COMP9242 Exam
Controlling the Speed of Virtual Time for Malware Deactivation
Paper 2

November 7, 2012

1 Summary

HyperSlow presents a mechanism for providers of Infrastructure as a Service (IaaS) to reduce the negative effects of processes running within a host VM that are identified as malware. It aims to provide more fine-grained control than typical systems which allow granularity at a per-VM level, giving the user the ability to slow down execution of malware running on a system on a per-process basis.

HyperSlow is implemented on top of the hypervisor Xen, and interfaces with a malware detection system and Xen's sheduler. By taking over the job of providing virtual timer interrupts to each virtual machine running on the system, it is able to trick a guest operating system (which trusts the timer interrupts it receives and uses them to calculate the time-slice given to each process) into reducing the timeslice given to the process identified as malware.

Assuming that a guest operating system expects a timer interrupt every t ms, and gives its processes a timeslice equal to at ms, where a is an integer, HyperSlow triggers Xen's timer interrupt faster than usual (say, every $\frac{1}{100}t$ ms) and presents an artificial system time (sped up by the same proportion) to the guest OS. The guest OS' scheduler will eventually run the next process in its list, without knowing about the disruption - the malware will have executed significantly less instructions than it would have otherwise, assuming it is a CPU-bound process.

HyperSlow builds on previous work such as FoxyLargo which also has the ability to control virtual CPU speed, but only on a per-VM granularity. In conjunction with the cited research on malware detection (Garfinkel et al, Jiang et al, Jones et al) it provides a solution that, when applied with these other systems, could form a fairly comprehensive malware detection and prevention scheme running on Xen.

2 Successes

By snooping the page table register of its host processor (CR3, on X86), HyperSlow is able to perform its time acceleration independantly of which guest operating system is running on it. No hooks are required by guest operating systems to support the technology, which sits within the privelaged domain dom0.

The system does a good job of slowing down CPU-bound processes.

3 Limitations

As they are generally not starved of CPU cycles, the HyperSlow system does very little to slow IO-bound processes (such as one sending spam emails).

VOIP applications rely on accurate timers for synchronising packets and rendering audio - when HyperSlow is active, there could be significant disruption to both packet timing and audio playback. Processes which rely on time-outs, like a login server, may also be impacted. In particular, some user-level applications may make assumptions about adjacent invocations of time functions such as *gettimeofday* returning non-negative time differences. However, the author does briefly acknowledge this shortcoming.

4 Criticisms

The implications of providing false time data to an operating system followed by a discontinuity in the system time are not explored, making it difficult to judge the real-world usefulness of HyperSlow. Perhaps the HyperSlow process could be reversed for the targeted VM to slow time, rather than create a discontinuity. However, this solution would still suffer from the problem affecting applications which require accurate timing at regular intervals.

The author should compare to a baseline system without malware or HyperSlow code running - what is the overhead of HyperSlow running under conditions where malware is not active? Are there any other side-effects of HyperSlow's interaction with Xen's scheduler compared to a native Xen system running without HyperSlow? Does the scheduler remain fair in the presense of many processes (and several malware processes)? This is not thoroughly explored in the paper.

"Deactivating" a process running on a client's virtual machine after receiving a false-positive detection could be a disaster for the client, and even violate any quality of service guarantees the provider promises. The system may be of dubious usefulness if the client is charged by CPU cycles or network bandwidth, as in the common IaaS setup - one could argue that it's the client's responsibility to decide what runs on their system, and their responsibility to keep it malware-free

Spam-mailing malware which is aware of HyperSlow may be able to detect its erratic scheduling in another thread and fork as necessary to attempt to maintain its email throughput.

5 Conclusions

HyperSlow exhibits an interesting method for malware prevention built into a hypervisor. Combined with existing research into malware detection algorithms, it could provide a useful malware detection/prevention system. Additional work could be done in investigating the impact of dynamically adjusting virtual timer interrupts on user-level applications that rely heavily on accurate time.