Draft NIST Special Publication 800-125A
Revision 1

Security Recommendations for ~~Hypervisor Deployment on Servers~~Server-based Hypervisor Platforms

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 NIST Special Publication 800-125A
Revision 1

Security Recommendations for Server-based Hypervisor Platforms

~~Hypervisor Deployment on Servers~~

Ramaswamy Chandramouli
Computer Security Division
Information Technology Laboratory

April 2018

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

National Institute of Standards and Technology
Walter Copan, NIST Director and Under Secretary of Commerce for Standards and Technology

Authority

National Institute of Standards and Technology Special Publication 800-125A Revision 1
Natl. Inst. Stand. Technol. Spec. Publ. 800-125A Rev. 1, 38 Pages (April 2018)
CODEN: NSPUE2

Public comment period: *April 11, 2018 through May 2, 2018*

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

All comments are subject to release under the Freedom of Information Act (FOIA).

Reports on Computer Systems Technology

Abstract

The Hypervisor platform is a collection of software modules that provides virtualization of hardware resources (such as CPU~~/GPU~~, Memory, Network and Storage) and thus enables multiple computing stacks (made of an operating system (OS) and ~~Application~~ 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. Further, the recommendations are in the context of a hypervisor deployed for server virtualization and not for other use cases such as embedded systems and desktops. Recommendations for secure configuration of a virtual network are dealt with in a separate NIST document (Special Publication 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. Special thanks to Andreas Bartelt from Bosch Center of Competence Security for valuable input regarding technologies for device virtualization. He also thanks Michael Bartock for his valuable review and feedback as a division reader. Last but not the least, he expresses his thanks to Isabel Van Wyk for her detailed editorial review.

Note to Reviewers

This revision includes additional technologies for device virtualization such as para-virtualization, passthrough and self-virtualizing hardware devices as well as associated security recommendations. Major content changes in this revision are in: Section 1.1, Section 2.2.2 and Section 5.

Table of Contents

EXECUTIVE SUMMARY v
1. INTRODUCTION, SCOPE & TARGET AUDIENCE 1
1.1 Hypervisor Baseline Functions 2
1.2 Scope of this document 3
1.3 Target Audience 4
1.4 Relationship to other NIST Guidance Documents 4
2. APPROACH FOR DEVELOPING SECURITY RECOMMENDATIONS 6
2.1 Hypervisor Platform Threat Sources 6
2.2 Potential Threats to Hypervisor Baseline Functions 7
3. SECURITY RECOMMENDATION FOR OVERALL PLATFORM INTEGRITY 11
4. SECURITY RECOMMENDATIONS HY-BF1 13
4.1 Hardware Assistance for Virtualization 13
4.2 VM Memory Allocation Scheduling Options 14
4.3 VM CPU Allocation Options 15
5. SECURITY RECOMMENDATIONS FOR HY-BF2 16
6. SECURITY RECOMMENDATIONS FOR HY-BF4 18
6.1 VM Image Management 18
6.2 VM Live Migration 18
6.3 VM Monitoring and Security Policy Enforcement 19
6.4 VM Configuration Management 21
6.5 Fine-grained Administrative Privileges for VM Management 21
7. SECURITY RECOMMENDATIONS FOR HY-BF5 23
7.1 Centralized Administration 23
7.2 Securing the Management Network 23
8. SECURITY RECOMMENDATION SUMMARY 25
Appendix A 26
Appendix B: Traceability of Security Recommendation to Hypervisor Baseline Functions 29
Appendix C: Glossary 34
Appendix D: References 35

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 and administrative overhead. The core software used for server virtualization ~~Server Virtualization is realized using a collection of software modules~~is called the Hypervisor which directly provides CPU and memory virtualization. Together with its supporting modules, it ~~that~~ enables virtualization of all hardware resources (e.g., CPU~~/GPU~~, Memory, Network and Storage) and thus enables multiple computing stacks called Virtual Machines (VMs) or Guests, each ~~made of~~hosting an OS (Guest OS) and application programs ~~called Virtual Machines (VMs) to be run~~, to be run on a single physical host. This physical host is referred to as Virtualized Host or Hypervisor Host. Since the hypervisor by itself cannot provide all functions needed for server virtualization, it has supporting software modules (e.g., device drivers) for devices (e.g., Network and Storage devices) virtualization in addition to management modules for VM lifecycle operations and hypervisor configuration. The hypervisor together with these supporting modules and the hosting hardware constitute the hypervisor platform. The hypervisor can be installed either directly on the hardware or bare metal (Type 1 Hypervisor) or on top of a full-fledged conventional OS called Host OS (Type 2 Hypervisor).

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 the 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 therefore require a focused set of security recommendations based on an analysis of threats to the secure execution of these functions.

Since there are 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, the tasks involved in each baseline function, the potential threats to the secure execution of the task, and the countermeasures that can provide assurance against exploitation of these threats in the form of security recommendations.

The following five are identified as baseline functions of a hypervisor:

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

Apart from providing security recommendations for ensuring the secure execution of the baseline functions listed above, a recommendation for ensuring the overall integrity of all components of a hypervisor platform is also provided. The recommendations cover both Type 1 and Type 2 hypervisors.

Secure execution of routine administrative functions for the physical host where the hypervisor is installed is not covered in this document. The protection requirements for countering physical access threats, as well as those for Guest OS and applications running on VMs and associated security recommendations, are also beyond the scope of this document. Further, the security recommendations pertain to hypervisors deployed for server virtualization and do not cover other use cases such as the use of hypervisor for desktops and embedded systems.

1. INTRODUCTION, SCOPE, AND TARGET AUDIENCE

The Hypervisor is ~~a collection of software modules~~the core software that provides server virtualization. Along with its supporting modules, it enables virtualization of all hardware resources (e.g., CPU~~/GPU~~, 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 a 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 (e.g., CPU, Memory, etc.) allocated to it. The VMs are also called "Guests," and the operating system (OS) running inside each of them is called "Guest OS." The resources associated with a VM are virtual resources as opposed to physical resources associated with a physical host. The hypervisor together with these supporting modules and the hosting hardware constitute the hypervisor platform.

The primary function of the hypervisor is to enforce guest OS isolation as well as controlled resource sharing among guest VMs.

~~The hypervisor forms part of the virtualization layer in a virtualized host and~~ Thus, it 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 (devices) across multiple VMs. While access to CPU and memory (to ensure process isolation) are handled directly by the hypervisor (through instruction set (CPU) virtualization and memory virtualization respectively with or without assistance from hardware), it handles the mediation of access to devices (devices virtualization) by calling on software modules running either in the kernel or in dedicated VMs called Device-driver VMs. ~~Therefore, all other functions needed to support virtualization such as emulation of network and storage devices and the management of VMs and the hypervisor itself can 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 or Service VMs). The~~The hypervisor can be installed either directly on the hardware or bare metal (Type 1 Hypervisor) or on top of a full-fledged conventional OS called Host OS (Type 2 Hypervisor).

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

The hypervisor baseline functions consist of:

- VM Process Isolation
- Devices ~~Emulation~~ Mediation and Access Control
- Execution of Privileged Operations by Hypervisor for Guest VMs
- VM Lifecycle Management
- 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 hardware (a physical host) to enable the operation of multiple virtual hosts (popularly known as VMs), commercial hypervisor offerings come with differing feature sets. 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). When VMMs are combined with the modules that provide OS-level services, such as the scheduling of VMs in the CPU, they are 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. In order to ensure VM process isolation, memory access from DMA capable devices needs to be under hypervisor control as well (e.g., via IOMMU). However, this function is considered under HY-BF2 since it pertains to devices mediation.
- HY-BF2: Devices Mediation & Access Control – making devices available to VMs (e.g., via emulation, para-virtualization, passthrough or self-virtualizing hardware devices) and controlling which VMs are allowed to access which devices (e.g., Network Interface Card (NIC), storage device such as IDE drive, etc.).
- ~~HY-BF2: Devices Emulation & Access Control – Emulating all Network and Storage (block) devices that are expected by different native drivers in VMs, mediating access to physical devices by different VMs~~
- HY-BF3: Direct Execution of ~~Privileged Operations for~~commands from Guest VMs – ~~Execution of certain~~Certain ~~operations~~ commands ~~invoked by~~from Guest OSs are executed directly by the hypervisor instead of being ~~executed directly by the host hardware because of their privileged nature;~~triggered through interrupts and context switching. This function applies to hypervisors that have implemented para-virtualization instead of full virtualization
- HY-BF4: VM Lifecycle management – All functions including creation and management of VM images, control of VM states (Start, Pause, Stop, etc.), VM migration, making snapshots, VM monitoring, and policy enforcement
- HY-BF5: Management of hypervisor platform – 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 and updates and patching to those modules.

The brief descriptions of the five baseline functions listed above are 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~~ Mediation & Access Control (HY-BF2)	~~Emulator software (e.g., QEMU)~~Device	Either in a dedicated VM (~~i.e., a parent partition or management VM~~called Device-driver VM) or in the hypervisor kernel itself

	emulator or Device driver	
Direct Execution of commands from Guest VMs ~~Execution of Privileged Operations for Guest VMs~~ (HY-BF3)	Hypervisor Kernel	Pertain to only para-virtualized hypervisors and handled by hypercall interfaces in that type of hypervisor
VM Lifecycle ~~Management~~ management (HY-BF4)	A management daemon	Installed on top of hypervisor kernel but runs in unprivileged mode
Management of ~~Hypervisor~~ hypervisor platform (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 by ~~either using~~ a software module that runs either ~~outside of the hypervisor (usually~~ in a dedicated ~~privileged~~ VM (called Device-driver VM) or in the hypervisor kernel itself. The functions HY-BF4 and HY-BF5 are performed by a module called management or service console or through a kernel module. Just like the module that performs the HY-BF2 function, 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 it or with an ultra-light OS used to present an API (shell and network access) with utility functions that facilitate performing only the hypervisor-specific configuration and administrative tasks.

1.2 Scope of this document

The architecture of a hypervisor deployed for server virtualization can be classified in different ways:

(a) Based on the entity over which the hypervisor installs – Type 1 Hypervisor or Type 2 Hypervisor (already described)

(b) Based on the type of ~~emulation~~virtualization
   a. Full Virtualization – Guest OS runs unmodified in the VM
   b. Para Virtualization – 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 are untrusted including the guest OS and its associated utilities (e.g., guest device drivers) that run in the kernel space and all applications that run in the user space
- The device drivers that are implemented within the hypervisor platform are untrusted unless they carry a security certification
- The hypervisor kernel component that provides isolation between VMs is trusted
- The host OS is trusted for Type 2 hypervisors
- The hardware of the hypervisor host is trusted

With the background information on hypervisor architecture and the assumed trust model, the scope of security recommendations in this document that pertain to the five baseline functions HY-BF1 through HY-BF5 covers the following:

- All tasks that relate to functions HY-BF1, HY-BF2, and HY-BF4
- HY-BF3, which relates to the handling of hypercalls in para-virtualized hypervisors, is a trusted function of the hypervisor and not included in the security recommendations

- All tasks under HY-BF5 are included, except for those related to the definition and configuration of virtual network (secure configuration of virtual networks is covered under a separate NIST document, SP 800-125B)

Also within the scope of this document are recommendations 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 (e.g., keeping patches current)
- Routine administration of Guest OS
- 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 (SP) 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 and Desktop Virtualization. Since then, Server Virtualization has found widespread adoption in IT data centers both for hosting in-house or on-premises (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 sets of one of the core layers of virtualization—the hypervisor—as well as 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 of its constituent modules) including the steps involved in the 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:

- A 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) is provided.
- Since real world deployment includes provisioning of VMs, all VM life-cycle operations, from creation and management of VM images to their administration using granular privileges, is covered.

- Recognizing that the hypervisor is a purpose-built Operating System (OS) kernel and that the security of a server OS depends upon its weakest link regardless of the distribution (e.g., driver software), security recommendations relating to these components have been provided as well.
- Recognizing that the hypervisor performs certain privileged operations without interference from any other entity in the virtualized host and that leveraging hardware support for these operations will make a significant difference to the overall security of hypervisor deployment, the security recommendations also improve performance when virtualization-specific functions (e.g., memory tables for multiple VMs) are offloaded (leveraged) to the processor instead of through software functions.
- All security recommendations are intended to provide assurance against exploitation of threats to tasks involved in the hypervisor's baseline functions.

2. APPROACH FOR DEVELOPING SECURITY RECOMMENDATIONS

Developing security recommendations for the deployment and use of a complex software such as the hypervisor requires knowledge of potential threats that, when exploited, would affect the three basic 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, the 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 noted that in some cases of large open-source and commercial software environments (e.g., Database Management System (DBMS) platform), the approach adopted for secure deployment and usage is to study the reports published in the public vulnerability databases for various product offerings, seek out available patches through online public forums or the software vendor, and look for recommended secure configuration settings (also via online public forums or the software 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 rather 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 supports multiple virtual hosts (virtual machines or VMs) that are generally nodes of a software-defined virtual network inside that physical host. In some cases, they could be nodes of an isolated network or sharing the host network. Based on this scenario, one can identify three basic sources of threats to a hypervisor platform, each of which 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 consoles

Threats from sources HY-TS1 and HY-TS3 are common to all server class software and are well known and 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.

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 reside 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, defining virtual local area networks (VLANs) or overlay networks, and 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:

Note that each threat is identified by the symbol HYP-T#, where HYP stands for hypervisor, T stands for threat, and # stands for the sequence number.

- Breach of Process Isolation - VM Escape (HYP-T1): Major threats to any hypervisor come from rogue VMs. Rogue VMs 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 to access. Possible reasons for this threat include (a) hypervisor design vulnerabilities or (b) malicious or vulnerable device drivers. Potential downstream impacts of a rogue VM taking control of the hypervisor include the installation of rootkits or attacks on other VMs on the same virtualized host.

- Breach of Network Isolation (HYP-T2): Potential threats to isolation include attacks such as IP or MAC address spoofing by a rogue VM and Traffic Snooping, or the interception of 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 for which they are not authorized.

- 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 on the hypervisor host.

2.2 Potential Threats to Hypervisor Baseline Functions

In this section, the tasks in each of the five hypervisor baseline functions (with the exception of HY-BF3) are examined, and the threats to the secure execution of those tasks are analyzed by relating to the causes identified in the previous section.

2.2.1 *Potential Threats to HY-BF1*

The primary threat to hypervisor's HY-BF1 function (VM Process Isolation) is breach of process isolation (HYP-T1). As mentioned in section 2.1, one of the causes for this threat is hypervisor design vulnerability. Some potential design vulnerabilities that pertain to this threat are discussed here with an explanation of the context under which they may manifest. Each vulnerability is identified by the symbol HYP-DV#, where HYP stands for hypervisor, DV stands for design vulnerability, and # stands for the sequence number.

- Virtual Machine Control Structure (HYP-DV1): To properly schedule an individual VM's tasks (i.e., vCPU tasks since each guest VM is allocated a set of virtual CPUs), the register states must be handled appropriately. To enable the 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 or critical instructions, send them to the VMM (hypervisor), and replace them with safer instructions 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 a guest OS being allowed to execute privileged instructions.

- Memory Management Unit-MMU (HYP-DV3): The hypervisor runs a software-based Memory Management Unit (MMU) that 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, a 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 a breach of memory isolation.

- Input/Output Memory Management Unit, IOMMU (HY-DV4): The hypervisor leverages the hardware I/O Memory Management Unit to enforce memory separation for device drivers and processes using direct memory access (DMA). This feature is built into the hypervisor and enabled in the hardware using a firmware switch. If unused, it may result in a vulnerability whereby the DMA could potentially be used as a common attack vector by one VM to overwrite physical memory used by other VMs and processes.

-

Out of these, the vulnerabilities HYP-DV1 and HYP-DV2 should be addressed through proper coding and testing of those modules. Therefore, no security protection measures can be applied at the deployment and usage stage. However, the memory violation vulnerability HYP-DV3 and DMA violation vulnerability HY-DV4 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 and DMA transfers through the re-mapping of DMA transfers, respectively. Due to these two vulnerabilities, the threat HYP-T1, a breach of process isolation, has been addressed through security recommendation HY-SR-2 in section 4.

Further, correct execution isolation requires that each VM obtains the proper memory and CPU resources necessary 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 the appropriate configuration of vCPU allocation options are addressed through security recommendations HY-SR-4 and HY-SR-5.

2.2.2 *Potential Threat to HY-BF2*

The applications executing in VMs need to access devices such as network and storage. Mediation of access to devices is handled in hypervisor hosts through device virtualization (also called IO virtualization). There are three common approaches to device virtualization: (a) Emulation, (b) Para-virtualization, and (c) Passthrough or self-virtualizing hardware devices.

In emulation, a code is implemented to present a virtual device that has a corresponding real (hardware) device for which the guest OS already has a driver for. This enables running of unmodified guests (VMs), thus implementing full virtualization. This emulation code runs in the hypervisor. An I/O call from a guest VM application (through its guest OS) is intercepted by the hypervisor kernel and forwarded to this code since guest VMs cannot access the physical devices directly under this setup. This emulation code traps all device access instructions and converts them to calls on the physical device driver for the physical device attached to the hypervisor host. ~~The main tasks under this function are: (a) emulation of storage and networking devices and (b) controlling access to them from various VMs. These tasks are handled by a device emulation and access code that run in the kernel. Any I/O call from a guest VM application is intercepted by the hypervisor kernel and forwarded to this code since guest VMs cannot typically access the physical devices directly unless they are assigned to it. This code emulates devices, mediates access to them, and~~It also multiplexes accesses from guest VMs' emulated virtual devices to the underlying physical device. ~~multiplexes the actual devices since each permitted VM has full access to the underlying physical device.~~ ~~The security recommendations relating to the 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 in section 5.~~

~~In addition to the faulty implementation of in-memory data structures for virtual devices, a potential threat to the secure execution of this function arises from faulty device driver code. The security recommendations relating to the 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 in section 5.~~ In the para-virtualization approach, the hypervisor presents to the guest an interface of an artificial device that has no corresponding hardware counterpart. This enables special, simplified hypervisor-aware I/O drivers (called para-virtualized drivers) to be installed in the guest. The calls from these para-virtualized device drivers in guest VMs are handled by another device driver (called back-end driver) which directly interfaces with the physical device and mediates access to that physical device from para-virtualized guests. In some instances, the calls from para-virtualized guest drivers are handled directly by the hypervisor through its hypercall interface (the corresponding calls are called hypercalls). Analysis of threats due to these hypercalls is provided in the next subsection.

The third approach to device virtualization, the passthrough approach (or direct device assignment), is deployed for situations where a VM needs exclusive access to a device (e.g., NIC, disk controller, HBA, USB controller, serial port, firewire controller, soundcard, etc) for performance reasons so as to be devoid of the overhead due to emulation. Since generally this is required for PCI devices, this is also called as PCI Passthrough. Since many of these devices have a memory-mapped interface, they can read or write directly to or from main memory and are also called Direct Memory Access (DMA) capable devices. To provide exclusive access to a DMA capable device for a VM, the memory pages of the device are mapped into guest VM's address space. The following is the threat due to DMA capable devices.

Threat due to DMA-capable hardware devices (HY-DV5): The security threat from DMA-capable device is that, since the VM controls the device, it can program the device to perform DMA operations directed at any physical (host) memory location, including the areas belonging to other VMs or the hypervisor [6]. Thus, the direct device assignment has the potential to subvert the isolation between VMs (rather making the MMU enforced isolation function (part of HY-BF1) meaningless).

Apart from three types of device virtualization described above, hypervisor hosts can support self-virtualizing hardware devices. These devices have interfaces that can export a set of virtual functions (VFs) corresponding to a physical function (PF). The hypervisor then can assign these VFs to multiple guest VMs, while it retains control of the PF. These devices conform to Single Root I/O Virtualization (SR-IOV) specification and thus enable DMA capable devices to be shared among VMs (as virtualization and multiplexing are done by the devices themselves) instead of being dedicated to a single VM as in passthrough mode.

2.2.3 *Potential Threat to HY-BF3*

~~Certain privileged operations (e.g., Memory Management) invoked by guest VMs are executed by the hypervisor handling them using mechanisms such as VM Exits (in which operations are processor-architecture specific) or Hypercalls (hypervisor-specific and similar to system calls to OS).~~ The previous subsection presented a scenario where the hypervisor has to execute certain instructions through its hypercall interface. A potential security issue with hypercalls is that the lack of~~Lack of~~ proper validation of ~~those~~ certain operations (e.g., not checking the scope, allowing a full dump of a VM's Virtual Machine Control Block, or input checking) ~~would~~ can potentially cause the entire ~~virtualized~~ hypervisor host to crash. This is again a design vulnerability that must be addressed through proper validation and testing of the relevant hypervisor code rather than through ~~deployment and usage tasks~~configuration or deployment procedures.

2.2.4 *Potential Threats to HY-BF4*

Potential threats to the secure execution of tasks under this function (i.e., VM Lifecycle Management) include:

- Presence of non-standard VM images in the library, including those with outdated OS versions and patches, which could result in any of the platform-level threats (HYP-T1 through HYP-T3)
- Presence of non-standard running VM instances due to their creation from non-standard images, restoration from snapshots, a drift from standard as a result of a lapse in monitoring, and updates that could result in any of the platform-level threats (HYP-T1 through HYP-T3)

In most instances, the management operations on VMs are performed using commands submitted through a GUI or a scripting environment, both of which are supported by a management daemon at the back-end. Secure execution of the above operations is addressed through security recommendations HY-SR9 through HY-SR18 in section 6.

2.2.5 *Potential Threats to HY-BF5*

The tasks under this function relate to the overall administration of a hypervisor host (i.e., virtualized host) and the hypervisor software and are usually performed through user-friendly web interfaces or network-facing virtual consoles. Threats to the secure execution of these tasks are common in any remote administration and are therefore 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 such as sensitivity of applications based on the set of hosted VMs, line of business or client in cloud service environments, etc. Thus, the security recommendations include a centralized management of hypervisor configuration (HY-SR-19) and a dedicated network segment for management traffic (HY-SR-20).

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 to preventing the server from 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. Instead, a specialized security fix, such as disabling the virtual NICs of VMs that use those ports, is needed.

~~4.~~ 3. SECURITY RECOMMENDATION FOR OVERALL PLATFORM INTEGRITY

Configuration changes, module version changes, and patches affect 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 that are part of the hypervisor stack can be trusted, 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 protects 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 trusted 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, trust in the hypervisor's components is established based on trusted hardware. In other words, a chain of trust from hardware to hypervisor is established with the initial component called *the root of trust*. This service can be provided by a hardware/firmware infrastructure of the hypervisor host that supports boot integrity measurement and the attestation process. In short, a measured launch environment (MLE) is needed in the hypervisor host.

Some hardware platforms provide support for MLE with firmware routines for measuring the identity (usually the hash of the binary code) of the components in a boot sequence. An example of a hardware-based cryptographic storage module that implements the measured boot process is the standards-based Trusted Platform Module (TPM), which has been standardized by the Trusted Computing Group (TCG) [4]. The three main components of a TPM are: (a) Root of Trust for Measurement (RTM) – makes integrity measurements (generally a cryptographic hash) and converts them into assertions, (b) Root of Trust for Integrity (RTI) - provides protected storage, integrity protection, and a protected interface to store and manage assertions, and (c) Root of Trust for Reporting (RTR) - provides a protected environment and interface to manage identities and sign assertions. The RTM measures the next piece of code following the boot sequence. The measurements are stored in special registers called Platform Configuration Registers (PCRs).~~The TPM has special registers called Platform Configuration Registers (PCRs) for storing the various measurements. Its architecture consists 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 component's identities, measurements, and other sensitive information. The RTM also measures the next piece of code following the boot sequence.~~

The measured boot process is briefly explained here using TPM as an example. 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 off control, a chain of trust is established. If the measurement chain continues through the entire boot sequence, the resultant PCR values reflect the measurement of all components.

The attestation process starts with the requester invoking, via an agent on the host, the TPM Quote command. It specifies an Attestation Identity Key (AIK) to perform the digital signature on the contents of the set of PCRs that 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 requester validates the signature and determines the trust of the launched components by comparing the measurements in the TPM quote with known good measurements.

The MLE can be incorporated in the hypervisor host as follows:

- 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 and to build a chain of trust, it should have a hardware-based module that supports an MLE. 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 utilize 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 into the hypervisor's code base since the kernel is the first module installed in a hypervisor boot up. The purpose of this pre-kernel module is to ensure the selection of the right authenticated module in the hardware that performs an orderly evaluation or measurement of the launch components of the hypervisor or any software launched on that hardware. The Tboot is an example of a mechanism that enables the hypervisor to take advantage of the MLE feature of the hardware.
- All hypervisor components that are intended to be part of the Trusted Computing Base (TCB) must be included within the scope of the MLE-enabling mechanism so that they are measured as part of their launch process.

The MLE feature with storage and reporting mechanisms on the hardware of the 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 with firmware, BIOS, hypervisor and hypervisor modules; comparing them to "known good values;" and reporting any discrepancies. If the measured boot process is to be extended to cover VMs and its contents (guest OS and applications), a software-based extension to the hardware-based MLE implementation within the hypervisor kernel is required. The security recommendation for ensuring a secure boot process for all components of a hypervisor platform can now be stated as follows:

Security Recommendation HY-SR-1: The hypervisor that is launched should be part of a platform and an overall infrastructure that contains: (a) hardware that supports an MLE with standards-based cryptographic measurement capabilities and storage devices and (b) an attestation process with the capability to provide a chain of trust starting from the hardware to all hypervisor components. Moreover, the measured elements should include, at minimum, the core kernel, kernel support modules, device drivers, and the hypervisor's native management applications for VM Lifecycle Management and Management of Hypervisor. The chain of trust should provide assurance that all measured components have not been tampered with and that their versions are correct (i.e., overall boot integrity). If the chain of trust is to be extended to guest VMs, the hypervisor should provide a virtual interface to the hardware-based MLE.

4. SECURITY RECOMMENDATION HY-BF1

To ensure the isolation of processes running in VMs, the following requirements must be met:

(a) The privileged commands or instructions from a Guest OS 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 managing memory access by multiple VMs.
(c) Memory allocation algorithms must ensure that payloads in all VMs are able to perform their functions.
(d) CPU~~/GPU~~ allocation algorithms must ensure that payloads in all VMs are able to perform their functions.

The requirements (a) and (b) can be met using software-based modules. However, hardware-based assistance for virtualization, such as Instruction Set Virtualization and Memory Virtualization, provide better assurance than software-based solutions in meeting those requirements and are therefore recommended in section 4.1. The hardware-assisted virtualization features are briefly discussed prior to stating the recommendations. The requirements (c) and (d) are meant to ensure the availability of application services running in VMs. The enablers are some features in memory allocation and CPU allocation algorithms, and their associated configuration parameters are stated as recommendations in sections 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 of which have four hierarchical privilege levels with Level 0 being the highest and Level 3 being the lowest. Additionally, 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 safety from at least any instruction set-type attacks by any Guest OS. However, VM escape can take place through normal networking protocols. This safety is ensured by allowing the hardware trapping privileged instructions to run in non-root mode and execution in root mode. Additionally, when the hypervisor does not have to perform additional functions (e.g., translating sensitive instructions using techniques such as binary translation), the code executing with privileges is reduced in the hypervisor, making the TCB smaller and enabling better assurance verification.

Memory Virtualization: Hardware-assisted memory virtualization is provided when the hardware enables the mapping of the Guest OS's physical addresses in their respective page tables to the host's physical addresses using hardware-based page tables instead of hypervisor-generated shadow page tables. The subsequent 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 include the following:

- 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 (e.g., Extended Page Tables, or EPT) 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 or root mode and guest or non-root mode. The host mode runs at a higher privilege than guest mode. The hypervisor code, which provides the baseline functionality HY-BF1 (processor allocation and memory management),

runs in host mode while the guest OS and applications in VMs run in guest mode. Hence any exploit code in guest OS cannot subvert the controls provided by the hypervisor code.

- A common threat in virtualization platforms involves a malicious VM accessing areas of memory belonging to other VMs. This is called a VM Escape attack. Hardware platforms with IOMMU provide safety against this through features such as Direct Memory Access (DMA) remapping, which limits allowed DMA access to the assigned protection domain (i.e., preventing a device from performing DMA beyond its allocated area).

~~Processors with virtualization extensions provide safety against this through features such as Direct Memory Access (DMA) remapping, which limits DMA access to what is valid for the VM (i.e., preventing a VM from performing DMA beyond its allocated area).~~

- The advantage of 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 an unmodified Guest OS, along with their native device drivers, can be run in VMs. The security implication of enabling this feature is that significantly more CVE data is available for a Guest OS, as well as patch versions and certified device drivers for each OS version.

Security Recommendation HY-SR-2: The hardware of the virtualized host should provide assistance for virtualization for instruction sets~~,~~ and memory management using MMU~~, and DMA transfers using IOMMU~~ since the hardware support provides the following security assurances that cannot be guaranteed with purely software-based virtualization:

- Better memory management controls can prevent attacks such as buffer overflow.
- The feature for re-mapping of DMA transfers in IOMMU provides better isolation of I/O devices. Further, the feature to directly assign I/O devices to a specific VM and enable direct access to those resources eliminates the need for providing emulated device drivers for that VM, thus reducing the size of ~~un~~trusted code.
- Guest OS code and hypervisor code execute in different processor modes, providing better isolation.
- Privilege-level isolation can provide better protection for device access mediation functions, and hardware-based memory protection can provide better VM-level protection.
- By supporting full virtualization, COTS versions of OSs can allow for easier patching and updating than having to perform the same operations on modified or 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 and verification.

4.2 VM Memory Allocation Scheduling Options

The hypervisor's memory scheduler is responsible for meeting the memory requirements for all workloads running in all VMs at all times. Like an OS, a typical hypervisor meets this requirement by using a combination of physical RAM and swap files called hypervisor kernel swap files. Further, a typical VM does not always require the entire memory it has been configured for. For these 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 that 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 in performance degradation of certain VM workloads that require a significant amount of memory.

Another factor affecting the availability of the virtualized host or 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 in which 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. Finally, there may be certain workloads that are time-sensitive, and the VMs hosting them should have some priority in getting the required memory resources compared to other running VMs. Therefore, a configuration option to specify a priority value for each VM should also exist.

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, as well as a limit to this value, and a priority value for obtaining the required RAM resource in situations of contention among multiple VMs. Further, the over-commit feature that enables the total configured memory for all VMs to exceed the host physical RAM should be disabled by default.

4.3 VM CPU Allocation Options

The security goal in VM CPU allocation is to guarantee availability for all VMs. This can be achieved by proper use of configuration options dealing with the allocation of physical resources such as CPU cores and CPU clock cycles. For example, one of the configuration options commonly available is to set a minimum CPU requirement, or 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. In a scenario where the hypervisor host has 6000 MHz of CPU capacity and the average reservation for each VM is 1000 MHZ, then no more than 6 VMs can be active in that hypervisor host. The reservation thus 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, or Limit, for the CPU cycles that each VM can use so that no single 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 be a feature to assign a priority score, or shares, to each VM. 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 hypervisor should have robust configuration features for provisioning virtual resources to all hosted VMs such that it does not exceed a key physical resource (e.g., number of CPU cores).

<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 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 access 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.~~Security recommendations for all three forms of device virtualization discussed in section 2.2.2 as well as for self-virtualized devices are provided in this section.~~ 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-6A (Emulation): ~~All device~~Because of the complexity of emulating a hardware device through software, emulation, apart from suffering performance penalties, also increases the size of the TCB especially in situations where the guest OS has native device drivers and the device emulation code runs as a kernel module with the same privilege level as the hypervisor. Hence emulation should only be used where complexity is manageable (e.g., USB host controller).

Security Recommendation HY-SR-6B (Para-virtualization): In situations where para-virtualized device drivers are used in VMs, mediation of access to physical devices should be enabled by running back-end device drivers (which control the physical device attached to the hypervisor host) in a dedicated VM rather than in the hypervisor. This facilitates running the back-end device driver code at a privilege level lower than that of the hypervisor. Additionally, the hypervisor platform should include hardware support in the form of I/O Memory Management Unit (IOMMU) for validating and translating access from the driver domain's underlying hardware device to host memory. The specific IOMMU feature that is mandatory is DMA remapping where the DMA call from a device to guest physical address (GPA) must be translated to host physical address (HPA) and then checked whether the HPA address falls within the protection domain assigned to that device. Combining these mechanisms enables reducing the size of TCB as well as reducing the impact of faulty device or device driver behavior (restricted to device-driver VM as opposed to the hypervisor).
~~drivers installed as part of a hypervisor platform should be configured to run as lower-privileged level process or user mode, rather than the privilege level of the hypervisor or kernel mode. If device drivers are run in the same privilege level as the hypervisor (e.g., kernel mode/kernel space), they should be implemented, designed, and tested using formal verification to guarantee that the drivers cannot compromise the security of hypervisor execution. This recommendation applies to any code running at the same privilege level as the hypervisor in the kernel (e.g., VMM).~~

Security Recommendation HY-SR-6C (Passthrough or self-virtualizing hardware devices): For situations, where VMs needs to be given dedicated access to DMA capable devices, the hypervisor platform should include hardware support in the form of I/O Memory Management Unit (IOMMU) for validating and translating all device access to host memory. This recommendation also applies to use of self-virtualizing hardware devices (based on SR-IOV specification). The specific IOMMU feature that is mandatory is DMA remapping where the DMA call from a device to guest physical address (GPA) must be translated to host physical address (HPA) and then checked whether the HPA address falls within the protection domain assigned to that device.

The following security recommendations are applicable irrespective of the type of device virtualization:

Security Recommendation HY-SR-7 (Device access): It should be possible to set up an Access Control List (ACL) to restrict the 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 VMs (semantically, a set of tasks) and/or have a feature to specify a whitelist, or 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 (DOS) attacks. Additionally, the proper use of resource limits localizes the impact of a DOS to the VM or the cluster for which the resource limit is defined.

~~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. Thus, only recommendations HY-SR-7 and HY-SR-8 are applicable to para-virtualized hypervisors.~~

6. SECURITY RECOMMENDATIONS FOR HY-BF4

6.1 VM Image Management

Since VM-based software (e.g., Guest OS, Middleware, and Applications) shares physical 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, but rather from VM Images. VM Images are templates used for creating running versions of VMs. An organization may have its own criteria for classifying the different VM Images it uses in its VM Library. Some commonly used criteria include: processor load (VM used for compute-intensive applications); memory load (VM used for memory-intensive applications, such as Database processing); and application sensitivity (VM running mission-critical applications utilizing mission-critical data). For each VM image type, the following practices must be followed to ensure that the resulting operational VMs are secure:

- Documentation on the Gold 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 an associated 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 have a secure protocol.

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

<u>Security Recommendation HY-SR-9</u>: Gold standard must be defined for VMs of all types, and VM Images that do not conform to the standard should not be allowed to be stored in the VM Image server or library. Images in the VM Image library should be periodically scanned for outdated OS versions and patches, which could result in a drift from the standard.

<u>Security Recommendation HY-SR-10</u>: 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.

<u>Security Recommendation HY-SR-11</u>: Permissions for checking into and out of images from the 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 or closed by a limited set of authorized administrators with passphrases of sufficient complexity.

<u>Security Recommendation HY-SR-12</u>: 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 present in all hypervisors, which 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, and host maintenance, upgrades, and patching. 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 storage), and network state.

The most common memory migration technique adopted in most hypervisors is 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 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. Storage migration 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. For example, a VM may continue to run on the host server while 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 aspects of hypervisor design. The storage migration function is an integral part of storage management and is applicable to 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 restrictions 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 consider are proper authentication and a secure network path for the migration process.

Security Recommendation HY-SR-13: During VM live migration, a secure authentication protocol must be employed; the credentials of the administrator performing the migration are 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 threats 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 and 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 with 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 a VM-based Security Monitoring and Intervention approach, software or a software-agent (i.e., a security tool) is run inside a VM to monitor security-relevant events. This approach is similar to running 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 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 and other application workloads running on that VM.

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

(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 that if the VM is migrated to some other physical host, the dedicated appliance must 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 for the following reasons:

- The set of VMs to be monitored is continuously in a state of flux since VMs are subject to migration from one virtualized host to another due to load balancing, performance, and even security reasons.
- If virtual LANs (VLANs) are used to provide communication-level isolation among VMs, the configuration of VLANs may undergo continuous change as the workload patterns shift 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 monitors and protects VMs (User VMs) is run outside of the VMs hosting business applications in a special 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 (e.g., 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 preferable solution since:

(a) It is not vulnerable to a flaw in the Guest OS.
(b) It 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.

Therefore, the security recommendations, with respect to creating the VM monitoring solution for the protection of the hypervisor, are as follows:

<u>Security Recommendation HY-SR-14</u>: There should be a mechanism for security monitoring, security policy enforcement of VM operations, and detecting malicious processes running inside VMs and malicious traffic going into 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.

<u>Security Recommendation HY-SR-15</u>: Solutions for Security Monitoring and security policy enforcement of VMs should be based outside of VMs and 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 or trusted VM.

<u>Security Recommendation HY-SR-16</u>: All antimalware tools (e.g., 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 native features that come with the hypervisor. The desired features for these tools are provided in the form of security recommendation below:

Security Recommendation 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.

6.5 Fine-grained Administrative Privileges for VM Management

Having the ability to assign fine-grained administrative permissions for the virtualized infrastructure enables the establishment 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:

- VM Administration Use Case 1: 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 go 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.

- VM Administration Use Case 2: A capacity planner assigned the task of determining the operating loads on various virtualized servers and the need for additional virtualized hosts may need permission to view 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 ability to grant view rights to the list of VMs in a virtualized host but deny the user the rights to interact with any of the visible objects.

- VM Administration Use Case 3: In virtualized data centers where VMs of different sensitivity levels are run on the same virtualized host, an administrator who is given administrative privileges at the hypervisor level should sometimes be prevented from accessing a specific VM because of the sensitive nature of the workload (i.e., set of applications) running on that VM. The desired capability in this scenario is to negate a permission, obtained through inheritance, for a specific child object.

- VM Administration Use Case 4: In some cases, assign permissions are needed 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., web server), 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 rather some arbitrary set of management functions such as Configure CD Media, Configure Floppy Media, Console Interaction, Device Connection, Power On, Power Off, Reset, or Suspend. 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., web server).

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

Security Recommendation HY-SR-18: The access control solution for VM administration should have a granular capability, both at the permission assignment level and 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 HY-BF5

Secure operation of administrative functions is critical for any server class software, and hypervisor is no exception to this. The outcome is a secure configuration that can provide the necessary protections against 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. Many configuration options relate functional features and performance. However, there are some options that have a 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. Although applicable to the hypervisor, these 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 and hypervisor hosts through enterprise virtualization management software.

Central management of all hypervisor platforms in the enterprise through enterprise virtualization management software (EVMS) is preferable since a gold-standard configuration for all hypervisors in the enterprise can be defined and easily enforced through EVMS. For any IT data center to operate efficiently, it is necessary to implement load balancing and fault tolerance measures, which can be realized by defining hypervisor clusters. Creation, assignment of application workloads, and management of clusters can be performed only with 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). Enterprise gold-standard hypervisor configurations for different types of workloads and clusters must be managed and enforced through EVMS. The gold-standard configurations should, at minimum, cover 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 allows for a software-defined communication fabric, or a virtual network, through its management console or command line interface (CLI). 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 the VMs residing inside it as its nodes. 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 must be considered:

- Providing selective connectivity or isolation between groups of VMs belonging to different logical groupings (e.g., different tenants in the case of an Infrastructure as a Service (IaaS) cloud service; different application tiers such as Web Server or Database Server; 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 (a node of the virtual network), which is used for performing key hypervisor baseline functions of VM lifecycle management (HY-BF4) and ~~management~~ Management of hypervisor platform (HY-BF5)

Out of the three functionalities stated above, selective connectivity and isolation between groups of VMs is required for providing security to the applications running on those VMs and therefore outside of 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 virtual network configuration is limited to providing protection for the network interfaces used for performing VM management and hypervisor administrative functions. A commonly adopted approach is to allocate a dedicated physical network interface card (NIC) for handling management traffic, and, if that is not feasible, a virtual network segment (vLAN ID) exclusively for it.

Security Recommendation HY-SR-20: Protection for hypervisor host and software administration functions should be ensured by allocating a dedicated physical NIC or, if that is not feasible, 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 virtualizes hardware resources to enable the execution of multiple computing stacks (VMs) with heterogeneous OSs and multiple applications hosted within them. Secure configuration of the hypervisor, together with its physical host (i.e., hypervisor host or virtualized host), is collectively called the hypervisor platform and is needed to provide a safe platform for the execution of mission-critical applications.

Since there are 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, the tasks involved in each baseline function, the potential threats to secure execution of the task, and to 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 the configuration of parameters of software modules in the hypervisor platform. These parameters include integrity metrics for software modules (e.g., device drivers and VM images), the setting of access controls (e.g., device access, VM image access, and VM administration), and the configuration of secure protocols (e.g., VM image server access and VM migration). The mapping of the security recommendations to a hypervisor's baseline functions is provided in Appendix B.

The trust model outlined in this document (refer section 1.2) assumes that the hardware of the hypervisor host is trusted. However, it must be mentioned that there have been reported case of attacks (e.g., side channel attacks regarding some implicitly shared hardware resources such as CPU caches and Translation Lookaside Buffers (TLB)). More recently published attacks concerning CPU-level performance optimizations (e.g., Spectre and Meltdown) also limit the assurance of trust on current hardware platforms used for hypervisor deployment."

Appendix A

Detailed descriptions of each of the five hypervisor baseline functions are provided below. As stated in the Introduction, 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. If DMA capable devices are used in the hypervisor host, memory access to those devices need to be controlled as well. However, this function is considered under HY-BF2 since it pertains to devices mediation.
- HY-BF2: Devices Mediation & Access Control – Mediates access to all devices (e.g., Network Interface Card (NIC), storage device such as IDE drive, etc.)
- HY-BF3: Direct Execution of commands from Guest VMs – Certain commands from Guest OSs are executed directly by the hypervisor instead of being triggered through interrupts and context switching. This function applies to hypervisors that have implemented para-virtualization instead of full virtualization.

~~HY-BF1: VM Process Isolation – Scheduling of VMs for execution, management of the application processes running in VMs (e.g., CPU and memory management), and context switching between various processor states during the running of applications in VMs~~

- ~~HY-BF2: Devices Emulation & Access Control – Emulating all network and storage (block) devices expected by different native drivers in VMs and 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~~hypervisor platform– 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 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. In order to ensure VM process isolation, memory access from DMA capable devices needs to be under hypervisor control as well (e.g., via IOMMU). However, this function is considered under HY-BF2 since it pertains to devices mediation.

~~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~~ Mediation & Access Control)

The applications executing in VMs need to access devices such as network and storage. Mediation of access to devices is handled in hypervisor hosts through device virtualization (also called IO virtualization). There are three common approaches to device virtualization: (a) Emulation, (b) Para-virtualization, and (c) Passthrough or self-virtualizing hardware devices.

In emulation, a code is implemented to present a virtual device that has a corresponding real (hardware) device for which the guest OS already has a driver for. This enables running of unmodified guests (VMs), thus implementing full virtualization. This emulation code runs in the hypervisor. An I/O call from a guest VM application (through its guest OS) is intercepted by the hypervisor kernel and forwarded to this code since guest VMs cannot access the physical devices directly under this setup. This emulation code traps all device access instructions and converts them to calls on the physical device driver for the physical device attached to the hypervisor host. It also multiplexes accesses from guest VMs' emulated virtual devices to the underlying physical device.

In the para-virtualization approach, the hypervisor presents to the guest an interface of an artificial device that has no corresponding hardware counterpart. This enables special, simplified hypervisor-aware I/O drivers (called para-virtualized drivers) to be installed in the guest. The calls from these para-virtualized device drivers in guest VMs are handled by another device driver (called back-end driver) which directly interfaces with the physical device and mediates access to that physical device from para-virtualized guests. In some instances, the calls from para-virtualized guest drivers are handled directly by the hypervisor through its hypercall interface (the corresponding calls are called hypercalls). Analysis of threats due to these hypercalls is provided in the next subsection.

The third approach to device virtualization, the passthrough approach (or direct device assignment), is deployed for situations where a VM needs exclusive access to a device (e.g., NIC, disk controller, HBA, USB controller, serial port, firewire controller, soundcard, etc) for performance reasons so as to be devoid of the overhead due to emulation. Since generally this is required for PCI devices, this is also called as PCI Passthrough. Since many of these devices have a memory-mapped interface, they can read or write directly to or from main memory and are also called Direct Memory Access (DMA) capable devices. To provide exclusive access to a DMA capable device for a VM, the memory pages of the device are mapped into guest VM's address space. The following is the threat due to DMA capable devices.

Apart from three types of device virtualization described above, hypervisor hosts can support self-virtualizing hardware devices. These devices have interfaces that can export a set of virtual functions (VFs) corresponding to a physical function (PF). The hypervisor then can assign these VFs to multiple guest VMs, while it retains control of the PF. These devices conform to Single Root I/O Virtualization (SR-IOV) specification and thus enable DMA capable devices to be shared among VMs (as virtualization and multiplexing are done by the devices themselves) instead of being dedicated to a single VM as in passthrough mode.

~~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 (Direct Execution of commands from Guest VMs~~Execution of Privileged Operations for Guest VMs~~):

Certain commands from Guest OSs are executed directly by the hypervisor instead of being triggered through interrupts and context switching. These commands are called hypercalls and are supported by a special interface in the hypervisor.This function applies only to hypervisors that have implemented para-virtualization instead of full virtualization.
~~Certain operations invoked by Guest OS kernels may have to be executed by the hypervisor because of their privileged nature. These calls are called 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 but not limited to*:*

- Creation of VMs conforming to a standard image, ensuring integrity of images and secure storage and retrieval of images; provisioning images with appropriate vCPU, RAM, network, and storage
- 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.
- Access control and management of snapshots

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~~hypervisor platform)

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 with standards-based cryptographic measurement capability and storage device and (b) Attestation process that should contain capabilities to take advantage of these to provide a chain of trust starting from the Hardware to all Hypervisor components. 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 chain of trust should provide assurance that all measured components have not been tampered with and that their versions are correct (i.e., overall boot integrity). If the chain of trust is to be extended to guest VMs, the hypervisor should provide a virtual interface to the hardware-based MLE.*	N/A
HY-SR-2	~~*The hardware of the virtualized host should provide assistance for virtualization (for instruction set, memory management (using MMU) and DMA transfers (using IOMMU)) as the hardware support provides the following security assurances that cannot be guaranteed with purely software-based virtualization because of the following:*~~ ~~• *Better memory management controls can prevent attacks such as buffer overflow.*~~ ~~• *The feature for re-mapping of DMA transfers in IOMMU provides better isolation of I/O devices. Further the feature to directly assign I/O devices to a specific VM (enabling direct access to those resources), eliminates the need for providing emulated device drivers for that VM, thus reducing the size of untrusted code.*~~ ~~• *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.*~~ *The hardware of the virtualized host should provide assistance for virtualization for instruction sets and memory management using MMU since the hardware support provides the following security assurances that cannot be guaranteed with purely software-based virtualization:*	HY-BF1 (VM Process Isolation)

	• *Better memory management controls can prevent attacks such as buffer overflow.* • *The feature for re-mapping of DMA transfers in IOMMU provides better isolation of I/O devices. Further, the feature to directly assign I/O devices to a specific VM and enable direct access to those resources eliminates the need for providing emulated device drivers for that VM, thus reducing the size of trusted code.* • *Guest OS code and hypervisor code execute in different processor modes, providing better isolation.* • *Privilege-level isolation can provide better protection for device access mediation functions, and hardware-based memory protection can provide better VM-level protection.* • *By supporting full virtualization, COTS versions of OSs can allow for easier patching and updating than having to perform the same operations on modified or 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 and verification.* ~~*hardware, the size of the hypervisor code will be small enabling better security attestation/verification.*~~	
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. Further, the over-commit feature (if available) that enables the total configured memory for all VMs to exceed the host physical RAM should be disabled by default.*	HY-BF1 (VM Process Isolation)
HY-SR-4	*The hypervisor should have robust configuration features for provisioning virtual resources to all hosted VMs in a way that it does not exceed a key physical resource such as number of CPU cores.*	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-6A, HY-SR-6B, HY-SR-6C	*Security Recommendation HY-SR-6A (Emulation): Because of the complexity of emulating a hardware device through software, emulation, apart from suffering performance penalties, also increases the size of the TCB especially in situations where the guest OS has native device drivers and the device emulation code runs as a kernel module with the same privilege level as the hypervisor. Hence emulation should only be used where complexity is manageable (e.g., USB host controller).* *Security Recommendation HY-SR-6B (Para-virtualization): In situations where para-virtualized device drivers are used in VMs, mediation of access to physical devices should be enabled by running back-end device drivers (which control the physical device attached*	HY-BF2 (Devices ~~Emulation~~ Mediation & Access Control)

	to the hypervisor host) in a dedicated VM rather than in the hypervisor. This facilitates running the back-end device driver code at a privilege level lower than that of the hypervisor. Additionally, the hypervisor platform should include hardware support in the form of I/O Memory Management Unit (IOMMU) for validating and translating access from the driver domain's underlying hardware device to host memory. The specific IOMMU feature that is mandatory is DMA remapping where the DMA call from a device to guest physical address (GPA) must be translated to host physical address (HPA) and then checked whether the HPA address falls within the protection domain assigned to that device. Combining these mechanisms enables reducing the size of TCB as well as reducing the impact of faulty device or device driver behavior (restricted to device-driver VM as opposed to the hypervisor). *Security Recommendation HY-SR-6C (Passthrough or self-virtualizing hardware devices): For situations, where VMs needs to be given dedicated access to DMA capable devices, the hypervisor platform should include hardware support in the form of I/O Memory Management Unit (IOMMU) for validating and translating all device access to host memory. This recommendation also applies to use of virtualization-enabled hardware devices (based on SR-IOV specification). The specific IOMMU feature that is mandatory is DMA remapping where the DMA call from a device to guest physical address (GPA) must be translated to host physical address (HPA) and then checked whether the HPA address falls within the protection domain assigned to that device.* ~~*All device drivers installed as part of Hypervisor [platform] should be configured to run as lower-privileged level process (user mode) than the privilege level of the hypervisor (kernel mode). If device drivers are run in same privilege level as hypervisor (kernel mode/kernel space), they should be implemented, designed and tested using formal verification to guarantee drivers cannot compromise security of hypervisor execution. (This recommendation applies to any code running at the same privilege level as the hypervisor in the kernel (e.g., VMM)).*~~	
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~~ Mediation & 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 (DOS) attacks. Further, the proper use of resource limits, localizes the impact of a DOS to the VM or the cluster for which the resource limit is defined.*	HY-BF2 (Devices ~~Emulation~~ Mediation & 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 in to and 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*	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 or trusted VM.~~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) despite 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 managed (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).*	HY-BF5 (Management of ~~Hypervisor~~hypervisor Platform)
HY-SR-20	*Protection for Hypervisor Host & Software administration functions should be ensured by allocating a dedicated physical NIC, or if that is not feasible, 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~~hypervisor Platform)

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. Ilie, *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. http://bth.diva-portal.org/smash/record.jsf?pid=diva2%3A834000
6. E. Bugnion, J. Nieh and D. Tsafrir**,** *Hardware and Software Support for Virtualization,*

~~5.~~