Critique of Configurations Everywhere: Implications for Testing and Debugging in Practice for CS5371 Soft Test for Mobile & Emb Sys

Jeremy Solmonson
School of Security Engineering
University of Colorado at Colorado Springs
Colorado Springs, CO 80922
Email: jsolmons@uccs.edu

***Abstract*—This is paper is a critique of "Configurations Everywhere: Implications for Testing and Debugging in Practice" In accordance with homework 5 requirements the critique will be based on: suggestion for acceptance, summary of paper, evaluation of paper, positive points, negatives points, and potential future work.**

I. Suggestion for Acceptance

I would recommend to accept this paper with minor revisions. The paper was well written and made some unique insights into software configurations.

II. Summary of Paper

This paper attempts to examine issues with debugging and fixing applications that fail due to conflicting configuration settings. Due to the numerous settings in current applications the debugging process can be challenging. The output from a fault or error is usually insufficient to isolate and fix the issue by using only the error information. To assist, the researches examined the configuration space of three applications. They found that large program are usually built with multiple programming languages, there are many different ways to modify a software configuration setting, and the configuration state usually cannot be determined upon failure. Additionally, they noted four lessons learned to help overcome these observations.

III. Evaluation

This paper was well organized and clearly articulated intricate concepts. The paper effectively discussed the experiment and tied together the results with lessons learned. The abstract could discuss more impact on why this research is more useful. I believe the intent is to help solve software errors and faulty systems. This was briefly discussed but not overtly stated within the abstract. While only three applications were analyzed, the level of detail and walk through analysis was easy to follow and readable. Overall I would give an exceptional rating to the readability of this paper.

While the readability was well done, the scope of the experiment was severely limited. The authors only assessed three applications, one of which was not disclosed in the paper. Although the results were displayed, we can assume the undisclosed application performed a similar function to Firefox or LibreOffice. If these three applications perform similar functions, then similar results can be expected from each application. As a result, the level of diversity was also limited in this experiment.

While the lessons learned stated briefly stated, "what needs to be done" they didn't address "how to do this." Specifically, what actions can future researchers do to improve up this model. The lessons learned was mostly targeted toward software developers on managing their own projects. A lot of manual work is required to perform the lessons learned outlined within this paper. The areas that researchers could focus toward provided minimal insight on how to achieve the desired result.

IV. Positive Points

+ High readability. This paper was well organized and articulated the details.

+ The lessons learned targeted areas for researchers and practitioners on next steps to improve the configuration domain.

V. Negative Points

- Limited experiment scope. Experiments should be conducted on a larger diverse set of software.

VI. Potential Future Work

The lessons learned identified four areas that could show improvement within the configuration domain. Two were targeted towards practitioners and two were targeted toward researchers. The two that were targeted toward researchers were: create software to better identify getter and setter functions and capture the configurations when an application fails.

I think the authors are in the right direction for the desired research directions, the ability to that end state can be challenging. The next focus of research would be to identify areas of code based on a set of conditions. While the authors specifically states getter and setter functions, the ability to

search through code that performs desired functions is valuable by itself. A program could be created to allow the author to input a set of unique conditions, and that code could be searched (dynamic, static, or both) to find sections of code that performs those functions. To the best of my knowledge, that does not exist.

As for the capturing configurations portion of future research, I do not see this as possible for every unique application. I think the authors of individual software will have to consider that as the program is developed. The ability to provide a platform that fits for all programs would be nearly impossible. Java is a good example as the bytecode was supposed to be able to run on any machine. The issue is the overhead to use java, and when applications require more speed, developers choose other languages.

As of this moment, I cannot think of any other direction to take this research that was not already mentioned.