Summary
HIDE: An Infrastructure for Efficiently Protecting Information Leakage on the Address Bus
Venkatesan Packirisamy
April 13, 2005

1 To be completed before class

What are the problems solved by this paper? (50 words)
Software piracy is a huge issue at present. XOM architecture proposed a way to have hardware support for copy and tamper proof software. But XOM cannot guarantee against information leakage via address bus. Though industry and academia does recognize the problem of protecting address bus, it is very hard because of the high overhead.

What are the approaches attempted by this paper? (50 words)
This paper proposes an infrastructure called HIDE - Hardware-support for leakage-Immune Dynamic Execution. They support chunk level protection of addresses with hardware support. A chunk is defined as one or more pages that are protected and permuted together. Different layers of security is built. Also the HIDE provides an application interface to control the protection mechanism.

What are the main conclusions of this paper? (50 words)
HIDE is able to be guarantee address bus protection that was lacking in XOM. The results show that HIDE guarantee 87% of the address sequence is protected and in which 95% of the accesses to code and static data are hidden. The performance overhead is atmost 1.5% in most cases. Also since HIDE provides a good interface, it makes the protection mechanism more flexible.

2 To be completed after class

Did this paper address an important issue? Explain. (100 words)
Software piracy is a very important problem by which many companies lose billions of dollars. Also application re-engineering can lead leakage of sensitive and copyrighted algorithms. XOM was proposed as an architecture framework which can prevent software piracy and the software theft problem. But it suffers from a major drawback that it does not ensure that the address bus is secure. So if the attacker can carefully manipulate the environment and cause cache misses, the CFG finger print of the program could be leaked. This can lead to software theft. Also the attacker could get the security key information of XOM and potentially lead to software piracy.

Are the proposed approaches valid? Describe its strength and weakness. (100 words)
In this paper they propose a mechanism called HIDE - hardware support for leakage immune dynamic execution. The basic idea is to avoid having any correlation between repeated memory addresses. So if there is no correlation between different accesses, then the attacker cannot get any information on the access pattern and the CFG, thus preventing the leakage. To prevent the correlation, we have remap the memory locations every time we access the memory. In this paper they propose to use chunks - here the permutations are performed only on a small set of addresses. Thus making it more efficient. Though the technique is foolproof, it can be very inefficient.
Do the results support the conclusions? Explain. (100 words)
At first the paper shows the importance of making the address bus secure. It is been shown that 70% of the code is reused, so most of the code is vulnerable to the attacks described in the problem. Also the paper shows an example where the data could be leaked showing the possibility of the security key itself being leaked and thus allowing software piracy. The results show that even with small chunks most of the code could be secured. It is shown that with 8K chunks, 75% of the code can be secured. It is shown that the performance overhead is only 1.5% in most benchmarks.

Describe the potential future works? (100 words)
Though the paper claims that the technique is very efficient, it does take a lot of hardware and for some programs the overhead could be much larger. Also it is not clear if the inter chunk transitions could lead to information leakage. So we should have a better mechanism than just using permutations. Also the permutations are assumed to be random. But they are not usually the case, information could be got by comparing the sequence with different randomizations methods. Also there are leakages possible at other locations like by monitoring the disk access patterns we could know which files or which records are updated. So the same approach could be extended for other scenarios.