Current Status of the Xenon Secure Hypervisor

John McDermott
Center for High-Assurance Computer Systems
Naval Research Laboratory
john.mcdermott@nrl.navy.mil

Friday, December 9, 2011
Project Goals

* investigate higher-assurance open source
* investigate run-time security of VMMs
Increasing Run-Time Security of a VMM

- add external run-time integrity verification
- add internal run-time integrity verification
- additional self-protection mechanisms, to increase tamper-resistance
- refactor the software, to decrease the number of residual security flaws
- use formal methods to decrease the number of residual security flaws
Issues 1

* ceteris paribus, adding software adds flaws and increases attack volume

* integrity verification challenged by sampling rate, coverage, assurance, and tamper resistance

* segment register rant (self-protection mechanisms)
Issues 2

* new self-protection mechanisms challenged by performance, restrictions on system programming, and reference monitor properties
* refactoring challenged by complexity of hardware and guests, performance, commodity, and lack of bling
* formal methods challenged by complexity of hardware and guests, performance, commodity, and size of target system
Related Work

* NoHype (Princeton)
* HyperSafe (NC State)
* NOVA (Tech. U. Dresden)
* CloudVisor (Fudan U.)
* Xoar (UBC, NSA)
* L4.verified (UNSW NICTA)
* embedded device separation kernel (NRL)
Refactoring

* simplification
* code base reduction
* subsetting
* must be able to run unmodified Windows7 guests
Simplification Patterns

* use complexity as a guide not a rule
* apply special patterns for Xen code
* replace/remove *gotos*
* static always_inline functions
* replace complex logic with simple state machines
Maximum Complexity

* Order of magnitude reduction from
  * 2954 (Xen) to
  * 112 (Xenon)

* Will be lower as work progresses
**Code Base Reduction**

* avoid high-security-risk features
* reduce cost and scope of 3rd party assessment
* reduce number of residual flaws
Significant Reductions

* dropped all CPU but x86_64
* dropped Intel HTT
* dropped transcendent memory
* dropped NUMA
* removed miscellaneous chunks of code
* dropping 32-bit VMM
* replacing XSM with MSM
Progress

* Xen - 181,174 SLOC
* Xenon - 128,082 SLOC
* measured by SciTools Understand
Size vs Complexity

Code Complexity Comparison

- Xen
- Xenon

Lines of Code vs Complexity
Performance

- no apparent penalty (slight gain)
- `kcbench -j 2 -n 10 # HAP enabled HVM Fedora 14 * 3`
- Xeon E31270 3.4 GHz, 16 GiB memory
- Xenon/Xen 258 sec / 270 sec
- similar results for Windows 7 guests
Keeping Pace

* Xenon is based on Xen 4.0
* need to keep pace with Xen community
  * Xen development benefits
  * keep up with hardware evolution
* adapted 194 patches from 13 April 2011 to 27 July 2011
Tools Are Important

* vi, emacs, cscope, hg, dot, bash, benchmarking tools
* mini-OS
* Understand (SciTools)
* CodeSurfer, CodeSonar (GrammaTech)
* CZT, Circus (formal methods)
Questions?