

The attached DRAFT document (provided here for historical purposes), released on April 11, 2018, has been superseded by the following publication:

Publication Number: **NIST Special Publication (SP) 800-125A Rev. 1**

Title: ***Security Recommendations for Server-based Hypervisor Platforms***

Publication Date: **June 2018**

- Final Publication: <https://doi.org/10.6028/NIST.SP.800-125Ar1> (which links to <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-125Ar1.pdf>).
- Related Information on CSRC:  
Final: <https://csrc.nist.gov/publications/detail/sp/800-125a/rev-1/final>

---

---

# Security Recommendations for Server-based Hypervisor Platforms

---

Ramaswamy Chandramouli

---

C O M P U T E R   S E C U R I T Y

---

Draft NIST Special Publication 800-125A  
Revision 1

# Security Recommendations for Server-based Hypervisor Platforms

Ramaswamy Chandramouli  
*Computer Security Division  
Information Technology Laboratory*

April 2018



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

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

67  
68 **Authority**  
69

70 This publication has been developed by NIST in accordance with its statutory responsibilities under the  
71 Federal Information Security Modernization Act (FISMA) of 2014, 44 U.S.C. § 3551 *et seq.*, Public Law  
72 (P.L.) 113-283. NIST is responsible for developing information security standards and guidelines, including  
73 minimum requirements for federal information systems, but such standards and guidelines shall not apply  
74 to national security systems without the express approval of appropriate federal officials exercising policy  
75 authority over such systems. This guideline is consistent with the requirements of the Office of Management  
76 and Budget (OMB) Circular A-130.

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

83  
84 National Institute of Standards and Technology Special Publication 800-125A Revision 1  
85 Natl. Inst. Stand. Technol. Spec. Publ. 800-125A Rev. 1, 38 Pages (April 2018)  
86 CODEN: NSPUE2  
87  
88  
89

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

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

100 Organizations are encouraged to review all draft publications during public comment periods and provide feedback to  
101 NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at  
102 <https://csrc.nist.gov/publications>.

103  
104 **Public comment period: April 11, 2018 through May 2, 2018**  
105

106 National Institute of Standards and Technology  
107 Attn: Computer Security Division, Information Technology Laboratory  
108 100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930  
109 Email: [sp800-125A-comments@nist.gov](mailto:sp800-125A-comments@nist.gov)

110  
111 All comments are subject to release under the Freedom of Information Act (FOIA).  
112

## 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 platform is a collection of software modules that provides virtualization of hardware resources (such as CPU, Memory, Network and Storage) and thus enables multiple computing stacks (made of an operating system (OS) and application programs) called Virtual Machines (VMs) to be run on a single physical host. In addition, it may have the functionality to define a network within the single physical host (called virtual network) to enable communication among the VMs resident on that host as well as with physical and virtual machines outside the host. With all this functionality, the hypervisor has the responsibility to mediate access to physical resources, provide run time isolation among resident VMs and enable a virtual network that provides security-preserving communication flow among the VMs and between the VMs and the external network. The architecture of a hypervisor can be classified in different ways. The security recommendations in this document relate to ensuring the secure execution of baseline functions of the hypervisor and are therefore agnostic to the hypervisor architecture. Further, the recommendations are in the context of a hypervisor deployed for server virtualization and not for other use cases such as embedded systems and desktops. Recommendations for secure configuration of a virtual network are dealt with in a separate NIST document (Special Publication 800-125B).

### Keywords

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

155

156

## Acknowledgements

157

158 The author, Ramaswamy Chandramouli wishes to thank his colleague Tim Grance for his  
159 personal input on the content and in helping with the logistics of the publication. Special thanks  
160 to Andreas Bartelt from Bosch Center of Competence Security for valuable input regarding  
161 technologies for device virtualization. He also thanks Michael Bartock for his valuable review  
162 and feedback as a division reader. Last but not the least, he expresses his thanks to Isabel Van  
163 Wyk for her detailed editorial review.

164

165

## Note to Reviewers

166

167

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

172

# Table of Contents

173  
174  
175 EXECUTIVE SUMMARY ..... v  
176 1. INTRODUCTION, SCOPE & TARGET AUDIENCE..... 1  
177 1.1 Hypervisor Baseline Functions ..... 1  
178 1.2 Scope of this document ..... 3  
179 1.3 Target Audience..... 4  
180 1.4 Relationship to other NIST Guidance Documents ..... 4  
181 2. APPROACH FOR DEVELOPING SECURITY RECOMMENDATIONS ..... 5  
182 2.1 Hypervisor Platform Threat Sources..... 5  
183 2.2 Potential Threats to Hypervisor Baseline Functions ..... 6  
184 3. SECURITY RECOMMENDATION FOR OVERALL PLATFORM INTEGRITY ..... 9  
185 4. SECURITY RECOMMENDATIONS HY-BF1 ..... 11  
186 4.1 Hardware Assistance for Virtualization ..... 11  
187 4.2 VM Memory Allocation Scheduling Options..... 12  
188 4.3 VM CPU Allocation Options ..... 13  
189 5. SECURITY RECOMMENDATIONS FOR HY-BF2..... 14  
190 6. SECURITY RECOMMENDATIONS FOR HY-BF4..... 15  
191 6.1 VM Image Management ..... 15  
192 6.2 VM Live Migration ..... 15  
193 6.3 VM Monitoring and Security Policy Enforcement ..... 16  
194 6.4 VM Configuration Management..... 18  
195 6.5 Fine-grained Administrative Privileges for VM Management ..... 18  
196 7. SECURITY RECOMMENDATIONS FOR HY-BF5..... 20  
197 7.1 Centralized Administration ..... 20  
198 7.2 Securing the Management Network..... 20  
199 8. SECURITY RECOMMENDATION SUMMARY ..... 22  
200 Appendix A ..... 23  
201 Appendix B: Traceability of Security Recommendation to Hypervisor Baseline Functions ..... 26  
202 Appendix C: Glossary ..... 30  
203 Appendix D: References ..... 31  
204  
205

## EXECUTIVE SUMMARY

Server Virtualization is now an established technology in data centers used as enterprise IT infrastructure and in those offered for cloud service as it brings about better utilization of hardware resources, saves/reduces physical space in data centers, and reduces power consumption and administrative overhead. The core software used for server virtualization is called the Hypervisor which directly provides CPU and memory virtualization. Together with its supporting modules, it enables virtualization of all hardware resources (e.g., CPU, Memory, Network and Storage) and thus enables multiple computing stacks called Virtual Machines (VMs) or Guests, each hosting an OS (Guest OS) and application programs, to be run on a single physical host. This physical host is referred to as Virtualized Host or Hypervisor Host. Since the hypervisor by itself cannot provide all functions needed for server virtualization, it has supporting software modules (e.g., device drivers) for devices (e.g., Network and Storage devices) virtualization in addition to management modules for VM lifecycle operations and hypervisor configuration. The hypervisor together with these supporting modules and the hosting hardware constitute the hypervisor platform. The hypervisor can be installed either directly on the hardware or bare metal (Type 1 Hypervisor) or on top of a full-fledged conventional OS called Host OS (Type 2 Hypervisor).

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

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

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

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

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

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



## 1. INTRODUCTION, SCOPE, AND TARGET AUDIENCE

The Hypervisor is the core software that provides server virtualization. Along with its supporting modules, it enables virtualization of all hardware resources (e.g., CPU, Memory, Network, and Storage) and thus enables multiple computing stacks (basically made of an OS and application programs) to be run on a single physical host. Such a physical host is called a Virtualized Host (also referred to as a Hypervisor Host in this document), and the individual computing stacks are encapsulated in an artifact called Virtual Machines (VMs). To be an independent executable entity, the definition of a VM should include resources (e.g., CPU, Memory, etc.) allocated to it. The VMs are also called “Guests,” and the operating system (OS) running inside each of them is called “Guest OS.” The resources associated with a VM are virtual resources as opposed to physical resources associated with a physical host. The hypervisor together with these supporting modules and the hosting hardware constitute the hypervisor platform.

The primary function of the hypervisor is to enforce guest OS isolation as well as controlled resource sharing among guest VMs. Thus, it plays many of the roles a conventional OS does on a non-virtualized host (server). Just as a conventional OS provides isolation between the various applications (or processes) running on a server, the hypervisor provides isolation between one or more VMs running on it. Also, similar to an OS, the hypervisor mediates access to physical resources (devices) across multiple VMs. While access to CPU and memory (to ensure process isolation) are handled directly by the hypervisor (through instruction set (CPU) virtualization and memory virtualization respectively with or without assistance from hardware), it handles the mediation of access to devices (devices virtualization) by calling on software modules running either in the kernel or in dedicated VMs called Device-driver VMs. The hypervisor can be installed either directly on the hardware or bare metal (Type 1 Hypervisor) or on top of a full-fledged conventional OS called Host OS (Type 2 Hypervisor).

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

The hypervisor baseline functions consist of:

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

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

### 1.1 Hypervisor Baseline Functions

While the basic function of a hypervisor is to virtualize hardware (a physical host) to enable the operation of multiple virtual hosts (popularly known as VMs), commercial hypervisor offerings come with differing feature sets. The modules that provide the same set of features are given different names in different product offerings. Hence, for accomplishing the goals of this document, it is necessary to identify a set of baseline features of a hypervisor that covers all functions for supporting hardware virtualization. In some instances, the module that just presents a set of virtualized resources to the VMs is called the Virtual Machine Manager

- 302 (VMM). When VMMs are combined with the modules that provide OS-level services, such as the scheduling  
 303 of VMs in the CPU, they are called the hypervisor. These hypervisor baseline features or functions are:
- 304 • HY-BF1: VM Process Isolation – Scheduling of VMs for execution, Management of the application  
 305 processes running in VMs such as CPU and Memory Management, context switching between various  
 306 processor states during the running of applications in VMs, etc. In order to ensure VM process isolation,  
 307 memory access from DMA capable devices needs to be under hypervisor control as well (e.g., via  
 308 IOMMU). However, this function is considered under HY-BF2 since it pertains to devices mediation.
  - 309 • HY-BF2: Devices Mediation & Access Control – making devices available to VMs (e.g., via  
 310 emulation, para-virtualization, passthrough or self-virtualizing hardware devices) and controlling  
 311 which VMs are allowed to access which devices (e.g., Network Interface Card (NIC), storage device  
 312 such as IDE drive, etc.).
  - 313 • HY-BF3: Direct Execution of commands from Guest VMs – Certain commands from Guest OSs are  
 314 executed directly by the hypervisor instead of being triggered through interrupts and context  
 315 switching. This function applies to hypervisors that have implemented para-virtualization instead of full  
 316 virtualization
  - 317 • HY-BF4: VM Lifecycle management – All functions including creation and management of VM images,  
 318 control of VM states (Start, Pause, Stop, etc.), VM migration, making snapshots, VM monitoring, and  
 319 policy enforcement
  - 320 • HY-BF5: Management of hypervisor platform – Defining some artifacts and setting values for various  
 321 configuration parameters in hypervisor software modules including those for configuration of a Virtual  
 322 Network inside the hypervisor and updates and patching to those modules.

323  
 324 The brief descriptions of the five baseline functions listed above are sufficient to guide the discussion in  
 325 the rest of the document. For interested readers, a detailed description of the above functions is given  
 326 in Appendix A.

327  
 328 The above functions are carried out by different hypervisor components or software modules. There are some  
 329 minor differences among hypervisor products in the way they distribute these functions. The mapping of these  
 330 functions to hypervisor components and the location of these components in overall hypervisor architecture  
 331 are given in Table 1 below:

Baseline Function	Component (Software Module)	Location
VM Process Isolation (HY-BF1)	Hypervisor Kernel	Either an OS kernel (along with a kernel module) itself or a component installed on a full-fledged OS (Host OS)
Devices Mediation & Access Control (HY-BF2)	Device emulator or Device driver	Either in a dedicated VM (called Device-driver VM) or in the hypervisor kernel itself
<u>Direct Execution of commands from Guest VMs</u> (HY-BF3)	Hypervisor Kernel	Pertain to only para-virtualized hypervisors and handled by hypercall interfaces in that type of hypervisor
VM Lifecycle management (HY-BF4)	A management daemon	Installed on top of hypervisor kernel but runs in unprivileged mode
Management of hypervisor platform (HY-BF5)	A set of tools with CLI (command line interface) or a GUI	A console or shell running on top of hypervisor kernel

333  
 334 Hypervisor products differ in the distribution of the above functions and in the name assigned to the  
 335 corresponding component. In general, functions HY-BF1 and HY-BF3 are offered by modules running in a  
 336 kernel collectively called “Hypervisor” while HY-BF2 is enabled by a software module that runs either in a  
 337 dedicated VM (called Device-driver VM) or in the hypervisor kernel itself. The functions HY-BF4 and HY-

338 BF5 are performed by a module called management or service console or through a kernel module. Just like  
339 the module that performs the HY-BF2 function, the console is a software layer that is generally not built  
340 into the hypervisor kernel but runs on top of it as a privileged VM and could be built either with a full-fledged  
341 OS installed inside it or with an ultra-light OS used to present an API (shell and network access) with utility  
342 functions that facilitate performing only the hypervisor-specific configuration and administrative tasks.  
343

## 344 1.2 Scope of this document

345  
346 The architecture of a hypervisor deployed for server virtualization can be classified in different  
347 ways:

- 348
- 349 (a) Based on the entity over which the hypervisor installs – Type 1 Hypervisor or Type 2  
350 Hypervisor (already described)
  - 351 (b) Based on the type of virtualization
    - 352 a. Full Virtualization – Guest OS runs unmodified in the VM
    - 353 b. Para Virtualization – Guest OS kernel must be modified to generate hypercalls
- 354

355 The trust model assumed for this document is as follows:

- 356
- 357 • All components in a VM are untrusted including the guest OS and its associated utilities (e.g.,  
358 guest device drivers) that run in the kernel space and all applications that run in the user space
  - 359 • The device drivers that are implemented within the hypervisor platform are untrusted unless they  
360 carry a security certification
  - 361 • The hypervisor kernel component that provides isolation between VMs is trusted
  - 362 • The host OS is trusted for Type 2 hypervisors
  - 363 • The hardware of the hypervisor host is trusted
- 364

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

- 368
- 369 • All tasks that relate to functions HY-BF1, HY-BF2, and HY-BF4
  - 370 • HY-BF3, which relates to the handling of hypercalls in para-virtualized hypervisors, is a  
371 trusted function of the hypervisor and not included in the security recommendations
  - 372 • All tasks under HY-BF5 are included, except for those related to the definition and  
373 configuration of virtual network (secure configuration of virtual networks is covered under a  
374 separate NIST document, SP 800-125B)
- 375

376 Also within the scope of this document are recommendations to ensure overall platform integrity.

377  
378 The security recommendations do not cover the following aspects:

- 379
- 380 • Hypervisor host user account management
  - 381 • Hypervisor host authentication and access control
  - 382 • Routine administration of Host OS (e.g., keeping patches current)
  - 383 • Routine administration of Guest OS
  - 384 • Security of Guest OSs running on VMs
  - 385 • Security of Applications/Services running on VMs
- 386

### 387 1.3 Target Audience

388

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

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

### 396 1.4 Relationship to other NIST Guidance Documents

397

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

406

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

414

- 415 • A focused set of security recommendations that are agnostic to the architecture for the deployment of  
416 Server Virtualization technology’s foundational component (i.e., the hypervisor) is provided.
- 417 • Since real world deployment includes provisioning of VMs, all VM life-cycle operations, from creation  
418 and management of VM images to their administration using granular privileges, is covered.
- 419 • Recognizing that the hypervisor is a purpose-built Operating System (OS) kernel and that the  
420 security of a server OS depends upon its weakest link regardless of the distribution (e.g., driver  
421 software), security recommendations relating to these components have been provided as well.
- 422 • Recognizing that the hypervisor performs certain privileged operations without interference from any  
423 other entity in the virtualized host and that leveraging hardware support for these operations will make  
424 a significant difference to the overall security of hypervisor deployment, the security recommendations  
425 also improve performance when virtualization-specific functions (e.g., memory tables for multiple  
426 VMs) are offloaded (leveraged) to the processor instead of through software functions.
- 427 • All security recommendations are intended to provide assurance against exploitation of threats to tasks  
428 involved in the hypervisor’s baseline functions.

429

## 2. APPROACH FOR DEVELOPING SECURITY RECOMMENDATIONS

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

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

It must be noted that in some cases of large open-source and commercial software environments (e.g., Database Management System (DBMS) platform), the approach adopted for secure deployment and usage is to study the reports published in the public vulnerability databases for various product offerings, seek out available patches through online public forums or the software vendor, and look for recommended secure configuration settings (also via online public forums or the software vendor websites). We do not adopt this approach in this document since the intended purpose is not to provide security recommendations for a specific open source or commercial hypervisor product offering but rather for the entire product class based on its baseline functions.

### 2.1 Hypervisor Platform Threat Sources

The hypervisor software is resident on a physical host that is connected to the enterprise network. It has the capability to be remotely administered. At the same time, it supports multiple virtual hosts (virtual machines or VMs) that are generally nodes of a software-defined virtual network inside that physical host. In some cases, they could be nodes of an isolated network or sharing the host network. Based on this scenario, one can identify three basic sources of threats to a hypervisor platform, each of which is identified by using the symbol HY-TS#:

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

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

The hypervisor controls VM access to physical hardware resources as well as provides isolation among VMs. VM access to hardware resources such as CPU and memory are directly controlled by the hypervisor while access to resources such as network and storage devices are controlled through modules (drivers) that reside in the kernel module or in a privileged VM (i.e., Management VM). The network isolation among VMs is

480 provided by assigning a unique IP or MAC address to each VM, defining virtual local area networks (VLANs)  
481 or overlay networks, and assigning the appropriate network identifier to each VM. The nature of threats to the  
482 hypervisor from rogue or compromised VMs can manifest in the following ways:  
483

484 Note that each threat is identified by the symbol HYP-T#, where HYP stands for hypervisor, T stands for  
485 threat, and # stands for the sequence number.  
486

- 487 • Breach of Process Isolation - VM Escape (HYP-T1): Major threats to any hypervisor come from rogue  
488 VMs. Rogue VMs manage to subvert the isolation function provided by the VMM/hypervisor to hardware  
489 resources such as memory pages and storage devices. In other words, the rogue or compromised VMs  
490 may access areas of memory belonging to the hypervisor or other VMs and storage devices they are not  
491 authorized to access. Possible reasons for this threat include (a) hypervisor design vulnerabilities or (b)  
492 malicious or vulnerable device drivers. Potential downstream impacts of a rogue VM taking control of the  
493 hypervisor include the installation of rootkits or attacks on other VMs on the same virtualized host.  
494
- 495 • Breach of Network Isolation (HYP-T2): Potential threats to isolation include attacks such as IP or MAC  
496 address spoofing by a rogue VM and Traffic Snooping, or the interception of virtual network traffic,  
497 intended for a VM on the same virtual network segment. The impact of the subversion of these network  
498 controls is loss of confidentiality. Some VMs will be viewing information for which they are not  
499 authorized.  
500
- 501 • Denial of Service (HYP-T3): Misconfigured or malicious VMs may be consuming a disproportionately  
502 high percentage of host resources, resulting in denial-of-service to other VMs on the hypervisor host.  
503

## 504 **2.2 Potential Threats to Hypervisor Baseline Functions**

505  
506 In this section, the tasks in each of the five hypervisor baseline functions (with the exception of HY-BF3) are  
507 examined, and the threats to the secure execution of those tasks are analyzed by relating to the causes identified  
508 in the previous section.  
509

### 510 **2.2.1 Potential Threats to HY-BF1**

511  
512 The primary threat to hypervisor's HY-BF1 function (VM Process Isolation) is breach of process isolation  
513 (HYP-T1). As mentioned in section 2.1, one of the causes for this threat is hypervisor design vulnerability.  
514 Some potential design vulnerabilities that pertain to this threat are discussed here with an explanation of the  
515 context under which they may manifest. Each vulnerability is identified by the symbol HYP-DV#, where  
516 HYP stands for hypervisor, DV stands for design vulnerability, and # stands for the sequence number.  
517

- 518 • Virtual Machine Control Structure (HYP-DV1): To properly schedule an individual VM's tasks (i.e.,  
519 vCPU tasks since each guest VM is allocated a set of virtual CPUs), the register states must be handled  
520 appropriately. To enable the saving and loading of the state of each vCPU, the hypervisor uses a data  
521 structure called Virtual Machine Control Structure (VMCS). Faulty implementation of this data structure  
522 has been known to cause hypervisor memory leaks.  
523
- 524 • Handling Sensitive Instructions (HY-DV2): On hardware platforms that do not provide assistance for  
525 virtualization, there should be a software mechanism to discover sensitive or critical instructions, send  
526 them to the VMM (hypervisor), and replace them with safer instructions using techniques such as binary  
527 translation before executing them on the hardware. Any error in not trapping the critical instructions or  
528 faulty translation may have security implications in the form of a guest OS being allowed to execute  
529 privileged instructions.

- 530
- 531 • Memory Management Unit-MMU (HYP-DV3): The hypervisor runs a software-based Memory  
532 Management Unit (MMU) that allocates a shadow page table for each VM since guest VMs cannot be  
533 granted direct access to the hardware-based MMU as that would potentially enable them to access memory  
534 belonging to the hypervisor and other co-hosted VMs (under some situations). However, a faulty  
535 implementation of software-based MMU could lead to disclosure of data in arbitrary address spaces, such  
536 as memory segments belonging to the hypervisor and co-located VMs, thus resulting in a breach of  
537 memory isolation.
- 538
- 539 • Input/Output Memory Management Unit, IOMMU (HY-DV4): The hypervisor leverages the hardware  
540 I/O Memory Management Unit to enforce memory separation for device drivers and processes using direct  
541 memory access (DMA). This feature is built into the hypervisor and enabled in the hardware using a  
542 firmware switch. If unused, it may result in a vulnerability whereby the DMA could potentially be used  
543 as a common attack vector by one VM to overwrite physical memory used by other VMs and processes.
- 544

545 Out of these, the vulnerabilities HYP-DV1 and HYP-DV2 should be addressed through proper coding and  
546 testing of those modules. Therefore, no security protection measures can be applied at the deployment and  
547 usage stage. However, the memory violation vulnerability HYP-DV3 and DMA violation vulnerability HY-  
548 DV4 can be addressed by hosting the hypervisor on a hardware platform that provides assistance for memory  
549 virtualization through a virtualization-aware hardware memory management unit and DMA transfers through  
550 the re-mapping of DMA transfers, respectively. Due to these two vulnerabilities, the threat HYP-T1, a breach  
551 of process isolation, has been addressed through security recommendation HY-SR-2 in section 4.

552

553 Further, correct execution isolation requires that each VM obtains the proper memory and CPU resources  
554 necessary for its hosting applications and that there is no denial of service. Ensuring adequate memory through  
555 proper configuration of memory allocation options is addressed through security recommendation HY-SR-3,  
556 and ensuring proper allocation of virtual CPUs through the appropriate configuration of vCPU allocation  
557 options are addressed through security recommendations HY-SR-4 and HY-SR-5.

558

### 559 **2.2.2 Potential Threat to HY-BF2**

560

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

565

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

574

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

581 hypercall interface (the corresponding calls are called hypercalls). Analysis of threats due to these hypercalls  
582 is provided in the next subsection.

583  
584 The third approach to device virtualization, the passthrough approach (or direct device assignment), is  
585 deployed for situations where a VM needs exclusive access to a device (e.g., NIC, disk controller, HBA, USB  
586 controller, serial port, firewire controller, soundcard, etc) for performance reasons so as to be devoid of the  
587 overhead due to emulation. Since generally this is required for PCI devices, this is also called as PCI  
588 Passthrough. Since many of these devices have a memory-mapped interface, they can read or write directly to  
589 or from main memory and are also called Direct Memory Access (DMA) capable devices. To provide  
590 exclusive access to a DMA capable device for a VM, the memory pages of the device are mapped into guest  
591 VM's address space. The following is the threat due to DMA capable devices.

592  
593 Threat due to DMA-capable hardware devices (HY-DV5): The security threat from DMA-capable device is  
594 that, since the VM controls the device, it can program the device to perform DMA operations directed at any  
595 physical (host) memory location, including the areas belonging to other VMs or the hypervisor [6]. Thus, the  
596 direct device assignment has the potential to subvert the isolation between VMs (rather making the MMU  
597 enforced isolation function (part of HY-BF1) meaningless).

598  
599 Apart from three types of device virtualization described above, hypervisor hosts can support self-virtualizing  
600 hardware devices. These devices have interfaces that can export a set of virtual functions (VFs) corresponding  
601 to a physical function (PF). The hypervisor then can assign these VFs to multiple guest VMs, while it retains  
602 control of the PF. These devices conform to Single Root I/O Virtualization (SR-IOV) specification and thus  
603 enable DMA capable devices to be shared among VMs (as virtualization and multiplexing are done by the  
604 devices themselves) instead of being dedicated to a single VM as in passthrough mode.

605  
606

### 607 **2.2.3 Potential Threat to HY-BF3**

608  
609 The previous subsection presented a scenario where the hypervisor has to execute certain instructions through  
610 its hypercall interface. A potential security issue with hypercalls is that the lack of proper validation of certain  
611 operations (e.g., not checking the scope, allowing a full dump of a VM's Virtual Machine Control Block, or  
612 input checking) can potentially cause the entire hypervisor host to crash. This is again a design vulnerability  
613 that must be addressed through proper validation and testing of the relevant hypervisor code rather than  
614 through configuration or deployment procedures.

615  
616

### 617 **2.2.4 Potential Threats to HY-BF4**

618  
619 Potential threats to the secure execution of tasks under this function (i.e., VM Lifecycle Management) include:

- 620
- 621 • Presence of non-standard VM images in the library, including those with outdated OS versions and  
622 patches, which could result in any of the platform-level threats (HYP-T1 through HYP-T3)
- 623 • Presence of non-standard running VM instances due to their creation from non-standard images,  
624 restoration from snapshots, a drift from standard as a result of a lapse in monitoring, and updates that  
625 could result in any of the platform-level threats (HYP-T1 through HYP-T3)
- 626

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

631



### 2.2.5 Potential Threats to HY-BF5

The tasks under this function relate to the overall administration of a hypervisor host (i.e., virtualized host) and the hypervisor software and are usually performed through user-friendly web interfaces or network-facing virtual consoles. Threats to the secure execution of these tasks are common in any remote administration and are therefore not addressed in this document. However, the core requirement in a data center with virtualized hosts is to have a uniform configuration for hypervisors based on different criteria such as sensitivity of applications based on the set of hosted VMs, line of business or client in cloud service environments, etc. Thus, the security recommendations include a centralized management of hypervisor configuration (HY-SR-19) and a dedicated network segment for management traffic (HY-SR-20).

Some conventional security fixes may not be practical in the case of hosts hosting a hypervisor. For example, in the case of a network attack on a physical server that is not virtualized, merely turning off the offending port is a solution to preventing the server from spamming the network with a bot attack. However, such a solution is not practical in the case of a hypervisor host since the same port in the physical network interface card of the hypervisor host could be shared by several running VMs. Instead, a specialized security fix, such as disabling the virtual NICs of VMs that use those ports, is needed.

## 3. SECURITY RECOMMENDATION FOR OVERALL PLATFORM INTEGRITY

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

Some hardware platforms provide support for MLE with firmware routines for measuring the identity (usually the hash of the binary code) of the components in a boot sequence. An example of a hardware-based cryptographic storage module that implements the measured boot process is the standards-based Trusted Platform Module (TPM), which has been standardized by the Trusted Computing Group (TCG) [4]. The three main components of a TPM are: (a) Root of Trust for Measurement (RTM) – makes integrity measurements (generally a cryptographic hash) and converts them into assertions, (b) Root of Trust for Integrity (RTI) - provides protected storage, integrity protection, and a protected interface to store and manage assertions, and (c) Root of Trust for Reporting (RTR) - provides a protected environment and interface to manage identities and sign assertions. The RTM measures the next piece of code following the boot sequence. The measurements are stored in special registers called Platform Configuration Registers (PCRs). The measured boot process is briefly explained here using TPM as an example. The measured boot process starts with the execution of a trusted immutable piece of code in the BIOS, which also measures the next piece of code to be executed. The result of this measurement is extended into the PCR of the TPM before the control is transferred to the next program in the sequence. Since each component in the sequence in turn measures the next before handing off control, a chain of trust is established. If the measurement chain continues through the entire boot sequence, the resultant PCR values reflect the measurement of all components.

684  
685 The attestation process starts with the requester invoking, via an agent on the host, the TPM Quote command.  
686 It specifies an Attestation Identity Key (AIK) to perform the digital signature on the contents of the set of  
687 PCRs that contain the measurements of all components in the boot sequence to quote and a cryptographic  
688 nonce to ensure freshness of the digital signature. After receiving the signed quotes, the requester validates  
689 the signature and determines the trust of the launched components by comparing the measurements in the  
690 TPM quote with known good measurements.

691  
692 The MLE can be incorporated in the hypervisor host as follows:  
693

- 694 • The hardware hosting the hypervisor is established as a root-of-trust, and a trust chain is established from  
695 the hardware through the BIOS and to all hypervisor components.
- 696 • For the hardware consisting of the processor and chipset to be established as the root-of-trust and to build  
697 a chain of trust, it should have a hardware-based module that supports an MLE. The outcome of launching  
698 a hypervisor in MLE-supporting hardware is a measured launch of the firmware, BIOS, and either all or  
699 a key subset of hypervisor (kernel) modules, thus forming a trusted chain from the hardware to the  
700 hypervisor.
- 701 • The hypervisor offering must be able to utilize the MLE feature. In other words, the hypervisor should be  
702 able to invoke the secure launch process, which is usually done by integrating a pre-kernel module into  
703 the hypervisor's code base since the kernel is the first module installed in a hypervisor boot up. The  
704 purpose of this pre-kernel module is to ensure the selection of the right authenticated module in the  
705 hardware that performs an orderly evaluation or measurement of the launch components of the hypervisor  
706 or any software launched on that hardware. The Tboot is an example of a mechanism that enables the  
707 hypervisor to take advantage of the MLE feature of the hardware.
- 708 • All hypervisor components that are intended to be part of the Trusted Computing Base (TCB) must be  
709 included within the scope of the MLE-enabling mechanism so that they are measured as part of their  
710 launch process.

711  
712 The MLE feature with storage and reporting mechanisms on the hardware of the virtualized host can be  
713 leveraged to provide boot integrity assurance for hypervisor components by measuring the identity of all  
714 entities in the boot sequence, starting with firmware, BIOS, hypervisor and hypervisor modules; comparing  
715 them to "known good values;" and reporting any discrepancies. If the measured boot process is to be extended  
716 to cover VMs and its contents (guest OS and applications), a software-based extension to the hardware-based  
717 MLE implementation within the hypervisor kernel is required. The security recommendation for ensuring a  
718 secure boot process for all components of a hypervisor platform can now be stated as follows:  
719

720 **Security Recommendation HY-SR-1:** The hypervisor that is launched should be part of a platform  
721 and an overall infrastructure that contains: (a) hardware that supports an MLE with standards-based  
722 cryptographic measurement capabilities and storage devices and (b) an attestation process with the  
723 capability to provide a chain of trust starting from the hardware to all hypervisor components.  
724 Moreover, the measured elements should include, at minimum, the core kernel, kernel support modules,  
725 device drivers, and the hypervisor's native management applications for VM Lifecycle Management  
726 and Management of Hypervisor. The chain of trust should provide assurance that all measured  
727 components have not been tampered with and that their versions are correct (i.e., overall boot integrity).  
728 If the chain of trust is to be extended to guest VMs, the hypervisor should provide a virtual interface to  
729 the hardware-based MLE.  
730

## 4. SECURITY RECOMMENDATION HY-BF1

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

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

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

### 4.1 Hardware Assistance for Virtualization

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

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

The security advantages of hardware-assisted virtualization platforms include the following:

- One of the potential security vulnerabilities for hypervisors is the buffer overflow attacks from VMs resident on the virtualized host platform. The hardware support for memory management (e.g., Extended Page Tables, or EPT) that comes as part of the hardware-assisted virtualization can be leveraged to prevent code execution from memory locations reserved for data storage, thus preventing buffer overflow attacks.
- Hardware extensions for Virtualization provide two modes of execution: host or root mode and guest or non-root mode. The host mode runs at a higher privilege than guest mode. The hypervisor code, which provides the baseline functionality HY-BF1 (processor allocation and memory management),

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

- 782 • A common threat in virtualization platforms involves a malicious VM accessing areas of memory  
783 belonging to other VMs. This is called a VM Escape attack. Hardware platforms with IOMMU provide  
784 safety against this through features such as Direct Memory Access (DMA) remapping, which limits  
785 allowed DMA access to the assigned protection domain (i.e., preventing a device from performing  
786 DMA beyond its allocated area).
- 787 • The advantage of hardware providing assistance for both forms of virtualization is that the emulation  
788 module of the hypervisor can present the true hardware architecture of the physical host instead of  
789 modified hardware architecture. The consequence of this feature is that an unmodified Guest OS, along  
790 with their native device drivers, can be run in VMs. The security implication of enabling this feature  
791 is that significantly more CVE data is available for a Guest OS, as well as patch versions and certified  
792 device drivers for each OS version.

793  
794 Security Recommendation HY-SR-2: The hardware of the virtualized host should provide assistance for  
795 virtualization for instruction sets and memory management using MMU since the hardware support provides  
796 the following security assurances that cannot be guaranteed with purely software-based virtualization:  
797

- 798 • Better memory management controls can prevent attacks such as buffer overflow.
- 799 • The feature for re-mapping of DMA transfers in IOMMU provides better isolation of I/O devices.  
800 Further, the feature to directly assign I/O devices to a specific VM and enable direct access to those  
801 resources eliminates the need for providing emulated device drivers for that VM, thus reducing the size  
802 of trusted code.
- 803 • Guest OS code and hypervisor code execute in different processor modes, providing better isolation.
- 804 • Privilege-level isolation can provide better protection for device access mediation functions, and  
805 hardware-based memory protection can provide better VM-level protection.
- 806 • By supporting full virtualization, COTS versions of OSs can allow for easier patching and updating than  
807 having to perform the same operations on modified or ported versions of OSs that are the only types that  
808 can be run on para-virtualized platforms.
- 809 • Since many features of virtualization are now available in hardware, the size of the hypervisor code will  
810 be small, enabling better security attestation and verification.  
811

## 812 **4.2 VM Memory Allocation Scheduling Options**

813  
814 The hypervisor's memory scheduler is responsible for meeting the memory requirements for all workloads  
815 running in all VMs at all times. Like an OS, a typical hypervisor meets this requirement by using a  
816 combination of physical RAM and swap files called hypervisor kernel swap files. Further, a typical VM does  
817 not always require the entire memory it has been configured for. For these reasons, it is a viable overall  
818 virtualization configuration decision to have the combined configured memory of all VMs running on a  
819 virtualized host to exceed the total physical RAM, provided that there are no memory-sensitive applications  
820 running in VMs. However, over-commit—the ratio of the total configured memory of VMs to host physical  
821 RAM—should not be too high as it may result in performance degradation of certain VM workloads that  
822 require a significant amount of memory.

823  
824 Another factor affecting the availability of the virtualized host or hypervisor for certain workloads in a VM is  
825 the ratio of the physical RAM size to kernel swap file size that is maintained by the memory scheduler of the  
826 hypervisor. Since a low ratio will deny execution of certain workloads for certain VMs, there should be a  
827 configuration option available in the hypervisor to specify a guaranteed physical amount of RAM for each  
828 VM. Also, in order to avoid a situation in which a particular VM makes use of the physical RAM for its entire  
829 configured memory, there should be a feature to specify a limit on the guaranteed physical RAM. Finally,

830 there may be certain workloads that are time-sensitive, and the VMs hosting them should have some priority  
831 in getting the required memory resources compared to other running VMs. Therefore, a configuration option  
832 to specify a priority value for each VM should also exist.

833  
834 Based on the above issues relating to hypervisor memory scheduling, the following are the security  
835 recommendations:

836  
837 Security Recommendation HY-SR-3: The hypervisor should have configuration options to specify a  
838 guaranteed physical RAM for every VM that requires it, as well as a limit to this value, and a priority  
839 value for obtaining the required RAM resource in situations of contention among multiple VMs.  
840 Further, the over-commit feature that enables the total configured memory for all VMs to exceed the  
841 host physical RAM should be disabled by default.

842

### 843 **4.3 VM CPU Allocation Options**

844

845 The security goal in VM CPU allocation is to guarantee availability for all VMs. This can be achieved by  
846 proper use of configuration options dealing with the allocation of physical resources such as CPU cores and  
847 CPU clock cycles. For example, one of the configuration options commonly available is to set a minimum  
848 CPU requirement, or reservation, in terms of clock cycles. The architectural parameter to be observed here  
849 is that the number of VMs that can be deployed can be no more than the ratio of the total CPU clock cycles  
850 that the hypervisor host can offer to the average reservation required by each VM. In a scenario where the  
851 hypervisor host has 6000 MHz of CPU capacity and the average reservation for each VM is 1000 MHz, then  
852 no more than 6 VMs can be active in that hypervisor host. The reservation thus sets a lower bound  
853 (guaranteed) on the CPU clock cycles required for each VM. Similarly, there should be a feature to set an  
854 upper bound, or Limit, for the CPU cycles that each VM can use so that no single VM (sometimes a rogue  
855 or a compromised one) consumes all CPU resources of the host and denies services to other co-resident VMs.  
856 Further, to facilitate scheduling of hypervisor host CPU clock cycles in situations where multiple VMs  
857 require clock cycles above the lower bound but below the upper bound, there should be a feature to assign a  
858 priority score, or shares, to each VM. Summarizing the above desired features for ensuring fair share for all  
859 VMs deployed, the security recommendations for VM CPU allocation are as follows:

860

861 Security Recommendation HY-SR-4: The hypervisor should have robust configuration features for  
862 provisioning virtual resources to all hosted VMs such that it does not exceed a key physical resource  
863 (e.g., number of CPU cores).

864

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

869

## 870 5. SECURITY RECOMMENDATIONS FOR HY-BF2

871

872 Security recommendations for all three forms of device virtualization discussed in section 2.2.2 as well as for  
873 self-virtualized devices are provided in this section.

874

875 Security Recommendation HY-SR-6A (Emulation): Because of the complexity of emulating a  
876 hardware device through software, emulation, apart from suffering performance penalties, also  
877 increases the size of the TCB especially in situations where the guest OS has native device drivers and  
878 the device emulation code runs as a kernel module with the same privilege level as the hypervisor.  
879 Hence emulation should only be used where complexity is manageable (e.g., USB host controller).

880

881 Security Recommendation HY-SR-6B (Para-virtualization): In situations where para-virtualized  
882 device drivers are used in VMs, mediation of access to physical devices should be enabled by  
883 running back-end device drivers (which control the physical device attached to the hypervisor host)  
884 in a dedicated VM rather than in the hypervisor. This facilitates running the back-end device driver  
885 code at a privilege level lower than that of the hypervisor. Additionally, the hypervisor platform  
886 should include hardware support in the form of I/O Memory Management Unit (IOMMU) for  
887 validating and translating access from the driver domain's underlying hardware device to host  
888 memory. The specific IOMMU feature that is mandatory is DMA remapping where the DMA call  
889 from a device to guest physical address (GPA) must be translated to host physical address (HPA) and  
890 then checked whether the HPA address falls within the protection domain assigned to that device.  
891 Combining these mechanisms enables reducing the size of TCB as well as reducing the impact of  
892 faulty device or device driver behavior (restricted to device-driver VM as opposed to the hypervisor).

893

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

902

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

905

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

910

911 Security Recommendation HY-SR-8 (Device Usage): It should be possible to set resource limits for  
912 network bandwidth and I/O bandwidth (e.g., disk read/write speeds) for each VM to prevent denial-of-  
913 service (DOS) attacks. Additionally, the proper use of resource limits localizes the impact of a DOS to  
914 the VM or the cluster for which the resource limit is defined.

915

916

917

## 918 6. SECURITY RECOMMENDATIONS FOR HY-BF4

### 919 6.1 VM Image Management

920  
921 Since VM-based software (e.g., Guest OS, Middleware, and Applications) shares physical memory of the  
922 virtualized host with hypervisor software, it is no surprise that a VM is the biggest source of all attacks  
923 directed at the hypervisor. In operational virtualized environments, VMs are rarely created from scratch, but  
924 rather from VM Images. VM Images are templates used for creating running versions of VMs. An  
925 organization may have its own criteria for classifying the different VM Images it uses in its VM Library.  
926 Some commonly used criteria include: processor load (VM used for compute-intensive applications);  
927 memory load (VM used for memory-intensive applications, such as Database processing); and application  
928 sensitivity (VM running mission-critical applications utilizing mission-critical data). For each VM image  
929 type, the following practices must be followed to ensure that the resulting operational VMs are secure:  
930

- 931 • Documentation on the Gold Image for each VM Image type. A Gold Image is defined by a set of  
932 configuration variables associated with the VM Image. The configuration variables should include, at  
933 the minimum, the Guest OS make, version, patch level, date of creation, number of vCPU cores, and  
934 memory size.
- 935 • Each VM Image in the VM Image Library must have an associated digital signature.
- 936 • Access privileges to the VM Image Library must be controlled through a robust access control  
937 mechanism.
- 938 • Access to the server storing VM Images should have a secure protocol.  
939

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

941  
942 Security Recommendation HY-SR-9: Gold standard must be defined for VMs of all types, and VM  
943 Images that do not conform to the standard should not be allowed to be stored in the VM Image server  
944 or library. Images in the VM Image library should be periodically scanned for outdated OS versions and  
945 patches, which could result in a drift from the standard.  
946

947 Security Recommendation HY-SR-10: Every VM Image stored in the image server should have a digital  
948 signature attached to it as a mark of authenticity and integrity, signed using trustworthy, robust  
949 cryptographic keys.  
950

951 Security Recommendation HY-SR-11: Permissions for checking into and out of images from the  
952 VM Image library should be enforced through a robust access control mechanism and limited to an  
953 authorized set of administrators. In the absence of an access control mechanism, VM image files  
954 should be stored in encrypted devices that can only be opened or closed by a limited set of authorized  
955 administrators with passphrases of sufficient complexity.  
956

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

### 959 6.2 VM Live Migration

960 Live migration is a functionality present in all hypervisors, which enables a VM to be migrated or moved  
961 from one virtualized host to another while the guest OS and applications on it are still running. This  
962 functionality provides key benefits such as fault tolerance, load balancing, and host maintenance, upgrades,  
963 and patching. In live migration, the state of the guest OS on the source host must be replicated on the

964 destination host. This requires migrating memory content, processor state, storage (unless the two hosts share  
965 a common storage), and network state.

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

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

983

984 Security Recommendation HY-SR-13: During VM live migration, a secure authentication protocol  
985 must be employed; the credentials of the administrator performing the migration are passed only to the  
986 destination host; the migration of memory content and processor state takes place over a secure network  
987 connection; and a dedicated virtual network segment is used in both source and destination hosts for  
988 carrying this traffic.

### 989 **6.3 VM Monitoring and Security Policy Enforcement**

990

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

994

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

998

999 In a VM-based Security Monitoring and Intervention approach, software or a software-agent (i.e., a security  
1000 tool) is run inside a VM to monitor security-relevant events. This approach is similar to running host-based  
1001 IDS. The advantage of this approach is that it provides good visibility and good context analysis for the code  
1002 running within the VM. However, because of the dependency of the security tool on the underlying Guest  
1003 OS, any attack on the latter will also disable the function of the security tool, thus disabling the  
1004 countermeasure. Another disadvantage of running the security tool as a virtualized workload is the  
1005 performance impact it will have on itself and other application workloads running on that VM.

1006

1007 Virtual Network-based Security Monitoring can come in two forms:

1008

- 1009 (a) A dedicated security appliance for protecting each VM;



- 1010 (b) A security appliance that runs in the virtual network and can protect  
1011 multiple VMs inside the hypervisor host.

1012  
1013 The dedicated security appliance is deployed in the virtual network in front of the monitored VM and  
1014 monitors all traffic going in and out of the VM. The main disadvantage of this approach is that if the VM is  
1015 migrated to some other physical host, the dedicated appliance must be migrated as well.

1016  
1017 A generic security appliance deployed on a virtual network and configured to monitor multiple VMs may  
1018 have to be continuously reconfigured for the following reasons:

- 1019  
1020 • The set of VMs to be monitored is continuously in a state of flux since VMs are subject to migration  
1021 from one virtualized host to another due to load balancing, performance, and even security reasons.  
1022 • If virtual LANs (VLANs) are used to provide communication-level isolation among VMs, the  
1023 configuration of VLANs may undergo continuous change as the workload patterns shift on VMs.  
1024 This may require re-configuration of the network traffic mirroring capabilities to ensure that all  
1025 virtual network traffic flows through the monitoring tool impacting the overall performance of the  
1026 workloads inside that virtualized host.

1027  
1028 In a hypervisor-based security monitoring solution, the security tool that monitors and protects VMs (User  
1029 VMs) is run outside of the VMs hosting business applications in a special security-hardened VM. A security  
1030 tool designed and configured to run in this mode is called Security Virtual Appliance (SVA). The SVA obtains  
1031 its visibility into the state of a VM (e.g., CPU, registers, memory, and I/O devices) as well as network traffic  
1032 amongst VMs and between VMs and the hypervisor through the *virtual machine introspection* API of the  
1033 hypervisor. This is the preferable solution since:

- 1034  
1035 (a) It is not vulnerable to a flaw in the Guest OS.  
1036 (b) It is independent of the Virtual Network Configuration and does not have to be reconfigured every  
1037 time the virtual network configuration changes due to migration of VMs or change in connectivity  
1038 among VMs resident on the hypervisor host.

1039  
1040 Therefore, the security recommendations, with respect to creating the VM monitoring solution for the  
1041 protection of the hypervisor, are as follows:

1042  
1043 Security Recommendation HY-SR-14: There should be a mechanism for security monitoring, security  
1044 policy enforcement of VM operations, and detecting malicious processes running inside VMs and  
1045 malicious traffic going into and out of a VM. This monitoring and enforcement mechanism forms the  
1046 foundation for building Anti-Virus (AV) and Intrusion Detection & Prevention System (IDPS)  
1047 solutions.

1048  
1049 Security Recommendation HY-SR-15: Solutions for Security Monitoring and security policy  
1050 enforcement of VMs should be based outside of VMs and leverage the virtual machine introspection  
1051 capabilities of the hypervisor. Generally, such solutions involve running a security tool as a Security  
1052 Virtual Appliance (SVA) in a security-hardened or trusted VM.

1053  
1054 Security Recommendation HY-SR-16: All antimalware tools (e.g., virus checkers, firewalls, and IDPS)  
1055 running in the virtualized host should have the capability to perform autonomous signature or reference  
1056 file updates on a periodic basis.

1057  
1058

## 1059 **6.4 VM Configuration Management**

1060  
1061 The configuration of every VM should be monitored and managed throughout its lifecycle. In most instances,  
1062 this is accomplished using dedicated third-party tools in addition to native features that come with the  
1063 hypervisor. The desired features for these tools are provided in the form of security recommendation below:  
1064

1065 Security Recommendation HY-SR-17: VM configuration management tools should have the capability  
1066 to compile logs and alert administrators when configuration changes are detected in any VM that is  
1067 being monitored.

## 1068 **6.5 Fine-grained Administrative Privileges for VM Management**

1069  
1070 Having the ability to assign fine-grained administrative permissions for the virtualized infrastructure enables  
1071 the establishment of different administrative models and associated delegations. To see the need for granular  
1072 permissions, it would be helpful to look at some use-case scenarios for administrative operations in the  
1073 virtualized infrastructure:  
1074

1075 • VM Administration Use Case 1: A quality assurance group wants to set up a few virtual machines  
1076 with some definite profiles (resource quotas such as Memory, CPUs) to test some applications that  
1077 may soon go into production. In this situation, it may be useful for one or more administrators  
1078 assigned exclusively to the quality assurance group to be given administrative permissions on  
1079 specific virtual machines set up for testing purposes.  
1080

1081 • VM Administration Use Case 2: A capacity planner assigned the task of determining the operating  
1082 loads on various virtualized servers and the need for additional virtualized hosts may need  
1083 permission to view the list of virtual machines in each of the virtualized hosts but not permissions  
1084 to perform any administrative operations on those VMs. In this situation, it is desirable to have the  
1085 ability to grant view rights to the list of VMs in a virtualized host but deny the user the rights to  
1086 interact with any of the visible objects.  
1087

1088 • VM Administration Use Case 3: In virtualized data centers where VMs of different sensitivity  
1089 levels are run on the same virtualized host, an administrator who is given administrative privileges  
1090 at the hypervisor level should sometimes be prevented from accessing a specific VM because of  
1091 the sensitive nature of the workload (i.e., set of applications) running on that VM. The desired  
1092 capability in this scenario is to negate a permission, obtained through inheritance, for a specific  
1093 child object.  
1094

1095 • VM Administration Use Case 4: In some cases, assign permissions are needed for a group of  
1096 administrators controlling a set of VMs for a particular organizational division or department. A  
1097 corollary to this type of administrative entity is the need for a class of administrators wanting to  
1098 administer VMs running a particular type of work load (e.g., web server), irrespective of its location  
1099 within the organizational structure. This class of administrators may not require the entire set of  
1100 administrative functions on a VM but rather some arbitrary set of management functions such as  
1101 Configure CD Media, Configure Floppy Media, Console Interaction, Device Connection, Power  
1102 On, Power Off, Reset, or Suspend. This scenario calls for the capability to create custom roles that  
1103 can contain an arbitrary set of permissions relating to a VM as well as the ability to create a custom  
1104 object that contains an arbitrary set of VMs carrying a particular type of workload (e.g., web server).  
1105

1106 Summing up the capabilities required in all four administrative scenarios, the overall security recommendation  
1107 with required permission granularity is as follows:

1108  
1109 Security Recommendation HY-SR-18: The access control solution for VM administration should have a  
1110 granular capability, both at the permission assignment level and the object level (i.e., the specification of  
1111 the target of the permission can be a single VM or any logical grouping of VMs based on function or  
1112 location). In addition, the ability to deny permission to some specific objects within a VM group (e.g.,  
1113 VMs running workloads of a particular sensitivity level) in spite of having access permission to the VM  
1114 group should exist.  
1115

## 7. SECURITY RECOMMENDATIONS FOR HY-BF5

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

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

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

### 7.1 Centralized Administration

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

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

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

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

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

### 7.2 Securing the Management Network

To connect multiple VMs to each other and to the enterprise network in which the virtualized host is a node, the hypervisor allows for a software-defined communication fabric, or a virtual network, through its

1164 management console or command line interface (CLI). This capability can be provided by a dedicated  
1165 management VM or directly in the hypervisor kernel through a kernel module. The virtual network is a software-  
1166 defined artifact that resides entirely within the virtualized host and has the VMs residing inside it as its nodes.  
1167 The components of this virtual network are (a) the virtual network interface cards (vNICs) that are defined  
1168 for each VM and provide connection for each VM to the virtual network; (b) the virtual switches that provide  
1169 selective connectivity among VMs and whose configuration determines the topology of the virtual network;  
1170 and (c) the physical network interface cards (pNICs) of the virtualized hosts that provide connectivity for  
1171 VMs to the enterprise network.

1172  
1173 While considering the security impact of the virtual network, the following three main functions must be  
1174 considered:

- 1175  
1176 • Providing selective connectivity or isolation between groups of VMs belonging to different logical  
1177 groupings (e.g., different tenants in the case of an Infrastructure as a Service (IaaS) cloud service;  
1178 different application tiers such as Web Server or Database Server; or different Line of Business  
1179 applications of an enterprise)
- 1180 • Dedicating subnets for key functions such as (a) migration of VMs from one hypervisor host to  
1181 another for security or performance reasons, (b) attaching network-based storage devices, and (c)  
1182 fault Tolerant Logging
- 1183 • Providing access to the management interface in the management VM (a node of the virtual  
1184 network), which is used for performing key hypervisor baseline functions of VM lifecycle  
1185 management (HY-BF4) and Management of hypervisor platform (HY-BF5)

1186  
1187 Out of the three functionalities stated above, selective connectivity and isolation between groups of VMs is  
1188 required for providing security to the applications running on those VMs and therefore outside of the scope of  
1189 this document. The same criteria apply to dedicating subnets for network-based storage administration. We  
1190 have already discussed secure VM migration under VM lifecycle management in section 6. Hence, our focus  
1191 on virtual network configuration is limited to providing protection for the network interfaces used for  
1192 performing VM management and hypervisor administrative functions. A commonly adopted approach is to  
1193 allocate a dedicated physical network interface card (NIC) for handling management traffic, and, if that is not  
1194 feasible, a virtual network segment (vLAN ID) exclusively for it.

1195  
1196 Security Recommendation HY-SR-20: Protection for hypervisor host and software administration  
1197 functions should be ensured by allocating a dedicated physical NIC or, if that is not feasible, placing the  
1198 management interface of the hypervisor in a dedicated virtual network segment and enforcing traffic  
1199 controls using a firewall (e.g., designating the subnets in the enterprise network from which incoming  
1200 traffic into the management interface is allowed).

1201  
1202

## 8. SECURITY RECOMMENDATION SUMMARY

1203  
1204  
1205  
1206  
1207  
1208  
1209

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

1210  
1211  
1212  
1213  
1214  
1215

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

1216  
1217  
1218  
1219  
1220  
1221  
1222

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

1223  
1224  
1225  
1226  
1227  
1228  
1229

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

## Appendix A

1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278

Detailed descriptions of each of the five hypervisor baseline functions are provided below. As stated in the Introduction, these baseline functions are:

- HY-BF1: VM Process Isolation – Scheduling of VMs for execution, Management of the application processes running in VMs such as CPU and Memory Management, context switching between various processor states during the running of applications in VMs, etc. If DMA capable devices are used in the hypervisor host, memory access to those devices need to be controlled as well. However, this function is considered under HY-BF2 since it pertains to devices mediation.
- HY-BF2: Devices Mediation & Access Control – Mediates access to all devices (e.g., Network Interface Card (NIC), storage device such as IDE drive, etc.)
- HY-BF3: Direct Execution of commands from Guest VMs – Certain commands from Guest OSs are executed directly by the hypervisor instead of being triggered through interrupts and context switching. This function applies to hypervisors that have implemented para-virtualization instead of full virtualization.
- HY-BF4: VM Lifecycle Management – This involves all functions from creation and management of VM images, control of VM states (Start, Pause, Stop, etc.), VM migration, VM monitoring and policy enforcement.
- HY-BF5: Management of hypervisor platform– This involves defining some artifacts and setting values for various configuration parameters in hypervisor software modules including those for configuration of a Virtual Network inside the hypervisor.

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

### A.1 HY-BF1 (VM Process Isolation)

Scheduling of VMs for execution, Management of the application processes running in VMs such as CPU and Memory Management, context switching between various processor states during the running of applications in VMs, etc. In order to ensure VM process isolation, memory access from DMA capable devices needs to be under hypervisor control as well (e.g., via IOMMU). However, this function is considered under HY-BF2 since it pertains to devices mediation.

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

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

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

1279 In the para-virtualization approach, the hypervisor presents to the guest an interface of an artificial device that  
1280 has no corresponding hardware counterpart. This enables special, simplified hypervisor-aware I/O drivers  
1281 (called para-virtualized drivers) to be installed in the guest. The calls from these para-virtualized device  
1282 drivers in guest VMs are handled by another device driver (called back-end driver) which directly interfaces  
1283 with the physical device and mediates access to that physical device from para-virtualized guests. In some  
1284 instances, the calls from para-virtualized guest drivers are handled directly by the hypervisor through its  
1285 hypercall interface (the corresponding calls are called hypercalls). Analysis of threats due to these hypercalls  
1286 is provided in the next subsection.

1287  
1288 The third approach to device virtualization, the passthrough approach (or direct device assignment), is  
1289 deployed for situations where a VM needs exclusive access to a device (e.g., NIC, disk controller, HBA, USB  
1290 controller, serial port, firewire controller, soundcard, etc) for performance reasons so as to be devoid of the  
1291 overhead due to emulation. Since generally this is required for PCI devices, this is also called as PCI  
1292 Passthrough. Since many of these devices have a memory-mapped interface, they can read or write directly to  
1293 or from main memory and are also called Direct Memory Access (DMA) capable devices. To provide  
1294 exclusive access to a DMA capable device for a VM, the memory pages of the device are mapped into guest  
1295 VM's address space. The following is the threat due to DMA capable devices.

1296  
1297 Apart from three types of device virtualization described above, hypervisor hosts can support self-virtualizing  
1298 hardware devices. These devices have interfaces that can export a set of virtual functions (VFs) corresponding  
1299 to a physical function (PF). The hypervisor then can assign these VFs to multiple guest VMs, while it retains  
1300 control of the PF. These devices conform to Single Root I/O Virtualization (SR-IOV) specification and thus  
1301 enable DMA capable devices to be shared among VMs (as virtualization and multiplexing are done by the  
1302 devices themselves) instead of being dedicated to a single VM as in passthrough mode.

1303

### 1304 **A.3 HY-BF3 (Direct Execution of commands from Guest VMs):**

1305

1306 Certain commands from Guest OSs are executed directly by the hypervisor instead of being triggered  
1307 through interrupts and context switching. These commands are called hypercalls and are supported by a  
1308 special interface in the hypervisor. This function applies only to hypervisors that have implemented para-  
1309 virtualization instead of full virtualization.

1310

### 1311 **A.4 HY-BF4 (VM Lifecycle Management)**

1312

1313 This encompasses all administrative operations on VMs throughout its life cycle. They include but not limited  
1314 to:

1315

- 1316 • Creation of VMs conforming to a standard image, ensuring integrity of images and secure storage and  
1317 retrieval of images; provisioning images with appropriate vCPU, RAM, network, and storage
- 1318 • Migration of VMs from one hypervisor host to another
- 1319 • Monitoring of VM execution and traffic flows into and out of VMs & overall configuration  
1320 management
- 1321 • Fine-grained access control for VM administration including the basic operations that alter the state of  
1322 VMs – Start, Pause, Stop etc.
- 1323 • Access control and management of snapshots

1324

1325 Management tasks are enabled using a management daemon which provides network interfaces. *These*  
1326 *interfaces are generally implemented not as part of the hypervisor kernel modules but on a privileged VM*  
1327 *(management VM) that is booted up as an integral part of the hypervisor platform boot process.*

1328



**A.5 HY-BF5 (Management of hypervisor platform)**

1329  
1330  
1331 These tasks include those that are involved in the configuration of the hypervisor host (virtualized host) and  
1332 the hypervisor software itself. Important tasks include: provisioning of VMs to hypervisor hosts, creating  
1333 and managing hypervisor clusters and configuration of the virtual network inside the hypervisor host. A  
1334 virtual network is a software-defined network inside the hypervisor host that enables connectivity among  
1335 VMs, as well as connectivity of VMs to external network (e.g., LAN, WAN, etc.).

1336  
1337  
1338

**Appendix B: Traceability of Security Recommendation to Hypervisor  
Baseline Functions**

NO	SECURITY RECOMMENDATION	BASELINE FUNCTION
HY-SR-1	<p><i>The hypervisor that is launched should be part of a platform and an overall infrastructure that contains: (a) Hardware that supports a MLE with standards-based cryptographic measurement capability and storage device and (b) Attestation process that should contain capabilities to take advantage of these to provide a chain of trust starting from the Hardware to all Hypervisor components. The measured elements (components) should include at the minimum the following: the core kernel, kernel support modules, device drivers and the hypervisor’s native management applications (for VM Lifecycle Management and Management of Hypervisor). The chain of trust should provide assurance that all measured components have not been tampered with and that their versions are correct (i.e., overall boot integrity). If the chain of trust is to be extended to guest VMs, the hypervisor should provide a virtual interface to the hardware-based MLE.</i></p>	N/A
HY-SR-2	<p><i>The hardware of the virtualized host should provide assistance for virtualization for instruction sets and memory management using MMU since the hardware support provides the following security assurances that cannot be guaranteed with purely software-based virtualization:</i></p> <ul style="list-style-type: none"> <li>• <i>Better memory management controls can prevent attacks such as buffer overflow.</i></li> <li>• <i>The feature for re-mapping of DMA transfers in IOMMU provides better isolation of I/O devices. Further, the feature to directly assign I/O devices to a specific VM and enable direct access to those resources eliminates the need for providing emulated device drivers for that VM, thus reducing the size of trusted code.</i></li> <li>• <i>Guest OS code and hypervisor code execute in different processor modes, providing better isolation.</i></li> <li>• <i>Privilege-level isolation can provide better protection for device access mediation functions, and hardware-based memory protection can provide better VM-level protection.</i></li> <li>• <i>By supporting full virtualization, COTS versions of OSs can allow for easier patching and updating than having to perform the same operations on modified or ported versions of OSs that are the only types that can be run on para-virtualized platforms.</i></li> <li>• <i>Since many features of virtualization are now available in hardware, the size of the hypervisor code will be small, enabling better security attestation and verification.</i></li> </ul>	HY-BF1 (VM Process Isolation)

HY-SR-3	<p><i>The hypervisor should have configuration options to specify a guaranteed physical RAM for every VM (that requires it) along with a limit to this value, and to specify a priority value for obtaining the required RAM resource in situations of contention among multiple VMs. Further, the over-commit feature (if available) that enables the total configured memory for all VMs to exceed the host physical RAM should be disabled by default.</i></p>	HY-BF1 (VM Process Isolation)
HY-SR-4	<p><i>The hypervisor should have robust configuration features for provisioning virtual resources to all hosted VMs in a way that it does not exceed a key physical resource such as number of CPU cores.</i></p>	HY-BF1 (VM Process Isolation)
HY-SR-5	<p><i>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.</i></p>	HY-BF1 (VM Process Isolation)
HY-SR-6A, HY-SR-6B, HY-SR-6C	<p><u><i>Security Recommendation HY-SR-6A (Emulation):</i></u> <i>Because of the complexity of emulating a hardware device through software, emulation, apart from suffering performance penalties, also increases the size of the TCB especially in situations where the guest OS has native device drivers and the device emulation code runs as a kernel module with the same privilege level as the hypervisor. Hence emulation should only be used where complexity is manageable (e.g., USB host controller).</i></p> <p><u><i>Security Recommendation HY-SR-6B (Para-virtualization):</i></u> <i>In situations where para-virtualized device drivers are used in VMs, mediation of access to physical devices should be enabled by running back-end device drivers (which control the physical device attached to the hypervisor host) in a dedicated VM rather than in the hypervisor. This facilitates running the back-end device driver code at a privilege level lower than that of the hypervisor. Additionally, the hypervisor platform should include hardware support in the form of I/O Memory Management Unit (IOMMU) for validating and translating access from the driver domain’s underlying hardware device to host memory. The specific IOMMU feature that is mandatory is DMA remapping where the DMA call from a device to guest physical address (GPA) must be translated to host physical address (HPA) and then checked whether the HPA address falls within the protection domain assigned to that device. Combining these mechanisms enables reducing the size of TCB as well as reducing the impact of faulty device or device driver behavior (restricted to device-driver VM as opposed to the hypervisor).</i></p> <p><u><i>Security Recommendation HY-SR-6C (Passthrough or self-virtualizing hardware devices):</i></u> <i>For situations, where VMs needs to be given dedicated access to DMA capable devices, the hypervisor platform should include hardware support in the form of I/O Memory Management Unit (IOMMU) for validating and translating all device access to host memory. This recommendation also applies to use of virtualization-enabled hardware devices (based on SR-IOV specification). The specific IOMMU feature that is mandatory is DMA</i></p>	HY-BF2 (Devices Mediation & Access Control)

	<i>remapping where the DMA call from a device to guest physical address (GPA) must be translated to host physical address (HPA) and then checked whether the HPA address falls within the protection domain assigned to that device.</i>	
HY-SR-7	<i>It should be possible to set up an Access Control List (ACL) to restrict access of each VM process to only the devices assigned to that VM. To enable this, the hypervisor configuration should support a feature to mark (label) VMs (semantically a set of tasks) and/or has a feature to specify a whitelist (list of allowable) of devices for each VM.</i>	HY-BF2 (Devices Mediation & Access Control)
HY-SR-8	<i>It should be possible to set resource limits for network bandwidth and I/O bandwidth (e.g., disk read/write speeds) for each VM to prevent denial of service (DOS) attacks. Further, the proper use of resource limits, localizes the impact of a DOS to the VM or the cluster for which the resource limit is defined.</i>	HY-BF2 (Devices Mediation & Access Control)
HY-SR-9	<i>Gold-standard must be defined for VMs of all types and VM Images not conforming to the standard should not be allowed to be stored in the VM Image server/library. Further images in the VM Image library should be periodically scanned for OS versions and patches going out of date and thus have drifted from the standard.</i>	HY-BF4 (VM Lifecycle Management)
HY-SR-10	<i>Every VM Image stored in the image server should have a digital signature attached to it as a mark of authenticity and integrity, signed using trustworthy, robust cryptographic keys.</i>	HY-BF4 (VM Lifecycle Management)
HY-SR-11	<i>Permissions for checking in to and checking out images from VM Image library should be enforced through a robust access control mechanism and limited to an authorized set of administrators. In the absence of an access control mechanism, VM image files should be stored in encrypted devices that can only be opened/closed by a limited set of authorized administrators with</i>	HY-BF4 (VM Lifecycle Management)
HY-SR-12	<i>Access to the server storing VM images should always be through a secure protocol such as TLS.</i>	HY-BF4 (VM Lifecycle Management)
HY-SR-13	<i>During VM live migration, care should be taken to see that a secure authentication protocol is used for performing live migration, that the credentials of the administrator performing the migration is passed only to the destination host, the migration of memory content and processor state takes place over a secure network connection and a dedicated virtual network segment is used in both source and destination hosts for carrying this traffic.</i>	HY-BF4 (VM Lifecycle Management)
HY-SR-14	<i>There should be a mechanism for security monitoring and security policy enforcement of VM operations –malicious processes running inside VMs and malicious traffic going in and out of a VM. This monitoring and enforcement mechanism forms the foundation for building Anti-Virus (AV) and Intrusion Detection &amp; Prevention System (IDPS) solutions.</i>	HY-BF4 (VM Lifecycle Management)

HY-SR-15	<i>Solutions for Security Monitoring and security policy enforcement of VMs should be based “outside of VMs” and should leverage the virtual machine introspection capabilities of the hypervisor. Generally, such solutions involve running a security tool as a Security Virtual Appliance (SVA) in a security-hardened or trusted VM..</i>	HY-BF4 (VM Lifecycle Management)
HY-SR-16	<i>All antimalware tools (virus checkers, firewalls and IDPS) running in the virtualized host should have the capability to perform autonomous signature or reference file updates on a periodic basis.</i>	HY-BF4 (VM Lifecycle Management)
HY-SR-17	<i>VM configuration management tools should have the capability to compile logs and alert administrators when configuration changes are detected in any VM that is being monitored.</i>	HY-BF4 (VM Lifecycle Management)
HY-SR-18	<i>The access control solution for VM administration should have the granular capability both at the permission assignment level as well as at the object level (i.e., the specification of the target of the permission can be a single VM or any logical grouping of VMs - based on function or location). In addition, the ability to deny permission to some specific objects within a VM group (e.g., VMs running workloads of a particular sensitivity level) despite having access permission to the VM group</i>	HY-BF4 (VM Lifecycle Management)
HY-SR-19	<i>The administration of all hypervisor installations in the enterprise should be performed centrally using an enterprise virtualization management system (EVMS). Further enterprise gold-standard hypervisor configurations for different types of workloads and clusters must managed (enforced) through EVMS. The gold-standard configurations should at the minimum cover the following aspects – CPU, Memory, Storage, Network bandwidth and Host OS hardening (if required).</i>	HY-BF5 (Management of hypervisor Platform)
HY-SR-20	<i>Protection for Hypervisor Host &amp; Software administration functions should be ensured by allocating a dedicated physical NIC, or if that is not feasible, by placing the management interface of the hypervisor in a dedicated virtual network segment and enforcing traffic controls using a firewall (e.g., designating the subnets in the enterprise network from which incoming traffic into the management interface is allowed).</i>	HY-BF5 (Management of hypervisor Platform)

**Appendix C: Glossary**

1340  
1341  
1342 Full Virtualization: A form of Virtualization in which the hypervisor presents virtualized resources  
1343 that reflect the architecture of the underlying hardware and hence unmodified guest OSs can be run.  
1344  
1345 Guest Operating System (OS): The operating system component of the execution stack of a  
1346 Virtual Machine (see below), others being Virtual Hardware, Middleware and Applications.  
1347  
1348 Hypervisor: A software built using a specialized kernel of an OS, along with supporting kernel modules  
1349 that provides isolation for various execution stacks represented by Virtual Machines (see below).  
1350  
1351 Virtualized Host: The physical host on which the virtualization software such as the Hypervisor  
1352 is installed. Usually, the virtualized host will contain a special hardware platform that assists  
1353 virtualization - specifically Instruction Set and Memory virtualization.  
1354  
1355 Virtual Machine (VM): A software-defined complete execution stack consisting of virtualized  
1356 hardware, operating system (guest OS), and applications.  
1357  
1358 Virtualization: A methodology for emulation or abstraction of hardware resources that enables  
1359 complete execution stacks including software applications to run on it.  
1360

## Appendix D: References

- 1361  
1362  
1363 1. *Mastering VMware vSphere 5.5*, Scott Lowe et al., Wiley Publishing Incorporated (2013)  
1364 2. *Running Xen: A Hands-On Guide to the Art of Virtualization*, J.N. Matthews et al., Prentice Hall  
1365 (2008)  
1366 3. *Building the Infrastructure for Cloud Security: A Solutions View*, R.Yeluri, and E.Castro-Leon,  
1367 Apress Media/Springer Science (2014)  
1368 4. *Trusted Platform Module (TPM) Main Specification:*  
1369 [http://www.trustedcomputinggroup.org/resources/tpm\\_main\\_specification](http://www.trustedcomputinggroup.org/resources/tpm_main_specification)  
1370 5. S.Shirinbab, L. Lundberg and D. Ilie, *Performance Comparison of KVM, VMware and Xenserver*  
1371 *using a Large Telecommunication Application, Proceedings of the Fifth International Conference*  
1372 *on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING)*, 2014. [http://bth.diva-](http://bth.diva-portal.org/smash/record.jsf?pid=diva2%3A834000)  
1373 [portal.org/smash/record.jsf?pid=diva2%3A834000](http://bth.diva-portal.org/smash/record.jsf?pid=diva2%3A834000)  
1374 6. E. Bugnion, J. Nieh and D. Tsafirir, *Hardware and Software Support for Virtualization*,  
1375