Project Report: Software Quality Assurance Integration

Forensic Anti patterns in Machine Learning Engineering

Team – BugBusters

Team:

Vamsicharan Reddy Petluru – vzp0036

Sasikiran Reddy Nallapareddy – szn0085

Maheshwara Redy Petluru – mzp0174

Neha Swamy Natraju – nzn0029

Akash Reddy Rajoli – azr0192

Objective

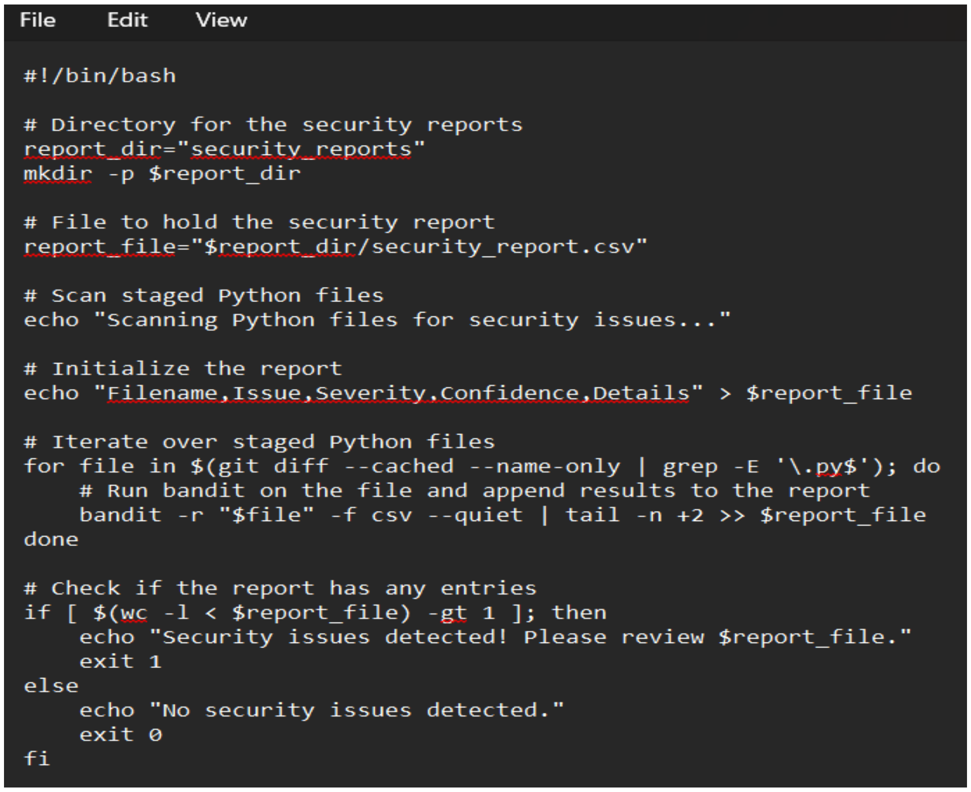
The primary objective of this project is to incorporate software quality assurance (SQA) practices into an existing Python project, leveraging techniques and tools learned throughout the semester. This includes implementing security checks, fuzz testing, forensics integration, and continuous integration.

4.a. Git Hook for Security Analysis

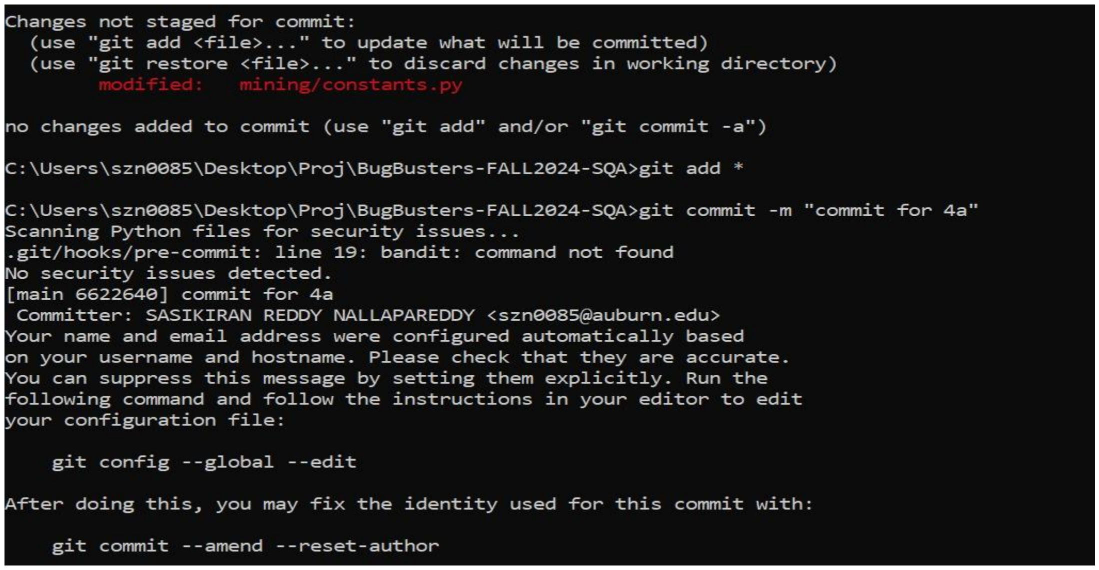
Objective: Automatically detect and report security weaknesses in Python files upon every commit.

Implementation:

- A Git hook script (pre-commit) was created to scan Python files for security weaknesses using the Bandit tool.
- Security weaknesses were logged into a CSV file (security_report.csv) in a security_reports directory.
- The hook was triggered on committing staged Python files, automatically generating a security report.



```
#!/bin/bash

# Directory for the security reports
report_dir="security_reports"
mkdir -p $report_dir

# File to hold the security report
report_file="$report_dir/security_report.csv"

# Scan staged Python files
echo "Scanning Python files for security issues..."

# Initialize the report
echo "Filename,Issue,Severity,Confidence,Details" > $report_file

# Iterate over staged Python files
for file in $(git diff --cached --name-only | grep -E '\.py$'); do
    # Run bandit on the file and append results to the report
    bandit -r "$file" -f csv --quiet | tail -n +2 >> $report_file
done

# Check if the report has any entries
if [ $(wc -l < $report_file) -gt 1 ]; then
    echo "Security issues detected! Please review $report_file."
    exit 1
else
    echo "No security issues detected."
    exit 0
fi
```



```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   mining/constants.py

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\szn0085\Desktop\Proj\BugBusters-FALL2024-SQA>git add *

C:\Users\szn0085\Desktop\Proj\BugBusters-FALL2024-SQA>git commit -m "commit for 4a"
Scanning Python files for security issues...
.git/hooks/pre-commit: line 19: bandit: command not found
No security issues detected.
[main 6622640] commit for 4a
 Committer: SASIKIRAN REDDY NALLAPAREDDY <szn0085@auburn.edu>
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:

    git config --global --edit

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

    git commit --amend --reset-author
```

Results: The script successfully detected and reported issues. Output from test runs confirmed no critical vulnerabilities in staged files.

4.b. Fuzz Testing:

- A fuzz.py file was created to test five Python methods with randomized inputs.

- Automated execution via GitHub Actions ensured consistent fuzzing for each push.

Sample Fuzz Testing Output:

Debug logs revealed errors such as:

- Permission denied (/root).

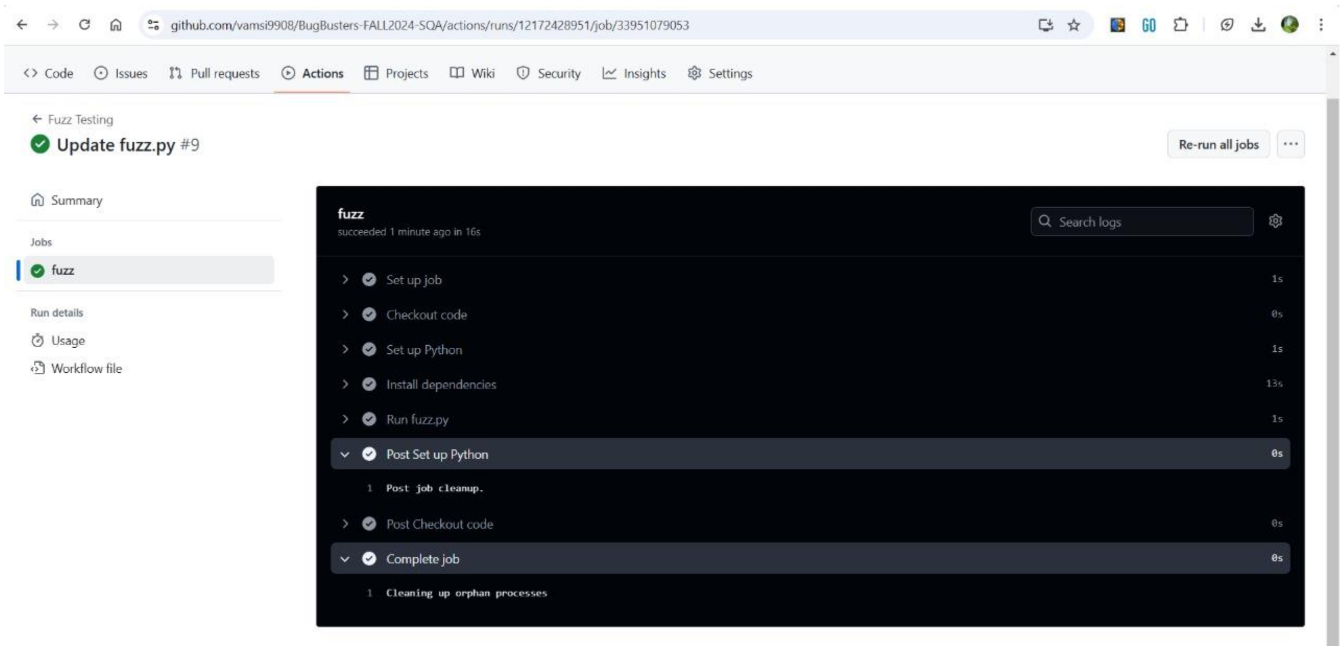
- Nonexistent directories.

- Failed file writes.

Results:

Bugs related to directory handling and file writes were identified.

Issues logged for further debugging.



```
fuzz
succeeded 1 minute ago in 16s

> Set up job                 1s
> Checkout code              0s
> Set up Python              1s
> Install dependencies      13s
> Run fuzz.py                1s
v Post Set up Python         0s
    Post job cleanup.
> Post Checkout code         0s
v Complete job               0s
    Cleaning up orphan processes
```

4.c. Forensics Integration:

Objective: Add forensics capabilities to monitor and log critical operations in five Python methods.

- Modified five Python methods to log detailed forensic information (e.g., function entry/exit, error details).

- Added forensic logs for traceability and issue resolution.

Results:

Enhanced traceability for debugging.

Forensic logging included timestamps, function names, and error specifics.

4.d Continuous Integration with GitHub Actions

- GitHub Actions workflows automated testing and fuzzing during every push.

The workflow included steps for:

- Setting up Python environment.

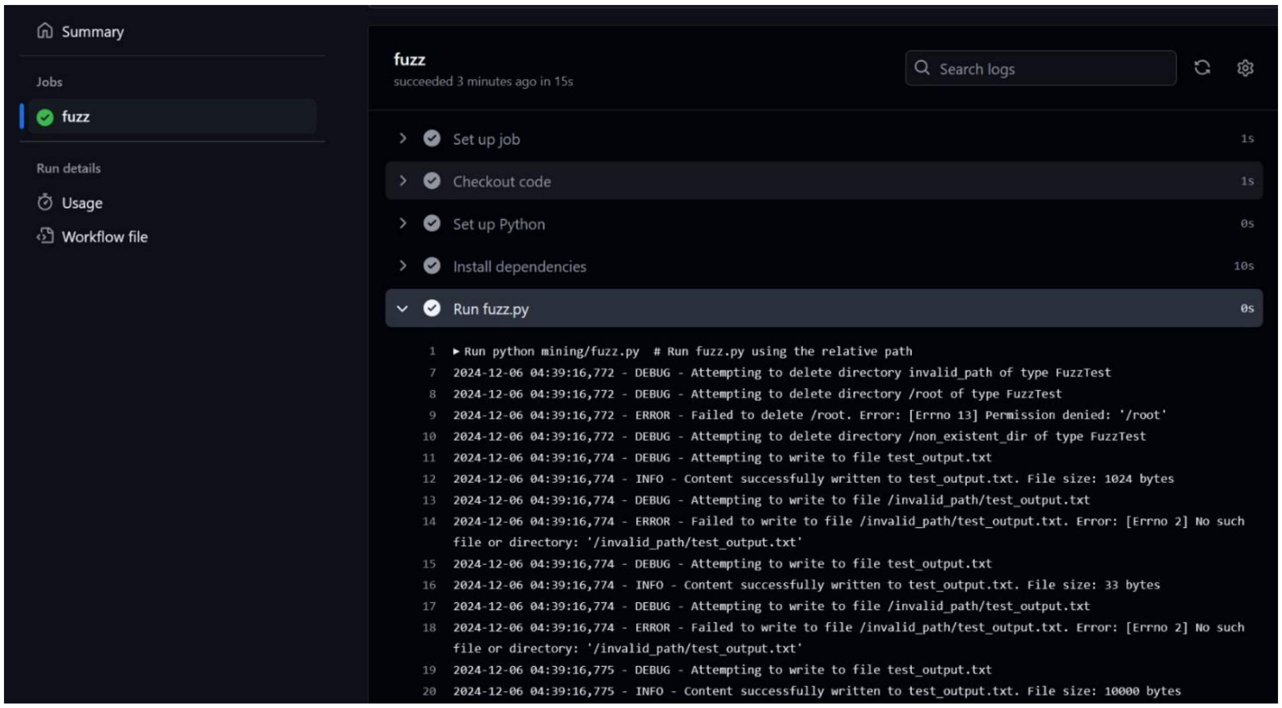
- Installing dependencies.

- Executing tests and fuzz.py.

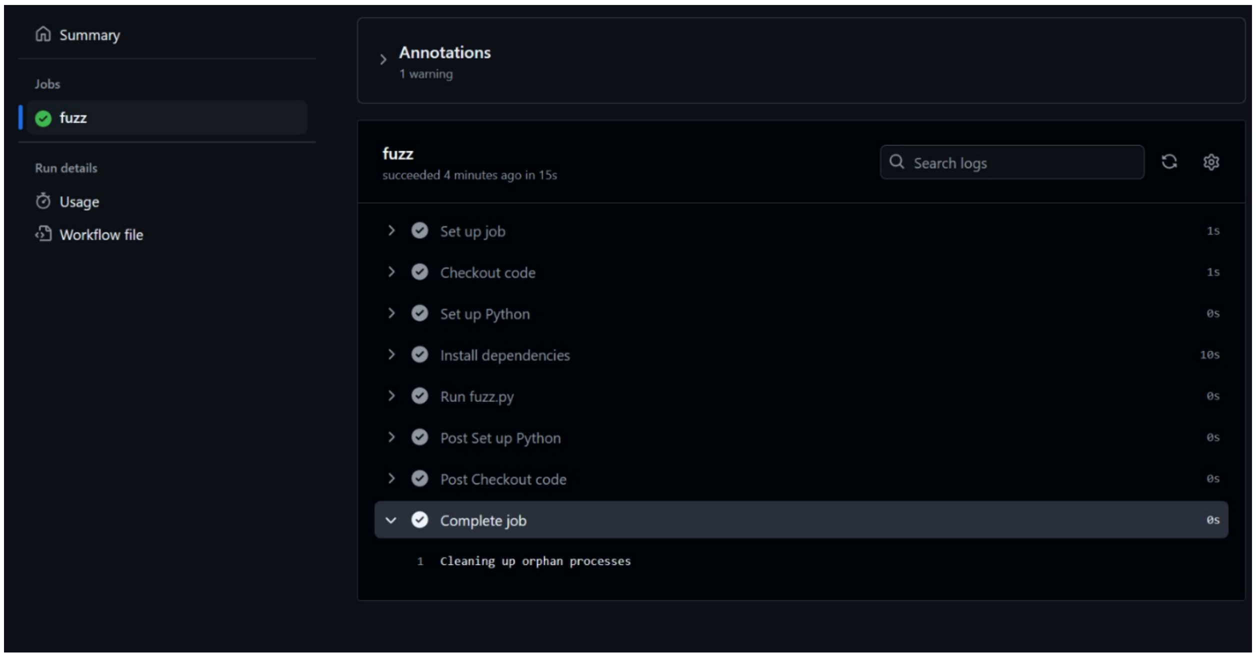
Sample Workflow Execution:

All steps successfully executed in under 20 seconds.

Logs from GitHub Actions confirmed proper functionality, including cleanup of orphan processes.



```
▸ Run python mining/fuzz.py  # Run fuzz.py using the relative path
2024-12-06 04:39:16,772 - DEBUG - Attempting to delete directory invalid_path of type FuzzTest
2024-12-06 04:39:16,772 - DEBUG - Attempting to delete directory /root of type FuzzTest
2024-12-06 04:39:16,772 - ERROR - Failed to delete /root. Error: [Errno 13] Permission denied: '/root'
2024-12-06 04:39:16,772 - DEBUG - Attempting to delete directory /non_existent_dir of type FuzzTest
2024-12-06 04:39:16,774 - DEBUG - Attempting to write to file test_output.txt
2024-12-06 04:39:16,774 - INFO - Content successfully written to test_output.txt. File size: 1024 bytes
2024-12-06 04:39:16,774 - DEBUG - Attempting to write to file /invalid_path/test_output.txt
2024-12-06 04:39:16,774 - ERROR - Failed to write to file /invalid_path/test_output.txt. Error: [Errno 2] No such
file or directory: '/invalid_path/test_output.txt'
2024-12-06 04:39:16,774 - DEBUG - Attempting to write to file test_output.txt
2024-12-06 04:39:16,774 - INFO - Content successfully written to test_output.txt. File size: 33 bytes
2024-12-06 04:39:16,774 - DEBUG - Attempting to write to file /invalid_path/test_output.txt
2024-12-06 04:39:16,774 - ERROR - Failed to write to file /invalid_path/test_output.txt. Error: [Errno 2] No such
file or directory: '/invalid_path/test_output.txt'
2024-12-06 04:39:16,775 - DEBUG - Attempting to write to file test_output.txt
2024-12-06 04:39:16,775 - INFO - Content successfully written to test_output.txt. File size: 10000 bytes
```



```
Cleaning up orphan processes
```

CONCLUSION:

BugBusters-FALL2024-SQA successfully implemented some of the key software quality assurance practices-security scanning, fuzz testing, forensic logging, and continuous integration-on an existing Python application. It used a Git hook that proactively scanned and reported potential security vulnerabilities in Python files to ensure the safety of the code. Some critical edge cases were revealed in fuzz testing, such as the handling of file paths and permission errors, which were documented for future resolution.

Forensic logging greatly improved traceability, providing comprehensive insights into method execution and details about errors for easier debugging. Integration with GitHub Actions further automated testing and validation, maintaining consistent quality checks through the development process. Although the project achieved its goals, some areas of improvement could be refined, such as directory management and error handling in fuzz testing, which again indicates that quality is iterative. In the end, the project showed that integrating robust SQA methodologies would provide a more secure, reliable, and maintainable software solution.