**Static Analysis Rules & Configuration Guide**

This document explains the static analysis rules, tools, and configuration used in your AI Code Reviewer project. It is designed to help any developer understand how code is checked for style and security issues, what patterns are flagged, and how to interpret the results.

**1. Tools Used**

**Flake8**

* **Purpose:** Checks Python code for style and quality issues (PEP8 compliance, unused imports, syntax errors, etc.).
* **Output:** Issues are grouped by filename, each entry shows the file, line, column, and error code/message.

**Bandit**

* **Purpose:** Scans Python code for common security vulnerabilities (dangerous functions, insecure imports, hardcoded passwords, etc.).
* **Output:** Issues are grouped by filename, each entry shows the file and a detailed description of the security concern.

**Custom Security Scan**

* **Purpose:** Adds extra checks for risky patterns not always covered by Bandit (e.g., use of pickle, yaml.load, os.system, input, etc.).
* **Output:** Issues are grouped by filename, each entry shows the file and the risky pattern found.

**2. How Static Analysis Works**

* The script recursively scans all .py files in the target directory.
* Each file is checked by Flake8 and Bandit.
* Custom regex-based checks supplement Bandit for additional risky patterns.
* Results are grouped by filename and stored in [static\_output.json](vscode-file://vscode-app/c:/Users/mdhnd/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ").

**3. Style Rules (Flake8)**

**Common Issues Detected:**

* **E501:** Line too long (>79 characters)
* **E302/E305:** Expected blank lines before/after functions/classes
* **W292:** No newline at end of file
* **F401:** Unused imports
* **E999:** Syntax errors

**Example Output:**

"run\_static.py": [

"run\_static.py: Line 12 Col 1: - E302 expected 2 blank lines, found 1",

"run\_static.py: Line 13 Col 80: - E501 line too long (84 > 79 characters)"

]

**4. Security Rules (Bandit + Custom)**

**Bandit Flags:**

* **B307:** Use of eval (dangerous, prefer ast.literal\_eval)
* **B105:** Hardcoded password strings
* **B404:** Use of subprocess module (potential command injection)
* **B603/B602:** Unsafe subprocess calls

**Custom Patterns Detected:**

* pickle.load (unsafe deserialization)
* yaml.load (unsafe YAML parsing)
* os.system (shell command execution)
* input(, exec(, eval( (dangerous code execution)
* subprocess.Popen, subprocess.call (potential command injection)
* open(..., w) (writing files, check for unsafe file handling)

**Example Output:**

*"run\_static.py": [*

*">> Issue: [B404:blacklist] Consider possible security implications associated with the subprocess module. ...",*

*"run\_static.py: Custom Security: Found risky pattern '\\bopen\\s\*\\(.\*w\\b' at position 5587"*

*]*

**5. Timeout Protection**

* Each static analysis tool is run with a timeout (default: 30–60 seconds).
* If a tool runs too long (e.g., infinite loop), it is stopped and an error is reported.

**6. Interpreting Results**

* **Files with issues:** Listed under "style" and/or "security" with details.
* **Files with no issues:** May appear with an empty list, or be omitted if filtered.
* **Custom findings:** Clearly marked as "Custom Security" in the output.

**7. Configuration**

* **Target Directory:** Change with --path argument (default: backend).
* **Timeouts:** Can be adjusted in the script (run\_command function).
* **Custom Patterns:** Add/remove regexes in custom\_security\_scan().

**8. How to Extend**

* **Add more Flake8 plugins** for additional style checks.
* **Add more Bandit plugins** or update Bandit configuration for stricter security.
* **Add more custom regex patterns** to custom\_security\_scan() for project-specific risks.

**9. Sample Output File (**[static\_output.json](vscode-file://vscode-app/c:/Users/mdhnd/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")**)**

*{*

*"style": {*

*"long\_line.py": [*

*"long\_line.py: Line 2 Col 80: - E501 line too long (134 > 79 characters)"*

*],*

*"good\_code.py": [*

*"good\_code.py: Line 4 Col 1: - E305 expected 2 blank lines after class or function definition, found 1"*

*]*

*},*

*"security": {*

*"eval\_danger.py": [*

*">> Issue: [B307:blacklist] Use of possibly insecure function ...",*

*"eval\_danger.py: Custom Security: Found risky pattern '\\beval\\s\*\\(' at position 30"*

*]*

*}*

*}*

**10. Summary**

* **Style issues** help maintain readable, maintainable code.
* **Security issues** help prevent vulnerabilities and unsafe practices.
* **Custom rules** catch project-specific risks.
* **Timeouts** ensure analysis is robust and does not hang.