( LCM % x1 == 0 && LCM % x2 == 0 ) {
                System.out.printf("The LCM of %d and %d is %d.", x1, x2, LCM);
              }
              ++LCM;
            }
          }


}
```

Manual Analysis:

- We observe that variable x3 is not used
- There is a while loop which will is not declared properly thus result in an infinite loop

SonarLint Analysis:



Results:

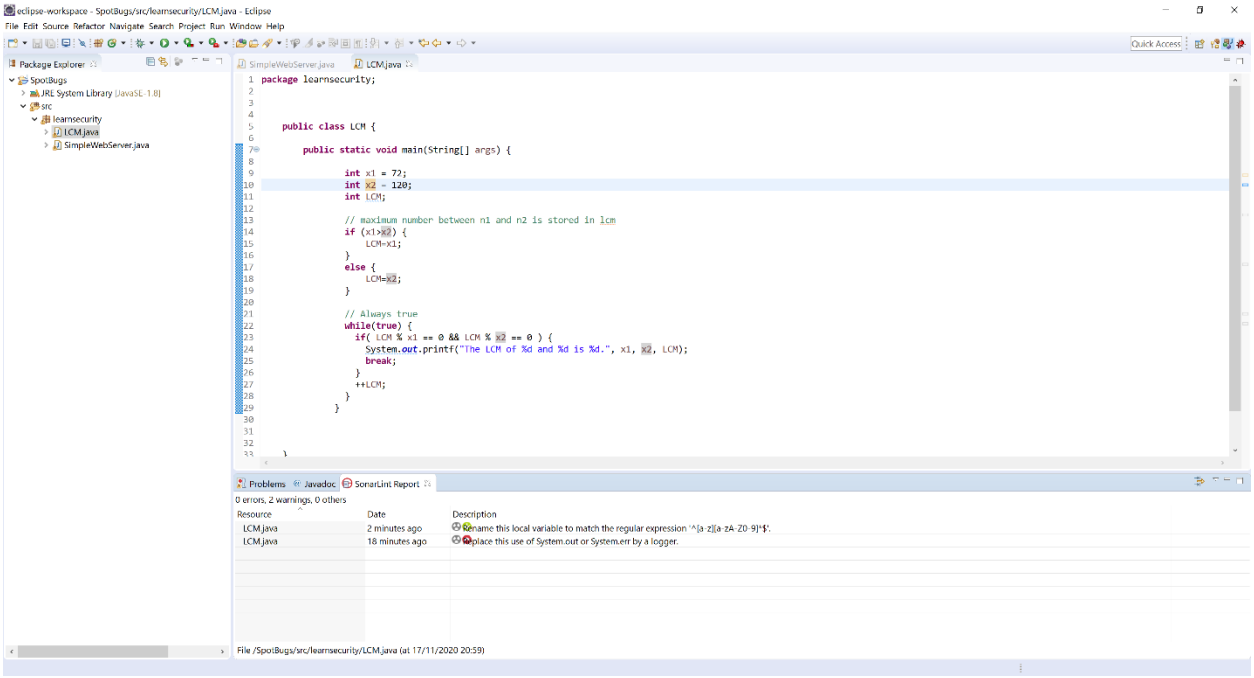
Number of Bugs-1

Number Of major Code smells-1

Number of Minor Code smells-4

Bug Fixes: we incorporate the bug fixes according to the SonartLint report

- We remove the unused variable x3
- Since the while loop is always true we have to find a way to end or break the loop thus we add a break statement
- We declare variable x2 in a new line as suggested.



We observe that after making the following changes the items which are reported as issues reduce from to 6 to 2 as shown in the image below