Protected Execution Facility:

Secure computing for Linux on OpenPOWER

Guerney D. H. Hunt
Research Staff Member

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 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
- All the software, firmware, and tooling is open source and will be made available on github. Some of the changes have already been posted.

Security Challenges

Increased prevalence of multi-tenant 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 software, etc.
- Components of these systems provide a large attack surface
- Unfortunately, these components can also contain a number of vulnerabilities and zero-day attacks

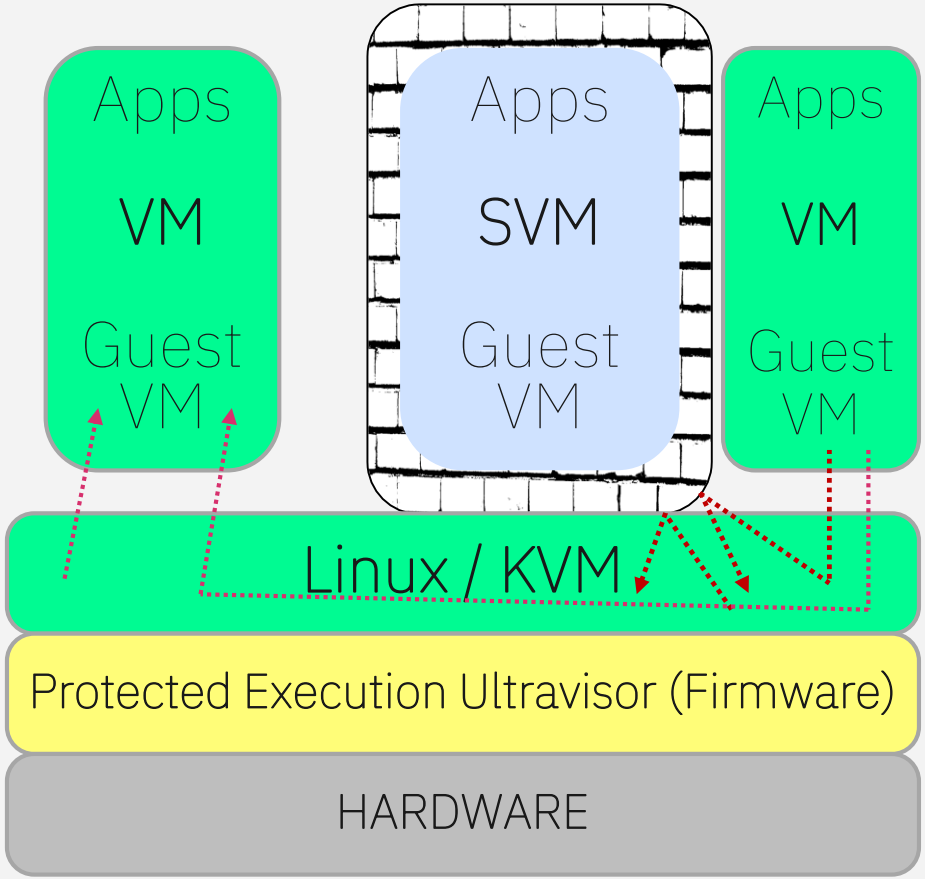
Objectives for Protected Execution Facility

- Introduce Secure Virtual Machines (SVMs)
  - Protect SVM against attacks
  - Protect confidentiality and integrity of SVM
  - Integration with Trusted Computing tooling
- Enable secrets to be inside (embedded) in SVM at creation
- Conversion of existing VMs into SVMs with new tooling
- Smaller Trusted Computing Base (TCB) leads to reduced attack surface
- Open Source ecosystem
- No limitations in amount of protected memory
- Existing application code can run in an SVM

Protected Execution Facility

For SVMs provide protection for sensitive code and data:

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



Creating and Running SVM

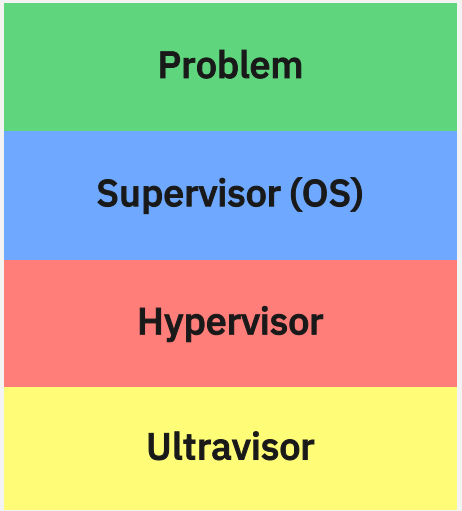
- **Creating**
  - Start with a regular/normal VM
  - Develop application
  - Run tooling
    - Collect public keys of authorized machines
    - Tooling confirms that file system is encrypted
    - Builds integrity information
  - Output is SVM

- **Running and SVM**
  - Starts like any normal VM
  - During the boot process SVM executes an Enter Secure Mode (ESM) syscall instruction
  - Ultravisor receives ESM and pointer to encrypted information
  - If access is gained, it checks the integrity of the SVM
    - If the integrity check passes execution continues
  - During execution the Ultravisor receives all interrupts from the SVM. It saves the state of the SVM and only reflects the information required to process the interrupt.

Architecture Overview

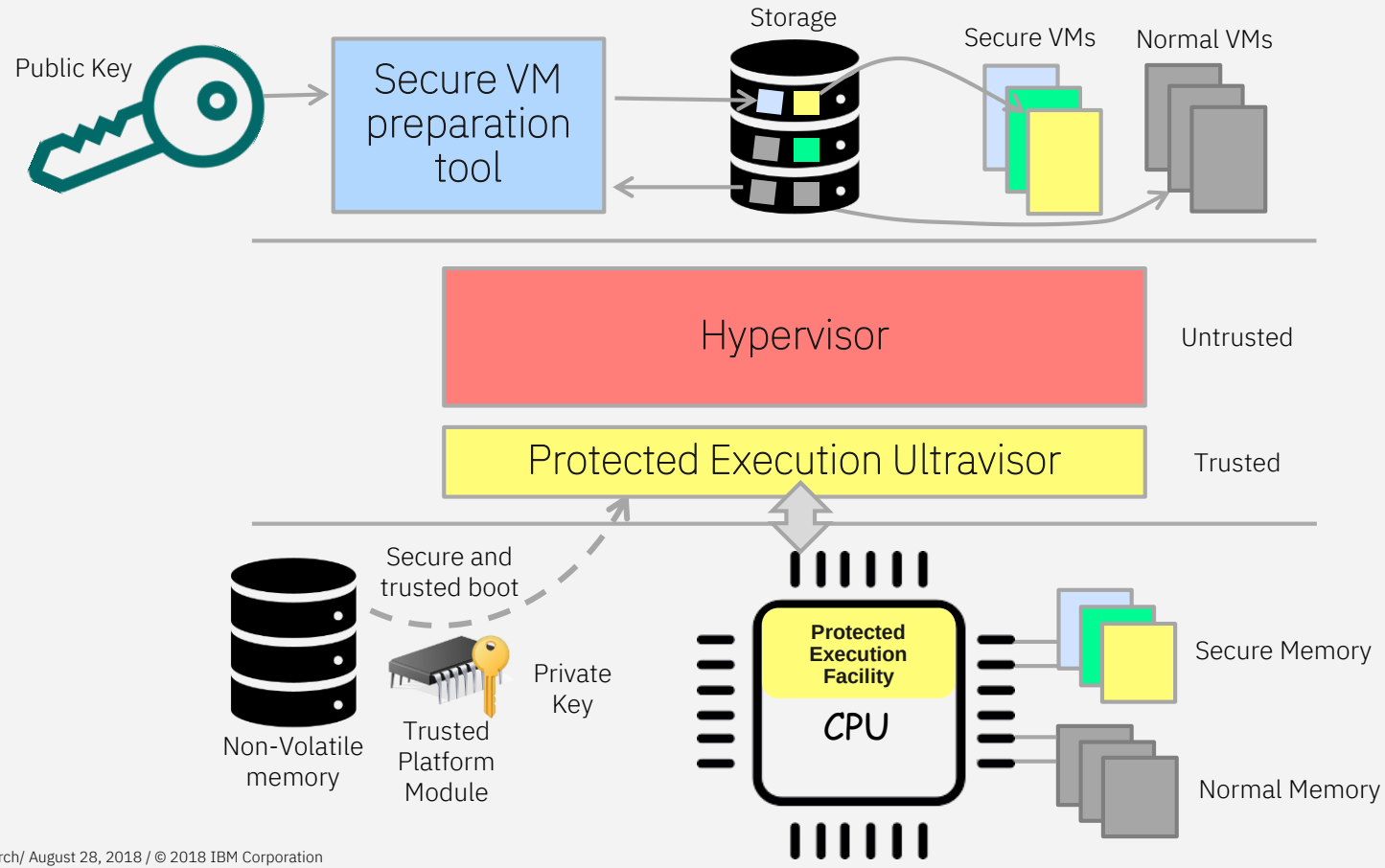
Base Principles

- Enable integrity and confidentiality protection for code and data
- Minimize the trusted computing base (TCB)
  - Processor (hardware changes), TPM, and Firmware (Hostboot, OPAL, & 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



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

Overview of architecture

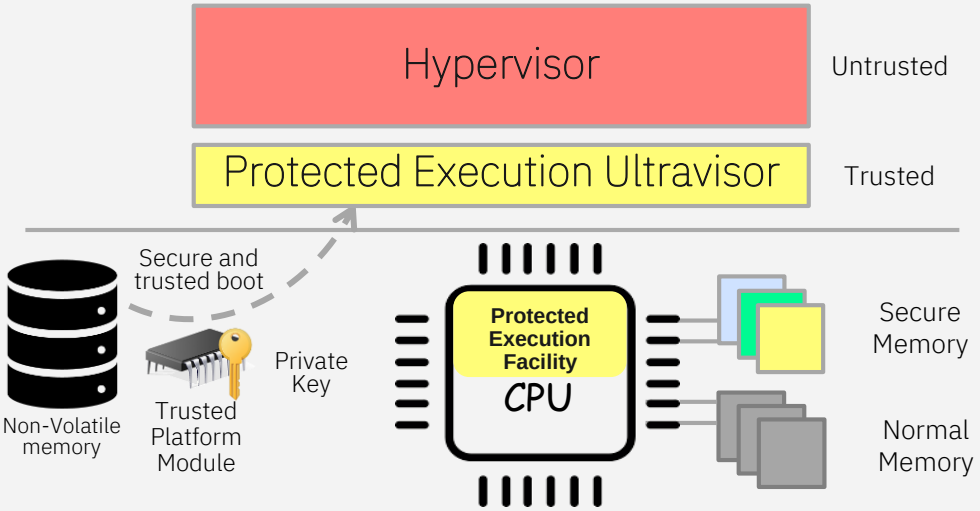


Overview of architecture

- 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

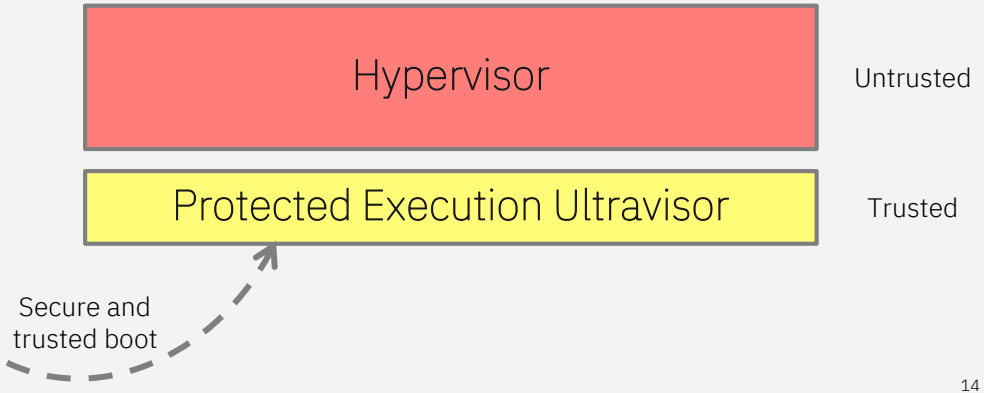
Architecture at the hardware/firmware level

- The private key of the machine remains in the TPM. The Ultravisor uses the TPM to get access to the symmetric key protecting 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 syscall is received, if the calling SVM has not been modified, the Ultravisor will transition it to secure mode



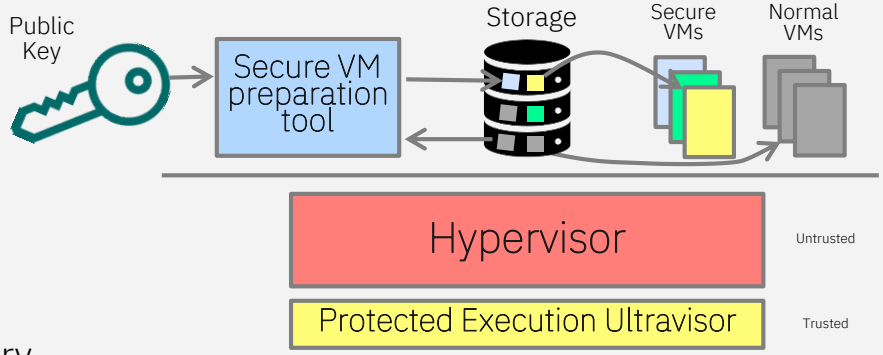
Architecture implication for the hypervisor

- 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 an SVM it will have to ask the Ultravisor to complete the return
- The Heterogeneous Memory Management (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
  - 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 use a new syscall instruction (ESM), directed to the Ultravisor, to transition into secure mode



Revocation

Revocation: Disabling an SVM from executing on a machine where it was previously authorized.

- In the case where an SVM was multiply authorized
  - Single machine
  - Multiple Machines

Who is revoking access?

- User of an SVM
- Creator of the SVM
- Owner of the Infrastructure

Should it be reversible?

Considerations:

Users it is not clear that a user needs any form of revocation because they can always delete the SVM and they are not required to run any SVM.

Creator: SVM can be designed so that the creator can revoke authorization. They can use, call home, license server, fixed time period, etc. All of these techniques can be securely embedded within an SVM.

Infrastructure Owner: The base approach is heavy handed requiring a lot of administration. More flexible approaches include some sort of revocation list.

- A revocation list (in some form) is required.
- Basis for identification has to be decided.

Limitations

First release

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

Protected Execution Facility does not currently support

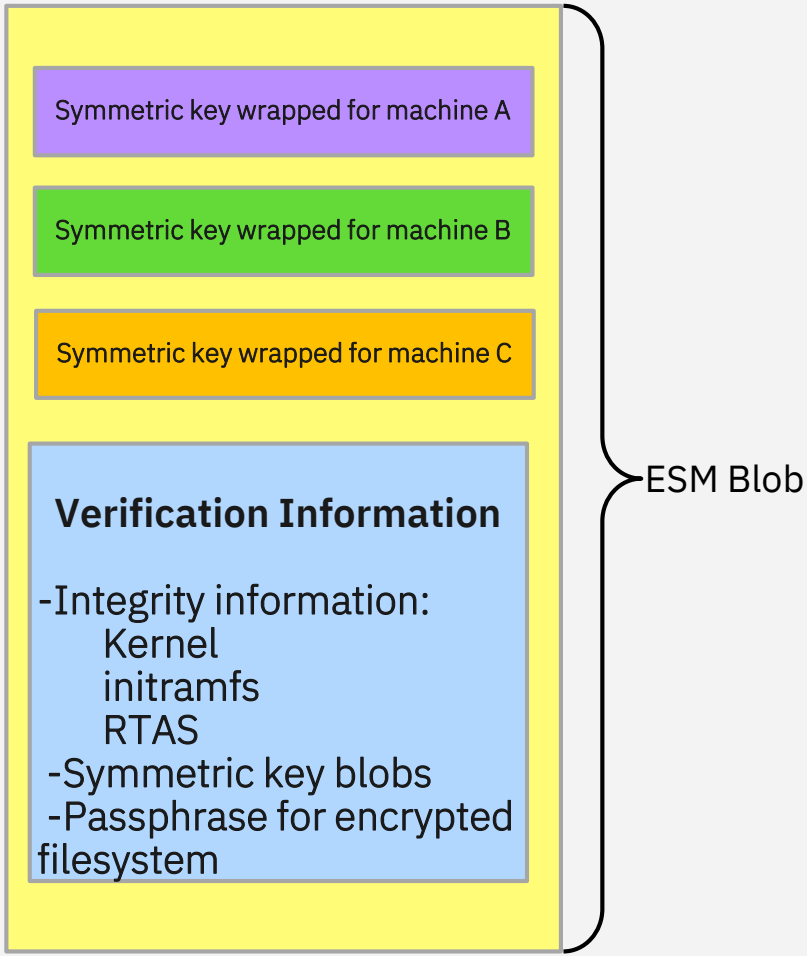
- Transaction memory (TM)
- Applications that use transaction memory if run in an SVMs will crash

Lower Level Details:
SVM, interfaces, kernel and hardware

Contents of ESM blob

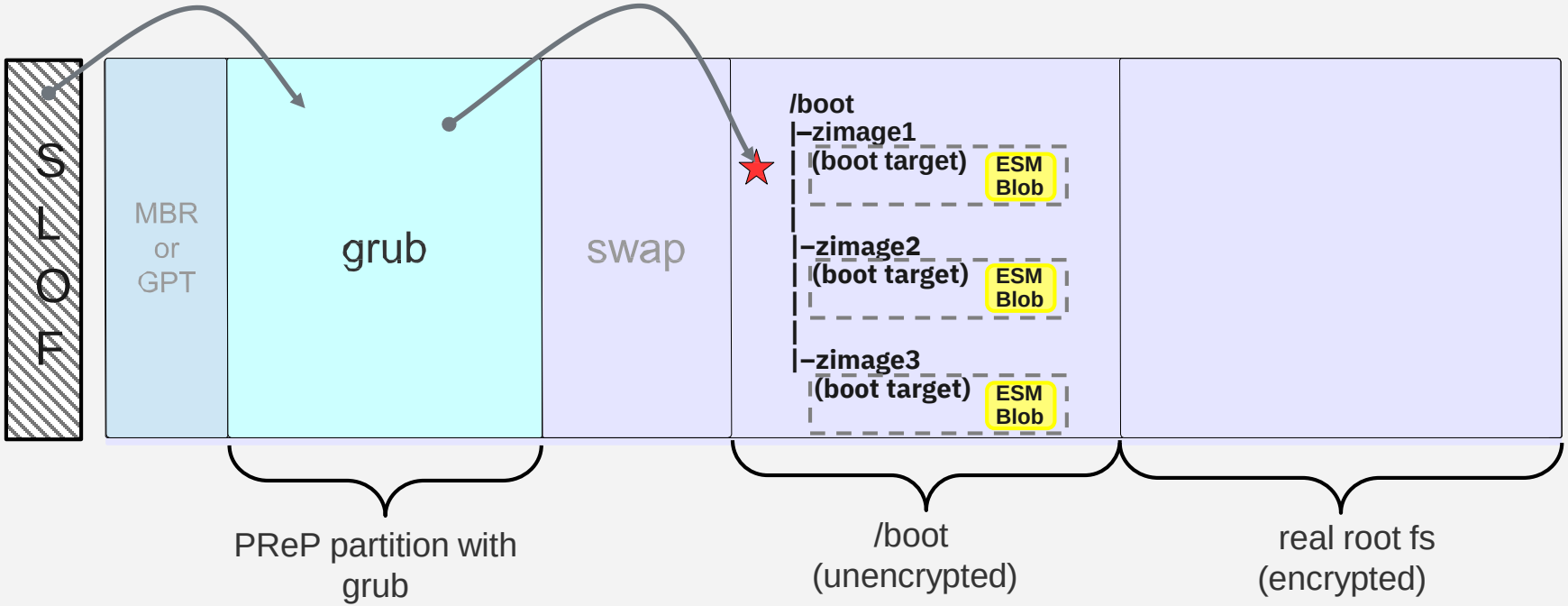
The symmetric key is encrypted with the public key of each machine the SVM is authorized to run on. Each blob is an index and a wrapped key. (index = hash of public key)

The ultravisor asks the TPM to use the private key of the machine (stored in TPM) to decrypt the symmetric key so it (ultravisor) can decrypt the verification information



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



Steps to start Secure VM

- Started as a Normal Virtual Machine
- At the end of prominit, SVM executes ESM Ultra Call
- Ultravisor copies VM into secure memory
- Searches for properly wrapped symmetric key
- If not found execution fails
- When found, decrypts verification information
- If decryption fails, verification fails
- Uses the verification information to confirm integrity of information
  - Kernel
  - Initram FS
  - RTAS information
- When Integrity confirmed return to SVM in secure mode
  - Pass phrase available through ultracall

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

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_SCOM
- UV_REGISTER_STOP_STATE
- UV_RESTRICTED_SPR_WRITE
- UV_PAGE_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 upstreamed in Linux4.18.0

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

https://patchwork.ozlabs.org/cover/973826/

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

https://lists.ozlabs.org/pipermail/linuxppc-dev/2018-August/177334.html

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 (OS)
1	0	1	problem
1	1	0	ultravisor
1	1	1	(reserved)

Normal

S	HV	PR	
0	0	0	privileged (OS)
0	0	1	problem
0	1	0	hypervisor
0	1	1	problem (HV)

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 (OS) 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

Summary of Protected Execution Facility

Feature	IBM Protected Execution Facility
Protection from	HV, other software, system admin
Security Domain	VM (at rest, in transit, or executing)
Application	No Changes to run in SVM
Guest OS	New Kconf options
Hypervisor	KVM must be para virtualized
Secure Memory	Integrity & Confidentiality
Embedded Secrets	Yes
Memory Size	Limited by available memory
Open Source	Yes

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

Questions?