# Control Flow Checking or Not? (for Soft Errors)

Aviral Shrivastava
School of Computing, Informatics and Decision Systems Engineering
Arizona State University
Aviral.Shrivastava@asu.edu

## 1. INTRODUCTION

Control Flow Checking (CFC) techniques were proposed to provide efficient protection from soft errors. The main idea is that most soft errors will eventually manifest as errors in the sequence of instruction execution. Therefore, just by making sure that the sequence of instructions executed (or the control flow of the program) is correct, then significant protection can be achieved. Note that a CFC technique by itself does not provide any protection – it merely provides the capability of detecting errors. Combined with a scheme to recover from errors (e.g., restart from the beginning; or in the case of regularly created checkpoints, continue from the last checkpoint when an error is detected), CFC techniques can provide protection from soft errors. In this paper, since we are interested in estimating the protection achieved by CFC techniques, we assume that there is some (but it does not matter which one) scheme to recover from the control flow error detected.

The arsenal of control flow based soft error protection techniques span across design layers from hardware [3, 6, 9], software [1, 8, 11, 13], and hardware-software hybrid techniques [4, 10, 14]. CFC techniques are attractive since they can often be implemented with much less overhead (as compared to full scale redundancy) and arguably provide a decent error coverage. Papers proposing CFC techniques perform fault injection tests and conclude that their technique is quite effective in combating soft errors.

In this work, we use the metric of *Vulnerability* [7] to quantitatively estimate the protection achieved by existing Control Flow Checking techniques. A bit is vulnerable in a certain cycle of execution, if a fault in it may cause a wrong result, otherwise, it is not vulnerable. Adding up the number of vulnerable bits in each cycle of the execution of a program, gives us the total vulnerability of the program execution. Higher vulnerability of program execution implies that the program execution is more susceptible to soft errors. The metric of vulnerability is more comprehensive, and does not require detailed compute-intensive fault injection experiments. In fact, the vulnerability of a program execution can be estimated in a single simulation run by tracking the events on each bit of the processor, and counting the number of vulnerable bits.

We estimate the vulnerability before and after the application of several CFC techniques. The basic idea is to analyze each vulnerable $< bit, cycle >$ in the original execution, and determine the control flow errors that it can cause. If any of the generated control flow error can be detected by the CFC, then the $< bit, cycle >$ is deemed to be *not vulnerable* in the presence of CFC.

Our results reveal that existing CFC techniques not only do not protect execution from soft errors, but in fact incur additional power and performance overheads. In particular, software only CFC protection schemes (CFCSS [8], CFCSS+NA [2], CEDA [11]) increase system vulnerability by 18% to 21% with 17% to 38% performance overhead. Hybrid CFC protection (CFEDC [4]) increases vulnerability by 5%. Even though the vulnerability remains almost the same for hardware-only CFC protection (CFCET [9]), they incur overheads of design cost, area, and power due to the hardware modifications required for their implementation.

## 2. REFERENCES

[1] Alkhalifa, Z., Nair, V. S. S., Krishnamurthy, N., and Abraham, J. A. Design and Evaluation of System-Level Checks for On-Line Control Flow Error Detection. *IEEE Trans. Parallel Distrib. Syst. 10*, 6 (June 1999), 627–641.

[2] Chao, W., Zhongchuan, F., Hongsong, C., Wei, B., Bin, L., Lin, C., Zexu, Z., Yuying, W., and Gang, C. CFCSS without Aliasing for SPARC Architecture. In *International Conference on Computer and Information Technology (CIT)* (29 2010-july 1 2010), pp. 2094 –2100.

[3] Eifert, J., and Shen, J. Processor Monitoring Using Asynchronous Signatured Instruction Streams. In *Twenty-Fifth International Symposium onFault-Tolerant Computing* (jun 1995), p. 106.

[4] Farazmand, N., Fazeli, M., and Miremadi, S. FEDC: Control Flow Error Detection and Correction for Embedded Systems without Program Interruption. In *Third International Conference on Availability, Reliability and Security* (march 2008), pp. 33 –38.

[5] Goloubeva, O., Rebaudengo, M., Sonza Reorda, M., and Violante, M. Soft-error detection using control flow assertions. In *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems* (nov. 2003), pp. 581 – 588.

[6] Madeira, H., and Silva, J. On-line signature learning and checking: experimental evaluation. In *Proceedings of 5th Annual European Computer Conference* (may 1991), pp. 642 –646.

[7] Mukherjee, S. S., Weaver, C., Emer, J., Reinhardt, S. K., and Austin, T. A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor. *IEEE/ACM International Symposium on Microarchitecture 0* (2003), 29.

[8] Oh, N., Shirvani, P., and McCluskey, E. Control-flow checking by software signatures. *IEEE Transactions on Reliability 51*, 1 (mar 2002), 111 –122.

[9] Rajabzadeh, A., and Miremadi, S. CFCET: A Hardware-based Control Flow Checking Technique in COTS Processors using Execution Tracing. *Microelectronics Reliability 46*, 5 (2006), 959–972.

[10] Saxena, N. R., and McCluskey, W. K. Control-Flow Checking Using Watchdog Assists and Extended-Precision Checksums. *IEEE Transactions on Computing 39*, 4 (Apr. 1990), 554–559.

[11] Vemu, R., and Abraham, J. CEDA: Control-Flow Error Detection Using Assertions. *IEEE Transactions on Computers 60*, 9 (Sept. 2011), 1233–1245.

[12] Vemu, R., Gurumurthy, S., and Abraham, J. ACCE: Automatic correction of control-flow errors. In *Test Conference, 2007. ITC 2007. IEEE International* (oct. 2007), pp. 1 –10.

[13] Venkatasubramanian, R., Hayes, J., and Murray, B. Low-cost on-line fault detection using control flow assertions. In *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE* (july 2003), pp. 137 – 143.

[14] Wilken, K., and Shen, J. P. Continuous Signature Monitoring: Efficient Concurrent-Detection of Processor Control Errors. In *Proceedings of the 1988 International Conference on Test* (Washington, DC, USA, 1988), ITC'88, IEEE Computer Society, pp. 914–925.

.