OpenPOWER™ Summit NA 2019

AUGUST 19-20 | SAN DIEGO

Protected Execution Facility:

Secure computing for Linux on OpenPOWER

Guerney Hunt - Research Staff Member, IBM (gdhh@us.ibm.com)
Ram Pai – Senior Software Engineer, IBM (pair@us.ibm.com/linuxram@us.ibm.com)
Michael Anderson – Senior Software Engineer, IBM (andmike@us.ibm.com)

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.

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)

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 including zero-day attacks

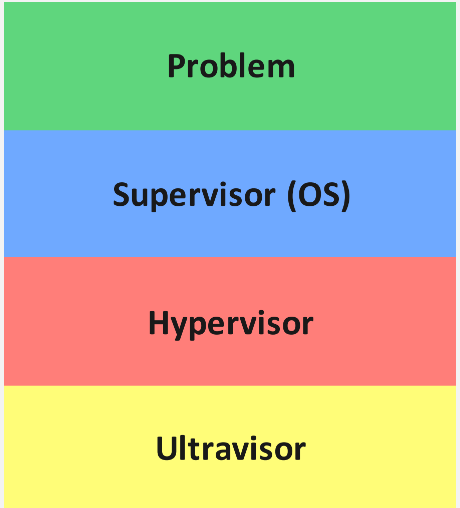
Objectives for Protected Execution Facility

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

Architecture Overview

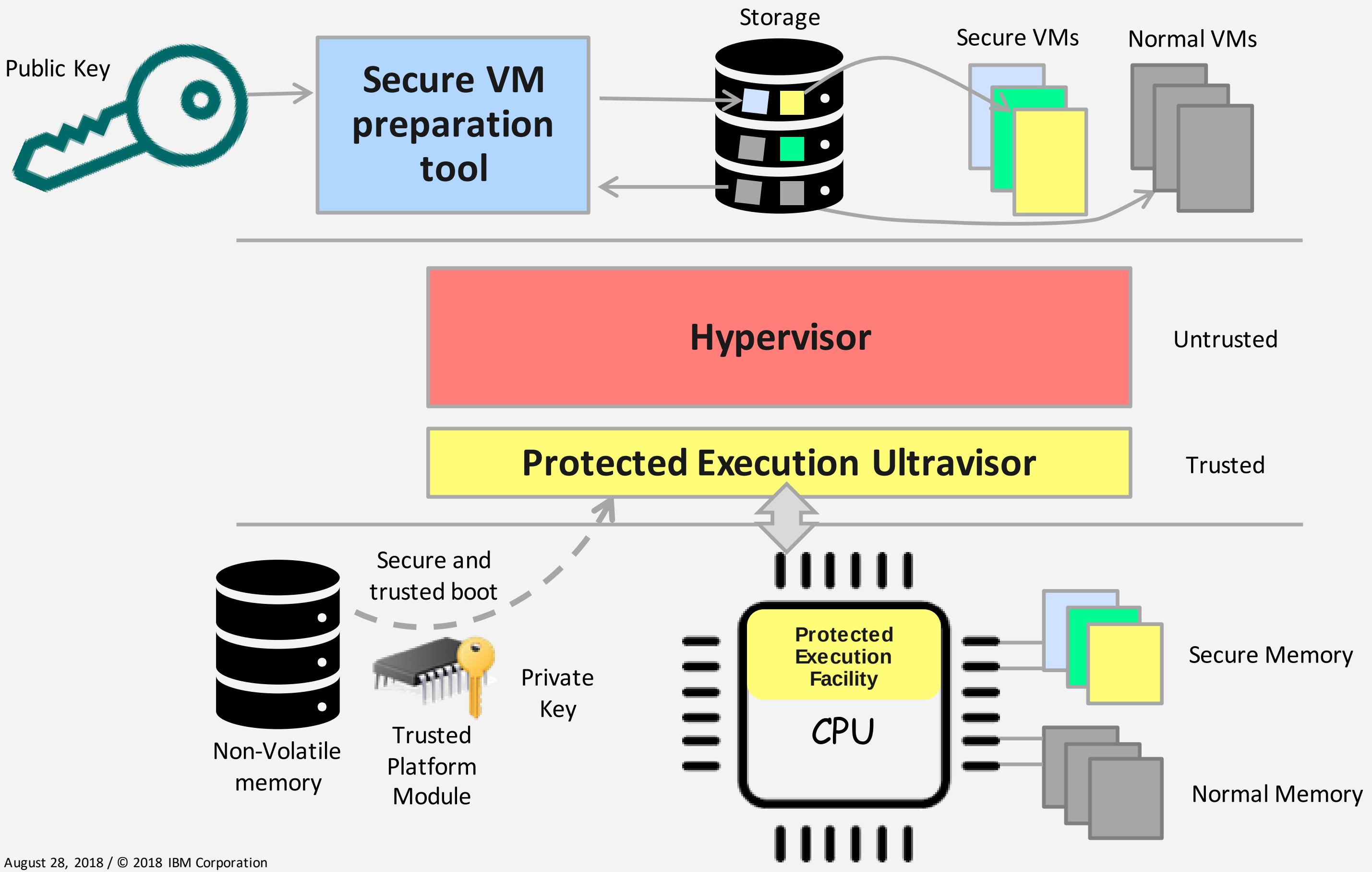
Base Principles

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



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

Overview of architecture

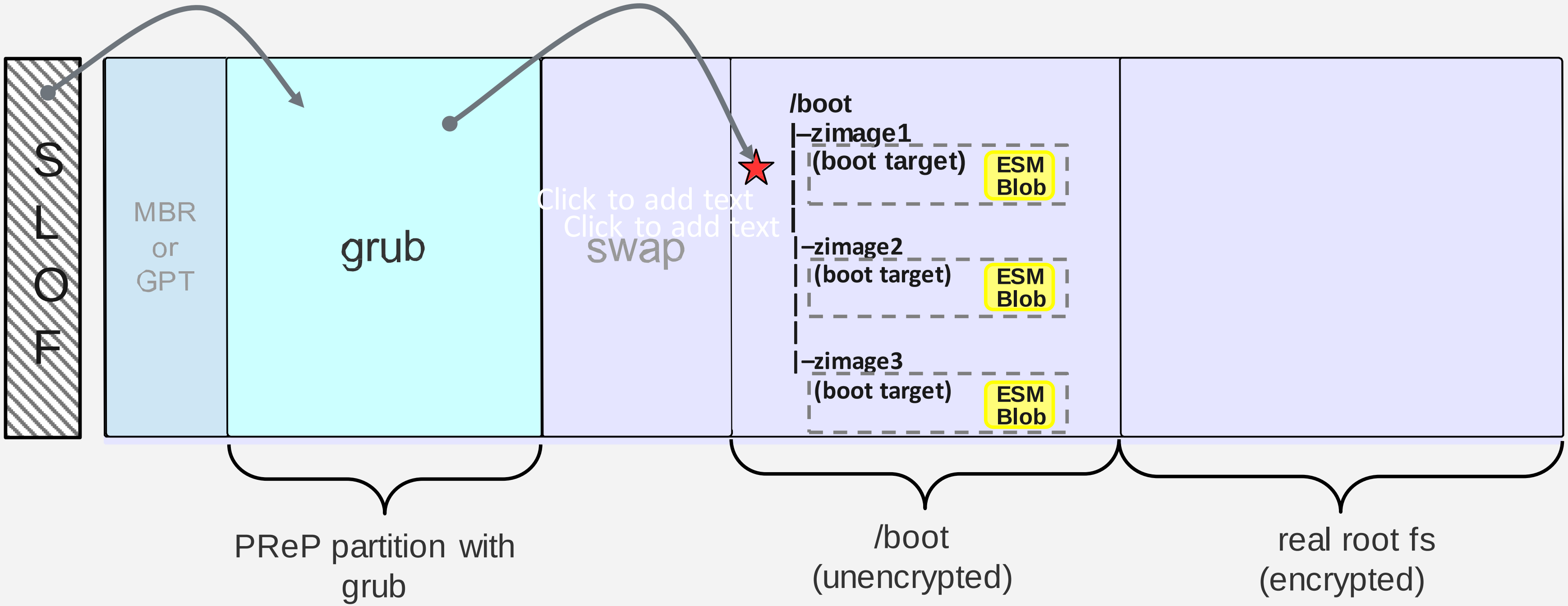


Overview of architecture

- 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

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
- Alternatively the tool can inject the ESM blob (data needed to assure SVM) in the initramfs

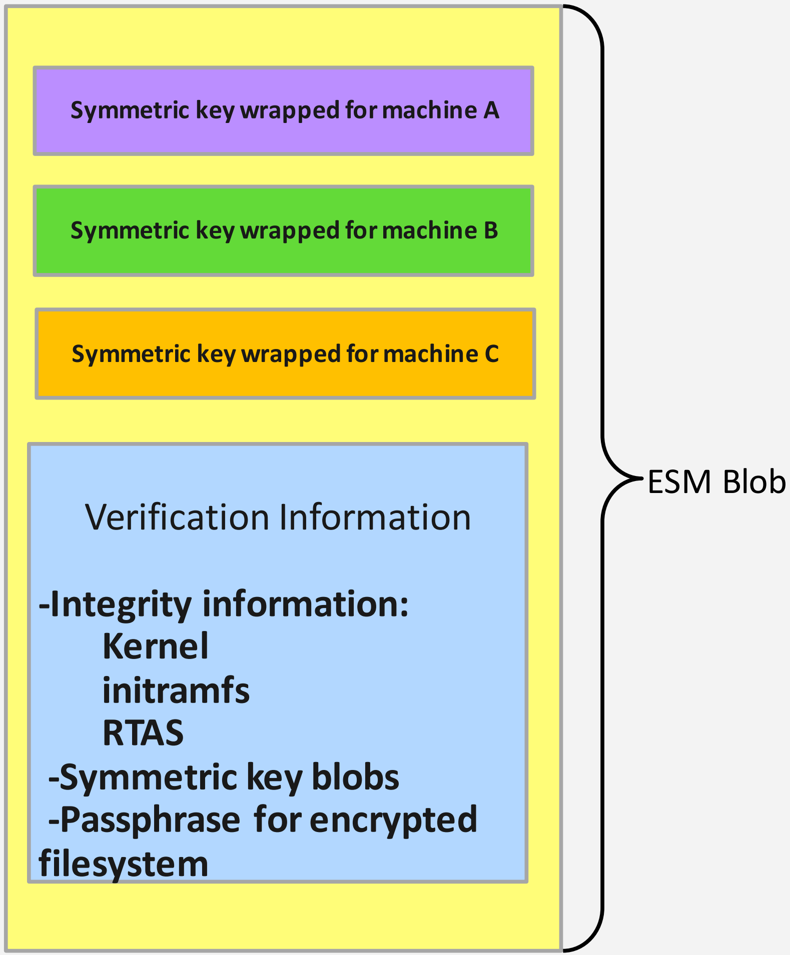


Contents of ESM operand

The symmetric key is encrypted with the public key of each machine the SVM is authorized to run on. Each lock box 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

Symmetric key blobs are customer supplied secrets



Hardware/Firmware Architecture:

Hardware changes and Ultravisor interfaces

Hardware changes

An address bit indicates a reference to secure memory

- Amount of secure memory is configurable

The MSR_S bit indicate that a running process is secure

$MSR_{S\ HV\ PR}$ determine privilege

Secure

S	HV	PR	
1	0	1	problem
1	0	0	privileged (OS)
1	1	0	ultravisor
1	1	1	(reserved)

Normal

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

New registers

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

New instruction

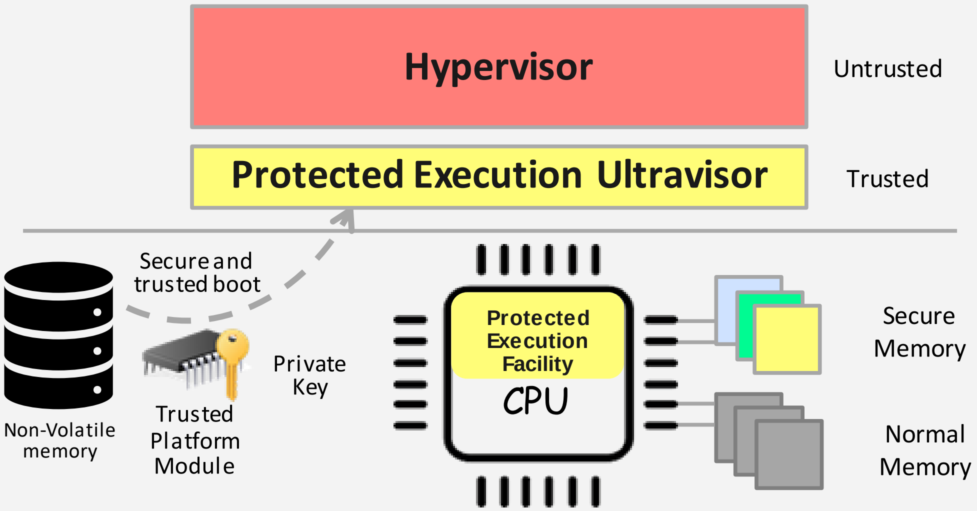
- URFID

- When MSR_S=1,

- All interrupts go to the Ultravisor
- Ultravisor responsible for handling or reflecting to the hypervisor as appropriate

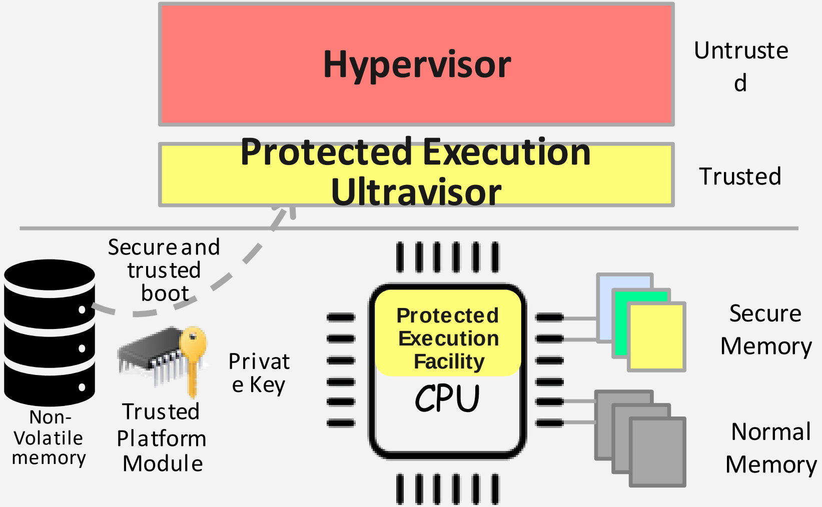
Architecture at the hardware/firmware level

- PEF relies on a root of trust. One such root of trust is the TPM available in OpenPOWER systems. The Ultravisor uses the TPM to get access to the symmetric key protecting the SVM.
  - Ultravisor has 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
- A new level of syscall is introduced, LEV=3, called an ultra call, which goes directly to the Ultravisor.
- When an ultra call is received, if the calling SVM has not been modified, the Ultravisor will transition it to secure mode



Results of changes

- When the processor is running in secure mode all interrupts are routed to the Ultravisor
- To protect the SVMs some resources which were previously hypervisor privileged are Ultravisor privileged.
- SVMs are protected at rest and can only run on an authorized machine
  - Authorization is determined by the SVM creator



Results continued

Unless a process is running in secure mode it cannot access secure memory

- Hypervisor must do an ultra call to access secure memory or ultravisor privileged resources.
- Hypervisor can only see secure memory encrypted (UV enforced)

– I/O systems cannot directly access secure memory

– SVM can request shared pages of memory with the hypervisor (no encryption protection).

– SVM can only address memory in its page table

- Hypervisor and NVM are protected from SVM.

– The Ultravisor only operates in response to an interrupt

- There is no timer interrupt that goes directly to the Ultravisor irrespective of the mode of the machine.

Ultravisor Resources

- The following resources have become Ultravisor privileged and require an Ultravisor interface to manipulate:

– Processor configurations registers (SCOMs)

– Stop state information

– PTCR and partition table entries (partition table is in secure memory).

- Paging for an SVM, sharing of memory with Hypervisor for an SVM. (Including VPA and virtual I/O.)

- The debug registers CIABR, DAWR, and DARWX become Ultravisor resources when SMFCTRL(D) is set. If SMFCTRL(D) is not set they do not work in secure mode. When set, reading and writing requires a Ultravisor call.

Software Architecture

Software Architecture

Hostboot

- Platform initialization

OPAL Boot-time services

- Device and Platform Initialization

Ultravisor

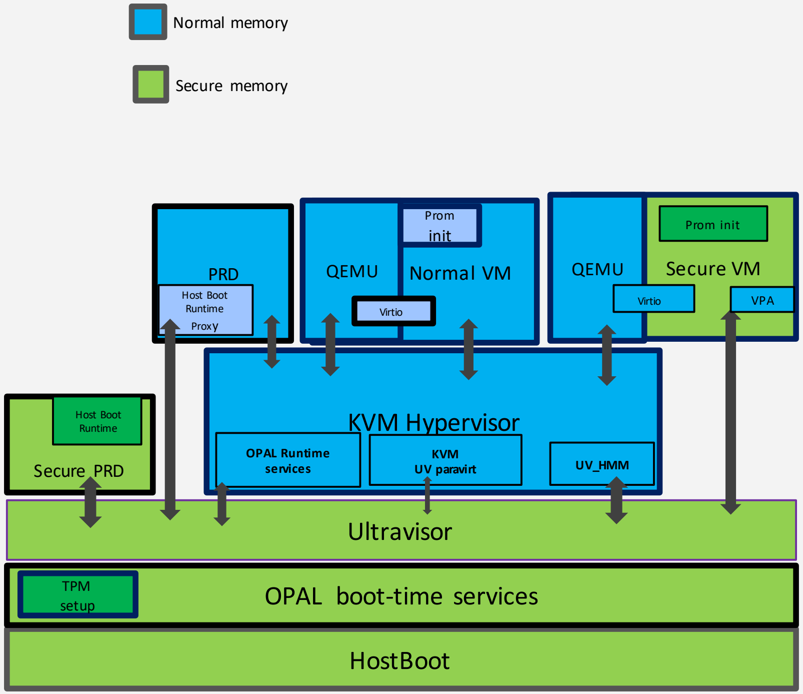
- Secure Memory Management
- Secure VM Services

Hypervisor

- VM Services

OPAL runtime services

- Platform and device management services



Software Architecture (continued)

Secure PRD/PRD/Host Boot Runtime/Host Boot Runtime Proxy

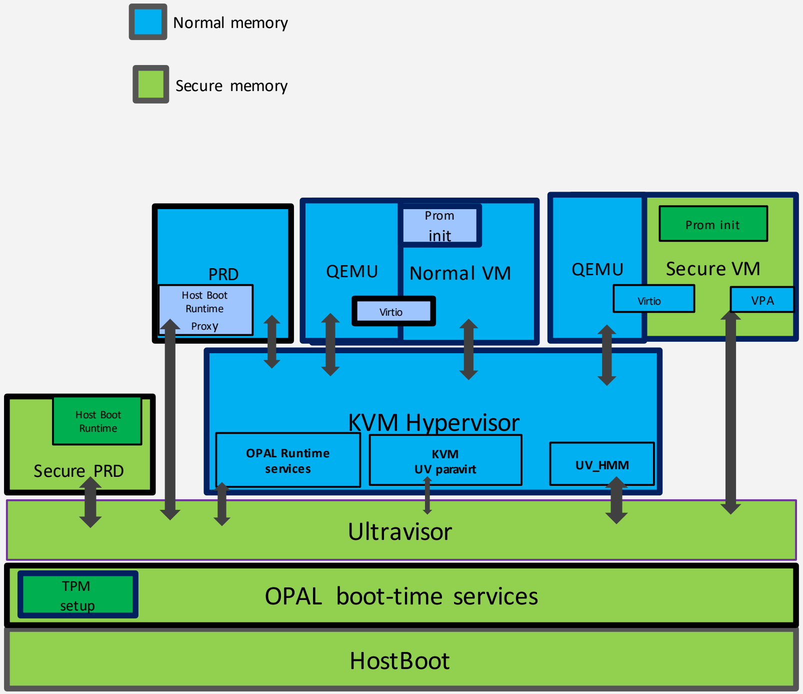
- Runtime Diagnostic and Corrective Service

Qemu - Virtual Platform and Device Emulator Process

- VM/SVM reside in its address space

UV_HMM

- Heterogenous memory manager for secure memory



Software Architecture(Continued)

Virtio

- Virtual Device Emulator

Prom_init

- Load and Initialize the kernel resources

VPA (Virtual Processor Area)

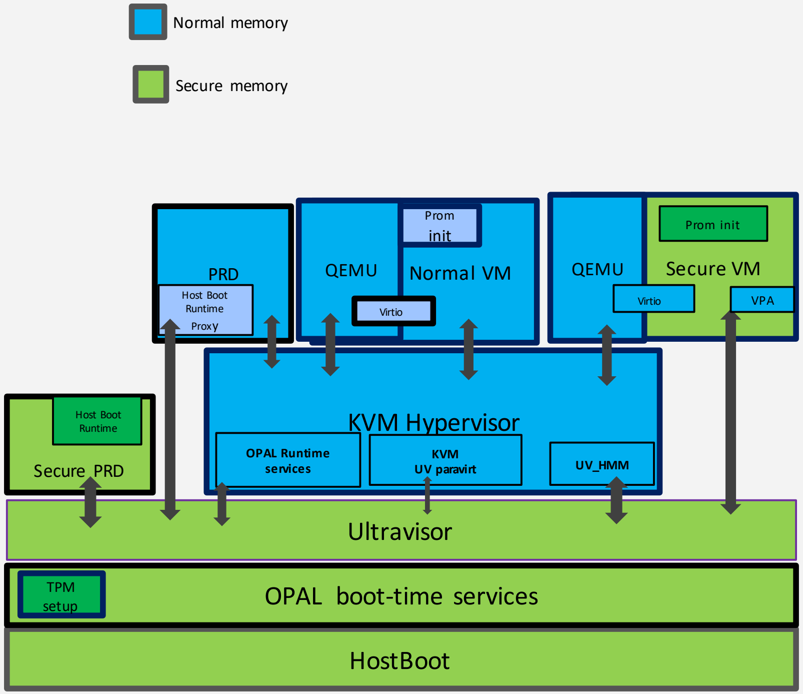
- Memory Area to communicate information between VM and Hypervisor

UV Paravirt

- Enables Hypervisor to run in a Ultravisor environment

TPM setup

- Initialize TPM



SVM Image build tools

ESM Blob generator

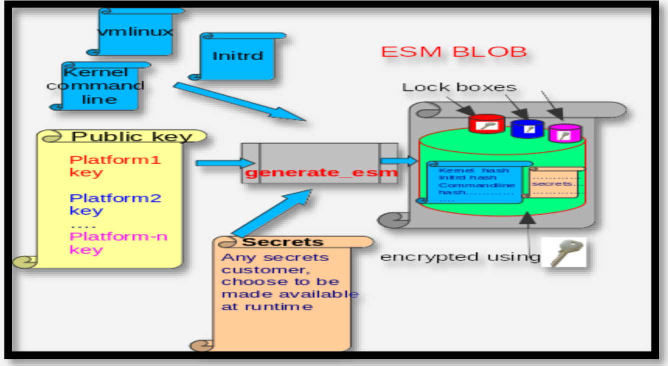
- Generates a binary blob that contains secrets and integrity information.

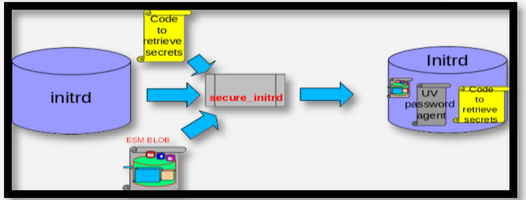
Initramfs Convertor

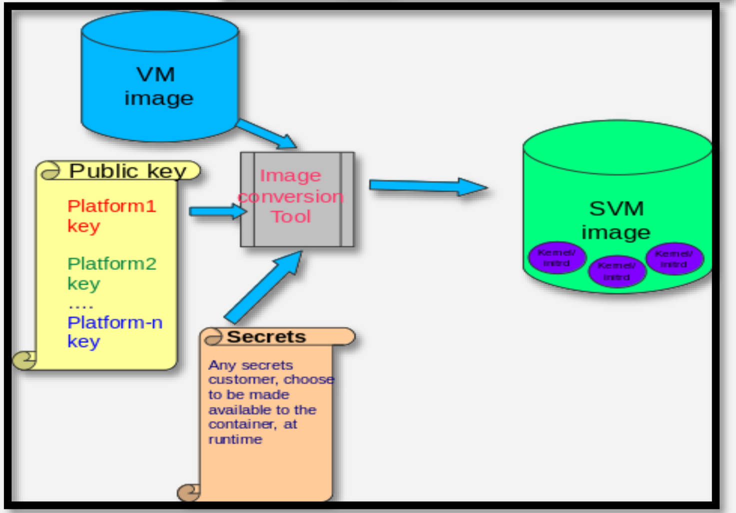
- Optionally, injects ESMblob into the initramfs.
- Makes a initramfs capable of securely mounting root filesystem.

VM image conversion tool

- Convert a VM image to Secure VM image
- Superset of the above two tools
- Creates ESMblob for each kernel/initramfs in the image.
- Injects ESMblob into its corresponding kernel/initramfs
- Regenerates boot table entries.







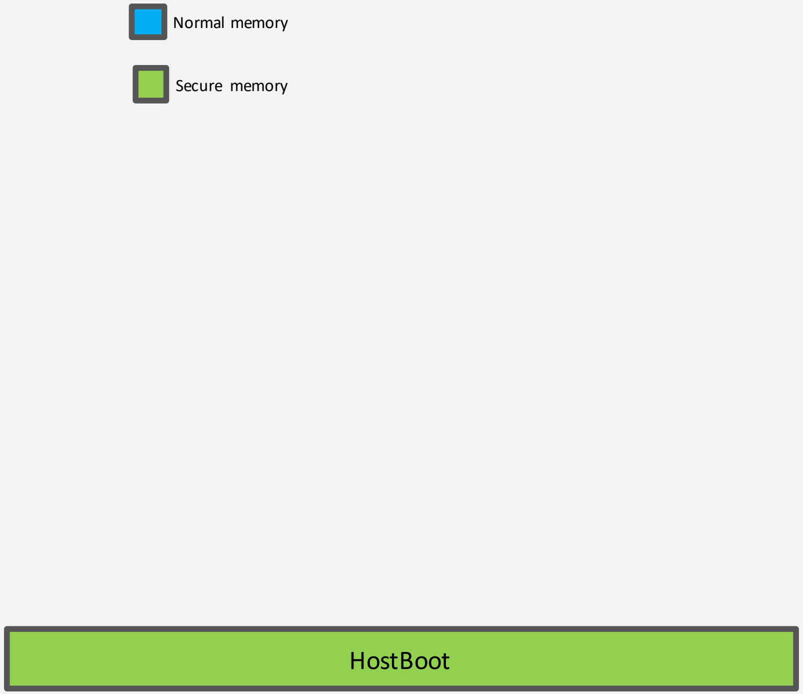
Setup and Boot Process

(a step by step narrated tour)

System Boot step 1

Power On

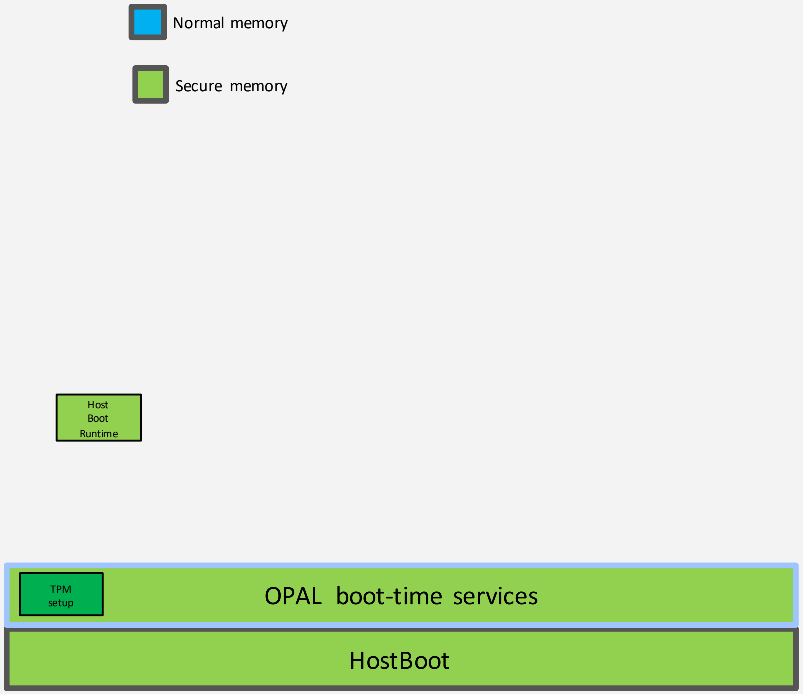
- Starts in Ultravisor Mode
- Loads HostBoot into the memory



System Boot step 2

Host Boot

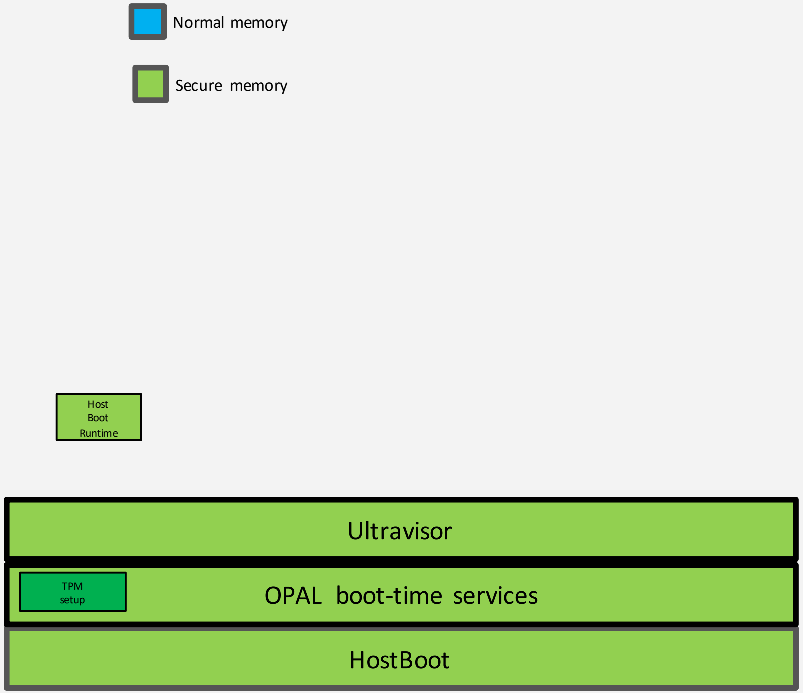
- Initializes the platform
- Configures Secure memory
- Create HDAT table
- Loads Host Boot Runtime
- Loads and executes OPAL



System Boot step 3

OPAL

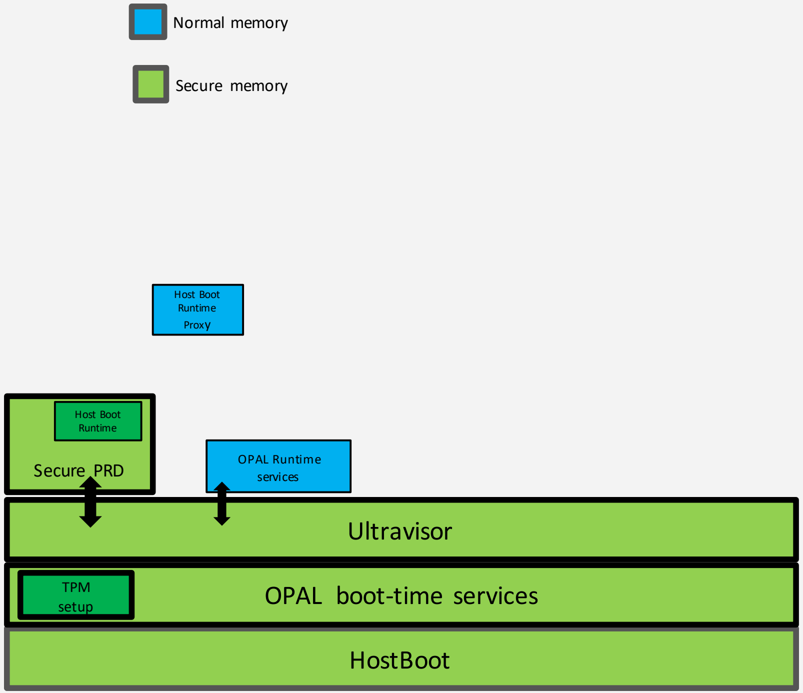
- Initializes the platform and devices
- Creates Device tree
- Initializes TPM with a TPM password
- Loads and executes Ultravisor
- Among other things, also passes the Device tree, TPM password to Ultravisor



System Boot step 4

Ultravisor

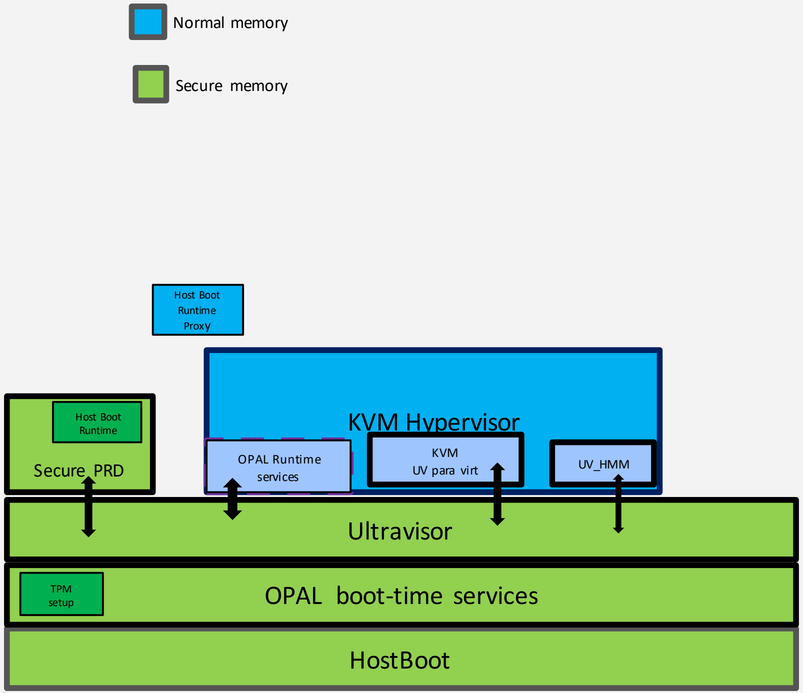
- Initializes secure memory.
- Sets up partition tables.
- Initialize data structures to
  - manage secure memory
  - secure virtual machines
  - Ultracalls/hypercall handling
- Synthesizes Secure PRD and initializes its contents
- Loads Host Boot runtime Proxy and initializes its contents
- Returns to OPAL switching the CPU mode to Hypervisor mode.



System Boot step 5

OPAL Runtime Services

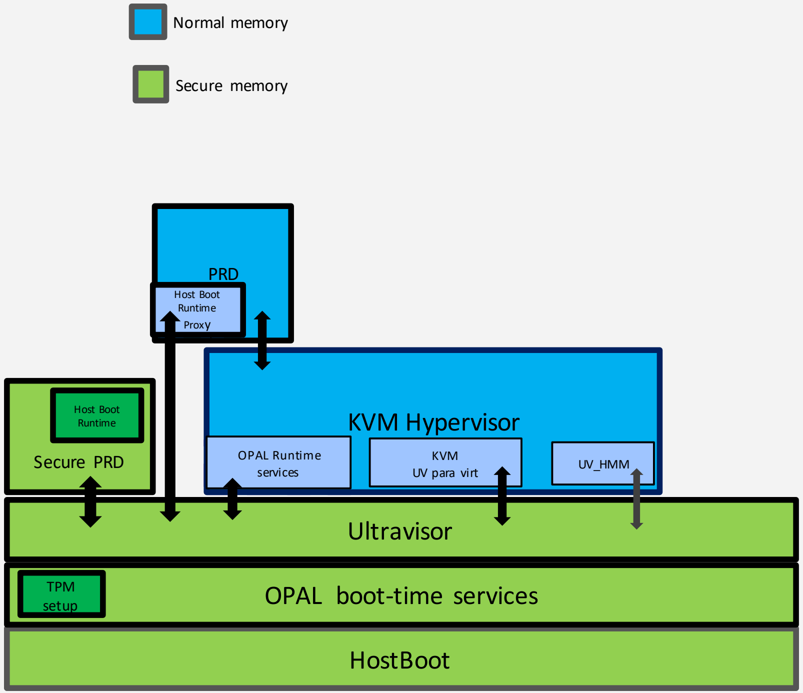
- Loads the Hypervisor.



System Boot step 6

Hypervisor

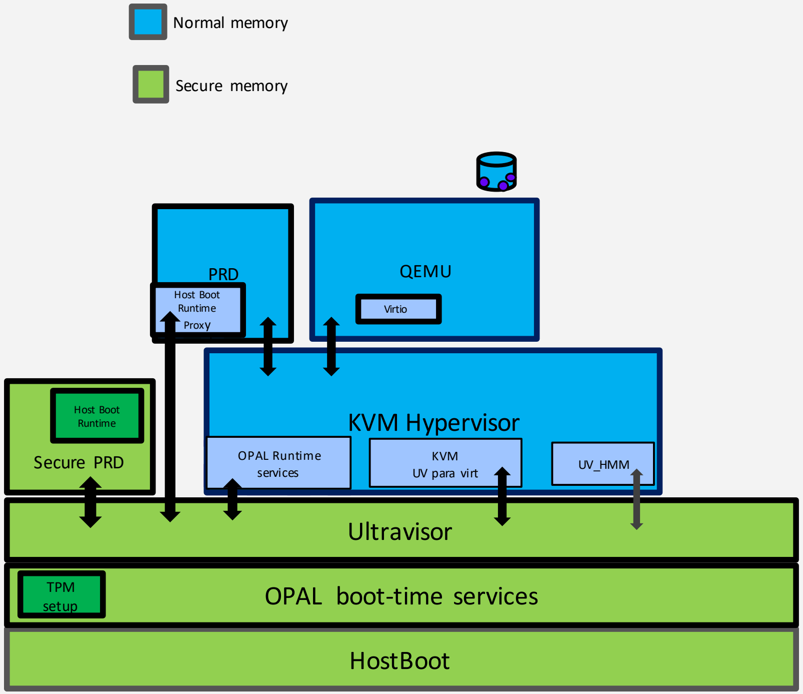
- Initializes normal memory.
- Initializes data structures to:
  - manage normal memory
  - normal virtual machines
  - hypercall handling
- Synthesizes PRD and initializes its contents
- Setup user space
- Loads and executes the init process



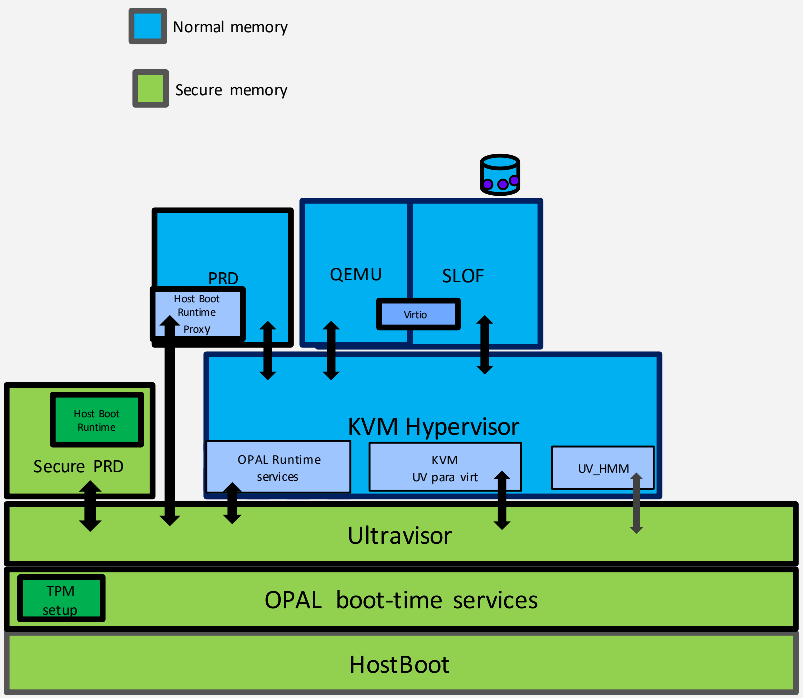
SVM boot step 1

User space

- Creates and deploys a Virtual Machine Image using QEMU.



SVM boot step 2



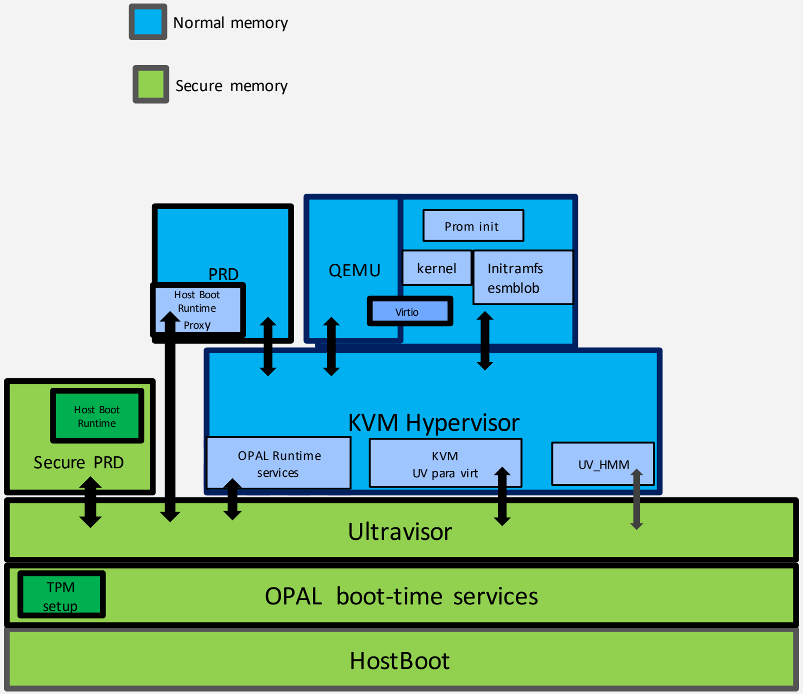
QEMU

- Registers a partition with the hypervisor.
- Initializes the virtio devices.
- Loads the firmware (SLOF) into its memory.
- Hands over control to SLOF in VM with the help of the hypervisor.

SVM boot step 3

Firmware(SLOF)

- Initializes its memory
- Loads the prom-init, kernel and initramfs/esm-blob into its memory.
- Hands over control to prom-init.
  - If grub is present, loads grub which in turn loads prom-init.



SVM boot step 4

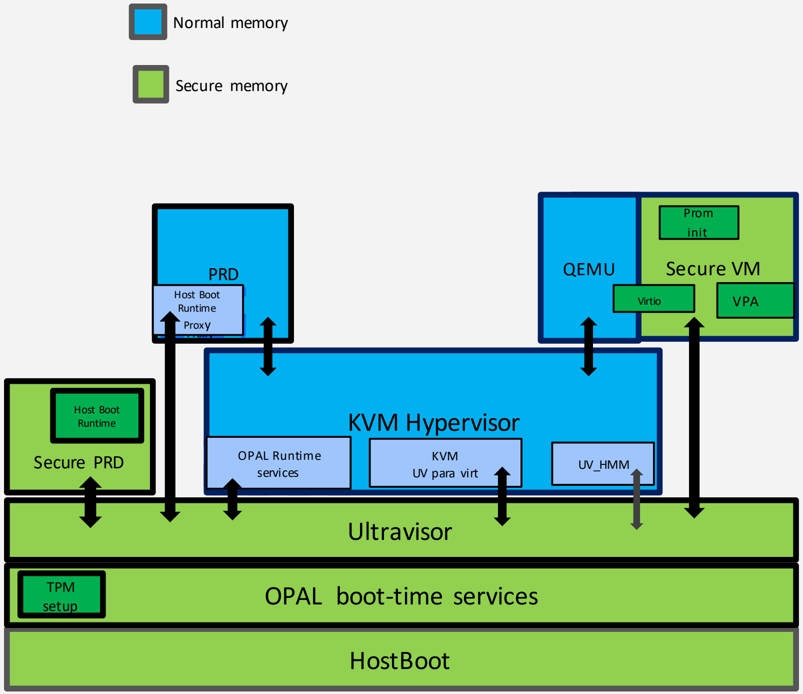
Prom init

- Initializes its datastructures.
- Ucalls to switch it to secure mode **

- Ultravisor
  - Allocates secure memory pages to the partition.
  - Moves the kernel, initramfs, and ESMblob to secure memory. (UV_HMM is invoked).
  - Unlocks the ESMblob with the help of the TPM.
  - If ESMblob checks-out, switches the partition to secure mode.
  - If ESMblob fails to check-out, terminates the partition.

- Hands over control to the kernel.

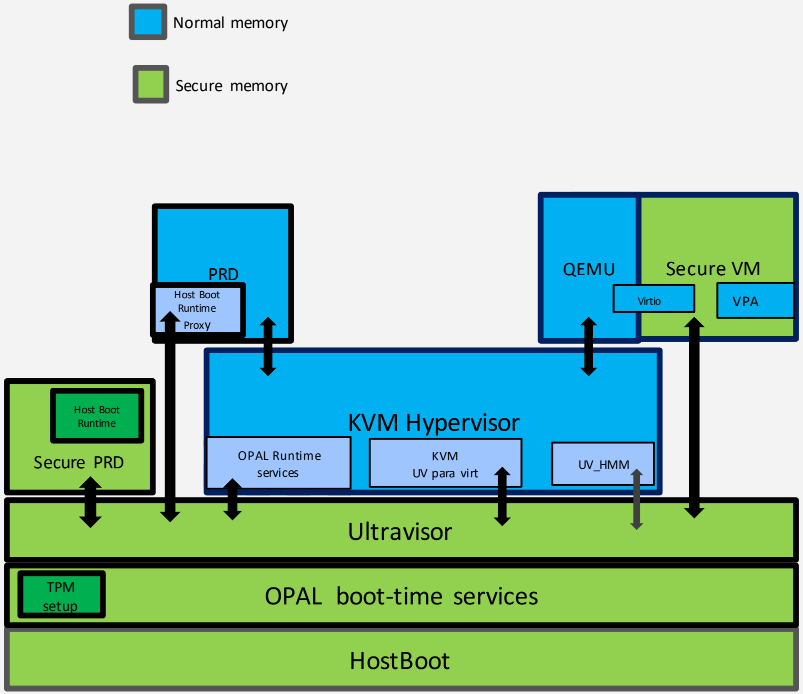
**applicable to secure VM, not to Normal VM



SVM boot step 5

Kernel (now running as an SVM)

- Initializes its data-structures.
- Ucalls to share VPA Pages with the Hypervisor**.
- Ucalls to share Virtio Pages with the Hypervisor**.
- Discards Prom-init
- Mounts initramfs
- Hands over control to the init process in the initramfs



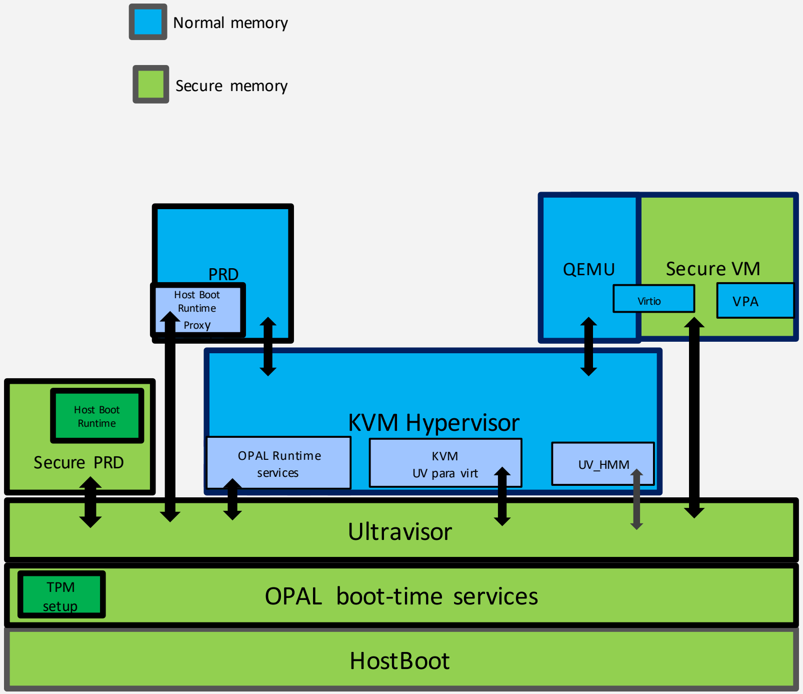
**applicable to secure VM, not to Normal VM

SVM boot step 6

Initramfs init process

- Ucalls to fetch the passphrase of the root filesystem.**
  - Ultravisor returns the passphrase from the ESMblob.
- Mounts the root filesystem.
- Executes the rest of the startup scripts.
- Reaches the login prompt!! Congrats!.

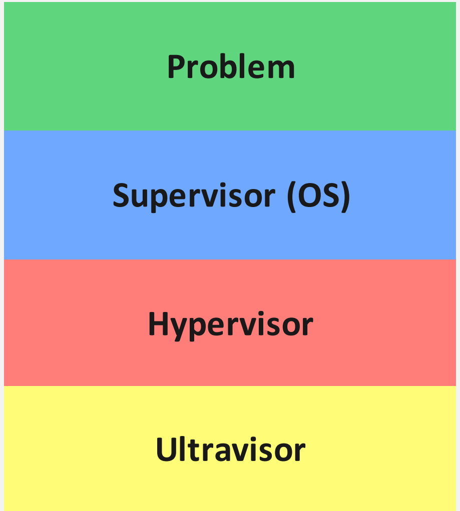
**applicable to secure VM, not to Normal VM



Summary

Summary

- Protected Execution Facility (PEF) provides a Trusted Execution Environment for POWER and OpenPOWER Linux systems.
- Enables Secure Virtual Machines (SVM).
- Normal VMs can be converted to SVMs.
- Entirely Opensource.
- A tech preview will be available prior to GA.



Questions?

Backup

Interfaces to the Ultravisor: ultra calls

An ultra call is a syscall instruction with Lev=2

These are the currently defined calls:

Used by Hypervisor

- UV_READ _SCOM
- UV_WRITE_SCOM
- UV_READ_MEM
- UV_WRITE_MEM
- 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

Used by SVM

- UV_SHARE_PAGE
- UV_UNSHARE_PAGE
- UV_UNSHARE_ALL_PAGES
- UV_ESM
- UV_ESMB_GETFILE

There will be additions to this list as we move forward**

Hypervisor Interfaces

New h-calls to support the Ultravisor:

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

There will be additions to this list as we move forward**

Limitations

First release

- Suspend/Resume/Migration.
- Over commit of Secure memory.
- Transaction memory facility for SVM applications.

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