Summary Part 2

HIDE: An Infrastructure for Efficiently Protecting Information Leakage on the Address Bus

Mrinal Nath (ID: 3307043)
April 14, 2005

Did this paper address an important issue? Explain.

XOM-based architectures have been proposed to provide copy- and tamper-protection to the software running on the XOM machine. However, the XOM architectures do not provide any security to the addresses being issued on the address bus. Due to this, one can snoop the address bus and accumulate a trace of the addresses being issued. Using this trace, one can reconstruct the CFG of a the program. It is shown that the CFG of a program or function/procedure is quite unique. Thus, if one can reconstruct the CFG of the program, then by matching this CFG with other known CFGs, one can figure out the functions being reused in the program, or the algorithm being used in the program.

Thus, the lack of security on the address bus can potentially nullify the security provided by the XOM-based architectures. This paper proposes techniques to provide significant amount of security on the address bus. Thus, (if one is paranoid about his/her IP rights) this paper is dealing with an important issue.

Are the proposed approaches valid? Describe its strength and weakness.

The basic idea to provide security to the address bus is to have a probabilistically fixed address sequence on the address bus. It is also important to avoid repeating address sequences. To achieve these goals, they propose a HIDE cache, in which blocks are locked for as long as possible, so that the same address need not go out on the bus. If a address has to be repeated on the bus, then they permute the memory addresses, so that the required word will no longer be at the old address.

Strengths

- The approach of permuting blocks within a chunk reduces the overhead required, while still being very effective at randomizing the addresses to avoid information leakage
- Compiler directed code and data layout optimizations can reduce the number of inter-chunk transitions and increase the security
- The authors suggest ways to manage the stack and heap securely at runtime
- The chunk size can be varied, which provides flexibility
- The performance overhead is (somehow) very small – at most 1.5%

Weaknesses

- The approach can not be directly applied to multiprocessor systems
- The power consumption can be greatly increased due to the proposed techniques
Do the results support the conclusions? Explain.

The main conclusion is that the HIDE approach provides strong security while having low overhead. According to the results, with 64K size chunks, 87% of the address sequence is protected, in which 95% of the accesses to code and data are hidden. The performance overhead is only 1.5%. Thus, the results do support the conclusions.

Describe the potential future works?

- How to extend this technique so that it can be applied to multiprocessor systems?
- Can the power consumption be lowered?