Protected Virtual Machines for s390x

Claudio Imbrenda

IBM

October 31 2019

Disclaimers

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:
IBM, the IBM logo, IBM Z, IBM z Systems, IBM z15, WebSphere, DB2 and Tivoli are trademarks of IBM Corporation in the United States and/or other countries. For a list of additional IBM trademarks, please see https://ibm.com/legal/copytrade.shtml.

The following are trademarks or registered trademarks of other companies: Java and all Java based trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries or both Microsoft, Windows,Windows NT and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both. Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. UNIX is a registered trademark of The Open Group in the United States and other countries or both. Linux is a trademark of Linus Torvalds in the United States, other countries, or both. Cell Broadband Engine is a trademark of Sony Computer Entertainment Inc. InfiniBand is a trademark of the InfiniBand Trade Association. Other company, product, or service names may be trademarks or service marks of others.

Linux penguin image courtesy of Larry Ewing (lewing@isc.tamu.edu) and The GIMP

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Users of this document should verify the applicable data for their specific environment. IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Information is provided “AS IS” without warranty of any kind. All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area. All statements regarding IBM’s future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Overview

Introduction

Protected Virtualization allows for virtual machines whose state is not observable or alterable by the hypervisor.

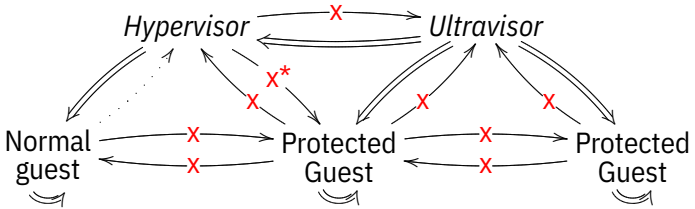
Motivations

- Protection against malicious operators
- Protection against malicious hypervisors
- Protection against bugs in the hypervisor
- Compliance

Architecture overview

Architecture overview

- Guest (trusted for itself)
- Hypervisor (untrusted)
- Ultravisor (trusted system-wide)



Ultravisor

The Ultravisor is a trusted entity, implemented in hardware and firmware.

- Takes over some of the tasks traditionally performed by the Hypervisor
- Decrypts and verifies the boot image of Protected Guests
- Protects the Guest from the Hypervisor and from other Guests
- Proxies the interactions between the Guest and the Hypervisor

Protection

- Guest memory is not accessible by the Hypervisor, unless shared
- Protected Guests can only access their own memory
- The Guest decides which memory to share with the Hypervisor (for I/O)
- The Guest-to-Host mappings are secured
- The Ultravisor checks and verifies the interactions between Hypervisor and Guest

What's left in the Hypervisor

- I/O and device model
- Scheduling
- Memory management
- Housekeeping for some instructions
- Many instructions are still handled by the Hypervisor (including I/O)

Lifecycle

Lifecycle – Hypervisor setup

1. Check for UVC (UltraVisor Call) instruction
2. Query Ultravisor
3. Opt in, donating some memory to the Ultravisor

Lifecycle – Protected Guest boot

1. The Guest boots in standard (non-secure) mode
2. The non-secure Guest loads the encrypted blob in memory
   - Unless the blob was already loaded by the Hypervisor
3. The Guest performs a “reboot into secure mode”
4. The Hypervisor issues the appropriate UV calls to set up a protected Guest
   1. Create secure configuration
   2. Create secure VCPUs
   3. Set configuration parameters
5. The Hypervisor issues the appropriate UV calls to unpack and decrypt the guest memory
   - The blob is encrypted with a public key
   - The private key is safely hidden in the hardware
6. The Hypervisor will now use a different format for the VCPU control block

Memory donation

At several stages during the lifecycle, the Hypervisor will need to donate some blocks of *contiguous* memory to the Ultravisor.

Area	*Type*	*Size*	
UV base storage	Absolute	Variable	can be big
Config base storage	Absolute	Fixed	small
Config virtual storage	Virtual	Fixed + Variable	can be big
CPU base storage	Absolute	Fixed	small

Lifecycle – Instruction interception

- Most fields in the CPU state description become reserved and unused
- Different Interception Codes than normal
  - Differentiation between interpretation and notification
- *Secure Instruction Data Area* used as a bounce buffer for small control blocks

Lifecycle – Interrupt injection

- Interrupt injection through state description
- Interrupt parameters in the Interception-Data block
  - Same locations used for interrupt intercepts
- Not all interrupts are always allowed
  - Only few program interrupts allowed to be injected
  - Only when interpreting some specific instructions
- Interrupt injection never allowed for masked interrupts

State Description

- Interception Parameters
  - Instruction text normalized
- General Registers
  - Each saved on a case by case basis
  - Saved in normalized locations
  - When appropriate copied back to the correct guest registers
- Control Registers
  - Only the interrupt-related bits are saved
- PSW
  - Only the interrupt bits and the wait-state bit are saved
- CC
  - Write-only
  - Checked only when a new value is expected from the Hypervisor
- Secure Instruction Data Area
  - Designation in the state description

Lifecycle – Guest

Bootloader / boot stub:

1. Check for Protected Virtualization IPL enhancements
2. Load encrypted blob
3. Perform a *reboot into secure mode*

Actual Guest kernel:

4. Check for UVC instruction and Query Ultravisor
5. Share the memory to use for the bounce buffers
6. Use bounce buffers (SWTLBIO) for VirtIO

Changes in Guest and Hypervisor

Changes in the bootloader / boot stub

- Load encrypted blob
- perform transition to secure mode

Changes in Guest

The architecture has been designed to require the smallest amount of changes in the Guest

- Query ultravisor information
- Need to share buffers for I/O
- Keyless mode

Already upstreamed!

Changes in KVM

- Query ultravisor information and memory donation
- Memory access IOCTLs
- Interrupt injection handling
- Special handling for several instructions
- Introduce UVCs in lifecycle

Upstream discussion started.

Changes in Qemu

- Add support for the “reboot into secure mode” functionality
- Interpretation of instructions needed fixups for secure mode

Conclusions

Side effects – takeaways

- Fixed some pre-existing bugs
- Improved overall architectural compliance

TODO

- Swap
- Migration

Questions?