DRAFT (2nd) NIST Special Publication 800-125A

Security Recommendations for Hypervisor Deployment

Ramaswamy Chandramouli

C O M P U T E R   S E C U R I T Y

National Institute of
Standards and Technology
U.S. Department of Commerce

DRAFT (2nd) NIST Special Publication 800-125A

Security Recommendations for Hypervisor Deployment

Ramaswamy Chandramouli
Computer Security Division
Information Technology Laboratory

September 2017

U.S. Department of Commerce
Wilbur L. Ross, Jr., Secretary

National Institute of Standards and Technology
Kent Rochford, Acting NIST Director and Under Secretary of Commerce for Standards and Technology

Authority

National Institute of Standards and Technology Special Publication 800-125A
Natl. Inst. Stand. Technol. Spec. Publ. 800-125A, 34 Pages (September 2017)
CODEN: NSPUE2

Public comment period: *September 14, 2017 through October 6, 2017*

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: sp800-125A-comments@nist.gov

Reports on Computer Systems Technology

Abstract

The Hypervisor is a collection of software modules that enables virtualization of hardware resources (such as CPU, Memory, Network and Storage) and thus enables multiple computing stacks (basically made of an OS and Application programs) called Virtual Machines (VMs) to be run on a single physical host. In addition, it may have the functionality to define a network within the single physical host (called virtual network) to enable communication among the VMs resident on that host as well as with physical and virtual machines outside the host. With all this functionality, the hypervisor has the responsibility to mediate access to physical resources, provide run time isolation among resident VMs and enable a virtual network that provides security-preserving communication flow among the VMs and between the VMs and the external network. The architecture of a hypervisor can be classified in different ways. The security recommendations in this document relate to ensuring the secure execution of baseline functions of the hypervisor and are therefore agnostic to the hypervisor architecture. Recommendations for secure configuration of virtual network are dealt with in a separate NIST document (SP 800-125B).

Keywords

Virtualization; Hypervisor; Virtual Machine; Virtual Network; Secure Configuration**;** Security Monitoring; Guest OS

Acknowledgements

The author, Ramaswamy Chandramouli wishes to thank his colleague Tim Grance for his personal input on the content and in helping with the logistics of the publication.

Table of Contents

EXECUTIVE SUMMARY

Server Virtualization is now an established technology in data centers used as enterprise IT infrastructure and in those offered for cloud service, as it brings about better utilization of hardware resources, saves/reduces physical space in data centers and reduces power consumption. Server Virtualization is realized using a collection of software modules called Hypervisor that enables virtualization of hardware resources (such as CPU, Memory, Network and Storage) and thus enables multiple computing stacks (basically made of an OS (called Guest OS) and application programs) called Virtual Machines (VMs) to be run on a single physical host.

At first glance it might appear that all activities related to secure management of a hypervisor and its hardware host (collectively called Hypervisor Platform) should consist of just the established state of art practices for any server class software and its hosting environment. However, closer examination reveals that functions for supporting hardware virtualization that a hypervisor provides have extensive security ramifications and hence require a focused set of security recommendations based on analysis of threats to the secure execution of these functions.

Due to multiple ways by which an architecture of a hypervisor can be classified, the approach taken in this document, is to identify the baseline functions that a hypervisor performs, identify the tasks involved in each baseline function, identify the potential threats to secure execution of the task and express the countermeasures that provide assurance against exploitation of these threats in the form of security recommendations.

The following five were identified as baseline functions of a hypervisor.

- VM Process Isolation
- Devices Emulation & Access Control
- Execution of Privileged Operations by Hypervisor for Guest VMs
- VM Lifecycle Management
- Management of Hypervisor

Apart from providing security recommendations for ensuring secure execution of above functions, a recommendation for ensuring the overall integrity of all components of a hypervisor platform is also provided.

Secure execution of routine administrative functions for the physical host where the hypervisor is installed is not covered in this document since they are addressed in several other NIST documents. The protection requirements for Guest OS and applications running on VMs and associated security recommendations are also beyond the scope of this document.

1. INTRODUCTION, SCOPE & TARGET AUDIENCE

The Hypervisor is a collection of software modules that enables virtualization of hardware resources (such as CPU, Memory, Network and Storage) and thus enables multiple computing stacks (basically made of an OS and application programs) to be run on a single physical host. Such a physical host is called a Virtualized Host (also referred to as Hypervisor Host in this document) and the individual computing stacks are encapsulated in an artifact called Virtual Machines (VMs). To be an independent executable entity, the definition of a VM should include resources such as CPU, Memory etc. allocated to it. The VMs are also called "Guests" and the operating system (OS) running inside each of them as "Guest OS". The resources associated with a VM are virtual resources as opposed to physical resources associated with a physical host.

The hypervisor forms part of the virtualization layer in a virtualized host and plays many of the roles a conventional OS does on a non-virtualized host (server). Just as a conventional OS provides isolation between the various applications (or processes) running on a server, the hypervisor provides isolation between one or more VMs running on it. Also, similar to an OS, the hypervisor mediates access to physical resources across multiple VMs. Hence, it is no surprise that hypervisors have as their foundation the kernel of an OS. All other functions needed to support virtualization- such as emulation of network & storage devices and management of VMs & hypervisor itself can all therefore be accomplished using kernel loadable modules (i.e., extending the kernel), though some hypervisor architectures accomplish these tasks using dedicated, privileged VMs (also called Management VMs).

At first glance it might appear that all activities related to secure management of a hypervisor and its hardware host (collectively called Hypervisor Platform) should consist of just the established state of art practices for any server class software and its hosting environment. However, closer examination reveals that functions for supporting hardware virtualization that a hypervisor provides have extensive security ramifications and hence require a focused set of security recommendations based on analysis of threats to the integrity of these functions. These functions that support hardware virtualization that a typical hypervisor provides are called hypervisor baseline functions in this document.

The hypervisor baseline functions consist of:

(a) VM Process Isolation
(b) Devices Emulation & Access Control
(c) Execution of Privileged operations by Hypervisor for Guest VMs
(d) VM Lifecycle management
(e) Management of hypervisor

A brief description of the above functions is given in section 1.1 below.

1.1 Hypervisor Baseline Functions

While the basic function of a hypervisor is to virtualize the hardware (a physical host) to enable running of multiple virtual hosts (popularly known as Virtual Machines (VM)), commercial hypervisor offerings come with differing feature sets. Also, the modules that provide the same set of features are given different names in different product offerings. Hence, for accomplishing the goals of this document, it is necessary to identify a set of baseline features of a hypervisor that covers all functions for supporting hardware virtualization. In some instances, the module that just presents a set of virtualized resources to the VMs is called the Virtual Machine Manager (VMM), and this together with the modules that provides OS-level services such as scheduling of VMs in the CPU is called the hypervisor. These hypervisor baseline features or functions are:

- HY-BF1: VM Process Isolation – Scheduling of VMs for execution, Management of the application processes running in VMs such as CPU and Memory Management, context switching between various processor states during the running of applications in VMs etc.
- HY-BF2: Devices Emulation & Access Control – Emulating all Network and Storage (block) devices that different native drivers in VMs are expecting, mediating access to physical devices by different VMs.
- HY-BF3: Execution of Privileged Operations for Guest VMs – Certain operations invoked by Guest OSs, instead of being executed directly by the host hardware, may have to be executed on its behalf by the hypervisor, because of their privileged nature.
- HY-BF4: VM Lifecycle management – This involves all functions from creation and management of VM images, control of VM states (Start, Pause, Stop, etc.),VM migration, VM monitoring and policy enforcement.
- HY-BF5: Management of hypervisor– This involves defining some artifacts and setting values for various configuration parameters in hypervisor software modules including those for configuration of a Virtual Network inside the hypervisor.

.

The brief description of the five baseline functions listed above is sufficient to guide the discussion in the rest of the document. For interested readers, a detailed description of the above functions is given in Appendix A.

The above functions are carried out by different hypervisor components or software modules. There are some minor differences among hypervisor products in the way they distribute these functions. The mapping of these functions to hypervisor components and the location of these components in overall hypervisor architecture are given in table 1 below:

Baseline Function	**Component (Software Module)**	**Location**
VM Process Isolation (HY-BF1)	Hypervisor Kernel	Either an OS kernel (along with a kernel module) itself or a component installed on a full-fledged OS (Host OS)
Devices Emulation & Access Control (HY-BF2)	Emulator software (e.g., QEMU)	Either in a dedicated VM (called as parent partition or management VM) or in the hypervisor kernel itself
Execution of Privileged Operations for Guest VMs (HY-BF3)	Hypervisor Kernel	These pertain to only paravirtualized hypervisors and are handled by hypercall interfaces in that type of hypervisor
VM Lifecycle Management (HY-BF4)	A management daemon	Installed on top of hypervisor kernel but runs in unprivileged mode
Management of Hypervisor (HY-BF5)	A set of tools with CLI (command line interface) or a GUI	A console or shell running on top of hypervisor kernel

Hypervisor products differ in the distribution of the above functions and in the name assigned to the corresponding component. In general, functions HY-BF1 and HY-BF3 are offered by modules running in a kernel collectively called “Hypervisor” while HY-BF2 is enabled either using a module called QEMU that runs outside the hypervisor (usually in a dedicated privileged VM) or in the hypervisor kernel itself in some instances. The functions HY-BF4 and HY-BF5 are performed by a module called management or service console or through a kernel module. Just like QEMU module, the console is a software layer that is generally not built into the hypervisor kernel but runs on top of it as a privileged VM and could be built either with a full-fledged OS installed inside in it or with a ultra-light OS used to present an API (shell and network access) with utility functions that facilitate performing just the hypervisor-specific configuration and administrative tasks.

1.2 Scope of this document

The architecture of a hypervisor can be classified in different ways:

(a) The entity over which the hypervisor installs – bare metal (directly on hardware) or over a full-fledged conventional Operating System (called Host OS). The resulting products are called Type 1 hypervisor and Type 2 hypervisor respectively

(b) Full Virtualization Vs Para Virtualization – In the former, the Guest OS runs unmodified in the VM, while in the latter the Guest OS kernel must be modified to generate hyper calls.

The trust model assumed for this document is as follows:

- All components in a VM is untrusted – the guest OS and its associated utilities (e.g., virtual device drivers) that run in the kernel space and all applications that run in the user space
- The physical device drivers that are part of the hypervisor software modules (that provide access to physical devices on the virtualized host) are untrusted unless they carry a security certification
- The hypervisor kernel component (that provides isolation between VMs) is trusted

With the background information on hypervisor architecture and the assumed trust model, the scope of security recommendations in this document with regard to the five baseline functions HY-BF1 through HY-BF5 are stated as follows:

- All features under HY-BF1, HY-BF2 and HY-BF4.
- HY-BF3 relates to handling of hypercalls in paravirtualized hypervisors. This is a trusted function of the hypervisor and hence not included in the security recommendations.
- All features under HY-BF5 are included, excepting those related to definition and configuration of virtual networks. Secure configuration of virtual networks is covered under a separate NIST document (SP 800-125B).

Also within the scope is a recommendation to ensure overall platform integrity.

The security recommendations do not cover the following aspects:

- Hypervisor host user account management
- Hypervisor host authentication and access control
- Routine administration of Host OS and Guest OS (e.g., keeping patches current)
- Security of Guest OSs running on VMs
- Security of Applications/Services running on VMs

1.3 Target Audience

The target audience for the security recommendations in this document is the following:

- The Chief Security Officer (CSO) or the Chief Technology Officer (CTO) of an Enterprise IT department in a private enterprise or government agency, who wants to develop a virtualization infrastructure to host various Line of Business (LOB) application systems on Virtual Machines (VM).
- The Managers of data centers who want to offer a virtualization infrastructure for hosting cloud offerings such as Infrastructure as a Service (IaaS) and who want to provide security assurance for that infrastructure to the cloud service clients.

1.4 Relationship to other NIST Guidance Documents

In terms of technology area, the NIST Guidance document that is related to this document is "NIST Special Publication 800-125 – Guide to Security for Full Virtualization Technologies". Consistent with the state of technology adoption at that time (SP 800-125 was published in January 2011), SP 800-125 provided higher-level security recommendations for use of components involved in two applications of virtualization paradigm – Server Virtualization & Desktop Virtualization. Since that time, the Server virtualization has found widespread adoption in IT data centers both for hosting in-house or on premise (enterprise) applications as well as for hosting applications and providing computing units for offering cloud services.

Accompanying this technology adoption trend is the increase in feature set of one of the core layer of virtualization – the hypervisor, as well as the market availability of the set of tools used for configuration and administration of the virtualized infrastructure spawned by the hypervisor. The objective of this document is to focus on the development of a set of security recommendations for deployment of the hypervisor (with all its constituent modules) including the steps involved in creation and provisioning of VMs. The distinguishing features of the set of security recommendations provided in this document in the context of similar NIST Guidance documents are given below:

- Focused set of security recommendations that are agnostic to the architecture for the deployment of Server Virtualization technology's foundational component– i.e., the Hypervisor.
- Since a realworld deployment includes provisioning of VMs, all VM lifecycle operations from creation and management of VM images to their administration using granular privileges is covered.
- Recognizing that the Hypervisor is nothing but an Operating System (OS) kernel and the security of a server OS – no matter the distribution – depends upon its weakest link, such as driver software, security recommendations relating to these components have been provided as well.
- Recognizing that the hypervisor has to perform certain privileged operations without interference from any other entity in the virtualized host and that leveraging hardware support for these operations will make a big difference to the overall security of hypervisor deployment. As an additional benefit, it also improves performance when virtualization specific functions (e.g., memory tables for multiple VMs) are offloaded (leveraged) to the processor instead of through software functions.
- Last but not the least, all security recommendations are intended to provide assurance against exploitation of threats to tasks involved in hypervisor's baseline functions.

2. APPROACH FOR DEVELOPING SECURITY RECOMMENDATIONS

Developing security recommendations for deployment and usage of a complex software such as the hypervisor requires knowledge of potential threats that when exploited would affect the basic three security properties of confidentiality, integrity and availability of hypervisor functions. The approach adopted for developing security recommendations for deployment of hypervisor in this document is as follows:

- Ensure the integrity of all components of the hypervisor platform – starting from the host BIOS to all software modules of the hypervisor. This is accomplished through a secure boot process outlined as recommendation HY-SR1 in section 3.
- Identify the threat sources in a typical hypervisor platform. The nature of threats from rogue or compromised VMs are briefly discussed (Section 2.1).
- For each of the five baseline functions HY-BF1 through HY-BF5 (with the exception of HY-BF3 (Execution of privileged operations by the hypervisor)), identify the different tasks under each function and for each of the tasks, identify the potential threats to the secure execution of the task. The counter measures that will provide assurance against exploitation of these threats form the basis for security recommendations (Section 2.2).

It must be mentioned that in some cases of large software environments (both open source and commercial) such as DBMS platform, the approach adopted for secure deployment/usage is to study the reports published in the public vulnerability databases for various product offerings, look for availability of patches (available through community/forum postings or from software vendor) or look for recommended secure configuration settings (again posted in public forums/blogs or vendor websites). We do not adopt this approach in this document since the intended purpose is not to provide security recommendations for a specific open source or commercial hypervisor product offering but for the entire product class based on its baseline functions.

2.1 Hypervisor Platform –Threat Sources

The hypervisor software is resident on a physical host that is connected to the enterprise network. It has the capability to be remotely administered. At the same time, it is supporting multiple virtual hosts (or virtual machines or VMs) that are nodes of a software-defined virtual network inside that physical host. Based on these scenario, one can identify three basic sources of threats for a hypervisor platform. (Each threat is identified by using the symbol HY-TS#).

- HY-TS1: Threats from and through the enterprise network in which the hypervisor host (virtualized host) resides.
- HY-TS2: Threats emanating from rogue or compromised VMs through channels such as shared hypervisor memory and virtual network inside the hypervisor host.
- HY-TS3: Threats from web interfaces to VM management daemon and hypervisor management console.

Threats from sources HY-TS1 and HY-TS3 are common to all server class software and are well known and are addressed in other NIST documents. Threats from source HY-TS2 is unique to the virtualization environment defined by the hypervisor. We look at the nature of threats from this threat source (consisting of VMs and the virtual network inside the hypervisor host) in the next subsection.

2.1.1 *Nature of Threats from VMs & Virtual Network*

The hypervisor controls VM access to physical hardware resources as well as provides isolation among VMs. VM access to hardware resources such as CPU and memory are directly controlled by the hypervisor while access to resources such as network and storage devices are controlled through modules (drivers) that are resident in the kernel module or in a privileged VM (i.e., Management VM). The network isolation among VMs is provided by assigning a unique IP or MAC address to each VM, by defining virtual local area networks (VLANs) or overlay networks and by assigning the appropriate network identifier to each VM. The nature of threats to the hypervisor from rogue or compromised VMs can manifest in the following ways:

Breach of Process Isolation - VM Escape (HYP-T1): The first threat to any hypervisor is from rogue VMs. The rogue VMs are the ones which manage to subvert the isolation function provided by the VMM/Hypervisor to hardware resources such as memory pages and storage devices. In other words, the rogue or compromised VMs may access areas of memory belonging to the hypervisor or other VMs and storage devices they are not authorized. The possible reasons for this threat are: (a) hypervisor design vulnerability (b) malicious or vulnerable device drivers. Potential downstream impact of a rogue VM taking control of the hypervisor is: (a) installing rootkits, and (b) attacking another VM on the same virtualized host.

Breach of Network Isolation (HYP-T2): Potential threats to isolation are from attacks such as: (a) IP or MAC address spoofing by a rogue VM (b) VLAN hopping – a rogue VM escaping the boundaries of its VLAN and (c) Traffic Snooping – intercepting virtual network traffic, intended for a VM on the same virtual network segment. The impact of the subversion of these network controls is loss of confidentiality – some VMs will be viewing information they are not authorized for.

Denial of Service (HYP-T3): Misconfigured or malicious VMs, may be consuming a disproportionately high percentage of host resources resulting in denial of service to other VMs in the hypervisor host.

Privileged Interfaces provided by Hypervisor (HYP-T4): Hypervisors provide privileged interfaces (generally called by the name Introspection API) to virtual security appliances (such as IPS/IDS). These interfaces could also become another target for exploitation by rogue/misconfigured VMs.

2.2 Potential Threats to Hypervisor Baseline Functions

We now proceed to look at the tasks in each of the five hypervisor baseline functions (HY-BF1 through HY-BF5) (with the exception of HY-BF3 (Execution of privileged operations by the hypervisor)), and analyze the threats to secure execution of those tasks in each baseline function by relating to the causes identified in the previous section.

2.2.1 *Potential threats to HY-BF1 function (VM Process Isolation)*

Threats to hypervisor's HY-BF1 function (VM Process Isolation) are due to the following:

- Faulty Implementation of some hypervisor modules
- Hypervisor Configuration Errors

The following are some design vulnerabilities that may have an impact on HY-BF1 function with an explanation of the context under which they may manifest.

Virtual Machine Control Structure (HYP-DV1): To schedule individual VM's tasks (i.e., vCPU tasks since each guest VM is allocated a set of virtual CPUs (vCPUs)) properly, the register states must be handled appropriately. To enable saving and loading of the state of each vCPU, the hypervisor uses a data structure called Virtual Machine Control Structure (VMCS). Faulty implementation of this data structure has been known to cause hypervisor memory leaks.

Handling Sensitive Instructions (HY-DV2): On hardware platforms that do not provide assistance for virtualization, there should be a software mechanism to discover sensitive (critical) instructions, send them to VMM (hypervisor) and replace them (using techniques such as binary translation) before executing them on the hardware. Any error in not trapping the critical instructions or faulty translation may have security implications in the form of guest OS being allowed to execute privileged instructions.

Memory Management Unit-MMU (HYP-DV3): The hypervisor runs a software-based Memory Management Unit (MMU) which allocates a shadow page table for each VM, since guest VMs cannot be granted direct access to the hardware-based MMU, as that would potentially enable them to access memory belonging to the hypervisor and other co-hosted VMs (under some situations). However faulty implementation of software-based MMU could lead to disclosure of data in arbitrary address spaces such as memory segments belonging to the hypervisor and co-located VMs, thus resulting in breach of memory isolation.

The collective threat due to the above three design vulnerabilities, is breach of process isolation (HYP-T1) Out of these, the vulnerabilities HYP-DV1 and HYP-DV2 are to be addressed through proper coding and testing of those modules and hence no security protection measures can be applied at the deployment and usage stage. However, the memory violation vulnerability HYP-DV3 can be addressed by hosting the hypervisor on a hardware platform that provides assistance for memory virtualization (through a virtualization-aware hardware memory management unit). This forms the basis for security recommendation HY-SR-2 in section 4.

Further, one of the requirements for correct execution isolation is that each VM gets the proper memory and CPU resources it wants for its hosting applications and that there is no denial of service. Ensuring adequate memory through proper configuration of memory allocation options is addressed through security recommendation HY-SR-3 and ensuring proper allocation of virtual CPUs through proper configuration of vCPU allocation options is addressed through security recommendations HY-SR-4 and HY-SR-5.

2.2.2 *Potential threats to HY-BF2 function (Devices Emulation & Access Control – such as Network and Storage (block) devices)*

The main task under this function is the emulation of storage and networking devices. This task is handled either by a module called QEMU or by a combination of the hypervisor kernel and loaded kernel modules. Any I/O call from a guest VM application is intercepted by the hypervisor kernel and forwarded to QEMU or VMM kernel module for handling since guest VMs (through their native device drivers – also called front-end drivers) cannot access the physical hardware device directly but only through backend device drivers (QEMU) located in the hypervisor. The QEMU can emulate a number of devices, mediates access to devices (by enforcing access policies) and multiplex the actual devices (since it has full access to the underlying physical device).

The potential threat to the secure execution of this function comes from (besides faulty implementation of in-memory data structures for virtual devices) faulty device driver code. The security recommendations relating to safe execution of device driver code, access control for devices and setting limits on I/O bandwidth are addressed through security recommendations HY-SR6, HY-SR-7 and HY-SR-8 respectively in section 5.

2.2.3 *Potential threats to HY-BF3 function (Execution of Privileged Operations for Guest VMs by the Hypervisor)*

Certain privileged operations (e.g., Memory Management) invoked by guest VMs are executed by the hypervisor handling them using mechanisms such as VM Exits (operations are processor-architecture specific) or Hypercalls (similar to system calls to OS and are hypervisor-specific). Lack of proper validation of those operations (not checking the scope such as allowing a full dump of a VM's Virtual Machine Control Block or input checking) would cause the entire virtualized host to crash. These potential attacks are again due to faulty hypervisor code and cannot be addressed through deployment and usage tasks.

2.2.4 *Potential threats to HY-BF4 function (VM Lifecycle Management)*

This function consists of tasks related to basic administrative operations for VMs. They include but not limited to:

- Creation of VM images conforming to a gold standard, ensuring integrity of images and secure storage and retrieval of images
- Migration of VMs from one hypervisor host to another
- Monitoring of VM execution and traffic flows into and out of VMs & overall configuration management
- Fine-grained access control for VM administration including the basic operations that alter the state of VMs – Start, Pause, Stop etc.

• The management operations on VMs are performed in most instances using a management daemon. Secure execution of above operations are addressed through security recommendations HY-SR9 through HY-SR18 in section 6.

2.2.5 *Potential threats to HY-BF5 function (Management of Hypervisor)*

The tasks under this function relate to overall administration of a hypervisor host (virtualized host) and the hypervisor software and are usually performed through user-friendly web interfaces or network-facing virtual consoles. The attacks on the secure execution of these tasks are nothing but those that pertain to any remote administration console and hence not addressed in this document. However, the core requirement in a data center with virtualized hosts is to have a uniform configuration for hypervisors based on different criteria - sensitivity of applications based on the set of hosted VMs, line of business or client (in cloud service environments) etc. Hence centralized management of hypervisor configuration (HY-SR-19) along with a dedicated virtual network segment for management traffic (HY-SR-20) are the security recommendations.

Some conventional security fixes may not be practical in the case of hosts hosting a hypervisor. For example, in the case of a network attack on a physical server that is not virtualized, merely turning off the offending port is a solution in the event of the server spamming the network with a bot attack. However, such a solution is not practical in the case of a hypervisor host since the same port in the physical network interface card of the hypervisor host could be shared by several running VMs.

3. SECURITY RECOMMENDATION FOR OVERALL PLATFORM INTEGRITY

Configuration changes, module version changes and patches change the content of the hypervisor platform components such as BIOS, hypervisor kernel and back-end device drivers running in the kernel. To ensure that each of these components are the safe ones to run as part of the hypervisor stack, it is necessary to check their integrity through a hardware-rooted attestation scheme that provides assurance of boot integrity. Checking integrity is done by cryptographically authenticating the hypervisor components that are launched. This authentication verifies that only authorized code runs on the system. Specifically, in the context of the hypervisor, the assurance of integrity that is provided is against tampering and low-level targeted attacks such as root kits. If the assertion of integrity is deferred to a trusted third party that fulfills the role of trust authority, the verification process is known as trusted attestation. Trusted attestation provides assurance that the code of the hypervisor components has not been tampered with. In this approach, we establish trust in the hypervisor's components based on trusted hardware. In other ways, we establish a chain of trust from hardware to hypervisor's with the initial component called the root of trust. Although it is possible to extend the chain of trust beyond the hypervisor to the VMs and the applications running inside them, in this guidance, we are restricting ourselves to authentication of only hypervisor components. This service can be provided by a hardware/firmware infrastructure of the hypervisor host that supports boot integrity measurement and attestation process. In other words, what is needed is a measured launch environment in the hypervisor host.

3.1 Implementing a Measured Launch Environment in the Hypervisor host

Some hardware platforms provide support for a measured launch environment (MLE) with firmware routines for measuring the identity (usually the hash of the binary code) of the components in a boot sequence. Such platforms also come equipped with a standards-based Trusted Platform Module (TPM) which contains the storage mechanisms (usually PCRs or Program Control Registers) for storing the results of those measurements and a reporting mechanism as well. These features (MLE with storage and reporting mechanisms) on a virtualized host can be leveraged to provide boot integrity assurance for hypervisor components by measuring the identity of all entities in the boot sequence starting from firmware, BIOS, hypervisor and hypervisor modules, comparing them to "known good values" and reporting any discrepancies.

The requirements for implementing a measured launch environment in the hypervisor host are the following:

- The hardware hosting the hypervisor is established as a root of trust and a trust chain is established from the hardware, through the BIOS and to all hypervisor components.
- For the hardware (consisting of the processor and chipset) to be established as the root of trust, it should have a hardware-based module that supports a measured launch environment (MLE) called the Root of Trust for Measurement (RTM). The outcome of launching a hypervisor in MLE-supporting hardware is a measured launch of the firmware, BIOS and either all or a key subset of hypervisor (kernel) modules, thus forming a trusted chain from the hardware to the hypervisor.
- The hypervisor offering must be able to take advantage of the MLE feature. In other words, the hypervisor should be able to invoke the secure launch process, which is usually done by integrating a pre-kernel module (since the kernel is the first module installed in a hypervisor boot up) into the hypervisor's code base. The purpose of this pre-kernel module is to ensure selection of the right authenticated module in the hardware that performs orderly evaluation (measurement) of the launch components of the hypervisor (or for that matter any software launched on that hardware). The most common mechanism to enable the hypervisor to take advantage of the MLE feature of the hardware is the Tboot.
- All hypervisor components that form part of the Trusted Computing Base (TCB) must be included under the scope of the Tboot mechanism so that they get measured as part of their launch process.

The measured elements (components) should include at the minimum the following: *the core kernel, kernel support modules, device drivers and the hypervisor's native management applications (for VM Lifecycle Management and Management of Hypervisor).*

- The various measurements (the identities of the components) that are included as part of the hypervisor launch process should be stored in a Trusted Platform Module (TPM) that has been standardized by the Trusted Computing Group (TCG) [4]. The TPM essentially contains the registers (called Platform Configuration Registers or PCRs) for storing the various measurements with its architecture consisting of two main components: Root of Trust for Storage (RTS) and Root of Trust for Reporting (RTR). The function of RTM is to make integrity measurements (generally a cryptographic hash) and send them to RTS. RTS then holds the components identities (measurements) and other sensitive information. In fact, the RTM measures the next piece of code following the boot sequence. The TPM must be an integral part of hypervisor host hardware.

The measured boot process starts with the execution of a trusted immutable piece of code in the BIOS which also measures the next piece of code to be executed. The result of this measurement is extended into the PCR of the TPM before the control is transferred to the next program in the sequence. Since each component in the sequence in turn measures the next before handing of control, there is a chain of trust established. If the measurement chain continues through the entire boot sequence, the resultant PCR values reflect the measurement of all files used.

The attestation process starts with the requester invoking, via an agent on the host, the TPM Quote command, specifying an Attestation Identity Key (AIK) to perform the digital signature on the contents of the set of PCRs (which contain the measurements of all components in the boot sequence) to quote, and a cryptographic nonce to ensure freshness of the digital signature. After receiving the signed quotes, the requestor validates the signature and determines the trust of the launched components by comparing the measurements in the TPM Quote with the known-good measurements.

The security recommendation based on the measured boot process that is part of a measured launch environment in the hypervisor platform can now be stated as follows:

<u>Security Recommendation HY-SR-1</u>: The hypervisor that is launched should be part of a platform and an overall infrastructure that contains: (a) Hardware that supports a MLE and standards-based TPM and (b) Attestation process that should contain capabilities to take advantage of these so as to provide a chain of trust starting from the Hardware to all Hypervisor components. The chain of trust provides assurance that all launched components (starting from BIOS, hypervisor and device drivers) have not been tampered with and that their versions are correct (i.e., overall boot integrity).

4. SECURITY RECOMMENDATIONS FOR VM PROCESS ISOLATION (HY-BF1)

To ensure isolation of processes running in VMs, the following requirements must be met.
(a) The privileged commands/instructions from Guest OSs to the host processor must be mediated such that the basic function of the VMM/hypervisor as the controller of virtualized resources is maintained.
(b) The integrity of the memory management function of the hypervisor host must be protected against attacks such as buffer overflows and illegal code execution especially in the presence of translation tables that are needed for handling memory access by multiple VMs.
(c) Memory allocation algorithms must ensure that payloads in all VMs are able to perform their functions.
(d) CPU allocation algorithms must ensure that payloads in all VMs are able to perform their functions.

The requirements (a) and (b) above can be met using software-based modules. However, hardware-based virtualization assist features such as Instruction Set Virtualization and Memory Virtualization provide better assurance than software-based solutions in meeting those requirements and hence are recommended in section 4.1. The above mentioned hardware-assisted virtualization features are briefly discussed prior to stating the recommendations. The requirements (c) and (d) above have to be met to ensure the availability of application services running in VMs and the enablers are some features in memory allocation and CPU allocation algorithms and their associated configuration parameters and these are stated as recommendations in section 4.2 and 4.3 respectively.

4.1 Hardware Assistance for Virtualization

Instruction Set Virtualization: Processor architectures that support Instruction Set Virtualization provide two modes of operation (root mode and non-root mode) each with 4 hierarchical privilege levels (Level 0 being the highest and Level 3 being the lowest). Further, among the two modes, the root mode has a higher privilege for executing CPU instructions than non-root mode. By running the hypervisor in root mode and VMs (Guests) OS in non-root mode (at privilege or ring level 0), the hypervisor is guaranteed to be safe at least from any instruction set-type attacks by any of the Guest OS's though VM escape can take place through normal networking protocols. This safety is ensured by the hardware trapping privileged instructions when running in non-root mode and allowing it to execute only in root mode. Also with the hypervisor not having to perform additional functions such as translating sensitive instructions (using techniques such as binary translation), the code executing with privileges is reduced in the hypervisor making the TCB smaller, enabling better assurance verification.

Memory Virtualization: Hardware-assisted memory virtualization is provided by the hardware enabling the mapping of the Guest OS's physical addresses in their respective Page Tables to Host's physical addresses using hardware-based page tables instead of hypervisor generated shadow page tables. The consequent reduction in privileged code executing this function provides the same security advantage mentioned for Instruction Set Virtualization above.

The security advantages of hardware-assisted virtualization platforms are:
- One of the potential security vulnerabilities for hypervisors is the buffer overflow attacks from VMs resident on the virtualized host platform. The hardware support for memory management such as Extended Page Tables (EPT), etc. that comes as part of the hardware-assisted virtualization can be leveraged to prevent code execution from memory locations reserved for data storage thus preventing buffer overflow attacks.
- Hardware extensions for Virtualization provide two modes of execution. Host (root) mode and Guest (non-root) mode. The host mode runs at a higher privilege than guest mode. The hypervisor code (providing the baseline functionality HY-BF1 (processor allocation + memory management) runs in host mode while guest OS's and applications in VMs run in guest mode. Hence any exploit code in guest OS cannot subvert the controls provided by hypervisor code.

- One of the common threats in virtualization platforms is a malicious VM accessing areas of memory belonging to other VMs. This is called a VM Escape attack. Processors with virtualization extensions provide safety against this through features such as Direct Memory Access (DMA) remapping that limits DMA access to what is valid for the VM (i.e., preventing a VM from performing DMA beyond its allocated area).
- Also the advantage of a hardware providing assistance for both forms of virtualization is that, the emulation module of the hypervisor can present the true hardware architecture of the physical host instead of modified hardware architecture. The consequence of this feature is that unmodified Guest OSs (along with their native device drivers) can be run in VMs. The security implication of enabling this feature is that a lot more CVE data is available for these OSs (along with patch versions) along with certified device drivers for each OS version.

***Security Recommendation HY-SR-2**: A Hypervisor platform with hardware assisted virtualization (both instruction set and memory management) provides greater security assurance than one with purely software assisted virtualization because of the following:*

- ***Better memory management controls can prevent attacks such as buffer overflow.***
- ***Guest OS code and hypervisor code execute in different processor modes providing better isolation***
- ***Better protection for device access mediation functions through privilege level isolation and better VM-level protection through hardware-based memory protection.***
- ***By supporting full virtualization, COTS versions of OSs can be run enabling easier patching/updating than having to perform the same operations on modified/ported versions of OSs that are the only types that can be run on para virtualized platforms.***
- ***Since many features of virtualization are now available in hardware, the size of the hypervisor code will be small enabling better security attestation/verification.***

4.2 VM Memory Allocation Scheduling Options

The hypervisor's memory scheduler has the responsibility to meet the memory requirements for all workloads running in all VMs at all times. A typical hypervisor meets this requirement (just like an OS) by using a combination of physical RAM and swap files (called hypervisor kernel swap files). Further, a typical VM does not require the entire memory it has been configured for at all times. Because of the above two reasons, it is a viable overall virtualization configuration decision to have the combined configured memory of all VMs running on a virtualized host to exceed the total physical RAM (provided there are no memory sensitive applications running in VMs). However, Over-commit, the ratio of the total configured memory of VMs to host physical RAM should not be too high as it may result not only in performance degradation but could also lead to a lack of availability of the virtualized host for certain VM workloads.

Another factor affecting the availability of the virtualized host/hypervisor for certain workloads in a VM is the ratio of the physical RAM size to kernel swap file size that is maintained by the memory scheduler of the hypervisor. Since a low ratio will deny execution of certain workloads for certain VMs, there should be a configuration option available in the hypervisor to specify a guaranteed physical amount of RAM for each VM. Also, in order to avoid a situation where a particular VM makes use of the physical RAM for its entire configured memory, there should be a feature to specify a limit on the guaranteed physical RAM as well. Further, there may be certain workloads that are time sensitive and hence, the VMs hosting them should have some priority in getting the required memory resources compared to other running VMs. Hence, a configuration option to specify a priority value for each VM should exist as well.

Based on the above issues relating to hypervisor memory scheduling, the following are the security recommendations:

<u>Security Recommendation HY-SR-3:</u> The hypervisor should have configuration options to specify a guaranteed physical RAM for every VM (that requires it) along with a limit to this value, and to specify a priority value for obtaining the required RAM resource in situations of contention among multiple VMs.

4.3 VM CPU Allocation Options

The security goal in VM CPU allocation is to guarantee availability for all VMs. This can be achieved in two ways. They are: (a) Proper choice of multi-virtual CPU (vCPU) VMs and (b) Proper configuration options in the hypervisor. While choosing a multi-vCPU VM, it is important to remember that most hypervisor kernels schedule CPU cycles simultaneously. Hence if a dual-vCPU VM places a request for CPU cycles, the request goes into a queue and the VM has to wait until the hypervisor recognizes the host has at least two cores with concurrent idle cycles. In terms of VM configuration options, the hypervisor should have a feature to set a minimum CPU requirement (called reservation) in terms of clock cycles. The architectural parameter to be observed here is that the number of VMs that can be deployed can be no more than the ratio of the total CPU clock cycles that the hypervisor host can offer to the average reservation required by each VM. For example, if the hypervisor host has 6000 MHz of CPU capacity and if the average reservation for each VM is 1000 MHZ, then no more than 6 VMs can be active in that hypervisor host. The reservation sets a lower bound (guaranteed) on the CPU clock cycles required for each VM. Similarly, there should be a feature to set an upper bound (called Limit) for the CPU cycles that each VM can use so that no one VM (sometimes a rogue or a compromised one) consumes all CPU resources of the host and denies services to other co-resident VMs. Further, to facilitate scheduling of hypervisor host CPU clock cycles in situations where multiple VMs require clock cycles above the lower bound but below the upper bound, there should a feature to assign a priority score (called shares) to each VM as well. Summarizing the above desired features for ensuring fair share for all VMs deployed, the security recommendations for VM CPU allocation are as follows:

<u>Security Recommendation HY-SR-4</u>: The number of virtual CPUs allocated to any VM deployed should be strictly less than the total number of cores in the hypervisor host.

<u>Security Recommendation HY-SR-5</u>: The hypervisor should provide features to specify a lower and upper bound for CPU clock cycles needed for every deployed VM as well as a feature to specify a priority score for each VM, to facilitate scheduling in situations of contention for CPU resources from multiple VMs.

5. SECURITY RECOMMENDATIONS FOR DEVICES EMULATION AND ACCESS CONTROL (HY-BF2)

The I/O calls from applications running in VMs invoke the virtual devices presented by VM. These I/O instructions are handled by a set of interfaces provided by the hypervisor as part of its device emulation and access function. The number of I/O interfaces depends on the number of emulated devices. The code implementing these interfaces can be run either in dedicated VMs or in the hypervisor kernel itself as a kernel module. Either way, this code calls on the device drivers installed in the hypervisor that accesses the physical devices connected to the hypervisor host. In most of the installations, this device emulation code is configured to run as non-privileged code. Hence its capacity to interfere with the normal operations of the hypervisor are limited. However, due security diligence is called for in the choice of device drivers that are installed in the hypervisor as well as in the controls set up for access and usage of devices.

***Security Recommendation HY-SR-6*: The I/O drivers installed as part of VMM/Hypervisor should be tested, certified and configured to run as non-privileged process. If possible a means to authenticate the driver before it is invoked should be provided.**

***Security Recommendation HY-SR-7 (Device access)*: It should be possible to set up an Access Control List (ACL) to restrict access of each VM process to only the devices assigned to that VM. To enable this, the hypervisor configuration should support a feature to mark (label) VMs (semantically a set of tasks) and/or has a feature to specify a whitelist (list of allowable) of devices for each VM.**

***Security Recommendation HY-SR-8 (Device Usage)*: It should be possible to set resource limits for network bandwidth and I/O bandwidth (e.g., disk read/write speeds) for each VM to prevent denial of service attacks.**

In the case of hypervisors implementing para virtualization, a custom OS device driver in VMs can directly access the device on the hypervisor host without the need for a device emulator module in the hypervisor. Hence only recommendations HY-SR-7 and HY-SR-8 are applicable for para virtualized hypervisors.

6. SECURITY RECOMMENDATIONS FOR VM LIFECYCLE MANAGEMENT (HY-BF4)

6.1 VM Image Management

Since VM-based software (such as Guest OS, Middleware and Applications) shares memory of the virtualized host with hypervisor software, it is no surprise that a VM is the biggest source of all attacks directed at the hypervisor. In operational virtualized environments, VMs are rarely created from scratch. They are created from VM Images. VM Images are templates used for creating running versions of VMs. An organization may have its own criteria for classifying different VM Images it uses in its VM Library. Some commonly used criteria include: - Processor load (VM used for compute-intensive applications), Memory load (VMs used for memory intensive applications such as Database processing), Application Sensitivity (VMs running mission-critical applications utilizing mission-critical data), etc. For each VM image type, the following practices must be followed to ensure that the resulting operational VMs created from it are secure:

- Documentation on the Golden Image for each VM Image type. A Gold Image is defined by a set of configuration variables associated with the VM Image. The configuration variables should include at the minimum, the Guest OS make, version, patch level, date of creation, number of vCPU cores and memory size.
- Each VM Image in the VM Image Library must have associated with it a digital signature.
- Access privileges to the VM Image Library must be controlled through a robust access control mechanism.
- Access to the server storing VM Images should be through a secure protocol

The security recommendations relating to the above practices are as follows:

***Security Recommendation HY-SR-9*: Gold-standard must be defined for VMs of all types and VM Images not conforming to the standard should not be allowed to be stored in the VM Image server/library. Further images in the VM Image library should be periodically scanned for OS versions and patches going out of date and thus have drifted from the standard.**

***Security Recommendation HY-SR-10:* Every VM Image stored in the image server should have a digital signature attached to it as a mark of authenticity and integrity, signed using trustworthy, robust cryptographic keys.**

***Security Recommendation HY-SR-11:* Permissions for checking out images from VM Image library should be enforced through a robust access control mechanism and limited to an authorized set of administrators. In the absence of an access control mechanism, VM image files should be stored in encrypted devices that can only be opened/closed by a limited set of authorized administrators with passphrase/key of sufficient complexity.**

***Security Recommendation HY-SR-12:* Access to the server storing VM images should always be through a secure protocol such as TLS.**

6.2 VM Live Migration

Live migration is a functionality that is present in all hypervisors that enables a VM to be migrated or moved from one virtualized host to another while the guest OS and applications on it are still running. This functionality provides key benefits such as fault tolerance, load balancing, host maintenance/upgrades/patching etc. In live migration, the state of the guest OS on the source host must be replicated on the destination host. This requires migrating memory content, processor state, storage (unless the two hosts share a common SAN storage) and network state.

The most common memory migration technique adopted in most hypervisors is a technique called pre-copy. In this approach, memory pages belonging to the VM are transferred to the destination host while the VM continues to run on the source host [5]. Memory pages that get modified during migration are sent again to the destination to ensure memory consistency. During this phase, the exact state of all the processor registers currently operating on the VM are also transferred and the migrating VM is suspended on the source host. Processor registers at the destination are modified to replicate the state at the source and the newly migrated VM resumes its operation. As far as storage migration is concerned, it is provided by a feature that allows admins to move a VM's file system from one storage location to another without downtime. This storage migration can even take place in situations where there is no VM migration. An example is a scenario where a VM continues to run on the host server, but the files that make up the VM are moved among storage arrays or LUNs.

In the process described above, the memory and processor-state migration functions are inherent part of hypervisor design. The storage migration function is an integral part of the storage management that is applicable in both virtualized and non-virtualized infrastructures. The network state is maintained after a VM migration because each VM carries its own unique MAC address and the migration process places some restriction on the migration target (e.g., the source and target host should be on the same VLAN). Hence from the security protection point of view, the only aspects to pay attention to are proper authentication and a secure network path for the migration process.

***Security Recommendation HY-SR-13:* During VM live migration, care should be taken to see that a secure authentication protocol is used for performing live migration, that the credentials of the administrator performing the migration is passed only to the destination host, the migration of memory content and processor state takes place over a secure network connection and a dedicated virtual network segment is used in both source and destination hosts for carrying this traffic.**

6.3 VM Monitoring and Security Policy Enforcement

Since VMs are prime sources of threat to the hypervisor, continuous monitoring of the state of VMs and the traffic going in and out of those VMs is necessary for: (a) controlling the type of traffic (b) intrusion detection/prevention and (c) detecting viruses and other malware. This function can be accomplished in two ways:

- VM-based Security Monitoring and Intervention Solution
- Security Monitoring and Intervention by a Hypervisor Module but enforcement of traffic rules at the point of a VM or at the virtual network object level (i.e., Virtual Switch's Port/Port Group)

In VM-based Security Monitoring and Intervention approach, a software or a software-agent (a security tool) is run inside a VM to monitor security-relevant events. This approach is similar to running a host-based IDS. The advantage of this approach is that it provides good visibility and good context analysis for the code running within the VM. However, because of the tight dependency of the security tool on the underlying Guest OS, any attack on the latter will also disable the function of the security tool (thus disabling the countermeasure). Another disadvantage of running the security tool as a virtualized workload is the performance impact it will have on itself as well as other application workloads running on that VM.

Virtual Network-based Security Monitoring can come in two flavors:

(a) A dedicated security appliance for protecting each VM.
(b) A security appliance that runs in the virtual network and can protect multiple VMs inside the hypervisor host.

The dedicated security appliance is deployed in the virtual network in front of the monitored VM and monitors all traffic going in and out of the VM. The main disadvantage of this approach is if the VM is migrated to some other physical host, the dedicated appliance has to be migrated as well.

A generic security appliance deployed on a virtual network (and configured to monitor multiple VMs), may have to be continuously reconfigured due to the following reasons:

(a) The set of VMs to be monitored is continuously in a state of flux as VMs are subject to migration from one virtualized host to another due to load balancing, performance and even security reasons.
(b) If virtual LANs (VLANs) are used to provide communication-level isolation among VMs, the configuration of VLANs may also be continuously undergoing change as the workload patterns keep changing on VMs. This may require re-configuration of the network traffic mirroring capabilities to ensure that all virtual network traffic flows through the monitoring tool impacting the overall performance of the workloads inside that virtualized host.

In a hypervisor-based security monitoring solution, the security tool that performs the monitoring and protection of VMs (User VMs) is run outside the VMs hosting business applications in a specially security-hardened VM. A security tool designed and configured to run in this mode is called Security Virtual Appliance (SVA). The SVA obtains its visibility into the state of a VM (CPU, registers, memory and I/O devices) as well as network traffic amongst VMs and between VMs and the hypervisor through the *virtual machine introspection* API of the hypervisor. This is the most preferable solution since:
(a) It is not vulnerable to a flaw in the Guest OS
(b) Is independent of the Virtual Network Configuration and does not have to be reconfigured every time the virtual network configuration changes due to migration of VMs or change in connectivity among VMs resident on the hypervisor host.

The security recommendations therefore, with respect to architecting the VM monitoring solution for the protection of the hypervisor are as follows:

***Security Recommendation HY-SR-14*: There should be a mechanism for security monitoring and security policy enforcement of VM operations –malicious processes running inside VMs and malicious traffic going in and out of a VM. This monitoring and enforcement mechanism forms the foundation for building Anti-Virus (AV) and Intrusion Detection & Prevention System (IDPS) solutions.**

***Security Recommendation HY-SR-15*: Solutions for Security Monitoring and security policy enforcement of VMs should be based "outside of VMs" and should leverage the virtual machine introspection capabilities of the hypervisor. Generally, such solutions involve running a security tool as a Security Virtual Appliance (SVA) in a security hardened (trusted) VM.**

***Security Recommendation HY-SR-16*: All antimalware tools (virus checkers, firewalls and IDPS) running in the virtualized host should have the capability to perform autonomous signature or reference file updates on a periodic basis.**

6.4 VM Configuration Management

The configuration of every VM should be monitored and managed throughout its lifecycle. In most instances, this is accomplished using dedicated third party tools in addition to using native features that come with the hypervisor. The desired features for these tools are provided in the form of security recommendation below:

<u>Security Recommendation HY-SR-17</u>: VM configuration management tools should have the capability to compile logs and alert administrators when configuration changes are detected in any VM that is being monitored.

6.5 Fine-grained Administrative Privileges for VM Management

Having the capability to assign fine-grained administrative permission for the virtualized infrastructure enables the setting up of different administrative models and associated delegations. To see the need for granular permissions, it would be helpful to look at some use case scenarios for administrative operations in the virtualized infrastructure:

- <u>VM Administration Use Case 1</u>: A Quality Assurance group wants to set up a few virtual machines with some definite profiles (resource quotas such as Memory, CPUs) to test some applications that may soon be going into production. In this situation, it may be useful for one or more administrators assigned exclusively to the Quality Assurance group to be given administrative permissions on specific virtual machines set up for testing purposes.
- <u>VM Administration Use Case 2</u>: A capacity planner assigned the task of determining the operating loads on the various virtualized servers and the need for additional virtualized hosts may need permission for viewing the list of virtual machines in each of the virtualized hosts but not permissions to perform any administrative operations on those VMs. In this situation, it is desirable to have the capability to view the list of VMs in a virtualized host but deny the user the capability to interact with any of the visible objects.
- <u>VM Administration Use Case 3</u>: In virtualized data centers, where VMs of different sensitivity levels are run on the same virtualized host, sometimes an administrator who is given administrative privileges at the hypervisor level should be prevented from accessing a specific VM because of the sensitive nature of the workload (set of applications) running in that VM. The desired capability in this scenario is to negate a permission (obtained through inheritance) for a specific child object.
- <u>VM Administration Use Case 4</u>: VM Administration Use Case 1 deals with a scenario where you need assign permissions for a group of administrators controlling a set of VMs for a particular organizational division or department. A corollary to this type of administrative entity is the need for a class of administrators wanting to administer VMs running a particular type of work load (e.g., webserver) irrespective of its location within the organizational structure. This class of administrators may not require the entire set of administrative functions on a VM but some arbitrary set of management functions such as: Configure CD Media, Configure Floppy Media, Console Interaction, Device Connection, Power On, Power Off, Reset, Suspend, etc. This scenario calls for the capability to create "Custom Roles" that can contain an arbitrary set of permissions relating to a VM as well as the ability to create a "Custom Object" that contains an arbitrary set of VMs carrying a particular type of workload (e.g., webserver).

Summing up the capabilities required in all four administrative scenarios, the overall security recommendation with required permission granularity is as follows:

<u>Security Recommendation HY-SR-18</u>: The access control solution for VM administration should have a granular capability both at the permission assignment level as well as at the object level (i.e., the specification of the target of the permission can be a single VM or any logical grouping of VMs - based on function or location). In addition, the ability to deny permission to some specific objects within a VM group (e.g., VMs running workloads of a particular sensitivity level) in spite of having access permission to the VM group should exist.

7. SECURITY RECOMMENDATIONS FOR MANAGEMENT OF HYPERVISOR (HY-BF5)

Secure operation of administrative functions is critical is for any server class software and hypervisor is no exception to this. The outcome is a secure configuration which can provide the necessary protection against any security violations. In the case of hypervisor, impact of insecure configuration can be more severe than in many server software instances since the compromise of a hypervisor can result in the compromise of many VMs operating on top of it. While the composition of the configuration parameters depends upon the design features of a hypervisor offering, the latitude in choosing the values for each individual parameter results in different configuration options. While many configuration options relate functional features and performance, there are some options that have direct impact on the secure execution of the hypervisor and it is those configuration options that are discussed in this document.

The following are some security practices that are generic for any server class software. Hence, although applicable to the hypervisor, they are not addressed in this document:

(a) Control of administrative accounts on the hypervisor host itself and least privilege assignment for different administrators
(b) Patch management for hypervisor software and host OS
(c) Communicating with the hypervisor through a secure protocol such as TLS or SSH

7.1 Centralized Administration

The administration of a hypervisor and hypervisor host can be performed in two ways.

- Having administrative accounts set up in each hypervisor host
- Centralized administration of all hypervisors/hypervisor host through an enterprise virtualization management software.

Management of all hypervisor platforms in the enterprise centrally through an enterprise virtualization management (EVMS) software is preferable since a gold-standard configuration for all hypervisors in the enterprise can be defined and easily enforced through EVMS. Further for any IT data center to operate efficiently, it is necessary to implement load balancing and fault tolerance measures, and this can be realized by defining hypervisor clusters. Creation, assignment of application workloads and management of clusters can be performed only from a centralized management software making the deployment and usage of an enterprise virtualization management software mandatory.

Hence the recommendation for the architecture for hypervisor administration is as follows:

***Security Recommendation HY-SR-19*: The administration of all hypervisor installations in the enterprise should be performed centrally using an enterprise virtualization management system (EVMS). Further enterprise gold-standard hypervisor configurations for different types of workloads and clusters must be developed and enforced through EVMS. The gold-standard configurations should at the minimum cover the following aspects – CPU, Memory, Storage, Network bandwidth and Host OS hardening (if required).**

7.2 Securing the Management Network

To connect multiple VMs to each other and to the enterprise network (in which the virtualized host is a node), the hypervisor provides the capability to define a software-defined communication fabric called a virtual network through its management console. This capability can be provided by a dedicated management VM or directly in the hypervisor kernel through a kernel module. The virtual network is a software-defined artifact that resides entirely within the virtualized host and has as its nodes, the VMs residing inside it. The components of this virtual network are: (a) the virtual network interface cards (vNICs) that are defined for each VM and provide connection for each VM to the virtual network, (b) the virtual switches that provide selective connectivity among VMs and whose configuration determines the topology of the virtual network and, (c) the physical network interface cards (pNICs) of the virtualized hosts that provide connectivity for VMs to the enterprise network.

While considering the security impact of the virtual network, the following three main functions have to be considered:

- Providing selective connectivity/isolation between groups of VMs belonging to different logical groupings – different tenants in the case of an Infrastructure as a Service (IaaS) cloud service, different application tiers such as Web Server, Database Server, etc. or different Line of Business applications of an enterprise.
- Dedicating subnets for key functions such as: (a) migration of VMs from one hypervisor host to another for security or performance reasons, (b) attaching network-based storage devices and (c) fault Tolerant Logging
- Providing access to the Management interface in the management VM (which is a node of the virtual network) which is used for performing key hypervisor baseline functions of VM lifecycle management (HY-BF4) and Management of hypervisor (HY-BF5).

Out of the three functionalities stated above, providing selective connectivity/isolation between groups of VMs is required for providing security to the applications running on those VMs and hence, outside the scope of this document. The same criteria apply to dedicating subnets for Network- based Storage administration. We have already discussed secure VM migration under VM lifecycle management in section 6. Hence, our focus on the virtual network configuration is limited to providing protection for the network interfaces used for performing VM management and Hypervisor administrative functions. A common approach adopted is to dedicate a virtual network segment (vLAN ID) exclusively for the management interface.

***Security Recommendation HY-SR-20*: Protection for Hypervisor Host & Software administration functions can be ensured by placing the management interface of the hypervisor in a dedicated virtual network segment and enforcing traffic controls using a firewall (e.g., designating the subnets in the enterprise network from which incoming traffic into the management interface is allowed).**

8. SECURITY RECOMMENDATION SUMMARY

The hypervisor is a complex server class software that performs the function of virtualization of hardware resources to enable execution of multiple computing stacks (VMs) with heterogeneous OSs and with multiple applications hosted in each of them. Hence secure configuration of the hypervisor together with its physical host (called hypervisor host or virtualized host), collectively called the hypervisor platform, is needed to provide a safe platform for execution of mission-critical applications.

Due to multiple ways by which an architecture of a hypervisor can be classified, the approach taken in this document, is to identify the five baseline functions that a hypervisor performs, identify the tasks involved in each baseline function, identify the potential threats to secure execution of the task and express the countermeasures that provide assurance against exploitation of these threats in the form of security recommendations.

Overall, twenty security recommendations are provided for secure deployment of hypervisors. All but two (HY-SR-1 and HY-SR-2) relate to configuration of parameters of software modules in the hypervisor platform. These parameters range from integrity metrics for software modules (device drivers and VM images), setting of access controls (device access, VM image access and VM administration), configuration of secure protocols (VM image server access and VM migration). The mapping of the security recommendations to a hypervisor's baseline functions is provided in Appendix B.

APPENDIX A

Here, a detailed description of each of the five hypervisor baseline functions is provided. Recalling from the Introduction Section, these baseline functions are:

- HY-BF1: VM Process Isolation – Scheduling of VMs for execution, Management of the application processes running in VMs such as CPU and Memory Management, context switching between various processor states during the running of applications in VMs, etc.
- HY-BF2: Devices Emulation & Access Control – Emulating all Network and Storage (block) devices that different native drivers in VMs are expecting, mediating access to physical devices by different VMs
- HY-BF3: Execution of Privileged Operations for Guest VMs – Certain operations invoked by Guest OSs, because of their privileged nature, may have to be executed by the hypervisor, instead of being executed directly on host hardware.
- HY-BF4: VM Lifecycle management – This involves all functions from creation and management of VM images, control of VM states (Start, Pause, Stop, etc.),VM migration, VM monitoring and policy enforcement..
- HY-BF5: Management of hypervisor– This involves defining some artifacts and setting values for various configuration parameters in hypervisor software modules including those for configuration of a Virtual Network inside the hypervisor.

A detailed description of the above baseline functions is given below:

A.1 HY-BF1 (VM Process Isolation)

Scheduling VMs on physical CPUs (by making the necessary translation from virtual CPU (vCPU) tasks to physical CPU tasks), Virtual Memory Management (such that a VM does not encroach on memory spaces allocated to other VMs and to the hypervisor itself) for multiple VMs (by leveraging the virtualization-aware hardware MMU), emulating the interrupt and timer mechanisms (that the motherboard provides to the physical machine), handling VM exits (e.g., intercepting I/O instructions and forwarding it to QEMU for handling) and Hypercalls (e.g., privileged calls (analogous to System calls – supported by hypervisors implementing para virtualization) made by Guest VMs for purposes such as managing hard disk partitions, altering memory pages using calls to memory management unit (MMU) etc.). *All tasks described so far are carried out by the hypervisor kernel or kernel extension modules*.

A.2 HY-BF2 (Devices Emulation & Access Control)

Since Guest VMs with different OSs run on a hypervisor platform, there must be a module that emulates devices for all device drivers available in the Guest OSs to support fully virtualized guests (guests with unmodified OS). This module is the QEMU code. The QEMU code generates one QEMU process for each running VM, performs the emulation of the device (corresponding to the native device driver in the Guest OS) and translates requests for that device to access requests for actual physical devices. Alternatively, the whole process described above can be performed by the hypervisor kernel directly. In the process, QEMU also enforces access control on the VM's right to access the device. Any I/O instruction originating from a guest OS is intercepted by the hypervisor kernel and forwarded to QEMU for performing the emulation and access control function. The QEMU is also responsible for relaying the output of the actual physical device back to the corresponding VM that made the I/O call. *From these discussions, it should be clear that all tasks under Device Emulation & Access Control are executed by the QEMU code generally residing in the privileged, dedicated VM.*

A.3 HY-BF3 (Execution of Privileged Operations for Guest VMs):

Certain operations invoked by Guest OS kernels may have to be executed by the hypervisor because of their privileged nature. These calls are Hypercalls and are analogous to OS system calls. Hyper calls are supported by hypervisors implementing para virtualization. Some Hypercalls may emanate from the privileged VM (used for Management of VMs and Administration of Hypervisor platform/software). Examples of Hypercalls are: call to Memory Management Unit (MMU), call for managing Disk partitions, etc.

A.4 HY-BF4 (VM Lifecycle Management)

This encompasses all administrative operations on VMs throughout its life cycle. They include*:* (a) Creation and Management of VM Images and (b) Setting VM states (Stop, Pause, Start). VM Management tasks are enabled using a management daemon which provides network interfaces. *These interfaces are generally implemented not as part of the hypervisor kernel modules but on a privileged VM (management VM) that is booted up as an integral part of the hypervisor platform boot process.*

A.5 HY-BF5 (Management of Hypervisor)

These tasks include those that are involved in the configuration of the hypervisor host (virtualized host) and the hypervisor software itself. Important tasks include: provisioning of VMs to hypervisor hosts, creating and managing hypervisor clusters and configuration of the virtual network inside the hypervisor host. A virtual network is a software-defined network inside the hypervisor host that enables connectivity among VMs, as well as connectivity of VMs to external network (e.g., LAN, WAN, etc.).

Appendix B: Traceability of Security Recommendation to Hypervisor Baseline Functions

NO	SECURITY RECOMMENDATION	BASELINE FUNCTION
HY-SR-1	*The hypervisor that is launched should be part of a platform and an overall infrastructure that contains: (a) Hardware that supports a MLE and standards-based TPM and (b) Attestation process that should contain capabilities to take advantage of these so as to provide a chain of trust starting from the Hardware to all Hypervisor components. The chain of trust provides assurance that all launched components (starting from BIOS, hypervisor and device drivers) have not been tampered with and that their versions are correct (i.e., overall boot integrity).*	N/A
HY-SR-2	*A Hypervisor platform with hardware assisted virtualization (both instruction set and memory management) provides greater security assurance than one with purely software assisted virtualization because of the following:* • *Better memory management controls can prevent attacks such as buffer overflow.* • *Guest OS code and hypervisor code execute in different processor modes providing better isolation* • *Better protection for device access mediation functions through privilege level isolation and better VM-level protection through hardware-based memory protection.* • *By supporting full virtualization, COTS versions of OSs can be run enabling easier patching/updating than having to perform the same operations on modified/ported versions of OSs that are the only types that can be run on para virtualized platforms.* • *Since many features of virtualization are now available in*	HY-BF1 (VM Process Isolation)
HY-SR-3	*The hypervisor should have configuration options to specify a guaranteed physical RAM for every VM (that requires it) along with a limit to this value, and to specify a priority value for obtaining the required RAM resource in situations of contention among multiple VMs.*	HY-BF1 (VM Process Isolation)
HY-SR-4	*The number of virtual CPUs allocated to any VM deployed should be strictly less than the total number of cores in the hypervisor host.*	HY-BF1 (VM Process Isolation)
HY-SR-5	*The hypervisor should provide features to specify a lower and upper bound for CPU clock cycles needed for every deployed VM as well as a feature to specify a priority score for each VM, to facilitate scheduling in situations of contention for CPU resources from multiple VMs.*	HY-BF1 (VM Process Isolation)
HY-SR-6	*The I/O drivers installed as part of VMM/Hypervisor should be tested, certified and configured to run as non-privileged process. If possible a means to authenticate the driver before it is invoked should be provided.*	HY-BF2 (Devices Emulation & Access Control)

HY-SR-7	*It should be possible to set up an Access Control List (ACL) to restrict access of each VM process to only the devices assigned to that VM. To enable this, the hypervisor configuration should support a feature to mark (label) VMs (semantically a set of tasks) and/or has a feature to specify a whitelist (list of allowable) of devices for each VM.*	HY-BF2 (Devices Emulation & Access Control)
HY-SR-8	*It should be possible to set resource limits for network bandwidth and I/O bandwidth (e.g., disk read/write speeds) for each VM to prevent denial of service attacks.*	HY-BF2 (Devices Emulation & Access Control)
HY-SR-9	*Gold-standard must be defined for VMs of all types and VM Images not conforming to the standard should not be allowed to be stored in the VM Image server/library. Further images in the VM Image library should be periodically scanned for OS versions and patches going out of date and thus have drifted from the standard.*	HY-BF4 (VM Lifecycle Management)
HY-SR-10	*Every VM Image stored in the image server should have a digital signature attached to it as a mark of authenticity and integrity, signed using trustworthy, robust cryptographic keys.*	HY-BF4 (VM Lifecycle Management)
HY-SR-11	*Permissions for checking out images from VM Image library should be enforced through a robust access control mechanism and limited to an authorized set of administrators. In the absence of an access control mechanism, VM image files should be stored in encrypted devices that can only be opened/closed by a limited set of authorized administrators with passphrase/key of sufficient complexity.*	HY-BF4 (VM Lifecycle Management)
HY-SR-12	*Access to the server storing VM images should always be through a secure protocol such as TLS.*	HY-BF4 (VM Lifecycle Management)
HY-SR-13	*During VM live migration, care should be taken to see that a secure authentication protocol is used for performing live migration, that the credentials of the administrator performing the migration is passed only to the destination host, the migration of memory content and processor state takes place over a secure network connection and a dedicated virtual network segment is used in both source and destination hosts for carrying this traffic.*	HY-BF4 (VM Lifecycle Management)

HY-SR-14	*There should be a mechanism for security monitoring and security policy enforcement of VM operations –malicious processes running inside VMs and malicious traffic going in and out of a VM. This monitoring and enforcement mechanism forms the foundation for building Anti-Virus (AV) and Intrusion Detection & Prevention System (IDPS) solutions.*	HY-BF4 (VM Lifecycle Management)
HY-SR-15	*Solutions for Security Monitoring and security policy enforcement of VMs should be based "outside of VMs" and should leverage the virtual machine introspection capabilities of the hypervisor. Generally, such solutions involve running a security tool as a Security Virtual Appliance (SVA) in a security hardened (trusted) VM.*	HY-BF4 (VM Lifecycle Management)
HY-SR-16	*All antimalware tools (virus checkers, firewalls and IDPS) running in the virtualized host should have the capability to perform autonomous signature or reference file updates on a periodic basis.*	HY-BF4 (VM Lifecycle Management)
HY-SR-17	*VM configuration management tools should have the capability to compile logs and alert administrators when configuration changes are detected in any VM that is being monitored.*	HY-BF4 (VM Lifecycle Management)
HY-SR-18	*The access control solution for VM administration should have the granular capability both at the permission assignment level as well as at the object level (i.e., the specification of the target of the permission can be a single VM or any logical grouping of VMs - based on function or location). In addition, the ability to deny permission to some specific objects within a VM group (e.g., VMs running workloads of a particular sensitivity level) in spite of having access permission to the VM group*	HY-BF4 (VM Lifecycle Management)
HY-SR-19	*The administration of all hypervisor installations in the enterprise should be performed centrally using an enterprise virtualization management system (EVMS). Further enterprise gold-standard hypervisor configurations for different types of workloads and clusters must be developed and enforced through EVMS. The gold-standard configurations should at the minimum cover the following aspects – CPU, Memory, Storage, Network and Host OS hardening (if required).*	HY-BF5 (Management of Hypervisor)
HY-SR-20	*Protection for Hypervisor Host & Software administration functions can be ensured by placing the management interface of the hypervisor in a dedicated virtual network segment and enforcing traffic controls using a firewall (e.g., designating the subnets in the enterprise network from which incoming traffic into the management interface is allowed).*	HY-BF5 (Management of Hypervisor)

Appendix C: Glossary

Full Virtualization: A form of Virtualization in which the hypervisor presents virtualized resources that reflect the architecture of the underlying hardware and hence unmodified guest OSs can be run.

Guest Operating System (OS): The operating system component of the execution stack of a Virtual Machine (see below), others being Virtual Hardware, Middleware and Applications.

Hypervisor: A software built using a specialized kernel of an OS, along with supporting kernel modules that provides isolation for various execution stacks represented by Virtual Machines (see below).

Virtualized Host: The physical host on which the virtualization software such as the Hypervisor is installed. Usually, the virtualized host will contain a special hardware platform that assists virtualization - specifically Instruction Set and Memory virtualization.

Virtual Machine (VM): A software-defined complete execution stack consisting of virtualized hardware, operating system (guest OS), and applications.

QEMU (Quick Emulator): A software module that is a component of the hypervisor platform that supports full virtualization by providing emulation of various hardware devices.

Virtualization: A methodology for emulation or abstraction of hardware resources that enables complete execution stacks including software applications to run on it.

Appendix D: References

1. *Mastering VMware vSphere 5.5*, Scott Lowe et al., Wiley Publishing Incorporated (2013)
2. *Running Xen: A Hands-On Guide to the Art of Virtualization*, J.N. Matthews et al., Prentice Hall (2008)
3. *Building the Infrastructure for Cloud Security: A Solutions View,* R.Yeluri, and E.Castro-Leon, Apress Media/Springer Science (2014)
4. *Trusted Platform Module (TPM) Main Specification:* http://www.trustedcomputinggroup.org/resources/tpm_main_specification
5. S.Shirinbab, l. Lundberg and D. Illie, *Performance Comparison of KVM, VMware and Xenserver using a Large Telecommunication Application*, Proceedings of the Fifth International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING), 2014.