Paper Review Form (1000 words maximum)
cs940: GhostRider: A Hardware-Software System for Memory Trace Oblivious Computation [1]

February 24, 2018

Paper Summary

In this paper, the authors present both a theoretical compilation model and its prototype hardware-software implementation to mitigate the severe performance penalties in using Oblivious RAM (ORAM) to protect sensitive remote computations from attackers with physical access to their infrastructure. The GhostRider solution exploits the lack of oblivious requirement for regular-pattern memory events to hold their corresponding data structures in the faster encrypted RAM (ERAM). Additional measures for determinism were implemented to further prevent information leak. With the aid of an annotated C-style language, the authors demonstrated that their compiler could achieve semantically-provable security against physical attacks. GhostRider's performance improvement from pure-ORAM solutions were demonstrated through simulated and prototype evaluations.

Pros and Cons

I believe that the paper has the following positive features:

- Through the introduction the authors presented a very clear objective of mitigating ORAM's performance penalties while further reducing information leak, without going into excessive depth in explaining past efforts.
- The additions of in-language sensitivity annotations and the specialised deterministic instruction set allowed a formal proof of security to be achieved without significantly compromising the generalisability of the solution.
- In performance evaluations, the authors covered all possible levels of specialisation (presences of ERAM and scratchpad) to present an in-depth analysis of performance variations.

I believe that the paper has the following negative features:

- The problem the authors were trying to solve was very narrow: most of organisations with sufficient resources to rewrite their codebase for secure remote execution would have sufficient resources to run computations in-house.
- The uses of specialised instruction set and ORAM integration depart radically from conventional computing hardware, making a realistic high performance implementation of the authors' solution unlikely.
- The solution's incomplete hardware prototype and significantly constrained programming requirements left behind significant future work to apply the architecture on real high-performance computing tasks.

The Problem/Motivation

The concept of "cloud computation" requires users of remote computing infrastructures to implicitly place trust on the physical security of the infrastructures, as attackers with physical access to the system can conduct direct and side-channel attacks. Prior solutions mostly placed computations entirely within large ORAM banks, which impose severe performance penalties to achieve oblivious memory events, and can still leak some side-channel information. Previous solutions such as Blanton et al. [2] optimised input data to allow faster ORAM execution as point solutions, and did not consider the program context. At the loss of some code generality (such as annotating variables and imposing more constrained semantics), the program context can be exploited to produce a faster ERAM-ORAM hybrid solution without compromising security.

The Solution/Approach

Evaluation

Your Opinion

Questions for the Authors

-
-

References

[1] C. Liu, A. Harris, M. Maas, M. Hicks, M. Tiwari, and E. Shi, "Ghostrider: A hardware-software system for memory trace oblivious computation," *ACM SIGPLAN Notices*, vol. 50, no. 4, pp. 87–101, 2015.

[2] M. Blanton, A. Steele, and M. Alisagari, "Data-oblivious graph algorithms for secure computation and outsourcing," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security.* ACM, 2013, pp. 207–218.