DRAFT NIST Special Publication 800-125-A

Security Recommendations for Hypervisor Deployment

Ramaswamy Chandramouli

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

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

DRAFT NIST Special Publication 800-125-A

Security Recommendations for Hypervisor Deployment

Ramaswamy Chandramouli
Computer Security Division
Information Technology Laboratory

October 2014

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie May, Acting Under Secretary of Commerce for Standards and Technology and Acting Director

Authority

This publication has been developed by NIST in accordance with its statutory responsibilities under the Federal Information Security Management Act of 2002 (FISMA), 44 U.S.C. § 3541 *et seq.*, Public Law 107-347. NIST is responsible for developing information security standards and guidelines, including minimum requirements for Federal information systems, but such standards and guidelines shall not apply to national security systems without the express approval of appropriate Federal officials exercising policy authority over such systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), *Securing Agency Information Systems*, as analyzed in Circular A-130, Appendix IV: *Analysis of Key Sections*. Supplemental information is provided in Circular A-130, Appendix III, *Security of Federal Automated Information Resources*.

Nothing in this publication should be taken to contradict the standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other Federal official. This publication may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright in the United States. Attribution would, however, be appreciated by NIST.

National Institute of Standards and Technology Special Publication 800-125-A
Natl. Inst. Stand. Technol. Spec. Publ. 800-125-A 34 Pages (October 2014)
CODEN: NSPUE2
This publication is available free of charge.

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by Federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, Federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. All NIST Computer Security Division publications, other than the ones noted above, are available at http://csrc.nist.gov/publications.

Public comment period: *October 20, 2014 through November 10, 2014*

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in Federal information systems. The Special Publication 800-series reports on ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

Abstract

The Hypervisor is a piece of software that provides abstraction of all physical resources (such as CPU, Memory, Network and Storage) and thus enables multiple computing stacks (consisting of an O/S, Middleware and Application programs) called Virtual Machines (VMs) to be run on a single physical host. In addition it may have the functionality to define a network within the single physical host (called virtual network) to enable communication among the VMs resident on that host as well as with physical and virtual machines outside the host. With all this functionality, the hypervisor has the responsibility to mediate access to physical resources, provide run time isolation among resident VMs and enable a virtual network that provides security-preserving communication flow among the VMs and between the VMs and the external network. To design a hypervisor with the core functionality described above, there are architectural options with each option presenting a different size of Trusted Computing Base (TCB) and hence different degree of ease in providing the required security assurance. Hence in providing security recommendations for the hypervisor, two different approaches have been adopted in this document– one approach based on architectural options that provide ease of security assurance and the second approach based on configuration choices that form part of its core administrative functions such as management of VMs, hypervisor host, hypervisor software and virtual networks.

Keywords

Virtualization; Hypervisor; Virtual Machine; Virtual Network; Secure Configuration**; Security Monitoring; Guest O/S**

Acknowledgements

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

Table of Contents

EXECUTIVE SUMMARY

Server Virtualization is being increasingly adopted in enterprise data centers as it brings about better utilization of hardware resources and reduces power consumption. The entity that provides this server virtualization is a piece of software called Hypervisor or Virtual Machine Monitor (VMM). The hypervisor provides abstraction of all physical resources (such as CPU, Memory, Network and Storage) and thus enables multiple computing stacks (each consisting of an O/S (called Guest O/S), Middleware and a set of Application programs) to be run on a single physical host (referred to virtualized host or hypervisor host in this document). The individual computing stack is called a Virtual Machine (VM).

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 consists of just the established state of art practices for any server-based software and its hosting environment. However, closer examination reveals that in order to develop security recommendations for deployment of hypervisors, the functionality that they deliver (i.e., the virtualization platform) should be analyzed from the following aspects:

- Hypervisor Platform architectural choices – the way various modules are inter-linked and the hardware platform
- Hypervisor Baseline functions – core functions that provide the virtualization functionality

Hypervisor Platform architecture can be classified along several dimensions with each dimension providing at least two different architectural choices:

(a) The entity over which the hypervisor installs – bare metal (directly on hardware) or over a full fledged Operating System (called Host O/S).
(b) Source of Support for functions such as processor and memory virtualization – Hardware or Software, and
(c) Hardware Support for Boot Integrity Assurance – YES or NO

The Hypervisor baseline functions consist of:

(a) Execution Isolation for VMs
(b) Devices Emulation & Access Control
(c) Execution of Privileged operations by Hypervisor for Guest VMs
(d) Management of VMs (also called VM Lifecycle Management)
(e) Administration of Hypervisor Platform and Hypervisor Software

The objective of this document is to examine the security implications of the above aspects (Hypervisor Platform architectural choices and baseline functions) and provide security recommendations for hypervisor deployments in an enterprise. In providing security recommendations for the hypervisor based on these two aspects, we adopt the following two different approaches in this document.

- The security recommendations with respect to hypervisor platform architectural choices merely highlight the ease of providing security assurance (due to size of attack surface, the size of trusted computing base (TCB) and hardware-assisted virtualization functions) in one architectural type compared to another and not with an intention to endorse any particular class of products.
- The security recommendations with respect to baseline functions are in terms of configuration choices, that ensure the secure execution of tasks performed under any of the five hypervisor baseline functions listed above.

The protection requirements for Guest O/Ss and applications running on VMs and associated security recommendations are beyond the scope of this document.

1. INTRODUCTION, SCOPE & TARGET AUDIENCE

The Hypervisor is a piece of software that provides abstraction of all physical resources (such as CPU, Memory, Network and Storage) and thus enables multiple computing stacks (consisting of an O/S, Middleware and Application programs) to be run on a single physical host. Such a physical host is called a Virtualized Host (also referred to as Hypervisor Host in this document) and the individual computing stacks are encapsulated in an artifact called Virtual Machines (VMs). In order to be an independent executable entity, the definition of a VM should include resources such as CPU, Memory etc allocated to it. The VMs are also called "Guests" and the operating system (O/S) running inside each of them as "Guest O/S". The resources associated with a VM are virtual resources as opposed to physical resources associated with a physical host.

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

It might appear from the above introduction that hypervisor is just another example of server-based software and hence all protection measures and security recommendations for it should mirror the state of art practices applicable for that class of software and its hosting environment. However, closer examination reveals that in order to develop security recommendations for deployment of hypervisors, the functionality that they deliver (i.e., the virtualization platform) should be analyzed from the following aspects:

- Hypervisor Platform architectural choices – the way various modules are inter-linked and the hardware platform
- Hypervisor Baseline functions – core functions that provide the virtualization functionality

1.1 Hypervisor Platform Architectural Choices

We define a Hypervisor Platform as one that consists of the following entities:

- The various software modules that provide virtualization of all hardware resources
- The hardware of the physical host (virtualized host)

The architecture of a hypervisor platform has several dimensions with at least two choices along each dimension. The dimensions and a brief description of choices within each dimension follows:

(a) The entity over which the hypervisor installs – bare metal (directly on hardware) or over a full fledged Operating System (called Host O/S).
(b) Source of Support for Virtualization functions – Hardware or Software
(c) Hardware Support for Boot Integrity Assurance – YES or NO

The entity over which the hypervisor installs: Some Hypervisors are installed directly on the hardware (or bare metal) while others require an underlying O/S (called Host O/S). The first type of hypervisors is called Type 1 while the second type is called Type 2.

Source of Support for Virtualization: In the context of this document, we consider only hypervisors that provide virtualization of the x86 and AMD hardware architecture supporting either 32-bit or 64-bit instructions. Within this x86 and AMD hardware offerings, design choices include those that provide hardware support for instruction set virtualization and memory virtualization and those that do not.

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

1.2 Hypervisor Baseline Functions

While the basic function of a Hypervisor is to virtualize a physical host to enable running of multiple virtual hosts (or Virtual Machines (VM)), commercial hypervisor offerings come with differing feature sets. Also the modules that provide the constitutent features are given different names in different product offerings. Hence, for the purpose of accomplishing the goals of this document, it is necessary to identify a set of baseline features of a hypervisor that provides all the required functionality of a virtualized host. The required functionality is also called Virtual Machine Manager (VMM) functionality. These baseline features or functions are:

- HY-BF1: Execution Isolation for Virtual Machines – Scheduling of VMs for execution, Management of the application processes running in VMs such as CPU and Memory Management, context switching between various processor states during the running of applications in VMs etc.
- HY-BF2: Devices Emulation & Access Control – Emulating all Network and Storage (block) devices that different native drivers in VMs are expecting, mediating access to physical devices by different VMs
- HY-BF3: Execution of Privileged Operations for Guest VMs – Certain operations invoked by Guest O/Ss, instead of being executed directly by the host hardware, may have to be executed on its behalf by the hypervisor, because of their privileged nature.
- HY-BF4: Management of VMs – Setting configuration parameters for VMs (VM Images) and control of VM states (Start, Pause, Stop etc).
- HY-BF5: Administration of Hypervisor Platform and Hypervisor Software – This involves setting of parameters for user interactions with the hypervisor host as well as hypervisor software and configuration of Virtual Network inside the hypervisor.

The brief description of the five baseline functions listed above is suffice 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.

Hypervisor offerings differ in the distribution of the above functions and in the name assigned to the corresponding module. In general, functions HY-BF1 and HY-BF3 are offered by modules running in a kernel collectively called "Hypervisor" while HY-BF2 is enabled using a module called QEMU that runs outside the hypervisor, usually in a dedicated privileged VM. The QEMU module spawns one process for each running VM. The functions HY-BF4 and HY-BF5 are performed by a module called management or service console. Just like QEMU module, this 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 O/S installed inside in it or with a ultra-light O/S used to present an API (shell and network access) with

utility functions that facilitates performing just the hypervisor-specific configuration and administrative tasks.

For the purpose of this document, we will assume the following baseline architecture (with respect to distribution of software functions) and a common terminology for the modules that provide one or more the baseline functions discussed above. The assumption of this architecture does not in any way affect the protection requirements for the module providing the baseline functionality and the associated security recommendation. In our baseline architecture, functionality HY-BF1 is provided by kernel module called the Virtual Machine Manager (VMM). Functions HY-BF2 and HY-BF3 are encapsulated within a privileged VM called the "Management VM", with the former function provided by a process called "QEMU" while the latter function provided by a "Management Daemon".

1.3 Scope of this document

The scope of security recommendations in this document covers the following entities. They are:

- A Reference Hypervisor Platform arrived at by selecting a choice from two of the hypervisor architectural dimensions.
- Hardware/Software components that provide all the hypervisor baseline functions.

Reference Hypervisor Platform: Based on the architectural choices in various dimensions along which a hypervisor platform can be classified, the Reference Hypervisor Platform that has been chosen as the basis for security recommendation is the following:
Type 1 (Bare Metal) Hypervisor resident on a Hardware that provides hardware support for both Instruction Set Virtualization and Memory Virtualization. It is assumed that all VMs that are going to be run on the hypervisor platform are fully virtualized Guests (Guest O/Ss are unmodified versions) host. An additional assumption is that device drivers run in Guest VMs are native to the Guest O/S version (i.e., no para-virtualized device drivers)

The reference hypervisor platform does not include a hardware base (consisting of a processor and chipset) that provides boot integrity assurance as such a type of hardware base is not widely available.

Hardware/Software Components that provide hypervisor baseline functions: All components involved in providing the five baseline functions HY-BF1 through HY-BF5.

The security recommendations do not cover the following aspects:

- Security of Guest O/Ss running on VMs
- Security of Applications/Services running on VMs

1.4 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.

1.5 Relationship to other NIST Guidance Documents

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

Accompanying this technology adoption trend is the increase in feature set of one of the core layer of virtualization – the hypervisor, as well as the market availability of the set of tools used for configuration and administration of the virtualized infrastructure spawned by the hypervisor. The objective of this document is to focus on the development of a set of security recommendations for deployment of the hypervisor and the related components of virtualization infrastructure (such as VM images, virtual network etc) that can built around the hypervisor. The distinguishing features of the set of security recommendations provided in this document in the context of similar NIST Guidance documents are given below:

- Focussed set of security recommendation for the deployment of the foundational component of the Server Virtualization technology – i.e., the Hypervisor
- Security recommendations for all components in a real-world virtualization infrastructure built around the Hypervisor deployment such as VM Images and Virtual Network.
- Recognizing that Hypervisor being a piece of server software, all the generic security considerations have been included as part of the recommendations. These include: (a) Keeping Patches upto date, protection of the execution instance using host-based Firewalls and IDPS, Analysis of Logs etc.
- Recognizing that the Hypervisor is nothing but a OS kernel and that the security of a server system configured with any OS distribution depends upon some weakest links such as the device driver software, security recommendations relating to these components have been provided as well.
- Recognizing that the hypervisor has to perform certain privileged operations without interference from any other entity in the virtualized host and that leveraging hardware support for these operations will make a big difference to the overall security of hypervisor deployment.
- Last but not the least, all security recommendations are traceable to one or more the hypervisor's baseline functionality, the secure execution of whose constitutent tasks, the security recommendations in this document are intended to provide protection.

2. APPROACH FOR DEVELOPING SECURITY RECOMMENDATIONS

Developing security recommendations for deployment and usage of a complex software such as the hypervisor requires knowledge of potential threats that when exploited would affect the basic three security properties of confidentiality, integrity and availability of hypervisor functions. There are two ways to obtain information regarding potential threats. The requisite protection measures that will counter the threats and hence form the basis for security recommendations can be achieved in the following ways:

- Choosing a particular hypervisor platform from among the architectural choices available
- Choosing configuration parameters for hypervisor components/modules that provide the baseline functions HY-BF1 through HY-BF5.

For each of the above two ways for choosing protection measures (and hence developing security recommendations), we adopt different approaches as follows:

(a) For choosing a hypervisor platform, we use some accepted security practices/metrics such as size of the code, attack surface based on exposed interfaces etc.

(b) For developing configuration parameters for hypervisor components/modules, we adopt an approach that consists of the following steps:
    - Identify the common interfaces exposed by a typical hypervisor implementation and the threat sources that could use those interfaces. We also provide selected examples of threats from those threat sources (Section 2.1)
    - For each of the five baseline functions HY-BF1 through HY-BF5, 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 (either an architectural choice or configuration setting) that will provide assurance against exploitation of these threats form the basis for security recommendations (Section 2.2).

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

2.1 Hypervisor Platform – Common Interfaces & Threat Sources

The hypervisor software is resident on a physical host that is connected to the enterprise network. It has the capability to be remotely administered. At the same time, it is supporting multiple virtual hosts (or virtual machines or VMs) that are nodes of a software-defined virtual network inside that physical host. Based on these features, one can identify three basic external interfaces for a hypervisor. They are:

- HY-CI1: One for hardware resources that are either an integral part of the physical host or connected to it (e.g., iSCSI drives)
- HY-CI2:One for Guest Systems (Virtual Machines) and
- HY-CI3:Management Console OS or interface

The threat sources leveraging these interfaces are the following:

- HY-TS1: Threats from and through the network in which the hypervisor host (virtualized host) resides
- HY-TS2:Threats emanating from rogue or compromised Virtual Machines (VMs) through channels such as shared hypervisor memory and the virtual network inside the hypervisor host
- HY-TS3: Threats from web interfaces to management consoles – used for VM Management as well as Hypervisor Administration

Threats from sources HY-TS1 and HY-TS3 are common to all server-based software and are well known. Threat from source HY-TS2 is unique to the virtualization environment defined by the hypervisor. We look at selected examples of the types of threats from this threat source (consisting of VMs and the virtual network residing inside the hypervisor host) in the next subsection.

2.1.1 *Selected Examples of Threats from VMs & Virtual Network*

The hypervisor controls VM access to physical hardware resources as well as provides isolation among VMs. VM access to hardware resources such as CPU and memory are directly controlled by the hypervisor while access to resources such as physical NICs and Storage drives are controlled through modules (drivers) that are resident in the kernel module or are in a privileged VM (i.e., Management VM). The isolation among VMs is provided by assigning a unique IP or MAC address to each VM, by defining virtual local area networks (VLANs) and by assigning each VM to a specific VLAN. The threats that could subvert the secure execution of these functions are:

VM Escape (HYP-T1): The first threat to any hyervisor is from rogue VMs. The rogue VMs are the ones which manage to subvert the access control function provided by the VMM/Hypervisor to hardware resources such as memory and storage. The possible reasons for this threat are: (a) misconfiguration of the hypervisor and/or guest VM container (b) malicious or vulnerable device drivers. Potential downstream impact of a rogue VM taking control of the hypervisor is: (a) installing rootkits and (b) attack on another VM on the same virtualized host.

Breaking the Isolation (HYP-T2): Potential threats to isolation are from attacks such as: (a) IP or MAC address spoofing by a rogue VM (b) VLAN hopping – a rogue VM escaping the boundaries of its VLAN and (c) Traffic Snooping – intercepting virtual network traffic, intended for VM on the same virtual network segment.

Resouce Starvation by rogue VMs (HYP-T3): Mis-configured or malicious VMs, may be consuming a disproportionately high percentage of host resources resulting in other VMs being denied (starved of) service.

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

The above threats are by no means exhaustive. However, they cover all areas of threat sources from inside the hypervisor host though not all types of threats.

2.2 Potential Threats related to Hypervisor Baseline Functions

We now proceed to look at the tasks in each of the five hypervisor baseline functions (HY-BF1 through HY-BF5) and how the threats to the secure execution of those tasks are channeled through these threat sources and common interfaces identified above.

2.2.1 *Potential threats on HY-BF1 function (Execution Isolation for Virtual Machines)*

The main tasks for the hypervisor under this function are*:*

- Scheduling of individual VM's tasks (i.e., vCPU tasks since each guest VM is allocated a set of virtual CPUs (vCPUs)) by handling register states appropriately. To enable saving and loading of the state of each vCPU, the hypervisor uses a data structure called Virtual Machine Control Structure (VMCS). Faulty implementation of this data structure has been known to cause hypervisor memory leaks.
- Hypervisor emulates interrupt and timer mechanisms that the motherboard normally provides to a physical machine. Faulty implementation of interrupt and timer related data structure has been known to cause denial of service attacks.

- The hypervisor has to run a software-based Memory Management Unit (MMU) in the form of a shadow page table for VM, since guest VMs cannot be granted direct access to the hardware-based MMU, as that would allow them to access memory belonging to the hypervisor and other co-hosted VMs. However faulty implementation of software-based MMU could lead to disclosure of data in arbitrary address spaces such as memory segments belonging to the hypervisor and co-located VMs.

Thus we see that attacks to secure execution of tasks under HY-BF1 function are to be addressed in hypervisor code and no security protection measures can be applied at the deployment and usage stage. However, security assurance for VM memory management can be obtained by using a hypervisor platform that provides virtualization-aware hardware memory management unit (also called a platform which provides support for hardware-assisted memory virtualization). This approach is realized through our security recommendation HY-SR-2.

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

The main task under this function is the emulation of storage and networking devices. This task is handled by a module called QEMU. Any I/O call from a guest VM application is intercepted by the hypervisor kernel and forwarded to QEMU for handling. This is due to the fact that guest VMs (through their native device drivers) cannot access the physical hardware device directly, given that the role of the hypervisor is to mediate user access to shared resources. Therefore the native drivers in VMs (also called front-end drivers) communicate with back-end drivers (in QEMU). The QEMU can emulate a number of devices, mediates access to devices (by enforcing access policies) and multiplex the actual devices (since it has full access to the underlying physical device).

The potential threat to the secure execution of this function comes from (besides faulty implementation of in-memory data structures for virtual devices) faulty device driver code. The security recommendation relating to this is spelt out in HY-SR-5 under section 4.

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

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

2.2.4 *Potential threats on HY-BF4 function (Management of VMs)*

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

- Setting up configuration parameters for a VM Image – these encompass virtual CPU topologies (e.g, number of vCPU cores) and priorities, memory quotas, assigned virtual devices etc.
- Altering the state of VMs – Start, Pause, Stop etc.

The management operations on VMs are performed directly from the hypervisor (if it is a monolithic software module) or from a Management VM (also called Parent Partition). The security protection measures and recommendations for protecting the secure execution of the tasks under this function are addressed in Section 5 of this document.

2.2.5 *Potential threats on HY-BF5 function (Administration of Hypervisor Host & Hypervisor Software)*

The tasks under this function relate to overall administration of hypervisor host (virtualized host) and the hypervisor software and are usually performed through user-friendly web interfaces or network-facing virtual consoles. The attacks on the secure execution of these tasks are nothing but those that pertain to any remote administration console. The set of security recommendations pertaining to these tasks are addressed in Section 6 of this document.

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

Remote administration of servers (hosts - whether virtualized or not) is enabled nowadays through installation of a program in the memory hardware called the BIOS rootkit. This program starts up, runs and resides in system memory and hence can survive other conventional attempts to get rid of malware, including reformatting or replacing the hard drive or even reflashing (updating) of the BIOS.

The security threat to BIOS rootkit comes from an attacker gaining administrative-level privileges to the host, and then flashing the BIOS over the Internet with malware-laden firmware. Although this threat is not exclusive to virtualized hosts, the impact of this attack is severe since a single virtualized host has multiple VMs running on it and in some instances comprising all tiers of an application such as Front-end (such as Web Server), Application Logic and Data Management software.

3. SECURITY RECOMMENDATIONS BASED ON HYPERVISOR PLATFORM ARCHITECTURAL CHOICES

By choosing a reference hypervisor platform, we have already eliminated the choices along two dimensions of its design. Nevertheless, we provide a justification for making those choices and state the resulting security recommendation formally. We also provide a security recommendation based on the third dimension – i.e., hypervisor hardware that supports an attestation scheme that provides assurance of boot integrity. The recommendations under this category merely highlight the ease of providing security assurance (due to size of attack surface, the size of trusted computing base (TCB) and hardware-based security enforcement etc) in one architectural choice compared to another and are not meant as endorsements for any particular class of hypervisor products.

3.1 Architectural Choices based on the Entity over which Hypervisor is installed

There are hypervisor software types that can be installed directly on the hardware or bare metal and those that can only be installed on an O/S base (called Host O/S). The former are called Type 1 hypervisor while the latter are called Type 2 hypervisors. From a security viewpoint, a Type 1 hypervisor is preferred due to absence of the Host O/S software stack and its associated vulnerabilities and the extra attack surface it may bring. Hence the security recommendation with respect to this mode of architectural choice is:

Security Recommendation - HY-SR-1: A Type 1 hypervisor provides more security assurance than a Type 2 hypervisor, due to the reduced attack surface (given the absence of Host O/S) and the consequent reduced list of vulnerabilities to be addressed.

3.2 Architectural Choices based on Source of Support for Virtualization (Hardware or Software)

There are hypervisor versions for installing on hardware platforms that provide hardware assisted virtualization capabilities as well as ones that can be installed on platforms that provide no virtualization features. In order to explain the rationale for these choices, a brief description of each of these forms of virtualization is in order:

(a) Instruction set Virtualization: Processor architectures that support Instruction set virtualization provide two modes of operation (root mode and non-root mode) each with 4 hierarchical privilege levels (Level 0 being the highest and Level 3 being the lowest). Further, among the two modes, the root mode has a higher privilege for executing CPU instructions than non-root mode. By running the hypervisor in root mode and VM's (Guest's) OS in non-root mode (at privilege or ring level 0), the hypervisor is guaranteed to be safe from any attacks by any of the Guest OSs. Also with the hypervisor not having to perform additional functions such as trap-and-emulate virtualiztion or binary translation, the code executing with privileges is reduced in the hypervisor making the TCB smaller enabling better assurance verification.

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

The security advantages of hardware-assisted virtualization platforms are:

- One of the potential security vulnerabilities for hypervisors is the buffer flow attacks from VMs resident on the Virtualized Host platform. The hardware support for memory management such as Extended Page Table etc. that comes as part of the hardware-assisted virtualization can be leveraged to prevent code execution from memory locations reserved for data storage thus preventing buffer flow attacks.
- Hardware extensions for Virtualization provide two modes of execution. Host (root) mode and Guest (non-root) mode. The host mode runs at a higher privilege than guest mode. The hypervisor code (providing the baseline functionality HY-BF1 (processor allocation + memory management) runs in host mode while guest O/Ss and applications in VMs run in guest mode. Hence any exploit code in guest O/S cannot subvert the controls provided by hypervisor code.
- One of the common threats in virtualization platforms is a malicious VM accessing areas of memory belonging to other VMs. This is called a VM Escape attack. Processors with virtualization extensions provide safety against this through features such as Direct Memory Access (DMA) remapping limiting DMA access to what is valid for the VM (i.e., preventing a VM from performing DMA beyond its allocated area).
- Also the advantage of a hardware providing assistance for both forms of virtualization is that, the emulation module of the hypervisor can present the true hardware architecture of the physical host instead of modified hardware architecture. The consequence of this feature is that unmodified Guest OSs (along with their native device drivers) can be run in VMs. The security implication of this enabling feature is that a lot more CVE data is available for these OSs and device drivers with the resultant patch management becoming faster (increasing availability) and robust.

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

- ***Better memory management controls can prevent attacks such as buffer overflow.***
- ***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 O/Ss can be run enabling easier patching/updating than having to perform the same operations on modified/ported versions of O/Ss 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.***

Recommendations HY-SR-1 & HY-SR-2 together forms the Reference Hypervisor Platform that forms the basis for security recommendation in this document. A schematic diagram of the building blocks of this architecture is given below in Figure 1:

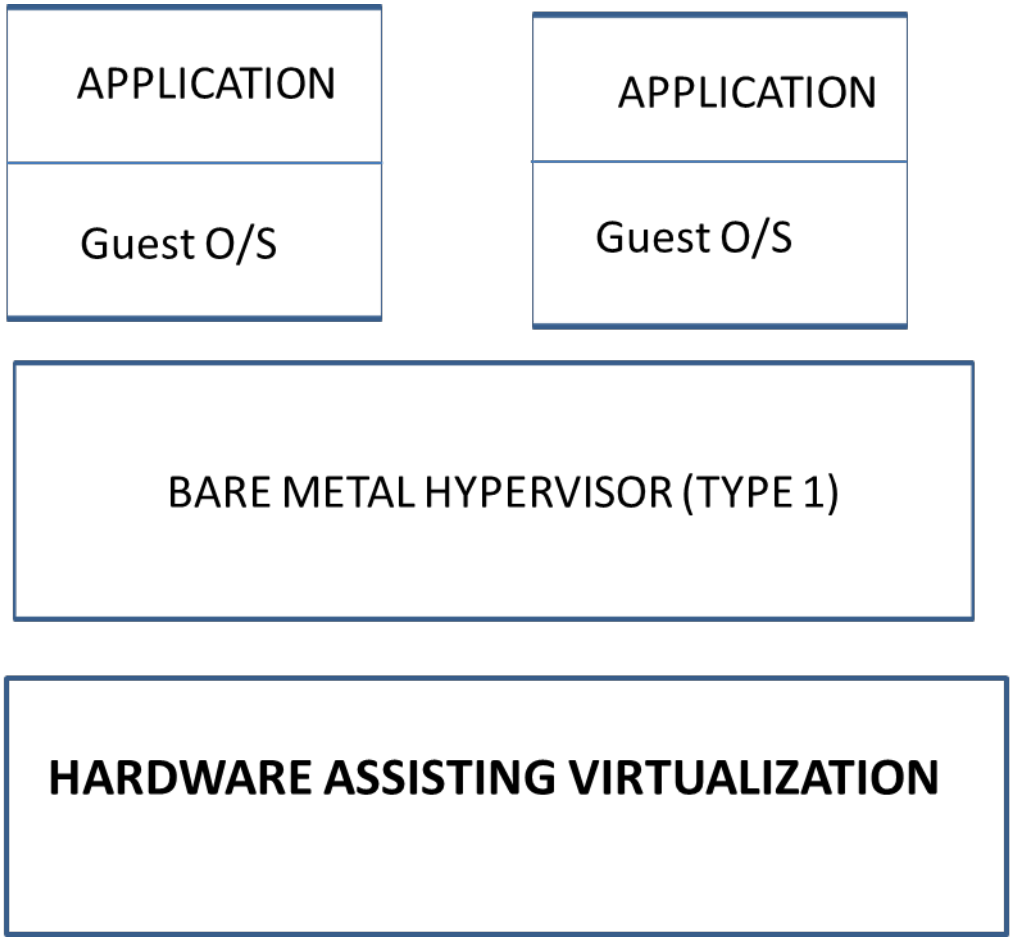

Fig 1: Reference Hypervisor Platform

3.3 Architectural Choices based on Hardware Support for Boot Integrity Assurance

Security assurance in the hypervisor can be obtained by having a hypervisor platform infrastructure that supports boot integrity measurements and attestation process. The boot integrity measurement and attestation process consists of the following:

- The hardware hosting the hypervisor is established as a root of trust and a trust chain is established from the hardware, through the BIOS and to all hypervisor components.
- For the hardware (consisting of the processor and chipset) to be established as the root of trust, it should have a hardware-based module that supports a measured launch environment (MLE) called the Root of Trust for Measurement (RTM). The outcome of launching a hypervisor in a MLE-supporting hardware is that you have a measured launch of the firware, BIOS and all the hypervisor components , thus forming a trusted chain from the hardware to the hypervisor.
- The hypervisor offering must be able to take advantage of the MLE feature. In other words, the hypervisor should be able to invoke the secure launch process, which is usually done by integrating a pre-kernel module (since the kernel is the first module installed in a hypervisor boot up) into the hypervisor's code base. The purpose of this pre-kernel module is to ensure selection of the right authenticated module in the hardware that performs orderly evaluation (measurement) of the launch components of the hypervisor (or for that matter any software launched on that hardware). The most common mechanism to enable the hypervisor to take advantage of MLE feature of the hardware is the Tboot.
- All hypervisor components that form part of the Trusted Computing Base (TCB) must be included under the scope of the Tboot mechanism so that they get measured as part of their launch process. The measured elements (components) should include at the minimum the following: *the core kernel, kernel support modules, device drivers and the hypervisor's native management applications (for VM Management and Hypervisor Host administration).*
- The various measurements (which are nothing but the identities of the components) that are taken as part of the hypervisor launch process should be stored in a standardized Trusted Platform Module (TPM) that conforms to TCG PC specification [TCG PC]. The TPM essentially contains the registers (called Platform Configuration Registers or PCRs) for storing the various measurements with its architecture consisting of two main components: Root of Trust for Storage (RTS) and Root of Trust for Reporting (RTR). The TPM must be an integral part of physical server hardware, which in many virtualization infrastructures consists of x86 architecture-based servers.

The following is an optional security recommendation if a hardware platform that provides support for virtualization has also features for Measured Launch Environment (MLE) (including TPM) and the hypervisor components launched on the platform has features for enabling support for MLE.

Security Recommendation HY-SR-3 (optional): The hypervisor that is launched should be part of a platform and an overall infrastructure that contains: (a) Hardware that supports a MLE and standards-based TPM and (b) Attestation process that should contain capabilities to take advantage of these so as to provide a chain of trust starting from the Hardware to all Hypervisor components. The chain of trust provides assurance that all launched components (starting from BIOS, hypervisor and device drivers) have not been tampered with and that their versions are correct (i.e., overall boot integrity).

There are situations where security provided through software in operating system, application and service layers could be circumvented. In this situation, providing security by creating a root of trust at the hardware layer is an option. In this approach, the launch of hypervisor components is cryptographically authenticated. This authentication verifies that only authorized code runs on the system. It thus provides assurance that the code of the hypervisor components has not been tampered with. In this approach, we establish trust in the Hypervisor's software components based on trusted hardware. In other ways, we establish a chain of trust from hardware to hypervisor's software components with the initial component called the root of trust. The justification for using hardware as the root of trust is due to the fact that for most attackers, the risk, cost and difficulty of tampering with hardware exceeds the potential benefits of

attempting to do so. Although it is possible to extend the chain of trust beyond the hypervisor to the VMs and the applications running inside them, in this guidance, we are restricting ourselves to authentication of hypervisor's launched components. Thus the objective of this exercise is therefore Hypervisor Platform Boot Integrity protection and the entities in the chain of trust therefore includes (starting from hardware) the firware, BIOS and the hypervisor components.Specifically, in the context of the hypervisor, the assurance of integrity that is provided is against tampering and low-level targeted attacks such as root kits. If the assertion of integrity is deferred to a trusted third party that fulfills the role of trust authority, the verification process is known as trust attestation.

The design of Trusted Platform Module (TPM) has been standardized by the Trusted Computing Group (TCG). To assert the integrity of pre-launched and launched components (which in our context is made up of firmware, BIOS and Hypervisor), the following steps are needed:

- A Measured boot process
- Provide assurance that the executed components are trusted components using a process called Attestation

The TPM supports the above steps using the following system elements:

- Root of Trust for Reporting (RTR)
- Root of Trust for Storage (RTS)

The function of RTM is to make integrity measurements (generally a cryptographic hash) and send them to RTS. RTS then holds the components identities (measurements) and other sensitive information. In fact, the RTM measures the next piece of code following in the boot sequence.

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 Platform Control Register (PCR) in the TPM before the control is transferred to the next program in the sequence. Since each component in the sequence in turn measures the next before handing of control, there is a chain of trust established. If the measurement chain continues through the entire boot sequence, the resultant PCR values reflect the measurement of all files used.

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

3.4 Architectural Choices based on footprint of the Management Console

A common feature in many hypervisor offerings is that the management console is not built into the core hypervisor module but runs on top of it as a privileged virtual machine (VM). The difference between this VM and other VMs that run applications is that the management console is tightly integrated with the hypervisor kernel and has a direct interface to communicate with it and it is also booted up along with the hypervisor executable. Thus when the client module of a hypervisor is connecting to the hypervisor it is in fact connecting to the management console and not to the hypervisor kernel module. There are two architectural choices for the design of management console.

- The management console may be built with a full fledged operating system inside it. The advantage of this design is that all operating system commands (plus any additional comments that the manufacturer has added for managing the hypervisor kernel) can be used when an administrator is logged into the management console. From an administrative viewpoint, the

management console can thus be used to perform all hypervisor configuration tasks. The downside is that the footprint in terms of the code size and the storage partitions required are relatively high.

- The management console could be built with some barebones features (with a Shell, API and some support for utility commands). Obviously such a management console cannot be as powerful as one that has a full fledged OS, but still often comes with enough features for management (to perform configuration and administrative tasks) as well as perform some troubleshooting operations as well. But the upside is that this type of management console has a smaller code and disk footprint.

In addition to the foot print, a factor that could increase the risk of attack is the number of exposed interfaces in the management console.

Security Recommendation HY-SR-4: A functional Hypervisor management console with smaller code and disk footprint and smaller number of exposed interfaces can provide better security assurance as it facilitates easier verification and also because of the fact that it presents a smaller attack surface.

4. SECURITY RECOMMENDATIONS FOR DEVICE EMULATION & ACCESS CONTROL (HY-BF2)

Device Drivers are the weakest links in any server security model and the hypervisor is no different in that respect. In order that device drivers do not become the source of security compromise, the following approaches have been found to be deployed:

- Certain hypervisor configuration features include the capability to specify acceptable sources for drivers. For example, the hypervisor could be configured to accept drivers only from sources who certify their drivers (or have certifications obtained from third-party assessors).
- Another approach taken is to run QEMU (device emulation code) process for each application VM in another unprivileged VM, so that the impact of a faulty device driver code does not affect the secure operation of other application VMs (if architecture permits).

Security Recommendation HY-SR-5: The hypervisor should have a boot configuration choice to disallow the user of non-certified drivers. Further, if architecture permits, the running of QEMU process for each application VM should be confined to an unprivileged VM so as to limit the impact of a faulty device driver code to the operation of the corresponding application VM.

5. SECURITY RECOMMENDATIONS FOR VM MANAGEMENT (HY-BF4)

5.1 VM Memory Allocation Scheduling Options

The hypervisor's memory scheduler has the responsibility to meet the memory requirements for all workloads running in all VMs at all times. A typical hypervisor meets this requirement (just like an OS) by using a combination of physical RAM and swap files (called hypervisor kernel swap files). Further a typical VM does not always require the entire memory it has been configured for at all times. Because of the above two reasons, it is a viable overall virtualization configuration decision to have the combined configured memory of all VMs running on a virtualized host to exceed the total physical RAM memory of the host. However, the ratio of the total configured memory of VMs to host physical RAM should not be too high as it may result not only in performance degradation but also availability of the virtualized host for certain VM workloads as well.

Another factor affecting the availability of the virtualized host/hypervisor for certain workloads for an individual 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 RAM for each VM. Also in order to avoid a situation where a particular VM makes use of the physical RAM for its entire configured memory, there should be a feature to specify a limit on the guaranteed physical RAM as well. Further, there may be certain workloads that are time sensitive and hence the VMs hosting them should have some priority in getting the required memory resources compared to other running VMs. Hence a configuration option to specify a priority value for each VM as well.

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

Security Recommendation HY-SR-6: The ratio of the combined configured memory of all VMs to the RAM memory of the virtualized host should not be very high. A typical ratio adopted is 1.5 : 1. In other words if a virtualized host has 64GB of RAM, then the combined configured memory of all VMs running on it should not exceed 96 GB.

Security Recommendation HY-SR-7: The hypervisor should have configuration options available to specify a guaranteed physical RAM for every VM (that requires it) along with a limit to this valu, and to specify a priority value for obtaining the required RAM resource in situations of contention among multiple VMs.

5.2 VM CPU Allocation Options

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

Security Recommendation HY-SR-8: The number of virtual CPUs allocated to any VM deployed should be strictly less than the total number of cores in the hypervisor host.

Security Recommendation HY-SR-9: 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.

5.3 VM Image Management

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

- Documentation on the Golden Image for each VM Image type. A Gold Image is defined by a set of configuration variables associated with the VM Image. The configuration variables should include among other things the Guest O/S make, version and patch level.
- Each VM Image in the VM Image Library must have associated with it a digital signature
- Access privileges to VM Image Library must be controlled through a robust access control mechanism
- VM Images must be stored in a storage location logically separate from the storage location where inactive or dormant VMs are stored

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

Security Recommendation HY-SR-10: The VM image library should reside outside of the hypervisor host, should have strict access control and each of the images checked out from the library should have a digital signature attached to it as a mark of authenticity and integrity.

- A Master White List of VMs in various statuses are required – Active, Suspended, Inactive/Dormant etc
- Periodical scan of Active VMs to ensure that they are operating safely. This is different from the monitoring of inter-VM traffic and it involves just the scanning of open ports and the memory stack.

5.4 VM Monitoring and Security Policy Enforcement

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

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

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

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

(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 into and from the VM. The main disadvantage of this approach is that if the VM is migrated to some other physical host, the dedicated appliance has to be migrated together.

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

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

In an hypervisor-based security monitoring solution, the security tool that performs the function of monitoring and protecting the VMs (let us call them as User VMs) is run outside of those VMs in a specially security-hardened VM. A security tool designed and configured to run in this mode is called Security Virtual Appliance (SVA). The SVA obtains its visibility into the state of a VM (including CPU state, registers, state of memory and I/O devices) as well as the network traffic between VMs and between VMs and the hypervisor through the *virtual machine introspection* API of the hypervisor. This is the most preferable solution since:

(a) Not vulnerable to a flaw in the Guest O/S
(b) Is independent of the Virtual Network Configuration and does not have to be reconfigured every time the virtual network configuration changes due to migration of VMs or change in connectivity among VMs resident on the hypervisor host.

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

Security Recommendation HY-SR-11: 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 VM. This monitoring and enforcement mechanism forms the foundation for building Anti-Virus (AV) and Intrusion Detection & Prevention (IDPS) solutions.

Security Recommendation HY-SR-12: 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 Security Virtual Appliance (SVA) in a security hardened (trusted) VM.

5.5 Granular Administrative Privileges for VM Management

Having the capability to assign granular administrative permission for the virtualized infrastructure enables setting up of different administrative models and associated delegations. To see the need for granular permissions, it would be helpful to look at some use case scenarios for some 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 be soon going for production. In this situation, it may be useful to one or more administrators assigned exclusively for Quality Assurance group to be given administrative permissions on specific virtual machines set up for the testing purpose.
- VM Administration Use Case 2: A capacity planner assigned the task of determining the various operating loads on the various virtualized servers and determining the need for additional for virtualized hosts may need permission for just viewing the list of virtual machines in each of the virtualized hosts but not permissions to perform any administrative operations on each of those VMs. In this situation, it is desirable to have the Capability to view the list of VMs in a virtualized host but deny the user the capability to interact with any of the visible objects
- VM Administration Use Case 3: In virtualized data centers, where VMs of different sensitivity levels are run on the same virtualized host, sometimes an administrator who is given administrative privileges at the hypervisor level should be prevented any access to a specific VM because of the sensitive nature of the workload (set of applications) running in that VM. The desired capability in this scenario is to negate a permission (obtained through inheritance) for a specific child object
- VM Administration Use Case 4: VM Administration Use Case 1 dealt with a scenario where you need assign permissions for a group of administrators controlling a set of VMs for a particular organizational division or department. A corollary to this type of administrative entity is the need to a class of administrators wanting to administer VMs running a particular type of work load (e.g., webserver) irrespective of its location within the organizational structure. This class of administrators may not require the entire set of administrative functions on a VM but some arbitrary set of management functions such as: Configure CD Media, Configure floppy media, Console interaction, Device Connection, Power On, Power Off, Reset, Suspend etc. This scenario calls for having the capability to create "Custom roles" that can contain arbitrary set of permissions relating to a VM as well as ability to create a "Custom Object" that contains an arbitrary set of VMs carrying a particular type of workload (e.g., webserver).

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

Security Recommendation HY-SR-13: 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, another desired capability for the access control solution is the ability to specify 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

6. SECURITY RECOMMENDATIONS FOR ADMINISTRATION OF HYPERVISOR HOST & HYPERVISOR SOFTWARE (HY-BF5)

The security of any software has to be ensured through robust administrative practices on it and its resident host. The Hypervisor is an example of a complex host-based software and hence proper administration of it using its configurable parameters as well as the hypervisor host (called virtualized host) is critical for the overall security of its intended functions. While the composition of the configuration parameters depends upon the design features of a hypervisor offering, the latitude in choosing the values for the individual available parameter results in different configuration options. While many configuration options affect the overall performance and fault tolerance capabilities of the hypervisor platform, there are some options that have direct impact on the security goals of confidentiality, integrity and availability and it is those configuration options that are discussed in this document.

6.1 Restricting local user Accounts on the Hypervisor Host

All commercial hypervisor offerings come with a Management server almost eliminating the need for local users and groups on each hypervisor host. However there are situations where tasks cannot be accomplished through the management server [18]. These situations are:

- The Management Server is down or the host housing the Management Server has been taken down for maintenance
- There is need to troubleshoot the hypervisor boot and configuration problems
- There is need to patch the hypervisor host O/S

In spite of the need for local privileged users and groups, the number of local privileged users and groups on the hypervisor host should be kept to the minimum (usually 2 or 3 is a good number - to account for the fact that one of the user accounts may be unavailable due to the absence or unreachability of one of the administrators). No end user accounts should exist on the hypervisor host.The security recommendation related to this practice is as follows:

Security Recommendation HY-SR-14: The number of user accounts (including privileged accounts) requiring direct access to hypervisor host should be limited to bare minimum (i.e., two or three).

6.2 Secure User Management & Authentication on each hypervisor host

There are generally two approaches available for management of users and groups for a hypervisor host. They are:

- Manage the users and groups associated with a hypervisor locally
- Manage the users and groups by integrating with a local directory infrastructure (e.g., Microsoft Active Directory) and using a directory-based authentication mechanism (e.g., Kerberos)

(a) Managing the users and groups associated with a hypervisor locally: In this approach the user accounts on the hypervisor host can be created using either:

- Service Console - for those hypervisors that come with one - specifically using the command line tools in the Service Console (or)
- Dedicated client interface

The module for creating user accounts through a service console or a dedicated client also includes the feature to set a password for the created account and also setting up basic access mode permissions such as shell access.

There are security issues associated with managing users and groups locally through the manual process outlined above. They are:

- When an administrative user having user accounts on some hypervisor host, quits the organization or moves over to a different division within the company, the user account associated with him/her has to be manually deleted in all hypervisor hosts. If it is not done properly, it has the potential to leave zombie accounts which can be exploited resulting in a security breach of the hypervisor host.
- Any changes to organization policy such as the password policy has to be enforced manually on each hypervisor leaving room for some mismatches in some hypervisor hosts.

(b) Managing the users and groups through a Directory Infrastructure: In this approach, the users and groups are still created through either the service console command line interface or through a dedicated hypervisor client but there are two differences:

- the administrative user account names created here are such that they match with those already present in the enterprise directory (though in some systems, it will be possible to simply point to the users and groups list in the directory thus avoiding duplication)
- No passwords are assigned while creating these user accounts
- A suitable command to modify the configuration to specify that the authentication will take place through a mechanism appropriate for the directory infrastructure (e.g., through the domain controller in an Active Directory infrastructure using an authentication mechanism such as Kerberos).

The advantage of this approach of managing the users of the hypervisor host through the directory infrastructure are as follows:

- User account changes (such as deletion) can be done centrally at the directory level. This way, any user account for a user no longer with the organization, though still present in the hypervisor host, cannot be used for logging in, since password and any other form of authentication has to be done at the directory infrastructure level and the latter will fail since such an user account no longer exists there.
- Password policies such as complexity, expiration times etc can be centrally defined and enforced
- Robust Authentication mechanisms that are not available locally in the hypervisor host can be set up because of integration of authentication function with the directory infrastructure

Security Recommendation HY-SR-15: The user accounts (including privileged accounts) on the hypervisor host must be integrated with the enterprise directory infrastructure in order to enable authentication through robust authentication protocols (e.g., Kerberos), enable enforcement of some corporate security policies (e.g., password policies) as well as handle changes to user account list (addition and deletion of user accounts).

6.3 Secure Configuration of Access through Remote Access Protocol

Hypervisor offering that come with full service console provide remote access capability through standardized remote access protocol such as Secure Shell (SSH). This capability can be used to access the command line interface provided by the service console to perform administrative operations on the hypervisor. However it is a safe practice to restrict the kinds of operations that can be performed through this generic remote access protocol. The way this can be achieved is by restricting SSH access only to certain administrative accounts and the root account with its extensive administrative privileges should not be part of that list. The associated security recommendation therefore is:

Security Recommendation HY-SR-16: The remote access protocol used to access the hypervisor service console should have configuration options available to: completely deny access (i.e., disable remote

access via specific protocols), deny hypervisor root acccount access and restrict access only to a specified list of administrative accounts.

6.4 Leveraging Features for automated security configuration

Some hypervisor offerings provide the capability to define "gold standard" settings for all aspects of hypervisor configuration (e.g., host profiles/templates), attach a profile/template to a specific hypervisor or a cluster (so that the hypervisor configuration parameters are automatically set with the values in the profile/template), check a hypervisor for conformance to a given host profile/template etc. The aspects of the hypervisor configuration include the following:

- Hypervisor CPU Configuration
- Hypervisor Memory Configuration
- Hypervisor Storage Configuration
- Hypervisor Network Configuration
- Hypervisor Hardening Configuration (e.g., disabling SSH, configuring NTP, etc)

The advantage of this feature is that it can be used to define a taxonomy or hierarchy or arbitrary designation of security levels based on combination of values of various configuration state variables. This can be used to enhance the security of hypervisors in the following two ways:

- These pre-configured settings can be used as templates to automatically set the values for configuration parameters of hypervisor. There could be multiple “Host Profiles”, depending upon different hypervisor host operating environments (or nature of workloads). This feature enables reduction of risk in hypervisors introduced due to errors in manual configurations.
- The various “Host Profiles” (Gold Configuration) can be used as baselines or benchmarks to evaluate (or perform “integrity checks”) the configuration settings in the existing running versions of the hypervisors and the findings of the resultant compliance/deviations reports can be utilized to selectively modify certain configuration parameters. This process thus improves the hypervisor's security profile.
- Leverage tools that can automatically remediate the hypervisor to remain in compliance with their defined profile/template.

Security Recommendation HY-SR-17: Use Hypervisor features that enable: (a) definition of a complete set of configuration settings (Gold Configuration) for a hypervisor deployment (b) automate application of those configuration settings to a new or existing hypervisor installation and (c) check compliance of existing hypervisor installation against those configuration settings, if available, in order to minimize manual configuration errors that may increase the security risk.

6.5 Security Considerations as Server-based Software

6.5.1 Keeping Patches Current

The hypervisor being another piece of software, all the manufacturer issued patches must be applied in a timely manner for the relevant versions. Very often, tools for patch management (downloading and installing patches) are often provided through modules in the hypervisor management server.

Security Recommendation HY-SR-18: A good hypervisor patch management practice (supported by a homegrown or an established COTS product) should be in place to keep the hypervisor current with all relevant patches.

6.5.2 Secure Configuration of Built-in Firewall

Some Hypervisor offerings that come with a full service console also come with an built-in firewall that can be used to control traffic in and out of service console. This type of firewall is intended to protect only the management interface of the hypervisor and not the VMs running on them. Further the type of protection is that of a stateless, packet filter, meaning that the firewall does not keep track of the conversations on the network and it simply evaluates each packet that goes through for conformance to service-related rule sets that specify the allowed ports. Hence the rule set consists of features to define incoming and outgoing connections and services (those that are designed to be run on that hypervisor offering) along with their respective TCP and UDP port numbers. Specifically the rule set should specify blocking of all incoming and outgoing traffic, only opening ports and protocols necessary for management of VMs and hypervisor host. These typically include ports that are used by the hypervisor services (some examples include ports needed for SSH client connection, hypervisor client connection, web browser connections, VM console connection etc) as well as any ports and protocols used for any special purpose security agents or third-party applications (e.g., monitoring status of physical server hardware).

Security Recommendation HY-SR-19: ***The built-in firewall for the hypervisor should only be configured to allow ports and protocols (network traffic) needed for enabled services in the hypervisor, such as management and specialized security agents and third-party applications.***

6.5.3 Standardized and Fault Tolerant Logging

All hypervisors come with logging capabilities. While installing and configuring hypervisors it should be ensured that the log daemon (or the equivalent running executable) is turned on. The following best practices are recommended for enabling the logging capabilities in hypervisors:

- The configuration of logging program in a hypervisor should be set up to store the log messages in an external server. This is critical since these messages may become inaccessible if the platform on which the hypervisor is resident is breached.
- Very commonly, the remote log server that receives logs from your hypervisor can be set up to receive these logs using either TCP or UDP. If the remote log server is resident in a remote data center (as opposed to being housed in a server within the same data center), then it is preferable to configure that remote log server to receive logs using the more reliable TCP, and if supported ver a secure channel such as TLS.
- It would be preferable to have logging software that generates logs conforming to a standard (e.g., syslog) format as opposed to one that has a proprietary log format. A standardized logging system has several security advantages compared to a proprietary log generator. They are:
    (a) There are many more open source and commercial analytical tools available for analyzing and reporting on logs in standardized formats as opposed to one with proprietary format. The analysis capability is very much critical for meeting compliance needs such as SOX, PCI DSS and HIPAA.
    (b) There are converters available for converting logs in many proprietary formats to the standardized log format. Examples are the converters that convert from “Windows Event Log” to standardized “syslog” format.
    (c) Further standardized logs facilitate consolidation of logs from different hypervisors products (in a multi-hypervisor virtualized data center) into a centralized repository to enable analysis and reporting for events for the virtualized data center. In fact, the collection and analysis of “syslog” records has spawned a separate security service provided by some Managed Security Service providers who attempt to apply analytics techniques (and sometimes artificial intelligence algorithms) to detect patterns and alert customers of problems.

Security Recommendation HY-SR-20: ***It would be preferable to have a hypervisor logging feature that generates logs in a standardized format to help leverage the use of tools with good analytical capabilities as well as the feature to transmit log records in real time over a secure channel to an external server for fault tolerance.***

6.6 Securing Virtual Network Configuration

To connect the multiple VMs with each other and to the enterprise network (in which the virtualized host is a node), the hypervisor provides the capability to define a software-defined communication fabric called a virtual network through its management console. This management console is not part of the hypervisor kernel but a dedicated management VM. The virtual network is a software-defined artifact that resides entirely within the virtualized host and has as its nodes the VMs residing inside it. The components of this virtual network are: (a) the virtual network interface cards (vNICs) that are defined for each VM and provide connection for each VM to the virtual network (b) the virtual switches that provide selective connectivity among VMs and whose configuration determines the topology of the virtual network and the last but not the least, (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 two main functions has to be considered:

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

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

Security Recommendation HY-SR-21*: Protection for VM Management and Hypervisor Host & Software administration functions can be ensured by placing the management interface of the hypervisor in a dedicated virtual network segment and enforcing traffic controls using a firewall (e.g., designating the subnets in the enterprise network from which incoming traffic into the management interface is allowed).*

Another virtual network configuration that is not related to any specific hypervisor baseline functionality but to the overall availability of the hypervisor platform, is one that ensures flow of communication traffic between VMs residing on the platform to the enterprise network without any bottleneck so that the overall availability and performance of hypervisor platform is not negatively impacted. This is usually accomplished by establishing multiple communication channels for a traffic emanating from a given VM to reach the enterprise network.

<u>Security Recommendation HY-SR-22</u>: Communication from a given VM to the enterprise (physical) network should be enabled by establishing multiple communication paths within the virtualized host. This is usually accomplished by providing multiple physical network adapters (pNICs) for traffic from a particular VM to reach the enterprise network.

7. SECURITY RECOMMENDATION SUMMARY

The mapping of the twenty-two security recommendations to hypervisor's baseline functions is provided in Appendix B. From the list of security recommendations in this document, it should be clear that many of the security recommendations have been necessitated by the unique functions that the hypervisor performs as well as the supporting features of the underlying hardware platform. In particular, the special protection measures that form the basis of many security recommendations stem from:

- The Hypervisor Platform Architectural Choices
- The need to manage multiple VMs (Configuration and State Setting) inside the Hypervisor Host
- Configuration choices in the Administration of Hypervisor Host & Hypervisor Software

As already mentioned in the introduction section, the security recommendations in this document have as its focus, the secure execution of tasks under the five baseline hypervisor functions. The secure operation of VMs (Guest O/S and resident applications) is beyond the scope of this document.

APPENDIX A

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

- HY-BF1: Execution Isolation for Virtual Machines – Scheduling of VMs for execution, Management of the application processes running in VMs such as CPU and Memory Management, context switching between various processor states during the running of applications in VMs etc.
- HY-BF2: Devices Emulation & Access Control – Emulating all Network and Storage (block) devices that different native drivers in VMs are expecting, mediating access to physical devices by different VMs
- HY-BF3: Execution of Privileged Operations for Guest VMs – Certain operations invoked by Guest O/Ss, instead of being executed directly by the host hardware, may have to be executed on its behalf by the hypervisor, because of their privileged nature.
- HY-BF4: Management of VMs – Setting configuration parameters for VMs (VM Images) and control of VM states (Start, Pause, Stop etc).
- HY-BF5: Administration of Hypervisor Platform and Hypervisor Software – This involves setting of parameters for user interactions with the hypervisor host as well as hypervisor software and configuration of Virtual Network inside the hypervisor.

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

A.1 HY-BF1 (Execution Isolation for Virtual Machines)

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) made by Guest VMs for purposes such as managing hard disk partitions, altering memory pages using calls to memory management unit (MMU) etc). *All tasks described so far are carried out by the hypervisor kernel or kernel extension modules*.

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

Since Guest VMs with different O/Ss run on a hypervisor platform, there must be a module that emulates devices for all device drivers available in the Guest O/Ss to support fully virtualized guests (Guests with unmodified O/S). 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 O/S) and translates requests for that device to access requests for actual physical devices running in the kernel of the privileged VM where QEMU runs. 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 O/S is intercepted by the hypervisor kernel and forwarded to QEMU for performing the emulation and access control function. The QEMU is also responsible for relaying the output of the actual physical device back to the corresponding VM that made the I/O call. *From these discussions, it should be clear that all tasks under Device Emulation & Access Control are executed by the QEMU code generally residing in the privileged, dedicated VM.*

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

Certain operations invoked by Guest O/S kernels may have to be executed by the hypervisor because of their privileged nature. These calls are Hypercalls and are analogous to O/S system calls. 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 parititions etc. *Just as for HY-BF1 tasks, Hypercalls are performed by the hypervisor kernel module by providing procedural interfaces*.

A.4 HY-BF4 (Management of VMs)

This encompasses all administrative operations on VMs and consists of two sets of tasks*:* (a) Configuration of Guest VMs and (b) Setting VM states. The VM configuration tasks include: assigning virtual devices (e.g., virtual network interface cards, virtual disks etc), main memory quotas, virtual CPU topologies and priorities etc. The tasks involved in setting VM states include commands such as Start, Pause, Stop etc. VM Management tasks are enabled using a management daemon which provides network interfaces that facilitate remote management of the entire hypervisor host. *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 (Administration of Hypervisor Platform and Hypervisor Software)

These tasks include those that are involved in the configuration of the hypervisor host (virtualized host) and the hypervisor software itself. An important aspect of hypervisor software configuration is the definition of a software-defined virtual network inside the hypervisor host to enable connectivity among VMs, as well as connectivity of VMs to external network (e.g., LAN, WAN etc). *These tasks are enabled by the same management interface used for VM Management (HY-BF4) that was described above*.

APPENDIX B

TRACEABILITY OF SECURITY RECOMMENDATION TO HYPERVISOR BASELINE FUNCTIONALITY

IDENTIFIER	SECURITY RECOMMENDATION	BASELINE FUNCTIONALITY/TASK
HY-SR-1	Bare Metal (Type 1) Hypervisor – Reduced Attack surface because of no Host O/S	N/A
HY-SR-2	Hypervisor Platform with hardware assisted virtualization (both Instruction Set and Memory)	HY-BF1 (Execution Isolation for VMs)
HY-SR-3	Hypervisor Platform that provides a Measured Launch Environment (MLE) and a standards-based TPM and an attestation process	N/A (Provides Boot Integrity Assurance rather than Assurance against a run-time functionality)
HY-SR-4	Hypervisor Management Console with a smaller code and disk footprint and smaller number of exposed interfaces	HY-BF4 & HY-BF5
HY-SR-5	(a) Disallowing non-certified drivers and (b) Running QEMU (device emulation code) process on unprivileged VMs (instead of in a Management or Privileged VM) if architecture permits	HY-BF2
HY-SR-6	Limiting the Ratio of Combined Virtual RAM allocated to VMs to Total Physical RAM of the Virtualized Host	HY-BF4
HY-SR-7	Ability to specify RAM allocations with Guaranteed,Upper Limit and Priority Values	HY-BF4
HY-SR-8	The number of virtual CPUs allocated to a VM should be strictly less than the total number of cores in the hypervisor host	HY-BF4
HY-SR-9	It should be possible to specify a lower and upper limit for CPU clock cycles for each VM and also a priority value for CPU access	HY-BF4
HY-SR-10	VM Image library should not be hosted on the hypervisor host and should have digitally signed VM images	HY-BF4
HY-SR-11 & HY-SR-12	Security Monitoring on VM Activities should cover: (a) Malicious processes running inside VMs and (b) Malicious traffic going in and out of VM. These monitoring functions should be based on tools residing outside any monitored VMs and should be based on VM Introspection API	HY-BF4
HY-SR-13	Access Control solution with granular permission assignment for VM Management – Permission at VM level, VM Group level, VM Group level with specific exceptions	HY-BF4

HY-SR-14	Limiting the number of user accounts with direct access to hypervisor host	HY-BF5
HY-SR-15	Integrating user accounts on the hypervisor host to an enterprise directory infrastructure to: (a) Enable Robust Multi-factor Authentication Protocols and (b) Maintain Integrity of User Account Maintenance	HY-BF5
HY-SR-16	Access to Hypervisor Management Console should be: (a) denied to root account and (b) restricted to a limited administrative accounts	HY-BF5
HY-SR-17	Use Hypervisor features that enable: (a) definition of a complete set of configuration settings (Gold Configuration) for a hypervisor deployment (b) automate application of those configuration settings to a new or existing hypervisor installation and (c) check compliance of existing hypervisor installation against those configuration settings, if available, in order to minimize manual configuration errors that may increase the security risk.	HY-BF5
HY-SR-18	A good hypervisor patch management practice (supported by a homegrown or an established COTS product) should be in place to keep the hypervisor current with all relevant patches.	HY-BF5
HY-SR-19	The built-in firewall for the hypervisor should only be configured for allowing traffic needed for enabled services in the hypervisor, such as Management and specialized security agents and third-party applications.	HY-BF5
HY-SR-20	It would be preferable to have a hypervisor logging feature that generates logs in a standardized format to help leverage the use of tools with good analytical capabilities as well as the feature to transmit log records in real time over a secure channel to an external server for fault tolerance.	HY-BF5
HY-SR-21	The management interface of the hypervisor should: (a) be placed in a dedicated virtual network segment and (b) access to that interface should only be allowed from designated subnets in the enterprise network	HY-BF5
HY-SR-22	Redundant communication channels from a VM to the enterprise (external) network by configuring virtual network paths from a given VM to multiple physical Network interface adapters of the virtualized host.	HY-BF5

APPENDIX C

Full Virtualization: A form of Virtualization which uses a hypervisor hardware platform with virtualization extensions and hence supports Virtual Machines (VMs) (see below) with unmodified Guest O/Ss to run on them.

Guest Operating System (O/S): 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 the kernel of an O/S, along with supporting kernel modules that provides separation 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 assist virtualization - specifically Instruction Set and Memory virtualization.

Virtual Machine (VM): A software-defined complete execution stack consisting of virtualized hardware, operating system, middleware 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

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