Spir-fuzz is a C++-based tool that automatically finds bugs in Vulkan drivers. It works by transforming the original shader into a new one that is semantically the same. Differences in the output of the new shader and the original one can be caused by bugs in the driver. Thus, my earlier on tasks include expanding the set of transformations by building more transformation classes and writing their corresponding tests and fuzzer passes and provide additional support to the SPIRV-tools repository:

SPIRV-Tools support

Transformations and fuzzer passes:

- #4326: Swap positions of two functions in a module
- #4376: Wrap Scalar operation into vector operation

Improvements and additional tests:

- #4253: Fix underflow problems in fuzzer pass
- #4304: Add tests for full coverage of access chain transformation
- #4284: Add tests for MaybeGet* functions in fuzzer utils

Our main task involves the WebGPU Shading Language, a new shading language featured by WebGPU. Since web browsers will have WebGPU, a secure implementation is crucial. To achieve a high test coverage, we use coverage-guided fuzzing. It uses program instrumentation to trace the code coverage reached by each input fed to a fuzz target. The information is then used to make informed decisions that maximize coverage, and thus increase the effectiveness of finding software bugs and security vulnerabilities. This project involves automatic fuzzing using LibFuzzer. Since LibFuzzer-based custom mutators mutate test cases in a domain-specific way, effective designing and implementing Tint-specific custom mutators are essential for this project to succeed, where Tint is a compiler for WGSL. My role specifically is to contribute to Tint AST (stands for abstract syntax tree) by providing different mutation classes and mutators (which are similar to the transformation classes and fuzzer passes for SPIRV-Tools) that introduce either semantic preserving or non-semantic preserving mutations of input. By expanding the set of mutations of the program, it is expected to increase the chance for us to find vulnerabilities and bugs in webGPU:

Tint AST

Issues:

- #1108: Wrap statement in if conditional
- #1110: Delete a statement
- #1111: Wrap unary operator

- #1119: Shuffle function parameters
- #1120: Transform if statement to for loops
- #1126: Duplicate statement
- #1127: Extend/contract the number of iterations for for loop statement.

Mutation classes:

- #60820: Change unary expression operator
- #61620: Change binary expression operator
- #61900: Delete statement
- #62001: Change if conditional wrapper to for loop statement
- #61200: Wrap statement in if statement
- #62000: Wrap unary operators
- WIP:
    - Extend/ Contract loops (might not be possible currently)
    - Duplicate Statement
    - Shuffle function parameters

Improvements:

- #61100: Improved node id map and find mutators structure
- #61381: Removed const constraint from GetNode return type

Additional tasks also include provide examples for ShaderTrap, an OpenGL shader runner that runs self-contained .shadertrap files. Different variants of atomic reduction examples are attained by adjusting the number of work groups and number of invocations.

ShaderTrap

Provide code examples:

- #66: Reduction operation by addition
- #71: Reduction operation by min and max
- #72: Reduction operation by bitwise and, or and xor