Critique of THeME Paper 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 "THeME: A System for Testing by Hardware Monitoring Events." In accordance with homework 0 requirements the critique will be based on: suggestion for acceptance, summary of paper, positive points, negatives points, and potential future work.**

I. Suggestion for Acceptance

I would strongly accept this paper. It provides a valuable contribution on options for testing within resource constrained environments.

II. Summary of Paper

Software testing techniques, such as test coverage analysis, require additional resources to perform the analysis. While this is usually accomplished through instrumentation, the overhead can be detrimental in a resource constrained environment. Instead of using instrumentation, hardware mechanisms can be used to augment the existing hardware and prevent additional code growth. Further multi-core processors can expedite the test. A design was created, Testing by Hardware Monitoring Events (THeME), to better understand how hardware can improve efficiencies in testing.

III. Positive Points

Easy to read and follow. The paper flowed quite well and explained the analysis along the way. Difficult concepts were well explained in relation to the papers contributions. For example, 3.2.1 Access via Polling could be removed, but explained the next session and why Interrupt Driven Access is more efficient.

The paper clearly identified the issues and contributions of the research. The papers contributions were listed in the introduction and bulleted format which is easy to identify. Further, the experiments objectives were also listed in bulleted format. The conclusion summarized the experiments results very neat and orderly.

The reason to conduct the experiment and the method taken seemed appropriate to solve the underlying question: Is a hardware approach more efficient than instrumentation? For the system under test, yes, it is more efficient. To become more definitive, a larger sample size of various architectures would need to be further explored.

IV. Negative Points

In section 2.1, Cache Misses (CM) and Branch Instruction Retired (BIR) are mentioned, but not further discussed. Instead the paragraph moves to LBRs which are explained and further discussed. Would recommend either removing CM or BIR or further elaborating on their involvement.

V. Potential Future Work

While this testing works on the SPEC2006, security related code such as obfuscation my hide the potential branch options. For example consider the below code.

```
cmp EAX, ECX
jne EAX
jmp EBX
```

The values in EAX and EBX must be known prior to analysis to the branch options. These values may vary based on previously executed branches leading to branch explosion. If the values are unknown, then this section of code will be examined on a conditional basis and code coverage can not be verified. As a result, a further examination may be useful on malicious programs.

Another potential area of exploration could be on unknown binaries. Specifically, could a similar process be executed without debugging information or optimizations? If so, then embedded systems that have malicious content could be examined as well. The experiment could compare known compiled binaries and perform similar testing. If so, this would be useful for black box security testing.