Project Report - WDESQA

Garrett Flowers, Houston Huggins, Patrick King, Walter Swan

[Project GitHub Link](https://github.com/Garrett12123/WDESQA-SQA2023-AUBURN)

Garrett - Developers can learn a lot about potential security flaws in their code by utilizing Git hooks and the Python script Bandit. Git hooks are programs that start or stop automatically before or after specific Git operations, like committing code changes. Because of how much I learnt about automating security checks on repositories, I really loved working on this aspect of the project. I'll be able to use Bandit a lot in the future.

Houston -

I primarily worked on part 4B, the fuzz.py file. I learned more about fuzzing and the extra experience in this strategy will be a valuable asset as I move into my career.

Patrick - I primarily worked on part 4C. I had to integrate forensics in 5 methods of our choice like we had to do in workshop 7. I chose 5 methods in the scanner.py file and implemented logging statements in those methods. With 4C being similar to workshop 7 I was able to have more practice and learn more about implementing logging statements while doing this part of the project. I also have not uploaded code to GitHub very often so it was nice to get more practice with that. I tend to forget the commands for git since I do not use it very often and more practice is always handy.

Walter -

I primarily worked on the ReadMe.md, project report, and general organization of the project. I learned a lot about teamwork, organization, and formatting principles on GitHub. Additionally, having a group project where we each push our own GitHub commits provided good hands-on experience with the website and process to learn from.

**Errors found by fuzzing:**

In the scanner.py file, the isValidUserName method will allow a username of “ “. The isValidPasswordName method also allows it. While this is not technically an error, it is usually inappropriate to have a blank space as a username or password.

In the graphtaint.py file, the getYAMLFiles and getSHFiles methods return an empty pair of brackets instead of any kind of error message when the string “errant message” is inputted. Both are expecting the path to directories. While this also isn’t technically an error, it does demonstrate that these methods lack complete input validation.

Also in the graphtaint.py file, the readBashAsStr method returns a FileNotFoundError if the string “errant input” is inputted instead of the path\_sh\_script value that is expected. This shows that there is incomplete input handling for the readBashAsStr method.