Research Statement

Tey Chee Meng
School of Information Systems, Singapore Management University
Email: cmtey.2008@smu.edu.sg
Updated on 9th December 2010

Background

Vulnerabilities in computer programs result in worms, viruses, computer espionage and cyber warfare. The fact that there exists no comprehensive solution to these issues limits the opportunities that the Internet can offer.

Current Research

We look into the building of a capability based hardware architecture that has the following features:

1. Buffer overflow prevention
2. Fast message passing
3. Isolated processes
4. Resistance to time and memory resource exhaustion

Buffer overflow prevention would solve a class of security vulnerabilities that rely on incorrect memory buffer use, either on the stack or on the heap. There are many solutions for BO prevention. However, we believe that a solution that is implemented at the lowest level would offer the most comprehensive solution.

Isolated processes allows protection of processes from one another. This is currently achievable between user mode processes on most mainstream operating systems. However, there is little protection from the kernel and other privileged processes. What we are doing here is to redesign the hardware-software interface such that even kernel/processes privileged enough to interface with IO does not have access to the other processes on the system.

The purpose of such a feature, is to reduce the burden of trust required of hardware operators and administrators. However, no software protection is viable if the administrator can physically swap out hardware or inspect memory. Hence this particular feature requires an accompanying set of physical security considerations.

Attacks may also seek to bring down a server, instead of attempting to control the server. 2 key ways is to either bring the program into an infinite loop or to make it consume an inordinate amount of memory. We include a feature in the hardware to signal a process based on attributes that the process passes to the hardware whenever the time or memory threshold is exceeded.

We also like to provide a formal hardware model for this architecture so that programs written for such an architecture can rely on it to prove their correctness.