AUBURN

UNIVERSITY

Software Quality Assurance (COMP 6710)

Project Report

Done By:
Koushik Govardhanam – kzg0099
Mounika Ghanta-mgz0144
Pavan Kalyan Annadevara- pza0045

Summary:

The goal of this project is to embed software quality assurance practices into an existing Python application. These efforts focus on improving the codebase's reliability, security, and maintainability by applying principles discussed in workshops. Key activities included in the project are as follows:

1. **Git Hooks Implementation:**
   A Git Hook was added to automatically scan Python files for security vulnerabilities whenever changes are committed. The system generates a result.csv file detailing any identified issues, ensuring developers are promptly alerted to potential weaknesses.
2. **Fuzz Testing:**
   A script named fuzz.py was developed to randomly test five selected methods using various input combinations. This script is integrated into GitHub Actions, enabling it to run automatically during commits and continually test the application for errors or inconsistencies.
3. **Forensic Logging:**
   Logging mechanisms were incorporated into five critical methods to track their usage. This includes capturing parameters, outputs, and execution flows, providing developers with detailed traceability to simplify debugging and improve overall transparency.
4. **Continuous Integration with GitHub Actions:**
   A set of workflows was configured to automate static analysis, fuzz testing, and linting whenever changes are pushed to the repository. These workflows ensure that errors and potential issues are identified early in the development cycle, streamlining quality assurance processes.

By integrating these activities, the project highlights a well-rounded approach to software quality assurance. It leverages static analysis, fuzz testing, forensic logging, and automation via continuous integration pipelines to uphold coding best practices. The outcomes and lessons learned through this integration are comprehensively documented in the accompanying project report.

Project for Software Quality Assurance (CSSE/6710) CODECRAFTERS-SQA2024-Auburn

Task 4.a: Setting Up a Pre-Commit Hook and Conducting Security Analysis with Bandit

The objective was accomplished by integrating a Git Hook into the development process. The Git Hook enables static analysis to operate seamlessly within the workflow, providing real-time detection of potential weaknesses whenever code changes are committed. This approach ensures a proactive and consistent enhancement of code security during development.

Repository Initialization:

- Initialized a Git repository and cloned it onto the local machine for development.
- Verified the repository structure to ensure readiness for customization.

Pre-Commit Hook Creation:

- Navigated to the ./git/hooks/ directory and reviewed the pre-commit.sample file.
- Used the sample file as a template to create a new file named pre-commit.
- Customized the new pre-commit script to include specific checks or validations to execute before commits.

Testing the Hook:

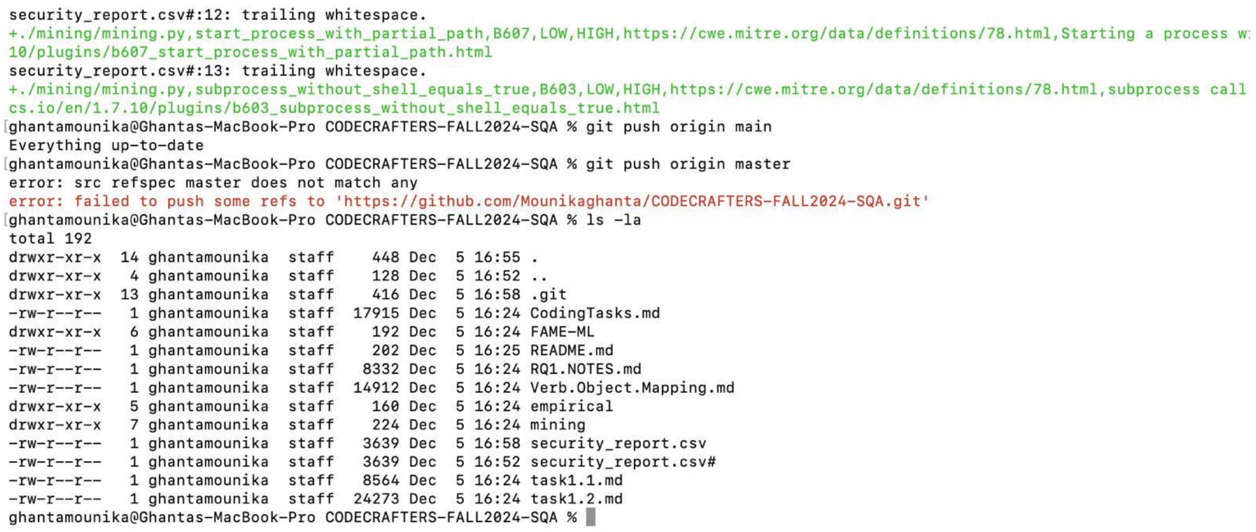
- Made changes to the report.py file in empirical within the repository to test the functionality of the pre-commit hook.
- Attempted a commit to confirm that the hook was correctly triggered and enforced the configured rules.

Security Analysis:

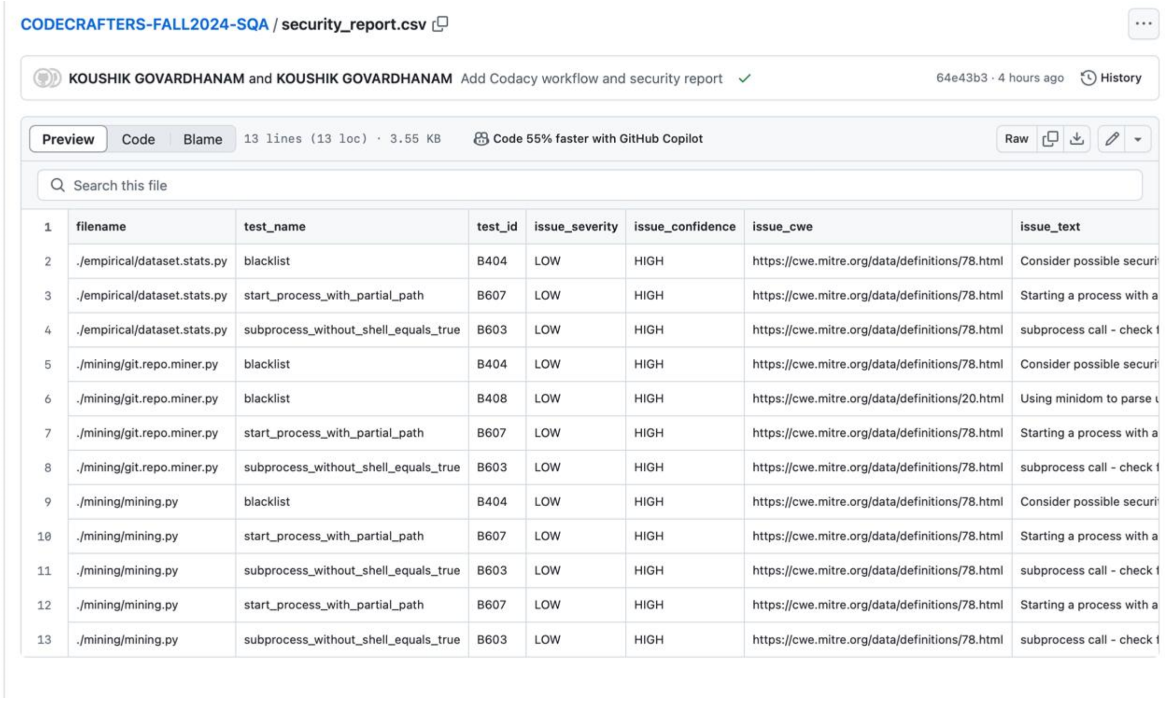
- Executed the bandit -r command to scan the repository for potential security vulnerabilities.
- Focused on identifying weaknesses in the updated report.py file.
- Documented the results provided by Bandit, including flagged warnings or suggestions for improvement.

Outcomes:

- The custom pre-commit hook was successfully implemented, tested, and validated.
- Bandit’s analysis provided actionable insights into potential vulnerabilities in the repository, particularly in the modified report.py file.
- These steps combined reinforce code quality and security in the repository, making it better equipped for reliable and secure development.
- The results are stored in security-report.csv file.



```
security_report.csv#:12: trailing whitespace.
+./mining/mining.py,start_process_with_partial_path,B607,LOW,HIGH,https://cwe.mitre.org/data/definitions/78.html,Starting a process w
10/plugins/b607_start_process_with_partial_path.html
security_report.csv#:13: trailing whitespace.
+./mining/mining.py,subprocess_without_shell_equals_true,B603,LOW,HIGH,https://cwe.mitre.org/data/definitions/78.html,subprocess call
cs.io/en/1.7.10/plugins/b603_subprocess_without_shell_equals_true.html
[ghantamounika@Ghantas-MacBook-Pro CODECRAFTERS-FALL2024-SQA % git push origin main
Everything up-to-date
[ghantamounika@Ghantas-MacBook-Pro CODECRAFTERS-FALL2024-SQA % git push origin master
error: src refspec master does not match any
error: failed to push some refs to 'https://github.com/Mounikaghanta/CODECRAFTERS-FALL2024-SQA.git'
[ghantamounika@Ghantas-MacBook-Pro CODECRAFTERS-FALL2024-SQA % ls -la
total 192
drwxr-xr-x  14 ghantamounika  staff    448 Dec  5 16:55 .
drwxr-xr-x   4 ghantamounika  staff    128 Dec  5 16:52 ..
drwxr-xr-x  13 ghantamounika  staff    416 Dec  5 16:58 .git
-rw-r--r--   1 ghantamounika  staff  17915 Dec  5 16:24 CodingTasks.md
drwxr-xr-x   6 ghantamounika  staff    192 Dec  5 16:24 FAME-ML
-rw-r--r--   1 ghantamounika  staff    202 Dec  5 16:25 README.md
-rw-r--r--   1 ghantamounika  staff   8332 Dec  5 16:24 RQ1.NOTES.md
-rw-r--r--   1 ghantamounika  staff  14912 Dec  5 16:24 Verb.Object.Mapping.md
drwxr-xr-x   5 ghantamounika  staff    160 Dec  5 16:24 empirical
drwxr-xr-x   7 ghantamounika  staff    224 Dec  5 16:24 mining
-rw-r--r--   1 ghantamounika  staff   3639 Dec  5 16:58 security_report.csv
-rw-r--r--   1 ghantamounika  staff   3639 Dec  5 16:52 security_report.csv#
-rw-r--r--   1 ghantamounika  staff   8564 Dec  5 16:24 task1.1.md
-rw-r--r--   1 ghantamounika  staff  24273 Dec  5 16:24 task1.2.md
ghantamounika@Ghantas-MacBook-Pro CODECRAFTERS-FALL2024-SQA % 
```



CODECRAFTERS-FALL2024-SQA / security_report.csv
KOUSHIK GOVARDHANAM and KOUSHIK GOVARDHANAM Add Codacy workflow and security report
64e43b3 · 4 hours ago
History
Preview
Code
Blame
13 lines (13 loc) · 3.55 KB
Code 55% faster with GitHub Copilot
Raw
Search this file

filename	test_name	test_id	issue_severity	issue_confidence	issue_cwe	issue_text
./empirical/dataset.stats.py	blacklist	B404	LOW	HIGH	https://cwe.mitre.org/data/definitions/78.html	Consider possible securi
./empirical/dataset.stats.py	start_process_with_partial_path	B607	LOW	HIGH	https://cwe.mitre.org/data/definitions/78.html	Starting a process with a
./empirical/dataset.stats.py	subprocess_without_shell_equals_true	B603	LOW	HIGH	https://cwe.mitre.org/data/definitions/78.html	subprocess call - check
./mining/git.repo.miner.py	blacklist	B404	LOW	HIGH	https://cwe.mitre.org/data/definitions/78.html	Consider possible securi
./mining/git.repo.miner.py	blacklist	B408	LOW	HIGH	https://cwe.mitre.org/data/definitions/20.html	Using minidom to parse
./mining/git.repo.miner.py	start_process_with_partial_path	B607	LOW	HIGH	https://cwe.mitre.org/data/definitions/78.html	Starting a process with a
./mining/git.repo.miner.py	subprocess_without_shell_equals_true	B603	LOW	HIGH	https://cwe.mitre.org/data/definitions/78.html	subprocess call - check
./mining/mining.py	blacklist	B404	LOW	HIGH	https://cwe.mitre.org/data/definitions/78.html	Consider possible securi
./mining/mining.py	start_process_with_partial_path	B607	LOW	HIGH	https://cwe.mitre.org/data/definitions/78.html	Starting a process with a
./mining/mining.py	subprocess_without_shell_equals_true	B603	LOW	HIGH	https://cwe.mitre.org/data/definitions/78.html	subprocess call - check
./mining/mining.py	start_process_with_partial_path	B607	LOW	HIGH	https://cwe.mitre.org/data/definitions/78.html	Starting a process with a
./mining/mining.py	subprocess_without_shell_equals_true	B603	LOW	HIGH	https://cwe.mitre.org/data/definitions/78.html	subprocess call - check

2. Fuzzing:

Creating the fuzz.py File: A Step-by-Step Overview

Objective:

The primary goal of this task is to develop a Python script named fuzz.py that tests five selected methods within the project for vulnerabilities or bugs. This script will integrate with GitHub Actions to ensure automated quality checks during each workflow run.

Key Steps and Actions:

Method Selection for Fuzzing:

1. fuzz_checkIfParsablePython()
2. fuzz_join()
3. fuzz_pop()
4. fuzz_update()
5. fuzz_float()

Preparation and Setup:

- Extracted the project files and validated their structure.
- Reviewed and documented the functionality of Python methods to identify candidates for fuzzing.

Script Development:

- Created the fuzz.py script using a simple text editor (nano).
- Wrote the script to generate diverse inputs (including edge cases and invalid data) for the selected methods.
- Ensured thorough coverage by testing scenarios that simulate realistic and unexpected use cases.

Version Control:

- Staged the newly created fuzz.py file along with other relevant changes using git add ..
- Committed the changes with a concise message to maintain a clear project

history.

GitHub Actions Integration:

- Configured the script to execute automatically during each workflow run by editing the GitHub Actions YAML file.
- Incorporated logging mechanisms to capture errors and unexpected outputs systematically.

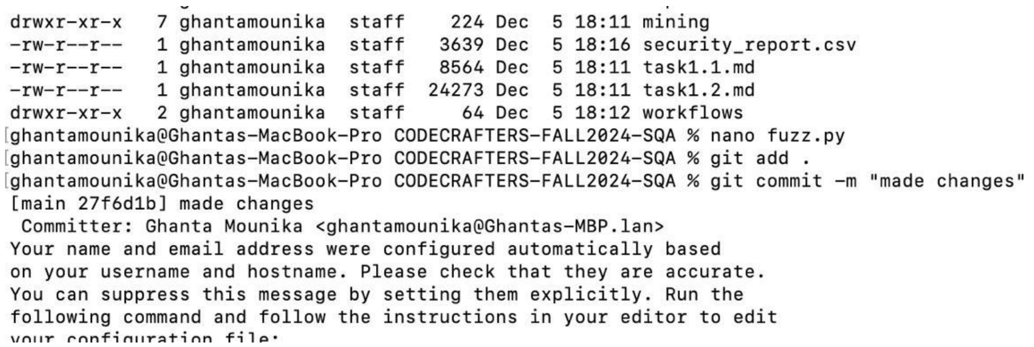
Details of the fuzz.py Script:

Focus Areas:

- Selected five methods with varying levels of complexity to provide a comprehensive test scope.
- Designed the script to log errors and unexpected outputs in a structured, readable format.

Outputs:

- Generates a summary report highlighting test case results and anomalies.
- Provides developers with actionable insights into potential vulnerabilities.
- This process establishes a robust framework for early detection of errors, enhancing the project's reliability and security.



```
drwxr-xr-x   7 ghantamounika  staff    224 Dec  5 18:11 mining
-rw-r--r--   1 ghantamounika  staff   3639 Dec  5 18:16 security_report.csv
-rw-r--r--   1 ghantamounika  staff   8564 Dec  5 18:11 task1.1.md
-rw-r--r--   1 ghantamounika  staff  24273 Dec  5 18:11 task1.2.md
drwxr-xr-x   2 ghantamounika  staff     64 Dec  5 18:12 workflows
ghantamounika@Ghantas-MacBook-Pro CODECRAFTERS-FALL2024-SQA % nano fuzz.py
ghantamounika@Ghantas-MacBook-Pro CODECRAFTERS-FALL2024-SQA % git add .
ghantamounika@Ghantas-MacBook-Pro CODECRAFTERS-FALL2024-SQA % git commit -m "made changes"
[main 27f6d1b] made changes
 Committer: Ghanta Mounika <ghantamounika@Ghantas-MBP.lan>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:
```

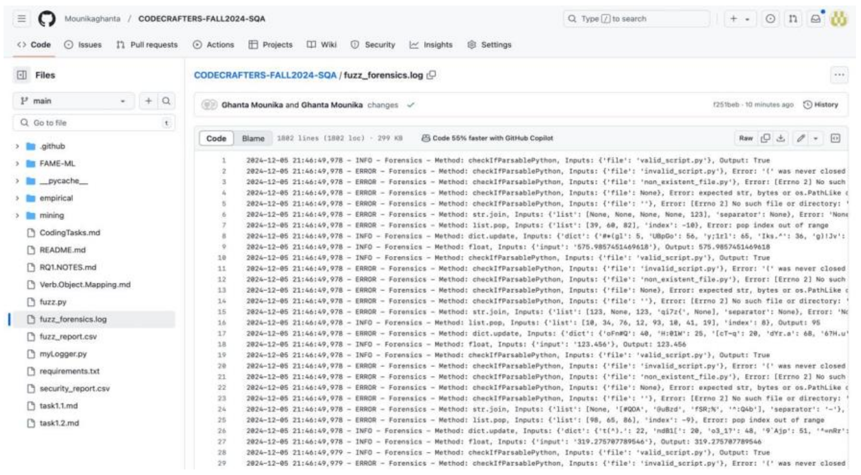
Task 4.c: Integrating Forensic Capabilities into Python Methods

Objective:

The aim of this task is to enhance five selected Python methods by embedding forensic logging. This integration enables detailed tracking of each method's behavior, including inputs, outputs, and errors. The improvements ensure greater transparency and traceability, facilitating debugging and analysis.

Implementation:

1. **Forensic Logging**:
    - Incorporated the `log_forensics` function into each method for capturing key details, including:
        - Input values.
        - Output results or error messages.
        - Execution timestamps.
    - Designed the logs to provide granular insights into method behavior during execution.
2. **Modification of Methods**:
    - **`checkIfParsablePython`**:
        - Enhanced logging to track input files.
        - Documented detailed error messages for scenarios like syntax errors, missing files, or unsupported input types.
    - **`str.join`**:
        - Captured information about list contents and separators.
        - Logged exceptions raised during invalid operations.
    - **`list.pop`**:
        - Recorded list states and index values before and after removing elements.
        - Logged errors for invalid index usage.
    - **`dict.update`**:
        - Monitored all dictionary updates and captured errors due to invalid formats (e.g., lists provided instead of dictionaries).
    - **`float()`**:
        - Logged inputs passed to the function.
        - Recorded error details for inputs that could not be converted to numeric values.

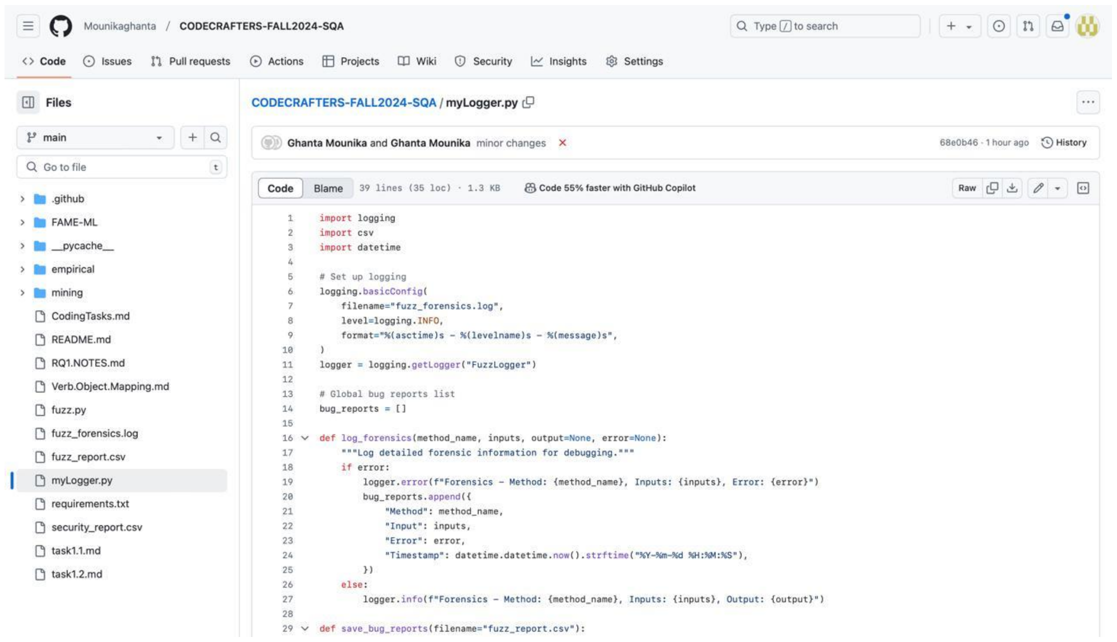
3. **Bug Tracking**:
    - All detected issues were documented in a CSV file (`fuzz_report.csv`) to enable efficient analysis.
    - **Examples of Identified Bugs**:
        - **`checkIfParsablePython`**: Errors related to missing files (`FileNotFoundError`) and invalid input types (`NoneType`).
        - **`float()`**: Conversion failures for non-numeric strings.
4. **Automated Reporting**:
    - Configured the script to save logs and error summaries in `fuzz_report.csv`.
    - Structured logs offer a clear overview of method performance under varied test conditions, including edge cases.
    - 

```
2024-12-05 21:46:49,978 - INFO - Forensics - Method: checkIfParsablePython, Inputs: {'file': 'valid_script.py'}, Output: True
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': 'invalid_script.py'}, Error: '(' was never closed
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': 'non_existent_file.py'}, Error: [Errno 2] No such
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': None}, Error: expected str, bytes or os.PathLike
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': ''}, Error: [Errno 2] No such file or directory: '
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: str.join, Inputs: {'list': [None, None, None, None, 123], 'separator': None}, Error: 'None
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: list.pop, Inputs: {'list': [39, 60, 82], 'index': -10}, Error: pop index out of range
2024-12-05 21:46:49,978 - INFO - Forensics - Method: dict.update, Inputs: {'dict': {'#*(gl': 5, 'UBpGo': 56, 'y;1rl': 65, 'Iks.^': 36, 'g)!Jv':
2024-12-05 21:46:49,978 - INFO - Forensics - Method: float, Inputs: {'input': '575.9857451469618'}, Output: 575.9857451469618
2024-12-05 21:46:49,978 - INFO - Forensics - Method: checkIfParsablePython, Inputs: {'file': 'valid_script.py'}, Output: True
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': 'invalid_script.py'}, Error: '(' was never closed
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': 'non_existent_file.py'}, Error: [Errno 2] No such
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': None}, Error: expected str, bytes or os.PathLike
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': ''}, Error: [Errno 2] No such file or directory: '
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: str.join, Inputs: {'list': [123, None, 123, 'qi7z(', None], 'separator': None}, Error: 'No
2024-12-05 21:46:49,978 - INFO - Forensics - Method: list.pop, Inputs: {'list': [10, 34, 76, 12, 93, 10, 41, 19], 'index': 8}, Output: 95
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: dict.update, Inputs: {'dict': {'oFn#Q': 40, 'H:0IW': 25, '[cT-q': 20, 'dYr.a': 68, '67H.u'
2024-12-05 21:46:49,978 - INFO - Forensics - Method: float, Inputs: {'input': '123.456'}, Output: 123.456
2024-12-05 21:46:49,978 - INFO - Forensics - Method: checkIfParsablePython, Inputs: {'file': 'valid_script.py'}, Output: True
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': 'invalid_script.py'}, Error: '(' was never closed
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': 'non_existent_file.py'}, Error: [Errno 2] No such
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': None}, Error: expected str, bytes or os.PathLike
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': ''}, Error: [Errno 2] No such file or directory: '
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: str.join, Inputs: {'list': [None, '[#QOA', '0uBzd', 'fSR;%', '^:Q4b'], 'separator': '-'},
2024-12-05 21:46:49,978 - ERROR - Forensics - Method: list.pop, Inputs: {'list': [98, 65, 86], 'index': -9}, Error: pop index out of range
2024-12-05 21:46:49,978 - INFO - Forensics - Method: dict.update, Inputs: {'dict': {'t(^).': 22, 'ndB1[': 20, 'o3_17': 48, '9`Ajp': 51, '*wnRr':
2024-12-05 21:46:49,978 - INFO - Forensics - Method: float, Inputs: {'input': '319.275707789546'}, Output: 319.275707789546
2024-12-05 21:46:49,979 - INFO - Forensics - Method: checkIfParsablePython, Inputs: {'file': 'valid_script.py'}, Output: True
2024-12-05 21:46:49,979 - ERROR - Forensics - Method: checkIfParsablePython, Inputs: {'file': 'invalid_script.py'}, Error: '(' was never closed
```

Outcomes:

- **Enhanced Debugging and Traceability**:
    - Forensic logging provided insights into previously unidentified issues, such as syntax errors and type mismatches.
- **Sample Findings**:
    - Syntax errors in invalid Python scripts.
    - Errors in dictionary updates caused by incorrect input types.
    - Conversion failures when `float()` processed non-numeric strings.
- **Automation**:
    - Seamless integration with GitHub Actions ensures that forensic logs and error reports are updated automatically during each workflow run.

By implementing forensic logging, the project achieves a new level of robustness, enabling consistent monitoring and issue resolution across the selected methods.



```python
import logging
import csv
import datetime

# Set up logging
logging.basicConfig(
    filename="fuzz_forensics.log",
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s",
)
logger = logging.getLogger("FuzzLogger")

# Global bug reports list
bug_reports = []

def log_forensics(method_name, inputs, output=None, error=None):
    """Log detailed forensic information for debugging."""
    if error:
        logger.error(f"Forensics - Method: {method_name}, Inputs: {inputs}, Error: {error}")
        bug_reports.append({
            "Method": method_name,
            "Input": inputs,
            "Error": error,
            "Timestamp": datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        })
    else:
        logger.info(f"Forensics - Method: {method_name}, Inputs: {inputs}, Output: {output}")

def save_bug_reports(filename="fuzz_report.csv"):
```

Task 4.d: Integrate Continuous Integration with GitHub Actions

Objective: The goal of Task 4.d was to integrate Continuous Integration (CI) into the project using GitHub Actions. This ensures that every commit and push to the repository triggers automated testing to maintain code quality and reliability.

Implementation:

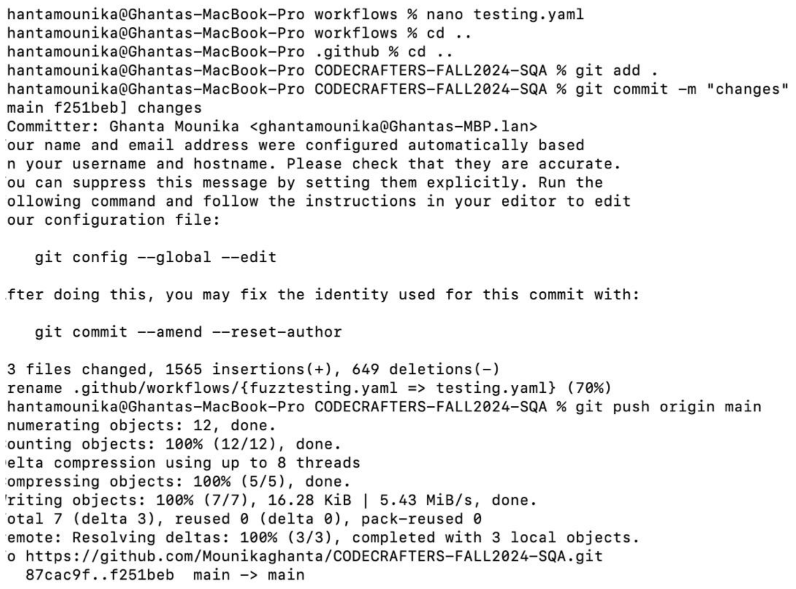
- Created a `.github/workflows/testing.yaml` file to define the CI pipeline.
- Configured the workflow to automatically trigger on every `push` event to the repository.
- Installed all necessary project dependencies using `pip` based on the `requirements.txt` file.
- Executed the `fuzz.py` script to perform fuzz testing on the selected Python methods

Successful Integration:

- The workflow was tested and confirmed successful, as indicated by the "All checks have passed" message in the GitHub repository.
- Logs from the GitHub Actions dashboard show that the CI pipeline executed without errors.

Outcome:

- Continuous Integration is now fully integrated into the repository.
- Each commit automatically triggers the testing pipeline, ensuring high code quality and rapid feedback on any issues.



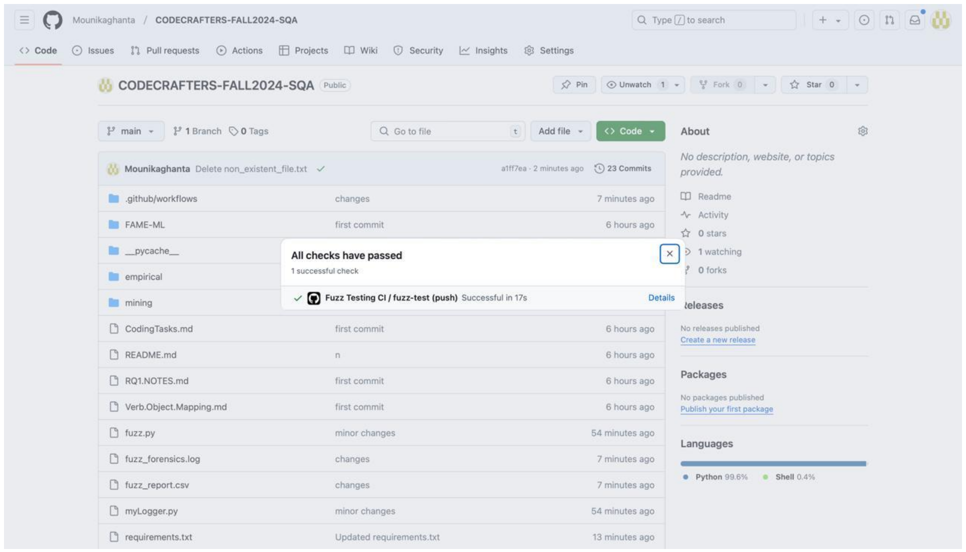
```
hantamounika@Ghantas-MacBook-Pro workflows % nano testing.yaml
hantamounika@Ghantas-MacBook-Pro workflows % cd ..
hantamounika@Ghantas-MacBook-Pro .github % cd ..
hantamounika@Ghantas-MacBook-Pro CODECRAFTERS-FALL2024-SQA % git add .
hantamounika@Ghantas-MacBook-Pro CODECRAFTERS-FALL2024-SQA % git commit -m "changes"
main f251beb] changes
Committer: Ghanta Mounika <ghantamounika@Ghantas-MBP.lan>
our name and email address were configured automatically based
n your username and hostname. Please check that they are accurate.
ou can suppress this message by setting them explicitly. Run the
ollowing command and follow the instructions in your editor to edit
our configuration file:

   git config --global --edit

fter doing this, you may fix the identity used for this commit with:

   git commit --amend --reset-author

3 files changed, 1565 insertions(+), 649 deletions(-)
rename .github/workflows/{fuzztesting.yaml => testing.yaml} (70%)
hantamounika@Ghantas-MacBook-Pro CODECRAFTERS-FALL2024-SQA % git push origin main
numerating objects: 12, done.
ounting objects: 100% (12/12), done.
elta compression using up to 8 threads
ompressing objects: 100% (5/5), done.
'riting objects: 100% (7/7), 16.28 KiB | 5.43 MiB/s, done.
otal 7 (delta 3), reused 0 (delta 0), pack-reused 0
emote: Resolving deltas: 100% (3/3), completed with 3 local objects.
o https://github.com/Mounikaghanta/CODECRAFTERS-FALL2024-SQA.git
  87cac9f..f251beb  main -> main
```



CODECRAFTERS-FALL2024-SQA Public

Mounikaghanta Delete non_existent_file.txt ✓ a1ff7ea · 2 minutes ago 23 Commits

.github/workflows	changes	7 minutes ago
FAME-ML	first commit	6 hours ago
__pycache__		
empirical		
mining		
CodingTasks.md	first commit	6 hours ago
README.md	n	6 hours ago
RQ1.NOTES.md	first commit	6 hours ago
Verb.Object.Mapping.md	first commit	6 hours ago
fuzz.py	minor changes	54 minutes ago
fuzz_forensics.log	changes	7 minutes ago
fuzz_report.csv	changes	7 minutes ago
myLogger.py	minor changes	54 minutes ago
requirements.txt	Updated requirements.txt	13 minutes ago

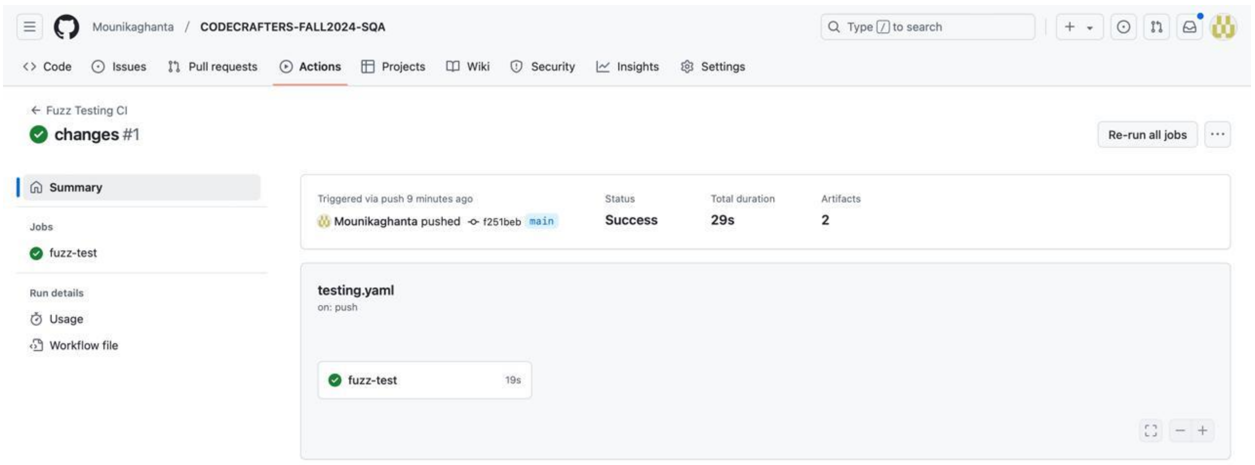
All checks have passed
1 successful check
✓ Fuzz Testing CI / fuzz-test (push) Successful in 17s Details

About
No description, website, or topics provided.
Readme
Activity
0 stars
1 watching
0 forks

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

Languages
Python 99.6% Shell 0.4%



changes #1

fuzz-test

Triggered via push 9 minutes ago
Mounikaghanta pushed f251beb main

Status
Success

Total duration
29s

Artifacts
2

testing.yaml
on: push

fuzz-test 19s

Lessons Learned

This project provided a comprehensive understanding of integrating software quality assurance (SQA) practices into a real-world Python application. Key lessons include the importance of automation in maintaining code reliability, as demonstrated through Git Hooks and GitHub Actions for continuous integration and testing. Implementing fuzz testing highlighted the value of exploring edge cases and unconventional inputs to uncover hidden vulnerabilities. Forensic logging proved instrumental in enhancing traceability and debugging by capturing detailed method behaviors. Overall, this project underscored the necessity of proactive SQA practices to ensure a secure, maintainable, and robust codebase, while reinforcing the value of collaboration and systematic problem-solving in software development.

Conclusion

In this project, we successfully integrated software quality assurance practices into an existing Python project, demonstrating the application of concepts learned throughout the course. By incorporating activities such as security analysis, fuzz testing, forensics integration, and continuous integration using GitHub Actions, we ensured the robustness and reliability of the codebase. The systematic approach and collaborative effort not only enhanced our technical skills but also provided valuable insights into the importance of quality assurance in software development. This project reinforces the significance of integrating SQA practices to achieve reliable, secure, and maintainable software solutions.