# Protected Execution Facility:

Secure computing for Linux on Power
and OpenPower

—

**Guerney D. H. Hunt**
Research Staff Member

IBM

# Acknowledgements

This work represents the view of the authors and does not necessarily represent the view of IBM. All design points disclosed herein are subject to finalization and upstream acceptance. The features described may not ultimately exist or take the described form in a product.

IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries. Linux is a registered trademark of Linus Torvalds. Other company, product, and service names may be trademarks or service marks of others.

This material contains some concepts that were initially developed during research sponsored by the Department of Homeland Security (DHS) Science and Technology Directorate, Cyber Security Division (DHS S&T/CSD) via BAA 11-02; the Department of National Defense of Canada, Defense Research and Development Canada (DRDC); and Air Force Research Laboratory Information Directorate via contract number FA8750-12-C-0243. The U.S. Government and the Department of National Defense of Canada, Defense Research and Development Canada (DRDC) are authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Department of Homeland Security; Air Force Research Laboratory; the U.S. Government; or the Department of National Defense of Canada, Defense Research and Development Canada (DRDC)

# Contents

# Team

- IBM Research, IBM  Cognitive Systems (POWER) including Linux Technology Centers.
- Our objective is to deliver the technology to the Power and OpenPower communities.
- Those involved in updating (patches) existing components and developing new components and tools will be pushing their commits to GitHub.

# Security Challenges

## Increased prevalence of multi-tenant and cloud computing models amplifies security concerns

- It is increasingly hard to verify the provenance and correctness of all software components like hypervisors, operating systems, privileged SW, etc.
- Components of these systems provide a large attack surface
- Unfortunately, these components can also contain a number of vulnerabilities and zero-day attacks

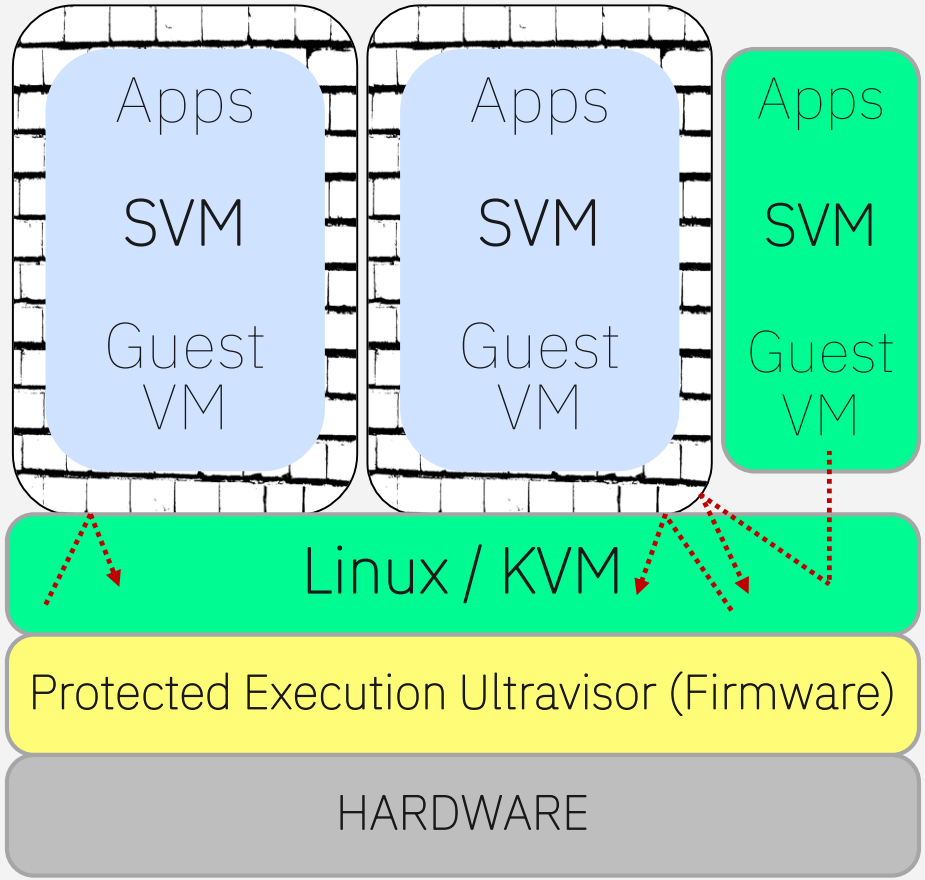
## Objectives

- Minimize the hardware and software that needs to be trusted - the TCB
- Provide provable protection against insider attacks: Malicious, “curious” or ”careless” administrators

# Protected Execution Facility

Provides protection for sensitive code and data:

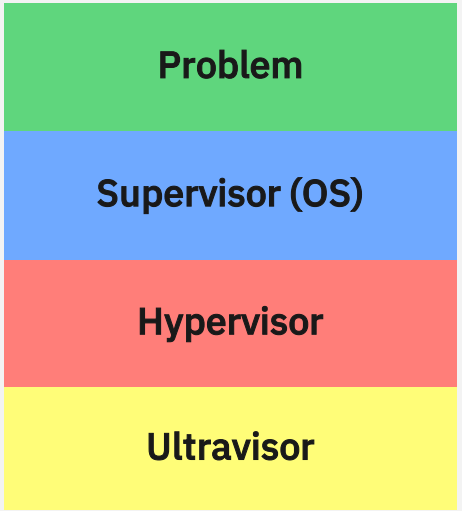
- From other software (applications, systems software)
- Rogue administrators
- Compromised hypervisor
- While in transit, executing, or stored on disk



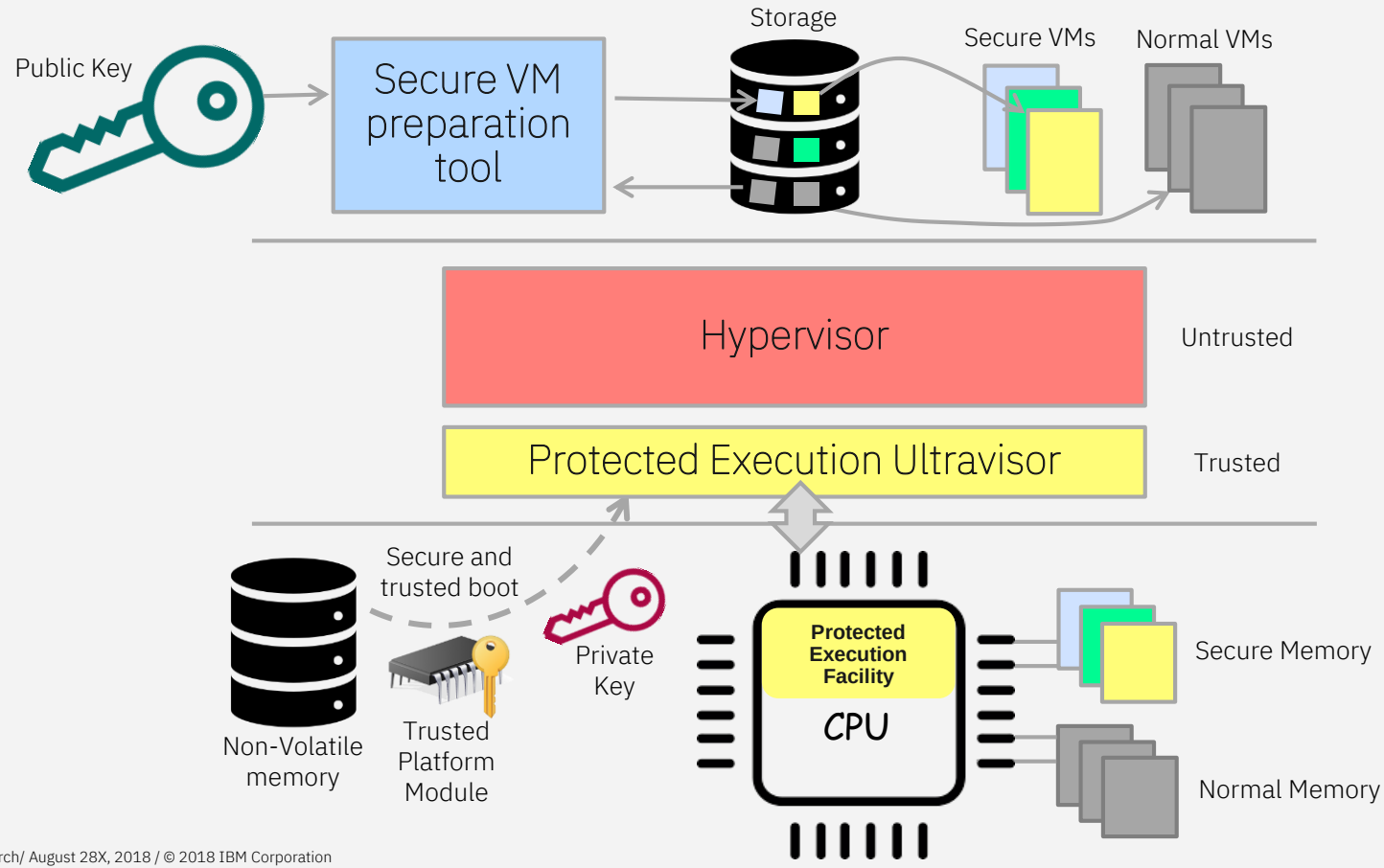
# Architecture Overview

# Base Principles

- Protect integrity and confidentiality of code and data
- Minimize the trusted computing base (TCB)
  - Processor (hardware changes), TPM, and Firmware (Ultravisor)
- Introduce Secure Memory, only accessible by secure VMs and Ultravisor
- Introduce new Power processor mode: “Ultravisor mode”
  - Higher privileged than hypervisor mode
  - Hardware and firmware are used to manage the new security feature
- Enable secure virtual machines (SVMs)
  - Normal VMs run on the same hardware



# Overview of architecture



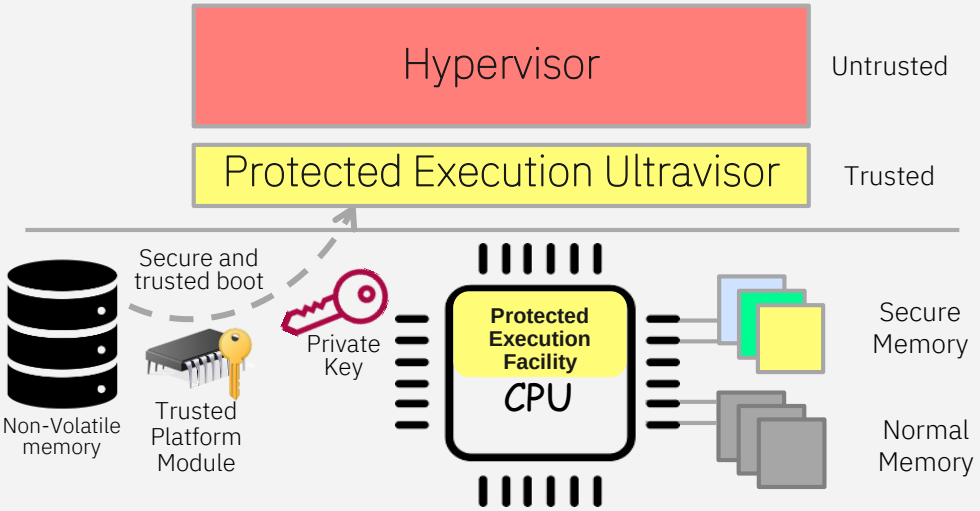
# Overview of architecture

- Protected Execution facility refers to the changes made to Power/OpenPower architecture
  - Each machine has a public private key pair
- Protected Execution Ultravisor is the firmware (which will be open source) part
- Secure VMs (SVM) and Normal VMs run on the same hardware
- Creating an SVM requires new tooling that will be open source
- SVMs execute in secure memory which is under the control of the Ultravisor
- The hypervisor and normal VMs cannot reference secure memory


IBM Research / August 28, 2018 / © 2018 IBM Corporation

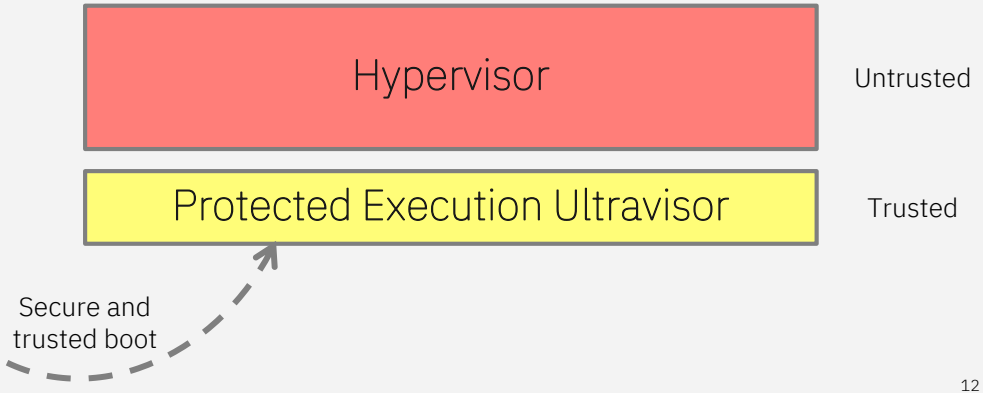
# Architecture at the hardware/firmware level

- The private key of the machine remains in the TPM. The Ultravisor uses the private key for decrypting parts of the SVM.
  - Ultravisor uses a secure channel to talk to the TPM
- The hardware separates memory into secure memory and normal memory
  - Only software running in secure mode can access secure memory
  - After boot, only the SVMs and Ultravisor run in secure mode
- When an ESM call is received, if the calling SVM has not been modified, the Ultravisor will transition it to secure mode



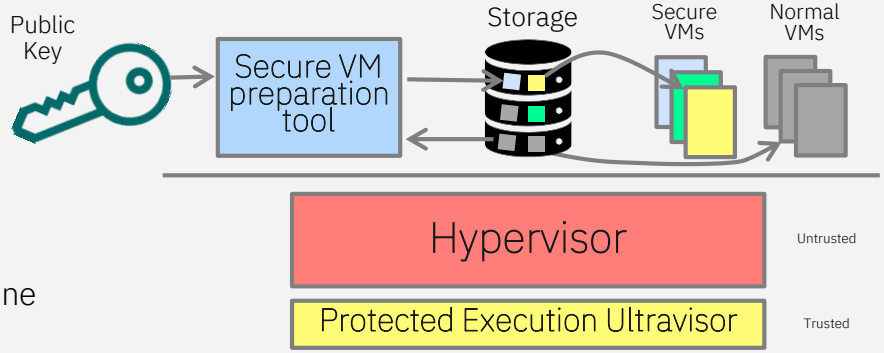
# Architecture  implication for the hypervisor

- The Ultravisor is higher privileged than the hypervisor.
- The hypervisor (Linux/KVM) has to be para virtualized to operate properly with the Ultravisor.
  - Most of these changes are in the architecture dependent sections of the hypervisor
- If the hypervisor needs to update the partitioned scoped page table it will have to ask the Ultravisor for assistance.
- If the hypervisor is returning to a SVM it will have to ask the Ultravisor to complete the return.
- HMM will be updated to help manage secure memory



# Architecture at the VM level

- Secure VMs (SVMs) and Normal VMs run on the same hardware. GRUB is boot loader.
- SVMs and VMs both get services from the hypervisor
  - All hypervisor calls from an SVM go to the Ultravisor which saves and protects state before reflecting the call to the hypervisor.
  - An SVM can share unprotected memory with the hypervisor
- SVMs are created with new tooling.
  - The creator of an SVM supplies the public key of every machine that the SVM is authorized to run on.
- Secure VMs start executing as a normal VM and, at the proper time, use a new syscall instruction directed to the Ultravisor to transition into secure mode.



# Limitations

**First release**

- Will not support
  - Suspend
  - Resume
  - Migration
  - Over commit of SVM memory
  - Dedicated devices to SVMs

**Protected Execution Facility does not support**

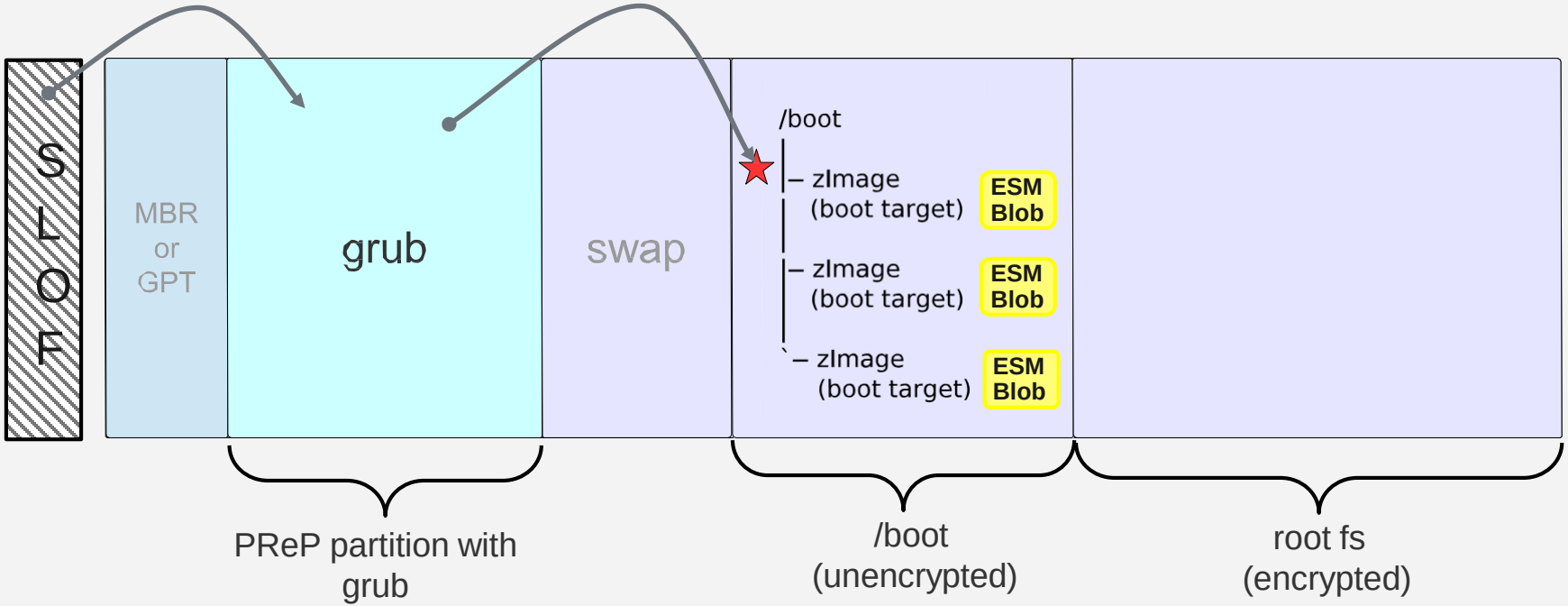
- Transaction memory
- VM that use transaction memory if converted to SVMs may crash while executing
  - If the tooling can detect that TM is required the conversion will fail.


IBM Research / August 28, 2018 / © 2018 IBM Corporation

# **Lower Level Details:**
SVM, interfaces, kernel and hardware

# SVM format and Booting

- Target OS kernels/initramfs in /boot are converted to zImage+ESM Blob
- Run “grub2-mkconfig” to point to new boot targets
- Target zimage provides information for Ultravisor to move VM into secure memory



# Boot Changes

**prom_init**

- Changes proposed to ensure that prom_init does not make changes components of the SVM and cause it to fail integrity checks

http://linuxppc.10917.n7.nabble.com/no-subject-td138496.html#a138497

**Wrapper**

- Changes proposed to enable an ESM Blob to be added to a pseries zImage

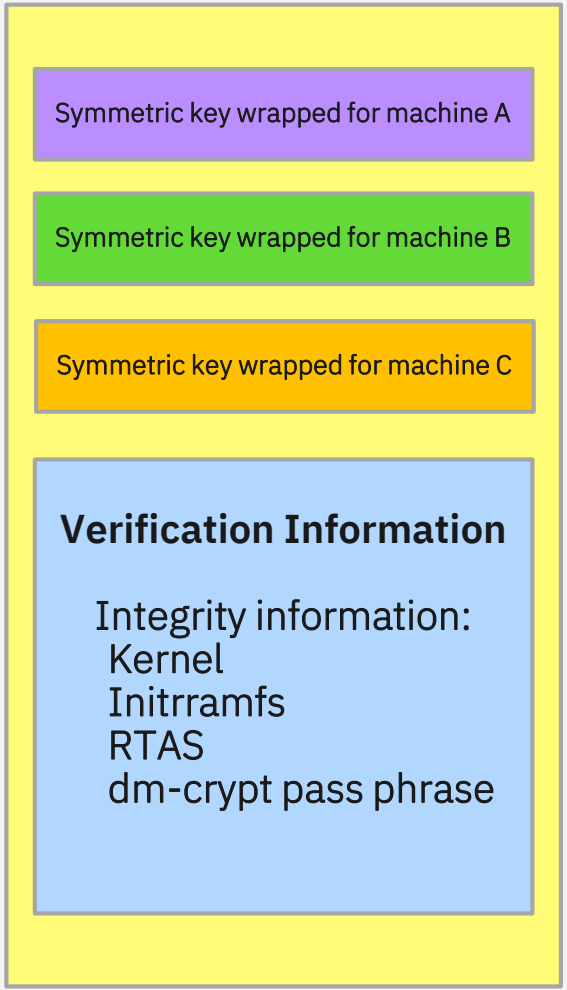
http://linuxppc.10917.n7.nabble.com/RFC-PATCH-powerpc-Add-support-for-adding-an-ESM-blob-to-the-zImage-wrapper-td138507.html

**grub2-mkconfig**

- Patch needed in grub2-mkconfig to discover and configure zImage targets

# Contents of ESM blob

Symmetric key encrypted by one or more public keys and the verification information.



# Interfaces to the Ultravisor: ultra calls

An ultra  call is a syscall instruction with Lev=2

These are the currently defined calls:

- UV_READ _SCOM
- UV_WRITE_SCO
- UV_REGISTER_STOP_STATE
- UV_RESTRICTED_SPR_WRITE
- UV_PAVE_OUT
- UV_PAGE_IN
- UV_PAGE_INVAL
- UV_WRITE_PATE
- UV_RETURN
- UV_REGISTER_MEM_SLOT
- UV_UNREGISTER_MEM_SLOT
- UV_SVM_TERMINATE
- UV_SHARE_PAGE
- UV_UNSHARE_PAGE
- UV_ESM

There probably will be changes to this list as we move forward

# KVM Changes

**New h-calls needed in KVM**

Several new h-calls need to be added to KVM to support the Ultravisor initially:

- H_SVM_INIT_START and H_SVM_INIT_DONE
- H_SVM_TERMINATE
- H_SVM_PAGE_IN and H_SVM_PAGE_OUT
- H_TPM_COMM

Other additions may be required.

**HMM-UV**

An additional ppc-specific driver is required for Ultravisor that exposes the secure memory management to the hypervisor (KVM)

- These changes are in addition to the HMM driver accepted in 4.18-rc6

Initial code is going through review/integration testing and is expected to be posted for external review in September 2018

# Kernel Changes

**virtio**

Changes needed to set up non-secure memory regions and establish bounce buffers in those regions to facilitate virtual I/O flow for SVMs

Proposed changes have been posted as RFC at https://lkml.org/lkml/2018/7/20/30

**VPA**

Changes needed to set up non-secure memory regions and establish private areas for communication between the hypervisor (KVM) and the SVM

Initial/proposed code developed and under discussion internally. Post to external community for discussion expected in August 2018

# Brief introduction to some of the hardware changes

An address bit indicates a reference to secure memory

- Amount of secure memory is configurable

The $MSR_S$ bit indicate running process is secure

MSR $_{S\ HV\ PR}$ determine privilege

Secure

| S | HV | PR | |
|---|---|---|---|
| 1 | 0 | 0 | privileged |
| 1 | 0 | 1 | problem |
| 1 | 1 | 0 | ultravisor |
| 1 | 1 | 1 | (reserved) |

Normal

| S | HV | PR | |
|---|---|---|---|
| 0 | 0 | 0 | privileged |
| 0 | 0 | 1 | problem |
| 0 | 1 | 0 | hypervisor |
| 0 | 1 | 1 | problem |

New registers

- SMFCTRL
- URMOR, USRR0, USRR1, USPRG0, USPRG1

New instruction

- URFID

When $MSR_S$=1, running either the Ultravisor or a secure VM in privilege or problem state.

- All hypervisor calls and interrupts go to the Ultravisor.
- Asynchronous interrupts go to the Ultravisor and are reflected to the hypervisor

# Summary

# Secure Execution Technologies

Minimal work for exiting code

Changes or additional work

Differing approaches to feature

| Step/decision | IBM | AMD | Intel | ARM |
|---|---|---|---|---|
| Name | Protected execution facility | Secure Encrypted Virtualization | Software Guard Extensions | TrustZone |
| Protection | Vulnerable HV, other software, system admin | Physical, vulnerable HV, other software, system admin | Physical, vulnerable HV, other software, system admin | From non secure world |
| Security Domain | VM/Container | VM/Container | Enclave | Secure World |
| Application | No Changes | No Changes | Software changes to use enclave | No Changes |
| Guest OS | changes to exploit | Guest manages encrypted memory pages | Software changes to use enclave | N/A |
| Hypervisor | KVM must be para virtualized | Software changes for keys, etc. | Software changes to use enclave | N/A |
| Secure Memory | Privilege protection | Encrypted | Encrypted | Privilege protection |
| Secure memory Integrity | Yes | No | Yes | N/A |
| Embedded Secrets | Yes | Yes | No | Yes? |

# Value of Protected Execution Facility

- Protects a Secure VM against attacks
- Smaller TCB (Trusted Computing Base) leads to reduced attack surface
- Open Source ecosystem
- Integration with Trusted Computing Tooling
- No limitations in amount of protected memory, no need to change application code, etc.

# Relevant IBM secure processor products and Research

## IBM 4758 cryptographic co-processor

- And it Successors: https://www-03.ibm.com/security/cryptocards/pciecc2/pdf/4767_PCIe_Data_Sheet.pdf

## IBM “Secure Blue” Secure Processor Technology

- https://researcher.watson.ibm.com/researcher/view_page.php?id=6904

## SecureBlue++/

- http://link.springer.com/chapter/10.1007%2F978-3-642-21599-5_13

## Secure Service Container secure execution technology on IBM Linux one

- https://www-03.ibm.com/press/us/en/pressrelease/53129.wss

## Access Control Monitor (ACM): Hardware-Support for end-to-end Trust

- Research project funded by US (DHS/AFRL) and Canadian governments
- Final Report: http://www.dtic.mil/dtic/tr/fulltext/u2/1026470.pdf