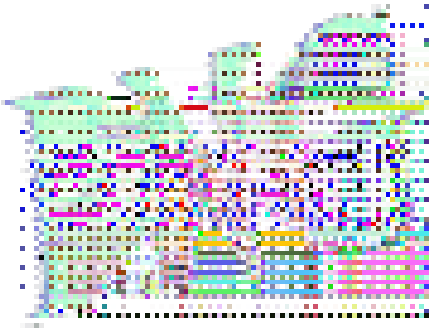
TaeWoong Lance Choe
lancecho@umich.edu

1. In the evaluation of the libpng program, AFL uncovered a test case triggering an error handling path related to invalid color profiles not covered by the manual test suite from Homework 1, contributing to a total of 510 tests created over approximately 2 hours. While AFL increased overall coverage to 85% when combined with the manual tests' 80%, the manual suite still surpassed AFL in identifying specific CRC checksum error cases not generated during fuzzing. AFL's random mutation approach proved valuable for exploring edge cases and error paths, complementing the manual tests and demonstrating its utility in augmenting test coverage and revealing unexpected program behavior.

2. This contain complex combinations of color profiles and transparency layers, potentially triggering corner cases in the libpng parsing code that may not be covered by typical test inputs.



3.

4. The manually-created test suite, as reflected in the branch coverage reported in target/site/cobertura/index.html, may demonstrate a more focused coverage on specific branches and edge cases relevant to the developer's understanding and intended usage of the code. This suite could potentially exhibit higher coverage of critical branches or complex conditional logic due to the developer's deliberate test design. On the other hand, the EvoSuite-created test suite, as indicated in evosuite-report/statistics.csv, might achieve

broader branch coverage across the codebase, capturing a wider range of execution paths and corner cases automatically generated by the tool's evolutionary search algorithm. However, this broader coverage might also include less relevant or redundant branches, leading to a lower depth of coverage on critical paths compared to the manually-created suite. In essence, while the manual suite may excel in depth, focusing on key branches, EvoSuite's suite might excel in breadth, covering a larger portion of the codebase but potentially with less depth on critical paths.

5. In examining the Element class in the jsoup library, EvoSuite likely produced higher coverage due to its systematic exploration of execution paths, including edge cases and less intuitive scenarios not covered by manual tests. EvoSuite may generate tests involving complex HTML structures, nested elements, or unusual attribute combinations that human testers might overlook. While EvoSuite's tests achieve broader coverage, they may lack the nuanced understanding and context of human testers, leading to less meaningful or relevant test cases with lower readability. Despite this, EvoSuite-generated tests can still help identify certain types of bugs, particularly related to edge cases or unexpected behavior not covered by manual tests, although human intervention remains crucial for ensuring comprehensive coverage and meaningful scenarios in the test suite.

6. In examining the Parser class in the jsoup library, EvoSuite yielded lower coverage compared to human tests, likely due to its limitations in handling complex parsing logic and understanding HTML parsing requirements contextually. While human-created tests encompass scenarios involving intricate HTML structures and specific parsing edge cases, EvoSuite's automated approach may struggle to generate such tests effectively. EvoSuite relies on search-based techniques to evolve test inputs towards coverage goals, but navigating the complex parsing rules and heuristics within the Parser class presents challenges. Additionally, EvoSuite's reliance on code instrumentation and static analysis may not capture all parsing nuances,

resulting in coverage gaps. One hypothesis for EvoSuite's limitation is that its search-based algorithm might get stuck in local optima or fail to explore certain paths due to the complexity of parsing logic and vast input space of HTML documents. This underscores the necessity of human intervention to design tests that thoroughly exercise parsing functionality, leveraging domain knowledge and understanding of parsing intricacies beyond EvoSuite's automated capabilities.

7. AFL excels in uncovering edge cases and unexpected behavior through fuzz testing, complementing manual test suites effectively. However, its reliance on input mutations may overlook specific bugs requiring precise input sequences or environmental conditions. EvoSuite automates unit test generation for Java programs, offering a systematic approach to achieving code coverage efficiently, although it may lack the nuanced understanding of human-designed tests. While AFL is beneficial for uncovering security vulnerabilities, EvoSuite streamlines test generation for Java projects. Combining both tools can enhance overall testing effectiveness and efficiency, leveraging AFL's fuzz testing for edge cases and EvoSuite for automating unit test generation.