**RELATED WORK on DIFT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Title | Authors | Lab | Publication date |
| 1 | Secure Program Execution Via DIFT | G. Edward Suh, Jaewook Lee, Srinivas Devadas | Computer Science and Artificial Intelligence Laboratory (CSAIL)  Massachusetts Institute of Technology | October 2004 |
| 2 | Raksha: A Flexible Information Flow Architecture for Software Security | Michael Dalton, Hari Kannan, Christos Kozyrakis | Computer Systems Laboratory, Stanford University | 2006 |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 | High-Performance Parallel Accelerator for Flexible and Efficient Run-Time monitoring (Harmoni) | Daniel Y. Deng and G. Edward Suh | Computer Systems Laboratory,  Cornell University | 2012 |

1. **Secure Program Execution via DIFT**

A HW mechanism using DIFT to protect from attacks. The main contribution of this work is it presents multiple security policies and an efficient tag management with different granularities (tags par page, per byte, per quad word). *Recognizing that real-world programs often validate their input through bounds checks, this design does not propagate the tag of an index if it is added to an untainted pointer with a pointer arithmetic instruction. This choice eliminates many false positive security exceptions but also allows for false negatives on common attacks such as return-into-libc[[1]](#endnote-1).* A 1 bit tag indicates whether the data is authentic (0) or spurious (1). To obtain efficient performance results Additional TLBs (Instr. and Data) and caches for tags (L1 and L2) were added to the design. The evaluation method was not studied.

1. **RAKSHA: A Flexible Information Flow Architecture for Software Security**

The main contributions of the article is the flexibility and practicality of the architecture. Furthermore, existing approaches at the time did not protect the OS code which was done in RAKSHA. In order to detect both high-level and low-level attacks and on both CISC/RISC Architectures, ISA instructions are decomposed into multiple simpler instructions. Four concurrent security policies can be applied using the TPR/TCR (Tag propagation register/tag check register) for each security policy. As there are 4 security policies, 4 tag bits are needed. The memory and registers are extended with 4 bits. The Main core needs to be modified and a custom Linux distribution is needed. The main drawback of RAKSHA Architecture is invasive modifications which is not so practical.

1. Hari Kannan Thesis (P. 48) [↑](#endnote-ref-1)