Configuration Manual

MSc Research Project
Masters in Cybersecurity

Ravali Chada
Student ID:23150335

School of Computing
National College of Ireland

Supervisor: Khadija Hafeez

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ravali Chada		
Student ID:	23150335		
Programme:	MSc Cybersecurity	**Year:**	2023-2024
Module:	Practicum		
Lecturer:	Khadija Hafeez		
Submission Due Date:	12 August 2024		
Project Title:	Automated Vulnerability Assessment Tool for Web Applications		
Word Count:	1439…………**Page Count:** 11		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Ravali Chada

Date: 12 August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Ravali Chada
Student ID:23150335

1 Integration Instructions

Step 1: Ensure Correct File Placement

1. **Directory Structure**:

Ensure the directory contains both the `VulnerabilityAssessmentTool.java` file and the `consolidated_vulnerability_scanner.py` file. The structure should look like this.

```
├─── VulnerabilityAssessmentTool.java

├─── VulnerabilityAssessmentTool.class

└─── consolidated_vulnerability_scanner.py
```

Step 2: Modify Java Code for Python Script Execution

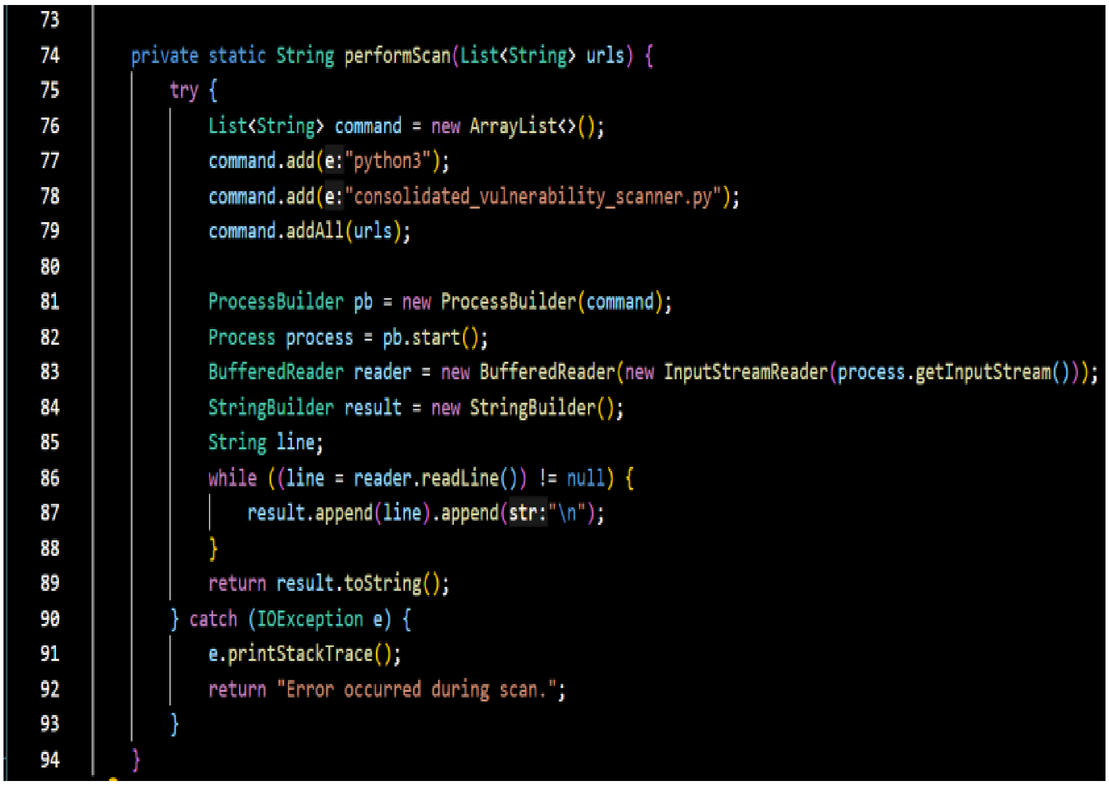
1. **Update Python Script Path in Java Code**:
   - Open the `VulnerabilityAssessmentTool.java` file in a text editor or an IDE.

Locate the `performScan` method. Ensure that the path to the Python script is correctly specified if the script is not in the same directory. Update the command list as needed:

```
private static String performScan(List<String> urls) {

         try {

             List<String> command = new ArrayList<>();

             command. Add("python3");

             command.add("path/to/consolidated_vulnerability_scanner.py"); // Update this path if necessary

             command.addAll(urls);


             ProcessBuilder pb = new ProcessBuilder(command);
```

```
                Process process = pb.start();
                BufferedReader reader = new BufferedReader(new
InputStreamReader(process.getInputStream()));
                StringBuilder result = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    result.append(line).append("\n");
                }
                return result.toString();
            } catch (IOException e) {
                e.printStackTrace();
                return "Error occurred during scan.";
            }
        }
```

- Save the changes to `VulnerabilityAssessmentTool.java`.



```
    private static String performScan(List<String> urls) {
        try {
            List<String> command = new ArrayList<>();
            command.add(e:"python3");
            command.add(e:"consolidated_vulnerability_scanner.py");
            command.addAll(urls);

            ProcessBuilder pb = new ProcessBuilder(command);
            Process process = pb.start();
            BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
            StringBuilder result = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                result.append(line).append(str:"\n");
            }
            return result.toString();
        } catch (IOException e) {
            e.printStackTrace();
            return "Error occurred during scan.";
        }
    }
```

Step 3: Compile and Run the Java Application

1. **Compile the Java Code**:
   - Open a terminal and navigate to the directory containing your Java file.

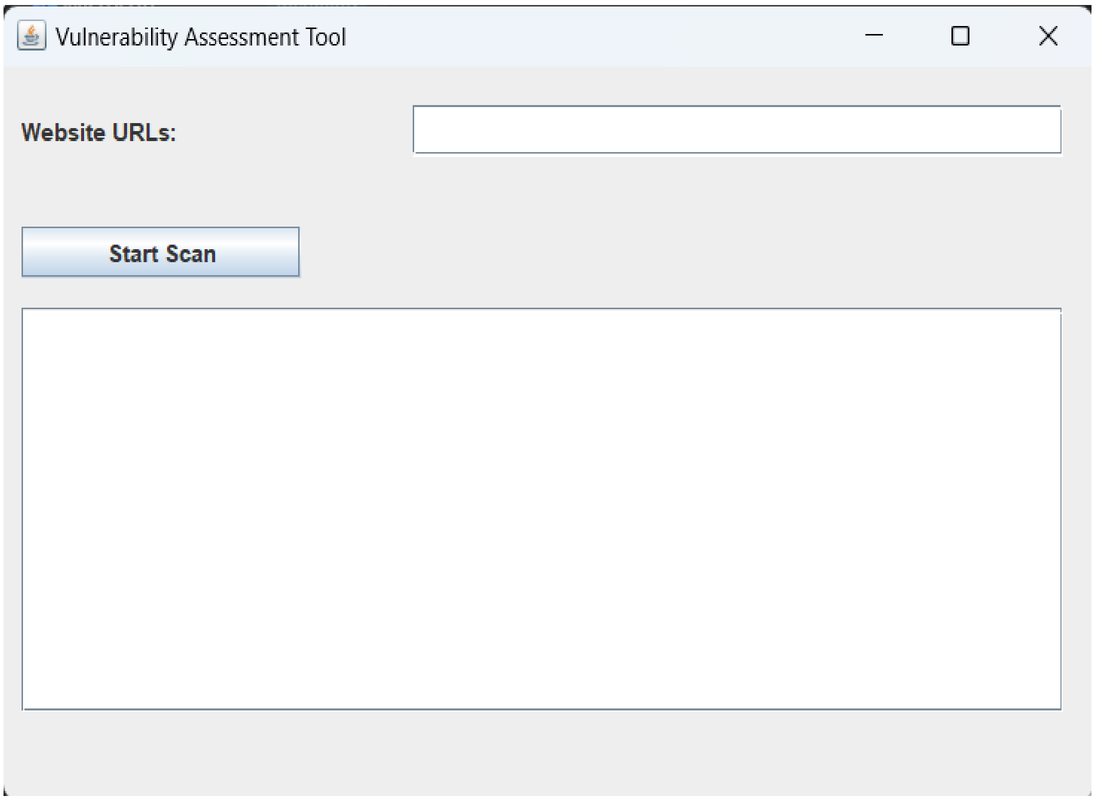
   Compile the Java code using the following command:

```
javac VulnerabilityAssessmentTool.java
```

2. **Run the Java Application**:

   Execute the Java application:

```
java VulnerabilityAssessmentTool
```

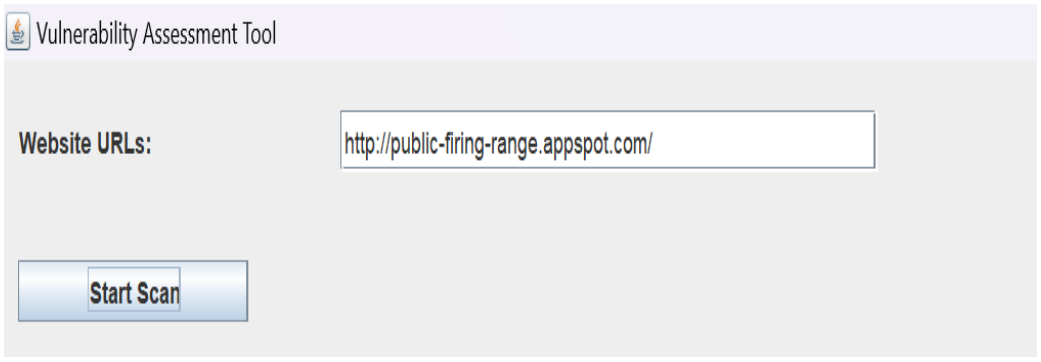


Step 4: Execute Vulnerability Scans Using the Integrated Tool

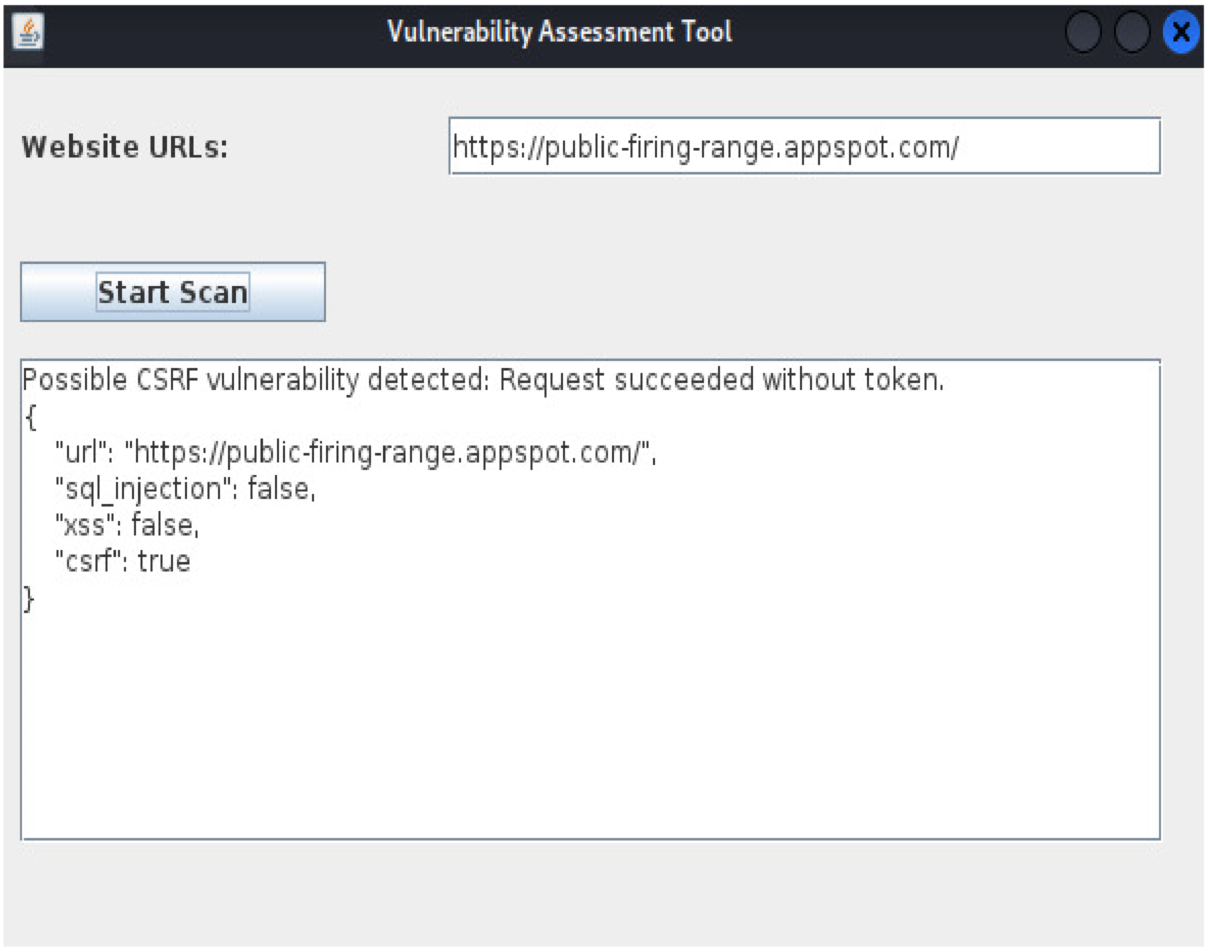
1. **Launch the Java Application**:
   - After running the `VulnerabilityAssessmentTool` class, a GUI window will appear.
2. **Input URLs**:

   In the text field labeled "Website URLs:", enter the URLs you want to scan, separated by commas. For example:

```
http://example.com, https://anotherexample.com
```

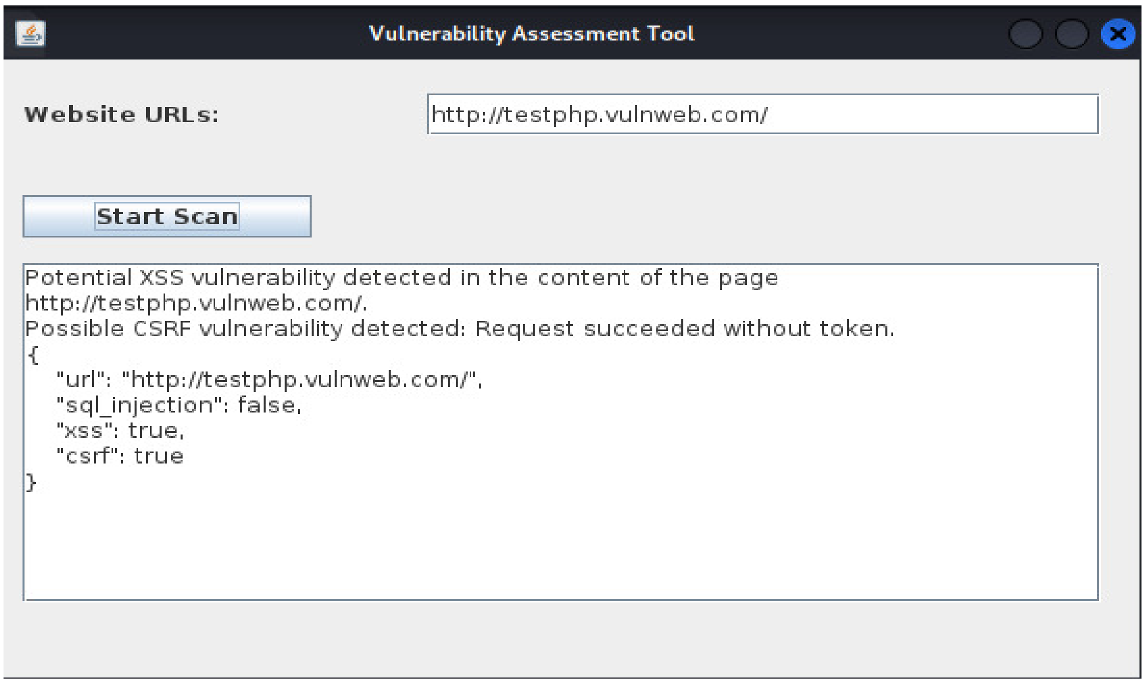

Vulnerability Assessment Tool
Website URLs:
http://public-firing-range.appspot.com/
Start Scan

3. **Start the Scan**:
    - Click the "Start Scan" button. The application will validate the URLs and, if they are valid, pass them to the Python script for scanning.
4. **View the Results**:
    - The scan results will be displayed in the text area within the GUI. A detailed report will also be saved in a file named `report. Json` in the current working directory.


Vulnerability Assessment Tool
Website URLs:
https://public-firing-range.appspot.com/
Start Scan

```
Possible CSRF vulnerability detected: Request succeeded without token.
{
   "url": "https://public-firing-range.appspot.com/",
   "sql_injection": false,
   "xss": false,
   "csrf": true
}
```

Another example:



```
Potential XSS vulnerability detected in the content of the page
http://testphp.vulnweb.com/.
Possible CSRF vulnerability detected: Request succeeded without token.
{
   "url": "http://testphp.vulnweb.com/",
   "sql_injection": false,
   "xss": true,
   "csrf": true
}
```

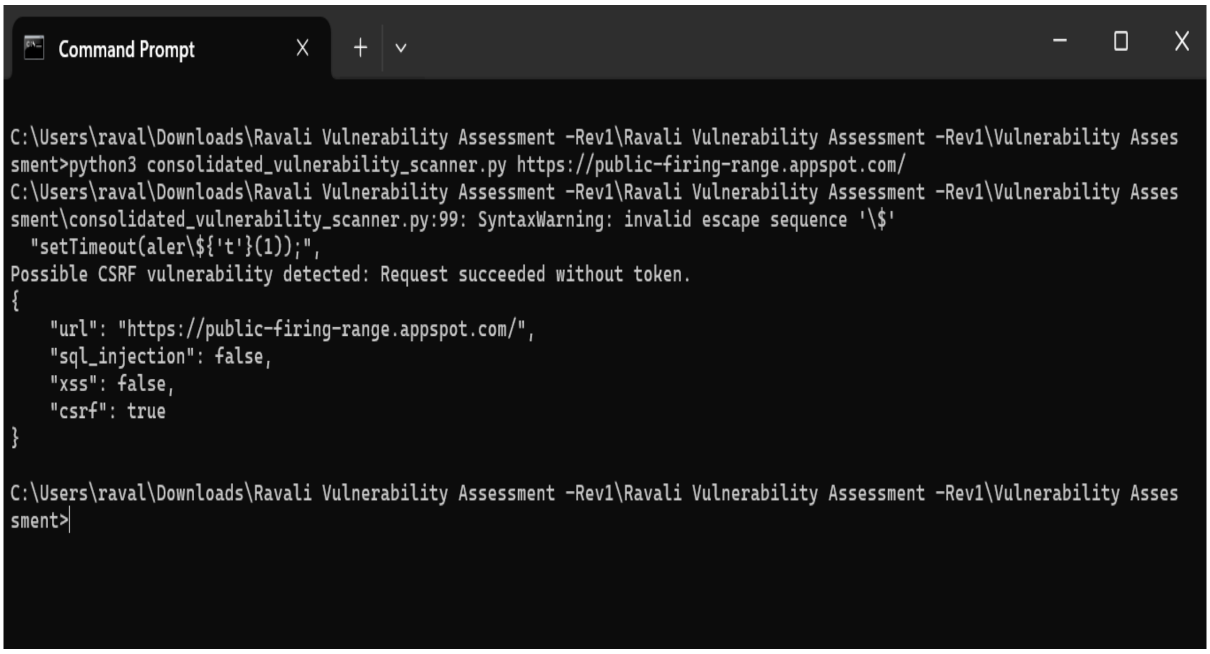
Step 5: Verify Python Script Execution

1. **Check Python Script Execution**:

   Ensure the Python script is executable by running it directly from the terminal:

```
python3 path/to/consolidated_vulnerability_scanner.py
http://example.com
```

   - This should return results like those displayed in the Java application's text area.



```
C:\Users\raval\Downloads\Ravali Vulnerability Assessment -Rev1\Ravali Vulnerability Assessment -Rev1\Vulnerability Asses
sment>python3 consolidated_vulnerability_scanner.py https://public-firing-range.appspot.com/
C:\Users\raval\Downloads\Ravali Vulnerability Assessment -Rev1\Ravali Vulnerability Assessment -Rev1\Vulnerability Asses
sment\consolidated_vulnerability_scanner.py:99: SyntaxWarning: invalid escape sequence '\$'
  "setTimeout(aler\${'t'}(1));",
Possible CSRF vulnerability detected: Request succeeded without token.
{
    "url": "https://public-firing-range.appspot.com/",
    "sql_injection": false,
    "xss": false,
    "csrf": true
}

C:\Users\raval\Downloads\Ravali Vulnerability Assessment -Rev1\Ravali Vulnerability Assessment -Rev1\Vulnerability Asses
sment>
```

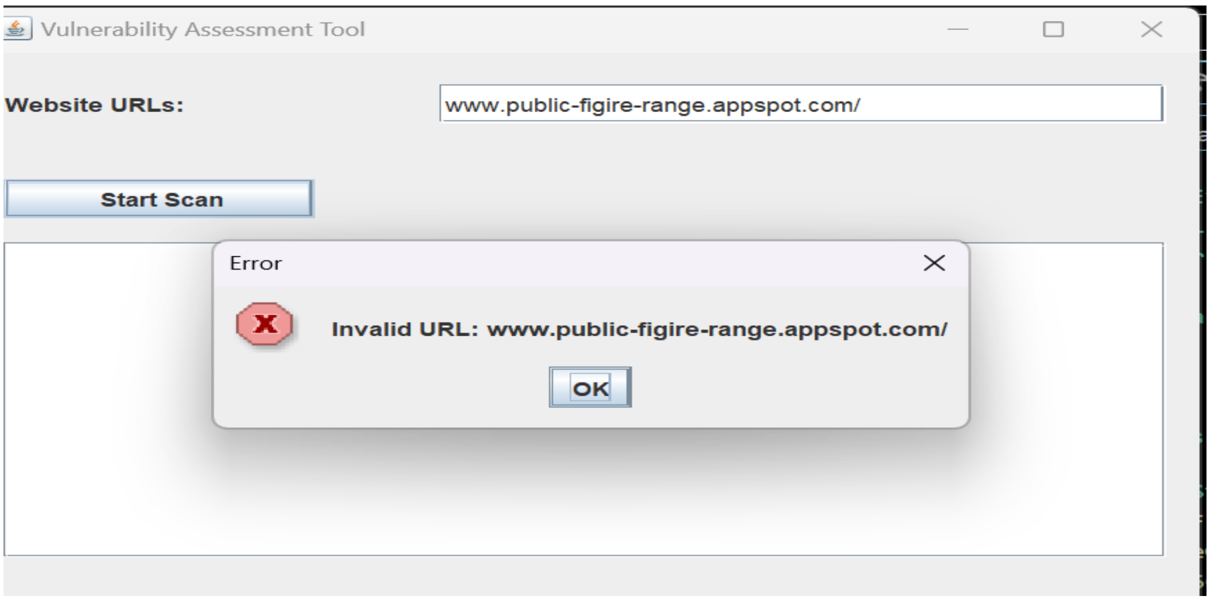
Step 6: Handle Common Issues

1. **Java Application Fails to Start**:
   - Ensure the JDK is correctly installed and the `JAVA_HOME` environment variable is set.
   - Verify the Java code is compiled without errors.
2. **Python Script Errors**:
   - Check that all required Python libraries (`requests`, `beautifulsoup4`, `json`, `re`) are installed.

Ensure the Python script has execution permissions:

```
chmod +x path/to/consolidated_vulnerability_scanner.py
```

3. **Invalid URLs**:
   - The application checks for valid URL formats. Ensure URLs are prefixed with `http://` or `https://`.


Vulnerability Assessment Tool

Website URLs: www.public-figire-range.appspot.com/

Start Scan

Error

Invalid URL: www.public-figire-range.appspot.com/

OK

By following these steps, the Java frontend will integrate with the Python backend, enabling vulnerability scans to be executed from the graphical interface.

2 Usage Guide

Launching the Application

1. **Run the Compiled Java Application**:
   - Open a terminal and navigate to the directory containing the compiled Java class file (`VulnerabilityAssessmentTool.class`).

Execute the Java application:

```
java VulnerabilityAssessmentTool
```

    - A graphical user interface (GUI) window titled "Vulnerability Assessment Tool" will appear.

Input URLs

1. **Enter URLs for Scanning**:
    - In the text field labeled **"Website URLs:"**, enter one or more URLs you wish to scan for vulnerabilities.

Ensure that the URLs are separated by commas if you enter multiple URLs. For example: `http://example.com, https://anotherexample.com`

- URLs must start with `http://` or `https://` to be considered valid.

Starting the Scan

1. **Initiate the Vulnerability Scan**:
    - After entering the URLs, click the **"Start Scan"** button below the text field.
    - The application will validate each URL. If a URL is invalid, an error message will be displayed indicating which URL is incorrect.

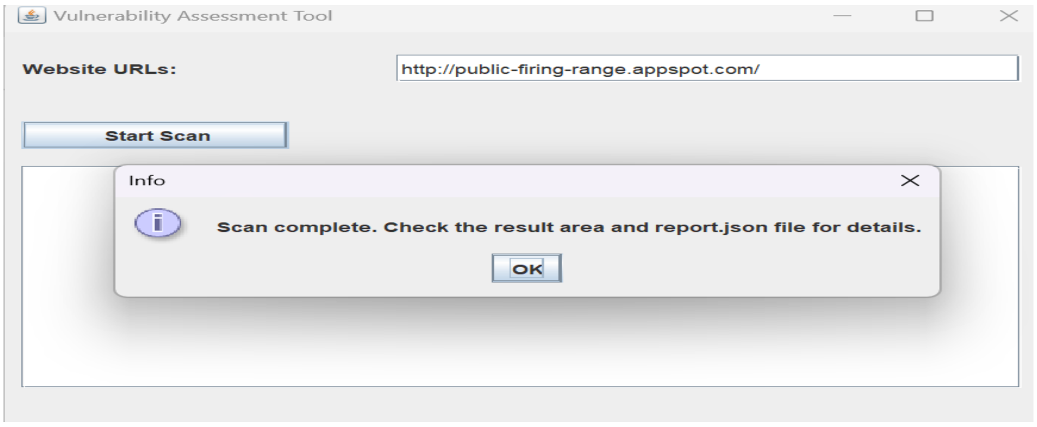
2. **Scan Execution**:
    - The Java application will call the Python script (`consolidated_vulnerability_scanner.py`) for valid URLs to perform the vulnerability scan.
    - A loading message or indicator may be displayed while the scan progresses.

Viewing the Results

1. **Results Display**:
    - Once the scan is complete, the results will be displayed in the text area within the GUI.
    - The text area provides a scrollable view to see the complete output of the scan.

2. **Interpreting the Results**:
    - The results will indicate whether vulnerabilities such as SQL injection, Cross-Site Scripting (XSS), or Cross-Site Request Forgery (CSRF) were detected for each URL.
    - Each detected vulnerability will be listed with details, including the payload that triggered the detection.

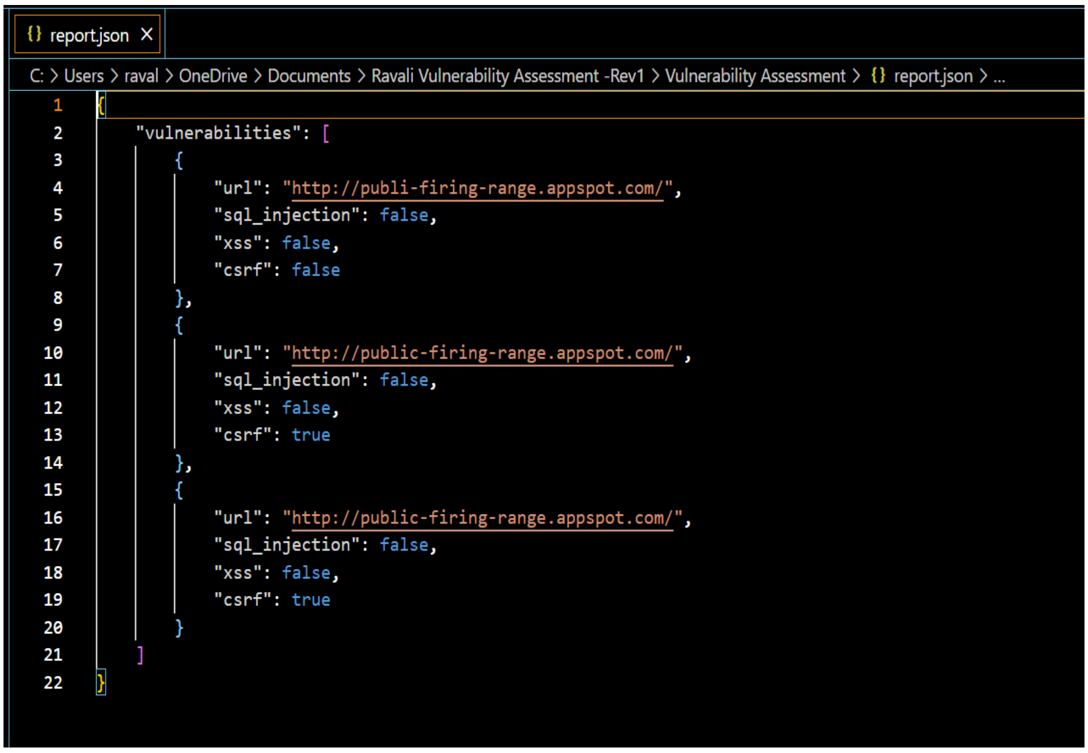
3. **Information Popup**:
    - A message box will appear informing you that the scan is complete and instructing you to check the result area and `report.json` file for detailed information.


Vulnerability Assessment Tool
Website URLs:
http://public-firing-range.appspot.com/
Start Scan
Info
OK

Scan complete. Check the result area and report.json file for details.

4. **Detailed Report**

1. **Accessing the Report**:
   - A detailed scan results report is saved in a file named `report.json` in the current working directory.
   - Open this file with any text editor or JSON viewer to see a structured and detailed account of the vulnerabilities detected.
2. **Understanding the Report Format**:
   - The `report.json` file contains an array of objects, each representing the scan results for a specific URL.
   - Each object includes the URL, and boolean flags indicating the presence of vulnerabilities (`sql_injection`, `xss`, `csrf`), and any error messages for invalid URLs.


{} report.json
C: > Users > raval > OneDrive > Documents > Ravali Vulnerability Assessment -Rev1 > Vulnerability Assessment > {} report.json > ...

```
{
    "vulnerabilities": [
        {
            "url": "http://publi-firing-range.appspot.com/",
            "sql_injection": false,
            "xss": false,
            "csrf": false
        },
        {
            "url": "http://public-firing-range.appspot.com/",
            "sql_injection": false,
            "xss": false,
            "csrf": true
        },
        {
            "url": "http://public-firing-range.appspot.com/",
            "sql_injection": false,
            "xss": false,
            "csrf": true
        }
    ]
}
```

3.**Common Issues and Troubleshooting**

1. **Java Application Fails to Start**:
   - Ensure the JDK is correctly installed and the `JAVA_HOME` environment variable is set.
   - Verify the Java code is compiled without errors.
2. **Python Script Errors**:
   - Check that all required Python libraries (`requests`, `beautifulsoup4`, `json`, `re`) are installed.

Ensure the Python script has execution permissions:

```
chmod +x path/to/consolidated_vulnerability_scanner.py
```

3. **Invalid URLs**:
   - Ensure URLs are correctly formatted and prefixed with `http://` or `https://`.

3 Troubleshooting

Java Application Issues

1. **Java Application Fails to Start**:
   - **Error Message**: `java.lang.NoClassDefFoundError`
     - **Solution**: Ensure the JDK is correctly installed and the `JAVA_HOME` environment variable is set.
     - **Steps**:
   - Check the JDK installation:
     ```
     java -version
     ```
   - Set the `JAVA_HOME` environment variable:
     ```
     export JAVA_HOME=/path/to/jdk

     export PATH=$JAVA_HOME/bin:$PATH
     ```
   - **Error Message**: `ClassNotFoundException`
     - **Solution**: Ensure the Java class file is in the correct directory and compiled without errors.
     - **Steps**:
   - Compile the Java code:
     ```
     javac VulnerabilityAssessmentTool.java
     ```
   - Run the Java application:
     ```
     java VulnerabilityAssessmentTool
     ```

Python Script Issues

1. **Python Script Execution Errors**:
   - **Error Message**: `ImportError: No module named 'requests'`
     - **Solution**: Install the required Python libraries.
     - **Steps**:

- Install the necessary libraries:
  `pip3 install requests beautifulsoup4`

- **Error Message**: `Permission denied`
  - **Solution**: Ensure the Python script has execution permissions.
  - **Steps**:
- Set the execute permission for the Python script:
  ```
  chmod +x
  path/to/consolidated_vulnerability_scanner.py
  ```

Invalid URLs

1. **Error Message**: `Invalid URL`
   - **Solution**: Ensure URLs are correctly formatted and prefixed with `http://` or `https://`.
   - **Steps**:
   - Verify the URL format before entering it into the application. URLs should be in the format:
     `http://example.com`
   - `https://anotherexample.com`

General Application Issues

1. **Scan Fails to Start**:
   - **Issue**: No output or response when clicking "Start Scan".
     - **Solution**: Ensure the paths to the Python script and other resources are correctly specified in the Java code.
     - **Steps**:
   - Verify and update the path to the Python script in the `performScan` method:
     ```
     command.add("path/to/consolidated_vulnerability_scan
     ner.py"); // Update this path if necessary
     ```
2. **Java Application Freezes or Crashes**:

Issue: The application becomes unresponsive during scanning.

- **Solution**: Ensure the system has sufficient resources and the scanned URLs do not cause the Python script to hang.
- **Steps**:
  - Monitor system resources and terminate any processes consuming excessive CPU or memory.
  - Test with a single, known-good URL to verify basic functionality.

3. **Unexpected Output or Incomplete Results**:

Issue: Scan results are not as expected or incomplete.

- **Solution**: Check for errors in the Python script output and ensure all necessary libraries and dependencies are installed correctly.

- **Steps**: Run the Python script directly with a test URL to identify any issues: `python3path/to/consolidated_vulnerability_scanner.py http://example.com`

Detailed Logging

1. **Enable Detailed Logging**:

Issue: Difficulty diagnosing problems without detailed logs.

- **Solution**: Add logging to Java and Python code to capture detailed execution information.
- **Steps**:
    - Add logging statements in the Java code to capture key events and errors.
    - Modify the Python script to log detailed information about the scanning process and any encountered issues.

References

[1] Oracle Corporation. (n.d.). *ProcessBuilder (Java Platform SE 11)*. Oracle. Retrieved August 12, 2024, from https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/ProcessBuilder.html

[2] Baeldung, E. (2019). *Execute Python scripts from Java*. Baeldung. Retrieved August 12, 2024, from https://www.baeldung.com/run-shell-command-in-java

[3] GeeksforGeeks. (n.d.). *Java-Python Integration: How to Run Python Code in Java*. GeeksforGeeks. Retrieved August 12, 2024, from https://www.geeksforgeeks.org/different-ways-to-execute-python-code-in-java/

[4] Luv2code Blog. (2019). *Running shell commands in Java using ProcessBuilder*. Luv2code. Retrieved August 12, 2024, from https://www.luv2code.com/2019/10/10/running-shell-commands-in-java-using-processbuilder/