Software Quality Assurance Final Project Report

Team Members: Trey Gaines, Caleb St.Germain

Team: Team24

Github repository: https://github.com/Calebstg/TEAM24-SQA2023-AUBURN/tree/main

Report

- The project involved a collective effort to conduct software forensics using logging. This was achieved through editing and introducing logging to 5 different Python methods in the repository. The edited methods are listed at the bottom, along with the generated log. Through this exercise, we gained knowledge about software forensics and how to implement logging in Python. We also gained insights into various types of attacks that logging can help prevent and detect, such as poisoning attacks and model tricking attacks.

- The process of implementing logging in Python methods allowed us to gain a deeper understanding of how software forensics works. We learned how to detect and prevent attacks using logging, which is an essential tool for identifying potential vulnerabilities and malicious activities in software applications. By editing these methods and creating a log, we were able to obtain valuable information about the functionality of the application, such as user activity and system behavior. Overall, this project helped us to develop our skills in software forensics and improved our understanding of how logging can be used to secure software applications.

- In addition to editing Python methods and implementing logging, we also added a static analysis security git hook to the repository. This hook runs the Bandit tool on all files in the repository during the commit process and generates a report in a file named "results.csv." This file is automatically added to the commit, providing a convenient way to track changes and monitor security. Through this exercise, we gained a better understanding of how git hooks work and how they can be used to enforce security policies in a software development project.

- By implementing these security measures, we learned about the importance of continuous integration in ensuring software security. The combination of static analysis security git hooks and logging helped us to identify and prevent potential security threats in the repository. This project helped us to develop our skills in software security and provided valuable experience in implementing security measures for software applications.

- As part of the project, we also performed 5 different fuzzing methods of our choice, which were implemented in the fuzz.py file. The output of the fuzzing was recorded in the fuzzingOutput.txt file. Through this exercise, we gained an understanding of the significance of identifying and addressing software vulnerabilities. Fuzzing involves testing a software system by providing it with unexpected or random input values to observe how it responds. This technique helped us recognize the importance of proper input validation to ensure that the software runs correctly.

- The results of the fuzzing methods in the fuzzingOutput.txt file showed the various errors that occurred due to unexpected inputs. This highlighted the importance of fuzzing in software development and emphasized the need for developers to have a failsafe in case unexpected inputs occur. By performing these fuzzing techniques, we gained valuable experience in identifying and addressing software vulnerabilities, helping us to improve our skills in software development and security. In addition, we also ran a Codacy Static Analysis.

Security Analysis Git Hook:

- Generated report log: https://github.com/Calebstg/TEAM24-SQA2023-AUBURN/blob/main/results.csv

Fuzzing:

- https://github.com/Calebstg/TEAM24-SQA2023-AUBURN/blob/main/fuzzingOutput.txt

Software Forensics:

- Generated report log: https://github.com/Calebstg/TEAM24-SQA2023-AUBURN/blob/main/SIMPLE-LOGGER.log
- Edited python file: https://github.com/Calebstg/TEAM24-SQA2023-AUBURN/blob/main/KubeSec-master/scanner.py