Summary
Hide: An infrastructer for Efficiently Protecting Information Leakage on the Address Bus.
Kiran Yella[ysula
April 13, 2005

1 To be completed before class
What are the problems solved by this paper? (50 words)
Inspite of employing various encryption techniques as in XOM, information can be leaked by tracking the addresses on the address bus. The CFG of a program can reveal about 80-90% of a program code due to software reuse and legacy softwares. Some accesses can even reveal secret information about the software and make the encrypting useless.

What are the approaches attempted by this paper? (50 words)
The authors propose to permute every memory location in the address space before it is accessed twice, thereby ensuring that the same address is not repeated twice on the address bus. The authors propose a Hide cache with a fetch queue and a permutation unit which does pre-permutation so as to remove the permutation from the critical path of execution.

What are the main conclusions of this paper? (50 words)
Using Chunks of size equal to a few pages reduces the overhead and the amount of permutations to be performed. Employing compiler guided layout optimizations for code and data further enhances the performance and security of the program. Information loss is prevented up to around 95% by employing Hide caches.

2 To be completed after class
Did this paper address an important issue? Explain. (100 words)
This paper addresses the issue of reducing information leakage on the address bus by about 95% with less than 5% loss in the performance. The CFG of a program can be constructed by looking at the addresses on the address bus, which then can be used to reverse engineer a program or leak vital information like the secret keys. The authors propose to mutate the address space randomly in chunks so that there can be no information correlated by tracking the addresses. The authors propose to even permute the address space at the beginning of the program so there can be no correlation between two runs.

Are the proposed approaches valid? Describe its strength and weakness. (100 words)
Strengths:
- Pre-permutation removes the permutation of memory space from the critical path.
- Using memory chunks reduces the address space that has to be permuted.
- Compiler Layout optimizations improve the performance the transition coverage which further enhances the security.
• Since the address space is permuted for every time an address is loaded into the cache, the attacker will have no information about the CFG.

Weakness:

• The authors require extra hardware to temporarily store the information and permute the chunks, which also adds complexity to implementing the units in hardware.
• The authors propose to use cache policies to decide which address space to permute using LRU. This complicates the cache replacement policy and is very difficult to implement in hardware.
• Using them as embedded processors is plausible but they use a lot of power.
• The make the processor behavior unpredictable for real time applications.

Do the results support the conclusions? Explain. (100 words)
The IPC of the default and chunk based optimized code is almost close to the base simplescalar case which proves that the technique is usable. The increase in the transition coverage for various chunk sizes indicates that the security level increases. The size of L2 cache as expected is crucial to reduce the number of capacity misses since the pages are locked once they are brought into the cache. Layout optimizations as expected improved the IPC and transition coverage.

Describe the potential future works? (100 words)
The authors can look to extend the scheme for multiprocessor caches. The authors also point out interprocessor communication is not yet handled. The authors can try to improve the performance by using prefetching to obtain addresses which might cause L2 cache misses. Since the system is implemented along with XOM processor based compartmentalized execution, the actual performance overhead of the whole integrated system has to be tested out.