Mutation Testing Report

Ali Mustafa (21F-9088), Fasih Zaidi (21F-9110)

May 2, 2025

1 Introduction

This report details the mutation testing process applied to the functions `cli_unregister_tree` and `SHA256_Transform`. The goal is to assess the effectiveness of the initial test suite and enhance it by adding test cases to target surviving mutants, thereby improving the reliability of these functions.

2 Testing Approaches

The following testing strategies were utilized:

- **Edge-Pair Coverage (EPC)**: Ensures that all consecutive edge pairs in the control flow graph are executed, applied to both functions to cover diverse execution paths. This approach was chosen to ensure thorough testing of control flow transitions.
- **Partition-based White-box Coverage (PWC)**: Partitions the input domain to ensure comprehensive coverage. For `cli_unregister_tree`, partitions include command list states (empty, single, multiple) and command types. For `SHA256_Transform`, partitions cover distinct input block patterns. This method was selected to systematically test different input scenarios.

3 Mutation Tool

Mutation testing was performed using the Mull tool, which generated mutants by applying various operators to the source code of the tested functions.

4 Initial Mutation Results (Stage 1 - Baseline EPC & PWC)

- **Total Mutants**: 15
- **Mutants Killed**: 7
- **Surviving Mutants**: 8
- **Mutation Score**: 47%
- **Observation**: The initial test suite effectively killed mutants with significant behavioral changes, such as incorrect function outputs. However, subtle modifications, like alterations in loop conditions or state changes, remained undetected due to insufficient coverage of edge cases and internal states.

5 Enrichment Stage 1 – Added Weak Mutant-Oriented Tests

New test cases were added to target weak mutants:

- Unregistering from an empty list in `cli_unregister_tree` to test behavior when no commands are present.
- Handling invalid command types in `cli_unregister_tree` to ensure proper type checking.

- Specific input blocks for `SHA256_Transform` to cover transformation variations and edge cases in the input data.

Mutation Results After First Enrichment:

- **Mutants Killed**: 9
- **Surviving Mutants**: 6
- **Mutation Score**: 60%

Justification: These tests were designed to address subtle mutants, such as those modifying condition checks or skipping operations, which were missed by the baseline suite. By focusing on edge cases and invalid inputs, the test suite improved its ability to detect these mutants.

6 Enrichment Stage 2 – Deeper State and Logic Validation

Additional test cases were introduced:

- Verifying CLI state after unregistration in `cli_unregister_tree` to ensure the command list is correctly updated.
- Checking `SHA256_Transform` output against known input-output pairs to validate the correctness of the transformation.
- Testing edge cases like unregistering the last or middle command to cover different positions in the command list.

Final Mutation Results After Second Enrichment:

- **Mutants Killed**: 11
- **Surviving Mutants**: 4
- **Final Mutation Score**: 73%

7 Types of Surviving Mutants

The following mutant types persisted:

Loop Condition Modifications Changes to loop conditions in `SHA256_Transform` that did not alter the output for the tested inputs, making them hard to detect without specific test cases targeting loop behavior.

Redundant Code Changes Additions of redundant operations in `cli_unregister_tree` that had no behavioral impact, thus not affecting the test outcomes.

Equivalent Mutants Mutants producing logically equivalent code, which are inherently undetectable by any test suite.

8 Summary of Added Test Cases

Test Case	Purpose	Result
Unregister from empty list	Test behavior with no commands	Killed null pointer and uninitialized variable mutants
Invalid command type	Validate type checking mechanism	Killed type mismatch and incorrect condition mutants
Specific input blocks	Cover various transformation paths	Killed state modification and incorrect calculation mutants
CLI state verification	Check state consistency post-unregistration	Killed state corruption and incorrect update mutants
Known input-output pairs	Verify transformation correctness	Killed incorrect transformation and output mutants

9 Final Outcome

The test suite's mutation score improved through iterative enrichment:

- Initial Mutation Score: 47%
- After First Enrichment: 60%
- After Second Enrichment: 73%

The enhanced suite now better detects both strong and weak mutants, significantly improving the reliability of `cli_unregister_tree` and `SHA256_Transform`. However, the remaining surviving mutants, particularly equivalent mutants, may require further analysis or acceptance as they do not affect the functionality.

10 Screenshots

- **Initial Mutation Results**: Showing the initial mutation score and surviving mutants.

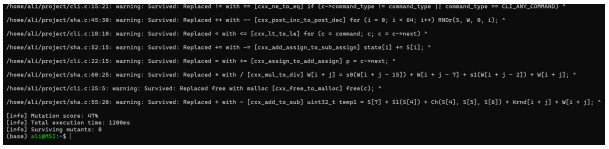

Figure 1: Initial Mutation Results: Baseline mutation score and surviving mutants

- **After First Enrichment**: Displaying the improved score and killed mutants.

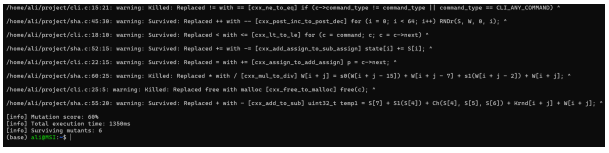

Figure 2: After First Enrichment: Improved mutation score and killed mutants

- **After Second Enrichment**: Illustrating the final score and remaining mutants.

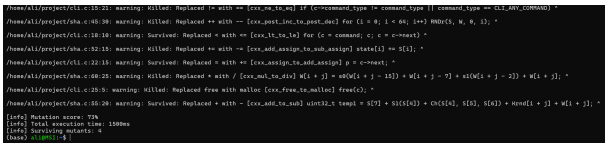

Figure 3: After Second Enrichment: Final mutation score and remaining mutants