Protection Profile for Virtualization Systemss

Version: 1.1.1

2021-08-19

National Information Assurance Partnership

Revision History

Version	Date	Comment
1.0	2016-11-17	Initial Publication
1.1	2021-06-14	Incorporate TDs, Reference TLS Package, Add Equivalency Guidelines, etc.
1.1.1	2021-08-19	Errata release. Fixes a few typos.

Contents

1 Introduction

1.1 Overview

The scope of this Protection Profile (PP) is to describe the security functionality of virtualization technologies in terms of [CC] and to define security functional and assurance requirements for such products. This PP is not complete in itself, but rather provides a set of requirements that are common to the PP-Modules for Server Virtualization and for Client Virtualization. These capabilities have been broken out into this generic 'base' PP due to the high degree of similarity between the two product types.

Due to the increasing prevalence of virtualization technology in enterprise computing environments and the shift to cloud computing, it is essential to ensure that this technology is implemented securely in order to mitigate the risk introduced by sharing multiple computers and their resident data across a single physical system.

1.2 Terms

The following sections list Common Criteria and technology terms used in this document.

1.2.1 Common Criteria Terms

Assurance	Grounds for confidence that a TOE meets the SFRs [CC].
Base Protection Profile (Base-PP)	Protection Profile used as a basis to build a PP-Configuration.
Collaborative Protection Profile (cPP)	A Protection Profile developed by international technical communities and approved by multiple schemes.
Common Criteria (CC)	Common Criteria for Information Technology Security Evaluation (International Standard ISO/IEC 15408).
Common Criteria Testing Laboratory	Within the context of the Common Criteria Evaluation and Validation Scheme (CCEVS), an IT security evaluation facility accredited by the National Voluntary Laboratory Accreditation Program (NVLAP) and approved by the NIAP Validation Body to conduct Common Criteria-based evaluations.
Common Evaluation Methodology (CEM)	Common Evaluation Methodology for Information Technology Security Evaluation.
Extended Package (EP)	A deprecated document form for collecting SFRs that implement a particular protocol, technology, or functionality. See Functional Packages.
Functional Package (FP)	A document that collects SFRs for a particular protocol, technology, or functionality.
Operational Environment (OE)	Hardware and software that are outside the TOE boundary that support the TOE functionality and security policy.
Protection Profile (PP)	An implementation-independent set of security requirements for a category of products.
Protection Profile Configuration (PP-Configuration)	A comprehensive set of security requirements for a product type that consists of at least one Base-PP and at least one PP-Module.
Protection Profile Module (PP-Module)	An implementation-independent statement of security needs for a TOE type complementary to one or more Base-PPs.
Security Assurance Requirement (SAR)	A requirement to assure the security of the TOE.
Security	A requirement for security enforcement by the TOE.

Functional Requirement (SFR)	
Security Target (ST)	A set of implementation-dependent security requirements for a specific product.
Target of Evaluation (TOE)	The product under evaluation.
TOE Security Functionality (TSF)	The security functionality of the product under evaluation.
TOE Summary Specification (TSS)	A description of how a TOE satisfies the SFRs in an ST.

1.2.2 Technical Terms

Administrator	Administrators perform management activities on the VS. These management functions do not include administration of software running within Guest VMs, such as the Guest OS. Administrators need not be human as in the case of embedded or headless VMs. Administrators are often nothing more than software entities that operate within the VM.
Auditor	Auditors are responsible for managing the audit capabilities of the TOE. An Auditor may also be an Administrator. It is not a requirement that the TOE be capable of supporting an Auditor role that is separate from that of an Administrator.
Domain	A Domain or Information Domain is a policy construct that groups together execution environments and networks by sensitivity of information and access control policy. For example, classification levels represent information domains. Within classification levels, there might be other domains representing communities of interest or coalitions. In the context of a VS, information domains are generally implemented as collections of VMs connected by virtual networks. The VS itself can be considered an Information Domain, as can its Management Subsystem.
Guest Network	See Operational Network.
Guest Operating System (OS)	An operating system that runs within a Guest VM.
Guest VM	A Guest VM is a VM that contains a virtual environment for the execution of an independent computing system. Virtual environments execute mission workloads and implement customer-specific client or server functionality in Guest VMs, such as a web server or desktop productivity applications.
Helper VM	A Helper VM is a VM that performs services on behalf of one or more Guest VMs, but does not qualify as a Service VM—and therefore is not part of the VMM. Helper VMs implement functions or services that are particular to the workloads of Guest VMs. For example, a VM that provides a virus scanning service for a Guest VM would be considered a Helper VM. For the purposes of this document, Helper VMs are considered a type of Guest VM, and are therefore subject to all the same requirements, unless specifically stated otherwise.
Host Operating System (OS)	An operating system onto which a VS is installed. Relative to the VS, the Host OS is part of the Platform. There need not be a Host OS, but often VSes employ a Host OS or Control Domain to support guest access to host resources. Sometimes these domains are themselves encapsulated within VMs.
Hypercall	An API function that allows VM-aware software running within a VM to invoke VMM functionality.
Hypervisor	The Hypervisor is part of the VMM. It is the software executive of the physical platform of a VS. A Hypervisor's primary function is to mediate access to all CPU and memory resources, but it is also responsible for either the direct management or the delegation of the management of all other hardware devices on the hardware platform.
Information Domain	See Domain.
Introspection	A capability that allows a specially designated and privileged domain to have visibility into another domain for purposes of anomaly detection or monitoring.

Management Network	A network, which may have both physical and virtualized components, used to manage and administer a VS. Management networks include networks used by VS Administrators to communicate with management components of the VS, and networks used by the VS for communications between VS components. For purposes of this document, networks that connect physical hosts and backend storage networks for purposes of VM transfer or backup are considered management networks.
Management Subsystem	Components of the VS that allow VS Administrators to configure and manage the VMM, as well as configure Guest VMs. VMM management functions include VM configuration, virtualized network configuration, and allocation of physical resources.
Operational Network	An Operational Network is a network, which may have both physical and virtualized components, used to connect Guest VMs to each other and potentially to other entities outside of the VS. Operational Networks support mission workloads and customer-specific client or server functionality. Also called a “Guest Network.”
Physical Platform	The hardware environment on which a VS executes. Physical platform resources include processors, memory, devices, and associated firmware.
Platform	The hardware, firmware, and software environment into which a VS is installed and executes.
Service VM	A Service VM is a VM whose purpose is to support the Hypervisor in providing the resources or services necessary to support Guest VMs. Service VMs may implement some portion of Hypervisor functionality, but also may contain important system functionality that is not necessary for Hypervisor operation. As with any VM, Service VMs necessarily execute without full Hypervisor privileges—only the privileges required to perform its designed functionality. Examples of Service VMs include device driver VMs that manage access to physical devices, VMs that provide life-cycle management and provisioning of Hypervisor and Guest VMs, and name-service VMs that help establish communication paths between VMs.
System Security Policy (SSP)	The overall policy enforced by the VS defining constraints on the behavior of VMs and users.
User	Users operate Guest VMs and are subject to configuration policies applied to the VS by Administrators. Users need not be human as in the case of embedded or headless VMs, users are often nothing more than software entities that operate within the VM.
Virtual Machine (VM)	A Virtual Machine is a virtualized hardware environment in which an operating system may execute.
Virtual Machine Manager (VMM)	A VMM is a collection of software components responsible for enabling VMs to function as expected by the software executing within them. Generally, the VMM consists of a Hypervisor, Service VMs, and other components of the VS, such as virtual devices, binary translation systems, and physical device drivers. It manages concurrent execution of all VMs and virtualizes platform resources as needed.
Virtualization System (VS)	A software product that enables multiple independent computing systems to execute on the same physical hardware platform without interference from one another. For the purposes of this document, the VS consists of a Virtual Machine Manager (VMM), Virtual Machine abstractions, a management subsystem, and other components.

1.3 Compliant Targets of Evaluation

A Virtualization System (VS) is a software product that enables multiple independent computing systems to execute on the same physical hardware platform without interference from one another. A VS creates a virtualized hardware environment (virtual machines or VMs) for each instance of an operating system permitting these environments to execute concurrently while maintaining isolation and the appearance of exclusive control over assigned computing resources. For the purposes of this document, the VS consists of a Virtual Machine Manager (VMM), Virtual Machine (VM) abstractions, a management subsystem, and other components.

A VMM is a collection of software components responsible for enabling VMs to function as expected by the software executing within them. Generally, the VMM consists of a Hypervisor, Service VMs, and other components of the VS, such as virtual devices, binary translation systems, and physical device drivers. It manages concurrent execution of all VMs and virtualizes platform resources as needed.

The Hypervisor is the software executive of the physical platform of a VS. A hypervisor operates at the highest CPU privilege level and manages access to all of the physical resources of the hardware platform. It exports a well-defined, protected interface for access to the resources it manages. A Hypervisor’s primary function is to mediate access to all CPU and memory resources, but it is also responsible for either the direct management or the delegation of the management of all other hardware devices on the hardware platform. This document does not specify any Hypervisor-specific requirements, though many VMM requirements

would naturally apply to a Hypervisor.

A Service VM is a VM whose purpose is to support the Hypervisor in providing the resources or services necessary to support Guest VMs. Service VMs may implement some portion of Hypervisor functionality, but also may contain important system functionality that is not necessary for Hypervisor operation. As with any VM, Service VMs necessarily execute without full Hypervisor privileges—only the privileges required to perform its designed functionality. Examples of Service VMs include device driver VMs that manage access to physical devices, VMs that provide life-cycle management and provisioning of Hypervisor and Guest VMs, and name-service VMs that help establish communication paths between VMs.

A Guest VM is a VM that contains a virtual environment for the execution of an independent computing system. Virtual environments execute mission workloads and implement customer-specific client or server functionality in Guest VMs, such as a web server or desktop productivity applications. A Helper VM is a VM that performs services on behalf of one or more Guest VMs, but does not qualify as a Service VM—and therefore is not part of the VMM. Helper VMs implement functions or services that are particular to the workloads of Guest VMs. For example, a VM that provides a virus scanning service for a Guest VM would be considered a Helper VM. The line between Helper and Service VMs can easily be blurred. For instance, a VM that implements a cryptographic function—such as an in-line encryption VM—could be identified as either a Service or Helper VM depending on the particular virtualization solution. If the cryptographic functions are necessary only for the privacy of Guest VM data in support of the Guest's mission applications, it would be proper to classify the encryption VM as a Helper. But if the encryption VM is necessary for the VMM to isolate Guest VMs, it would be proper to classify the encryption VM as a Service VM. For the purposes of this document, Helper VMs are subject to all requirements that apply to Guest VMs, unless specifically stated otherwise.

1.3.1 TOE Boundary

Figure 1 shows a greatly simplified view of a generic Virtualization System and Platform. TOE components are displayed in Red. Non-TOE components are in Blue. The Platform is the hardware, firmware, and software onto which the VS is installed. The VMM includes the Hypervisor, Service VMs, and VM containers, but not the software that runs inside Guest VMs or Helper VMs. The Management Subsystem is part of the TOE, but may or may not be part of the VMM.

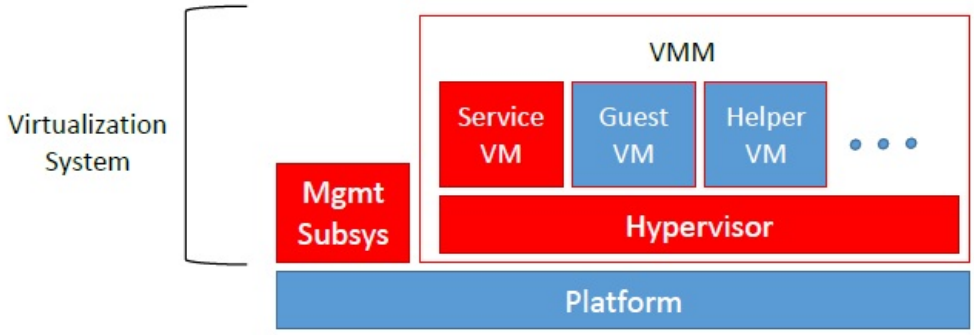


Figure 1: Virtualization System and Platform

For purposes of this Protection Profile, the Virtualization System is the TOE, subject to some caveats. The Platform onto which the VS is installed (which includes hardware, platform firmware, and Host Operating System) is not part of the TOE. Software installed with the VS on the Host OS specifically to support the VS or implement VS functionality is part of the TOE. General purpose software—such as device drivers for physical devices and the Host OS itself—is not part of the TOE, regardless of whether it supports VS functionality or runs inside a Service VM or control domain. Software that runs within Guest and Helper VMs is not part of the TOE.

In general, for virtualization products that are installed onto "bare metal," the entire set of installed components constitute the TOE, and the hardware constitutes the Platform. Also in general, for products that are hosted by or integrated into a commodity operating system, the components installed expressly for implementing and supporting virtualization are in the TOE, and the Platform comprises the hardware and Host OS.

1.3.2 Requirements Met by the Platform

Depending on the way the VS is installed, functions tested under this PP may be implemented by the TOE or by the Platform. There is no difference in the testing required whether the function is implemented by the TOE or by the Platform. In either case, the tests determine whether the function being tested provides a level of confidence acceptable to meet the goals of this Profile with respect to a particular product and platform. The equivalency guidelines are intended in part to address this TOE vs. Platform distinction, and to ensure that confidence in the evaluation results do not erode between instances of equivalent products on equivalent platforms—and also, of course, to ensure that the appropriate testing is done when the distinction is significant.

1.3.3 Scope of Certification

Successful evaluation of a Virtualization System against this profile does not constitute or imply successful evaluation of any Host Operating System or Platform—no matter how tightly integrated with the VS. The Platform, including any Host OS, supports the VS through provision of services and resources. Specialized VS components installed on or in a Host OS to support the VS may be considered part of the TOE. But general-

purpose OS components and functions—whether or not they support the VS—are not part of the TOE, and thus are not evaluated under this PP.

1.3.4 Product and Platform Equivalence

The tests in this Protection Profile must be run on all product versions and Platforms with which the Vendor would like to claim compliance—subject to this Profile's equivalency guidelines (see Appendix Appendix H - Equivalency Guidelines).

1.4 Use Cases

This Base-PP does not define any use cases for virtualization technology. Client Virtualization and Server Virtualization products have different use cases and so these are defined in their respective PP-Modules.

2 Introduction

The purpose of equivalence in PP-based evaluations is to find a balance between evaluation rigor and commercial practicability--to ensure that evaluations meet customer expectations while recognizing that there is little to be gained from requiring that every variation in a product or platform be fully tested. If a product is found to be compliant with a PP on one platform, then all equivalent products on equivalent platforms are also considered to be compliant with the PP.

A Vendor can make a claim of equivalence if the Vendor believes that a particular instance of their Product implements PP-specified security functionality in a way equivalent to the implementation of the same functionality on another instance of their Product on which the functionality was tested. The Product instances can differ in version number or feature level (model), or the instances may run on different platforms. Equivalency can be used to reduce the testing required across claimed evaluated configurations. It can also be used during Assurance Maintenance to reduce testing needed to add more evaluated configurations to a certification.

These equivalency guidelines do not replace Assurance Maintenance requirements or NIAP Policy #5 requirements for CAVP certificates. Nor may equivalency be used to leverage evaluations with expired certifications.

This document provides guidance for determining whether Products and Platforms are equivalent for purposes of evaluation against the Protection Profile for Virtualization (VPP) when instantiated with either the Client or Server PP-Module.

Equivalence has two aspects:

1. ***Product Equivalence:*** Products may be considered equivalent if there are no differences between Product Models and Product Versions with respect to PP-specified security functionality.
2. ***Platform Equivalence:*** Platforms may be considered equivalent if there are no significant differences in the services they provide to the Product--or in the way the platforms provide those services--with respect to PP-specified security functionality.

The equivalency determination is made in accordance with these guidelines by the Validator and Scheme using information provided by the Evaluator/Vendor.

3 Conformance Claims

Conformance Statement

A Security Target must claim exact conformance to this Protection Profile, as defined in the CC and CEM addenda for Exact Conformance, Selection-Based SFRs, and Optional SFRs (dated May 2017).

The following PPs and PP-Modules are allowed to be specified in a PP-Configuration with this PP-Module with this PP.

- PP-Module for Client Virtualization Systems, Version 1.1
- PP-Module for Server Virtualization Systems, Version 1.1

CC Conformance Claims

This PP is conformant to Parts 2 (extended) and 3 (extended) of Common Criteria Version 3.1, Release 5 [CC].

PP Claims

This PP does not claim conformance to any other PP.

Package Claims

This PP is Functional Package for TLS-conformant. This PP is Functional Package for Secure Shell-conformant.

4 Security Problem Description

4.1 Threats

T.DATA_LEAKAGE

It is a fundamental property of VMs that the domains encapsulated by different VMs remain separate unless data sharing is permitted by policy. For this reason, all Virtualization Systems shall support a policy that prohibits information transfer between VMs.

It shall be possible to configure VMs such that data cannot be moved between domains from VM to VM, or through virtual or physical network components under the control of the VS. When VMs are configured as such, it shall not be possible for data to leak between domains, neither by the express efforts of software or users of a VM, nor because of vulnerabilities or errors in the implementation of the VMM or other VS components.

If it is possible for data to leak between domains when prohibited by policy, then an adversary on one domain or network can obtain data from another domain. Such cross-domain data leakage can, for example, cause classified information, corporate proprietary information, or personally identifiable information to be made accessible to unauthorized entities.

T.UNAUTHORIZED_UPDATE

It is common for attackers to target outdated versions of software containing known flaws. This means it is extremely important to update VS software as soon as possible when updates are available. But the source of the updates and the updates themselves must be trusted. If an attacker can write their own update containing malicious code they can take control of the VS.

T.UNAUTHORIZED_MODIFICATION

System integrity is a core security objective for Virtualization Systems. To achieve system integrity, the integrity of each VMM component must be established and maintained. Malware running on the platform must not be able to undetectably modify VS components while the system is running or at rest. Likewise, malicious code running within a virtual machine must not be able to modify Virtualization System components.

T.USER_ERROR

If a Virtualization System is capable of simultaneously displaying VMs of different domains to the same user at the same time, there is always the chance that the user will become confused and unintentionally leak information between domains. This is especially likely if VMs belonging to different domains are indistinguishable. Malicious code may also attempt to interfere with the user's ability to distinguish between domains. The VS must take measures to minimize the likelihood of such confusion.

T.3P_SOFTWARE

In some VS implementations, functions critical to the security of the TOE are by necessity performed by software not produced by the virtualization vendor. Such software may include physical device drivers, and even non-TOE entities such as Host Operating Systems. Since this software has the same or similar privilege level as the VS, vulnerabilities can be exploited by an adversary to compromise the VS and VMs. Where possible, the VS should mitigate the results of potential vulnerabilities or malicious content in third-party code on which it relies. For example, physical device drivers (potentially the Host OS) could be encapsulated within VMs in order to limit the effects of compromise.

T.VMM_COMPROMISE

The VS is designed to provide the appearance of exclusivity to the VMs and is designed to separate or isolate their functions except where specifically shared. Failure of security mechanisms could lead to unauthorized intrusion into or modification of the VMM, or bypass of the VMM altogether, by non-TOE software, such as that running in Guest or Helper VMs or on the host platform. This must be prevented to avoid compromising the VS.

T.PLATFORM_COMPROMISE

The VS must be capable of protecting the platform from threats that originate within VMs and operational networks connected to the VS. The hosting of untrusted—even malicious—domains by the VS cannot be permitted to compromise the security and integrity of the platform on which the VS executes. If an attacker can access the underlying platform in a manner not controlled by the VMM, the attacker might be able to modify system firmware or software—compromising both the VS and the underlying platform.

T.UNAUTHORIZED_ACCESS

Functions performed by the management layer include VM configuration, virtualized network configuration, allocation of physical resources, and reporting. Only certain authorized system users (administrators) are allowed to exercise management functions or obtain sensitive information from the TOE.

Virtualization Systems are often managed remotely over communication networks. Members of these networks can be both geographically and logically separated from each other, and pass through a variety of other systems which may be under the control of an adversary, and offer the opportunity for communications to be compromised. An adversary with access to an open management network could inject commands into the management infrastructure or extract sensitive information. This would provide an adversary with administrator privilege on the platform, and administrative control over the VMs and virtual network connections. The adversary could also gain access to the management network

by hijacking the management network channel.

T.WEAK_CRYPTO
To the extent that VMs appear isolated within the VS, a threat of weak cryptography may arise if the VMM does not provide good entropy to support security-related features that depend on entropy to implement cryptographic algorithms. For example, a random number generator keeps an estimate of the number of bits of noise in the entropy pool. From this entropy pool random numbers are created. Good random numbers are essential to implementing strong cryptography. Cryptography implemented using poor random numbers can be defeated by a sophisticated adversary. Such defeat can result in the compromise of Guest VM data and credentials, and of VS data and credentials, and can enable unauthorized access to the VS or VMs.

T.UNPATCHED_SOFTWARE
Vulnerabilities in outdated or unpatched software can be exploited by adversaries to compromise the VS or platform.

T.MISCONFIGURATION
The VS may be misconfigured, which could impact its functioning and security. This misconfiguration could be due to an administrative error or the use of faulty configuration data.

T.DENIAL_OF_SERVICE
A VM may block others from system resources (e.g., system memory, persistent storage, and processing time) via a resource exhaustion attack.

4.2 Assumptions

These assumptions are made on the Operational Environment (OE) in order to be able to ensure that the security functionality specified in the PP can be provided by the TOE. If the TOE is placed in an OE that does not meet these assumptions, the TOE may no longer be able to provide all of its security functionality.

A.PLATFORM_INTEGRITY
The platform has not been compromised prior to installation of the VS.

A.PHYSICAL
Physical security commensurate with the value of the TOE and the data it contains is assumed to be provided by the environment.

A.TRUSTED_ADMIN
TOE Administrators are trusted to follow and apply all administrator guidance.

A.NON_MALICIOUS_USER
The user of the VS is not willfully negligent or hostile, and uses the VS in compliance with the applied enterprise security policy and guidance. At the same time, malicious applications could act as the user, so requirements which confine malicious applications are still in scope.

4.3 Organizational Security Policies

This document does not define any additional OSPs

5 Security Objectives

5.1 Security Objectives for the TOE

O.VM_ISOLATION

VMs are the fundamental subject of the system. The VMM is responsible for applying the system security policy (SSP) to the VM and all resources. As basic functionality, the VMM must support a security policy that mandates no information transfer between VMs.

The VMM must support the necessary mechanisms to isolate the resources of all VMs. The VMM partitions a platform's physical resources for use by the supported virtual environments. Depending on customer requirements, a VM may need a completely isolated environment with exclusive access to system resources or share some of its resources with other VMs. It must be possible to enforce a security policy that prohibits the transfer of data between VMs through shared devices. When the platform security policy allows the sharing of resources across VM boundaries, the VMM must ensure that all access to those resources is consistent with the policy. The VMM may delegate the responsibility for the mediation of resource sharing to select Service VMs; however in doing so, it remains responsible for mediating access to the Service VMs, and each Service VM must mediate all access to any shared resource that has been delegated to it in accordance with the SSP.

Both virtual and physical devices are resources requiring access control. The VMM must enforce access control in accordance with system security policy. Physical devices are platform devices with access mediated via the VMM per the O.VMM_Integrity objective. Virtual devices may include virtual storage devices and virtual network devices. Some of the access control restrictions must be enforced internal to Service VMs, as may be the case for isolating virtual networks. VMMs may also expose purely virtual interfaces. These are VMM specific, and while they are not analogous to a physical device, they are also subject to access control.

The VMM must support the mechanisms to isolate all resources associated with virtual networks and to limit a VM's access to only those virtual networks for which it has been configured. The VMM must also support the mechanisms to control the configurations of virtual networks according to the SSP.

O.VMM_INTEGRITY

Integrity is a core security objective for Virtualization Systems. To achieve system integrity, the integrity of each VMM component must be established and maintained. This objective concerns only the integrity of the VS—not the integrity of software running inside of Guest VMs or of the physical platform. The overall objective is to ensure the integrity of critical components of a VS.

Initial integrity of a VS can be established through mechanisms such as a digitally signed installation or update package, or through integrity measurements made at launch. Integrity is maintained in a running system by careful protection of the VMM from untrusted users and software. For example, it must not be possible for software running within a Guest VM to exploit a vulnerability in a device or hypercall interface and gain control of the VMM. The vendor must release patches for vulnerabilities as soon as practicable after discovery.

O.PLATFORM_INTEGRITY

The integrity of the VMM depends on the integrity of the hardware and software on which the VMM relies. Although the VS does not have complete control over the integrity of the platform, the VS should as much as possible try to ensure that no users or software hosted by the VS can undermine the integrity of the platform.

O.DOMAIN_INTEGRITY

While the VS is not responsible for the contents or correct functioning of software that runs within Guest VMs, it is responsible for ensuring that the correct functioning of the software within a Guest VM is not interfered with by other VMs.

O.MANAGEMENT_ACCESS

VMM management functions include VM configuration, virtualized network configuration, allocation of physical resources, and reporting. Only authorized users (administrators) may exercise management functions.

Because of the privileges exercised by the VMM management functions, it must not be possible for the VMM’s management components to be compromised without administrator notification. This means that unauthorized users cannot be permitted access to the management functions, and the management components must not be interfered with by Guest VMs or unprivileged users on other networks—including operational networks connected to the TOE.

VMMs include a set of management functions that collectively allow administrators to configure and manage the VMM, as well as configure Guest VMs. These management functions are specific to the VS and are distinct from any other management functions that might exist for the internal management of any given Guest VM. These VMM management functions are privileged, with the security of the entire system relying on their proper use. The VMM management functions can be classified into different categories and the policy for their use and the impact to security may vary accordingly.

The management functions are distributed throughout the VMM (within the VMM and Service VMs). The VMM must support the necessary mechanisms to enable the control of all management functions according to the system security policy. When a management function is distributed among multiple

Service VMs, the VMs must be protected using the security mechanisms of the Hypervisor and any Service VMs involved to ensure that the intent of the system security policy is not compromised. Additionally, since hypercalls permit Guest VMs to invoke the Hypervisor, and often allow the passing of data to the Hypervisor, it is important that the hypercall interface is well-guarded and that all parameters be validated.

The VMM maintains configuration data for every VM on the system. This configuration data, whether of Service or Guest VMs, must be protected. The mechanisms used to establish, modify and verify configuration data are part of the VS management functions and must be protected as such. The proper internal configuration of Service VMs that provide critical security functions can also greatly impact VS security. These configurations must also be protected. Internal configuration of Guest VMs should not impact overall VS security. The overall goal is to ensure that the VMM, including the environments internal to Service VMs, is properly configured and that all Guest VM configurations are maintained consistent with the system security policy throughout their lifecycle.

Virtualization Systems are often managed remotely. For example, an administrator can remotely update virtualization software, start and shut down VMs, and manage virtualized network connections. If a console is required, it could be run on a separate machine or it could itself run in a VM. When performing remote management, an administrator must communicate with a privileged management agent over a network. Communications with the management infrastructure must be protected from Guest VMs and operational networks.

O.PATCHED_SOFTWARE
The VS must be updated and patched when needed in order to prevent the potential compromise of the VMM, as well as the networks and VMs that it hosts. Identifying and applying needed updates must be a normal part of the operating procedure to ensure that patches are applied in a timely and thorough manner. In order to facilitate this, the VS must support standards and protocols that help enhance the manageability of the VS as an IT product, enabling it to be integrated as part of a manageable network (e.g., reporting current patch level and patchability).

O.VM_ENTROPY
VMs must have access to good entropy sources to support security-related features that implement cryptographic algorithms. For example, in order to function as members of operational networks, VMs must be able to communicate securely with other network entities—whether virtual or physical. They must therefore have access to sources of good entropy to support that secure communication.

O.AUDIT
An audit log must be created that captures accesses to the objects the TOE protects. The log of these accesses, or audit events, must be protected from modification, unauthorized access, and destruction. The audit log must be sufficiently detailed to indicate the date and time of the event, the identify of the user, the type of event, and the success or failure of the event.

O.CORRECTLY_APPLIED_CONFIGURATION
The TOE must not apply configurations that violate the current security policy.
The TOE must correctly apply configurations and policies to a newly created Guest VM, as well as to existing Guest VMs when applicable configuration or policy changes are made. All changes to configuration and to policy must conform to the existing security policy. Similarly, changes made to the configuration of the TOE itself must not violate the existing security policy.

O.RESOURCE_ALLOCATION
The TOE will provide mechanisms that enforce constraints on the allocation of system resources in accordance with existing security policy.

5.2 Security Objectives for the Operational Environment

This PP-Module does not define any objectives for the OE.

OE.CONFIG
TOE administrators will configure the VS correctly to create the intended security policy.

OE.PHYSICAL
Physical security, commensurate with the value of the TOE and the data it contains, is provided by the environment.

OE.TRUSTED_ADMIN
TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.

OE.NON_MALICIOUS_USER
Users are trusted to not be willfully negligent or hostile and use the VS in compliance with the applied enterprise security policy and guidance.

5.3 Security Objectives Rationale

This section describes how the assumptions, threats, and organizational security policies map to the security objectives.

Table 1: Security Objectives Rationale

Threat, Assumption, or OSP	Security Objectives	Rationale
T.DATA_LEAKAGE	O.VM_ISOLATION	Logical separation of VMs and enforcement of domain integrity prevent unauthorized transmission of data from one VM to another.
	O.DOMAIN_ INTEGRITY	Logical separation of VMs and enforcement of domain integrity prevent unauthorized transmission of data from one VM to another.
T.UNAUTHORIZED_ UPDATE	O.VMM_ INTEGRITY	System integrity prevents the TOE from installing a software patch containing unknown and potentially malicious code.
T.UNAUTHORIZED_ MODIFICATION	O.VMM_ INTEGRITY	Enforcement of VMM integrity prevents the bypass of enforcement mechanisms and auditing ensures that abuse of legitimate authority can be detected.
	O.AUDIT	Enforcement of VMM integrity prevents the bypass of enforcement mechanisms and auditing ensures that abuse of legitimate authority can be detected.
T.USER_ERROR	O.VM_ISOLATION	Isolation of VMs includes clear attribution of those VMs to their respective domains which reduces the likelihood that a user inadvertently inputs or transfers data meant for one VM into another.
T.3P_SOFTWARE	O.VMM_ INTEGRITY	The VMM integrity mechanisms include environment-based vulnerability mitigation and potentially support for introspection and device driver isolation, all of which reduce the likelihood that any vulnerabilities in third-party software can be used to exploit the TOE.
T.VMM_COMPROMISE	O.VMM_ INTEGRITY	Maintaining the integrity of the VMM and ensuring that VMs execute in isolated domains mitigate the risk that the VMM can be compromised or bypassed.
	O.VM_ISOLATION	Maintaining the integrity of the VMM and ensuring that VMs execute in isolated domains mitigate the risk that the VMM can be compromised or bypassed.
T.PLATFORM_ COMPROMISE	O.PLATFORM_ INTEGRITY	Platform integrity mechanisms used by the TOE reduce the risk that an attacker can ‘break out’ of a VM and affect the platform on which the VS is running.
T.UNAUTHORIZED_ ACCESS	O.MANAGEMENT_ ACCESS	Ensuring that TSF management functions cannot be executed without authorization prevents untrusted subjects from modifying the behavior of the TOE in an unanticipated manner.
T.WEAK_CRYPTO	O.VM_ENTROPY	Acquisition of good entropy is necessary to support the TOE's security-related cryptographic algorithms.
T.UNPATCHED_ SOFTWARE	O.PATCHED_ SOFTWARE	The ability to patch the TOE software ensures that protections against vulnerabilities can be applied as they become available.
T.MISCONFIGURATION	O.CORRECTLY_ APPLIED_ CONFIGURATION	Mechanisms to prevent the application of configurations that violate the current security policy help prevent misconfigurations.
T.DENIAL_OF_SERVICE	O.RESOURCE_ ALLOCATION	The ability of the TSF to ensure the proper allocation of resources makes denial of service attacks more difficult.
A.PLATFORM_ INTEGRITY	OE.PHYSICAL	If the underlying platform has not been compromised prior to installation of the TOE, its integrity can be assumed to be intact.
A.PHYSICAL	OE.PHYSICAL	If the TOE is deployed in a location that has appropriate physical safeguards, it can be assumed to be physically secure.
A.TRUSTED_ADMIN	OE.TRUSTED_ ADMIN	Providing guidance to administrators and ensuring that individuals are properly trained and vetted before being

		given administrative responsibilities will ensure that they are trusted.
A.NON_MALICIOUS_ USER	OE.NON_ MALICIOUS_ USER	If the organization properly vets and trains users, it is expected that they will be non-malicious.
	OE.CONFIG	If the TOE is administered by a non-malicious and non-negligent user, the expected result is that the TOE will be configured in a correct and secure manner.

6 Security Requirements

6.1 Security Functional Requirements

6.1.1 Security Audit (FAU)

FAU_ARP.1 Security Audit Automatic Response

FAU_ARP.1.1

The TSF shall take [**assignment**: *list of actions*] upon detection of a potential security violation.
Application Note: In certain cases, it may be useful for Virtualization Systems to perform automated responses to certain security events. An example may include halting a VM which has taken some action to violate a key system security policy. This may be especially useful with headless endpoints when there is no human user in the loop.
The potential security violation mentioned in FAU_ARP.1.1 refers to FAU_SAA.1.

FAU_GEN.1 Audit Data Generation

FAU_GEN.1.1

The TSF shall be able to generate an audit record of the following auditable events:

a. Start-up and shutdown of audit functions
b. [**All administrative actions relevant to claimed SFRs as defined in the Auditable Events Table from the Client and Server PP-Modules**]
c. [**Auditable events defined in**]
d. **[selection: *Auditable events defined in*** for Strictly Optional SFRs, *Auditable events defined in for Objective SFRs*, *Auditable events defined in for Selection-Based SFRs*, *Auditable events for the TLS listed in figure Table* , *Auditable events defined in the audit table for the Secure Shells PP*, *no other auditable events*]

FAU_GEN.1.2

The TSF shall record within each audit record at least the following information:

a. Date and time of the event
b. Type of event
c. Subject and object identity (if applicable)
d. The outcome (success or failure) of the event
e. [**Additional information defined in**]
f. **[selection: *Additional information defined in*** for Strictly Optional SFRs, *Additional information defined in for Objective SFRs*, *Additional information defined in for Selection-Based SFRs*, *Additional information for the TLS listed in figure Table* , *Additional information defined in the audit table for the Secure Shells PP*, *no other information*]

Application Note: The ST author can include other auditable events directly in ; they are not limited to the list presented. The ST author should update the table in FAU_GEN.1.2 with any additional information generated. “Subject identity” in FAU_GEN.1.2 could be a user id or an identifier specifying a VM, for example.

Appropriate entries from , , and should be included in the ST if the associated SFRs and selections are included.

The entry for FDP_VNC_EXT.1 refers to configuration settings that attach VMs to virtualized network components. Changes to these configurations can be made during VM execution or when VMs are not running. Audit records must be generated for either case.

The intent of the audit requirement for FDP_PPR_EXT.1 is to log that the VM is connected to a physical device (when the device becomes part of the VM's hardware view), not to log every time that the device is accessed. Generally, this is only once at VM startup. However, some devices can be connected and disconnected during operation (e.g., virtual USB devices such as CD-ROMs). All such connection/disconnection events must be logged.

The following table contains the events enumerated in the auditable events table for the TLS Functional Package. Inclusion of these events in the ST is subject to selection above, inclusion of the corresponding SFRs in the ST, and support in the FP as represented by a selection in the table below.

Table 2: : Auditable Events for the TLS Functional Package

FCS_TLSC_EXT.1	Failure to establish a session.	Reason for failure.
FCS_TLSC_EXT.1	Failure to verify presented identifier.	Presented identifier and reference identifier.
FCS_TLSC_EXT.1	Establishment/termination of a TLS session.	Non-TOE endpoint of connection.
FCS_TLSS_EXT.1	Failure to establish a session.	Reason for failure.
FCS_DTLSC_EXT.1	Failure of the certificate validity check.	Issuer Name and Subject Name of certificate.
FCS_DTLSS_EXT.1	Failure of the certificate validity check.	Issuer Name and Subject Name of certificate.

FAU_SAA.1 Potential Violation Analysis

FAU_SAA.1.1

The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the enforcement of the SFRs.

FAU_SAA.1.2

The TSF shall enforce the following rules for monitoring audited events:

a. Accumulation or combination of [**assignment**: *subset of defined auditable events*] known to indicate a potential security violation;
b. [**assignment**: *any other rules*].

Application Note: The potential security violation described in FAU_SAA.1 can be used as a trigger for automated responses as defined in FAU_ARP.1.

FAU_SAR.1 Audit Review

FAU_SAR.1.1

The TSF shall provide [*administrators*] with the capability to read [*all information*] from the audit records.

FAU_SAR.1.2

The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

FAU_STG.1 Protected Audit Trail Storage

FAU_STG.1.1

The TSF shall protect the stored audit records in the audit trail from unauthorized deletion.

FAU_STG.1.2

The TSF shall be able to [*prevent*] unauthorized modifications to the stored audit records in the audit trail.
Application Note: The evaluation activity for this SFR is not intended to imply that the TOE must support an administrator's ability to designate individual audit records for deletion. That level of granularity is not required.

FAU_STG_EXT.1 Off-Loading of Audit Data

FAU_STG_EXT.1.1

The TSF shall be able to transmit the generated audit data to an external IT entity using a trusted channel as specified in FTP_ITC_EXT.1.

FAU_STG_EXT.1.2

The TSF shall [**selection**: *drop new audit data, overwrite previous audit records according to the following rule: [**assignment**: rule for overwriting previous audit records], [**assignment**: other action]*] when the local storage space for audit data is full.
Application Note: An external log server, if available, might be used as alternative storage space in case the local storage space is full. An 'other action' could be defined in this case as 'send the new audit data to an external IT entity'.

6.1.2 Cryptographic Support (FCS)

FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.1.1

The TSF shall generate **asymmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm [**selection**: *RSA schemes using cryptographic key sizes [2048-bit or greater] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3] ECC schemes using ["NIST curves" P-256, P-384, and [**selection**: P-521, no other curves] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4] FFC schemes using cryptographic key sizes [2048-bit or greater] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.1]]. FFC Schemes using Diffie-Hellman group 14 that meet the following: [RFC 3526]FFC Schemes using safe primes that meet the following: ['NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes"]*] ~~and specified cryptographic key sizes [assignment: cryptographic key sizes] that meet the following: [assignment: list of standards]~~.

Application Note: The ST author selects all key generation schemes used for key establishment and device authentication. When key generation is used for key establishment, the schemes in FCS_CKM.2.1 and selected cryptographic protocols shall match the selection. When key generation is used for device authentication, the public key is expected to be associated with an X.509v3 certificate.

If the TOE acts as a receiver in the RSA key establishment scheme, the TOE does not need to implement RSA key generation.

FCS_CKM.2 Cryptographic Key Distribution

FCS_CKM.2.1

The TSF shall ~~distribute cryptographic keys~~ **implement functionality to perform cryptographic key establishment** in accordance with a specified cryptographic key establishment method: [**selection**: *RSA-based key establishment schemes that meets the following: RSAES-PKCS1-v1_5 as specified in Section 7.2 of RFC 8017, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.2", Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", Finite field-based key establishment schemes that meets the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", Key establishment scheme using Diffie-Hellman group 14 that meets the following: RFC 3526*] ~~that meets the following [assignment: list of standards]~~.

FCS_CKM_EXT.4 Cryptographic Key Destruction

FCS_CKM_EXT.4.1

The TSF shall cause disused cryptographic keys in volatile memory to be destroyed or rendered unrecoverable.

Application Note: The threat addressed by this element is the recovery of disused cryptographic keys from volatile memory by unauthorized processes.

The TSF must destroy or cause to be destroyed all copies of cryptographic keys created and managed by the TOE once the keys are no longer needed. This requirement is the same for all instances of keys within TOE volatile memory regardless of whether the memory is controlled by TOE manufacturer software or by third-party TOE modules. The evaluation activities are designed with flexibility to address cases where the TOE manufacturer has limited insight into the behavior of third-party TOE components.

The preferred method for destroying keys in TOE volatile memory is by direct overwrite of the memory occupied by the keys. The values used for overwriting can be all zeros, all ones, or any other pattern or combination of values significantly different than the value of the key itself such that the keys are rendered inaccessible to running processes.

Some implementations may find that direct overwriting of memory is not feasible or possible due to programming language constraints. Many memory- and type-safe languages provide no mechanism for programmers to specify that a particular memory location be accessed or written. The value of such languages is that it is much harder for a programming error to result in a buffer or heap overflow. The downside is that multiple copies of keys might be scattered throughout language-runtime memory. In such cases, the TOE should take whatever actions are feasible to cause the keys to become inaccessible—freeing memory, destroying objects, closing applications, programming using the minimum possible scope for variables containing keys.

Likewise, if keys reside in memory within the execution context of a third-party module, then the TOE should take whatever feasible actions it can to cause the

keys to be destroyed.

Cryptographic keys in non-TOE volatile memory are not covered by this requirement. This expressly includes keys created and used by Guest VMs. The Guest is responsible for disposing of such keys.

FCS_CKM_EXT.4.2

The TSF shall cause disused cryptographic keys in non-volatile storage to be destroyed or rendered unrecoverable.
Application Note: The ultimate goal of this element is to ensure that disused cryptographic keys are inaccessible not only to components of the running system, but are also unrecoverable through forensic analysis of discarded storage media. The element is designed to reflect the fact that the latter may not be wholly practical at this time due to the way some storage technologies are implemented (e.g., wear-leveling of flash storage).

Key storage areas in non-volatile storage can be overwritten with any value that renders the keys unrecoverable. The value used can be all zeros, all ones, or any other pattern or combination of values significantly different than the value of the key itself.

The TSF must destroy all copies of cryptographic keys created and managed by the TOE once the keys are no longer needed. Since this is a software-only TOE, the hardware controllers that manage non-volatile storage media are necessarily outside the TOE boundary. Thus, the TOE manufacturer is likely to have little control over—or insight into—the functioning of these storage devices. The TOE must make a “best-effort” to destroy disused cryptographic keys by invoking the appropriate platform interfaces—recognizing that the specific actions taken by the platform are out of the TOE’s control.

But in cases where the TOE has insight into the non-volatile storage technologies used by the platform, or where the TOE can specify a preference or method for destroying keys, the destruction should be executed by a single, direct overwrite consisting of pseudorandom data or a new key, by a repeating pattern of any static value, or by a block erase.

For keys stored on encrypted media, it is sufficient for the media encryption keys to be destroyed for all keys stored on the media to be considered destroyed.

FCS_COP.1/Hash Cryptographic Operation (Hashing)

FCS_COP.1.1/Hash

The TSF shall perform [*cryptographic hashing*] in accordance with a specified cryptographic algorithm [**selection**: *SHA-1*, *SHA-256*, *SHA-384*, *SHA-512*, *SHA-3-224*, *SHA-3-256*, *SHA-3-384*, *SHA-3-512*] and message digest sizes [**selection**: *160*, *256*, *384*, *512 bits*] that meet the following: [**selection**: *FIPS PUB 180-4* "*Secure Hash Standard*", *ISO/IEC 10118-3:2018*]
Application Note: Per NIST SP 800-131A, SHA-1 for generating digital signatures is no longer allowed, and SHA-1 for verification of digital signatures is strongly discouraged as there may be risk in accepting these signatures. It is expected that vendors will implement SHA-2 algorithms in accordance with SP 800-131A.

The intent of this requirement is to specify the hashing function. The hash selection shall support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used (for example, SHA 256 for 128-bit keys).

FCS_COP.1/KeyedHash Cryptographic Operation (Keyed Hash Algorithms)

FCS_COP.1.1/KeyedHash

The TSF shall perform [*keyed-hash message authentication*] in accordance with a specified cryptographic algorithm [**selection**: *HMAC-SHA-1*, *HMAC-SHA-256*, *HMAC-SHA-384*, *HMAC-SHA-512*, *SHA-3-224*, *SHA-3-256*, *SHA-3-384*, *SHA-3-512*] and cryptographic key sizes [**assignment**: *key size (in bits) used in HMAC*] and message digest sizes [**selection**: *160 bits*, *256 bits*, *384 bits*, *512 bits*] that meet the following: [***FIPS Pub 198-1, "The Keyed-Hash Message Authentication Code," and FIPS Pub 180-4, “Secure Hash Standard"***].
Application Note: The selection in this requirement must be consistent with the key size specified for the size of the keys used in conjunction with the keyed-hash message authentication.

FCS_COP.1/Sig Cryptographic Operation (Signature Algorithms)

FCS_COP.1.1/Sig

The TSF shall perform [*cryptographic signature services (generation and verification)*] in accordance with a specified cryptographic algorithm [**selection**: *RSA schemes using cryptographic key sizes [2048-bit or greater] that meet the following: [FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section*

*4]ECDSA schemes using [“NIST curves” P-256, P-384 and [**selection**: P-521, no other curves]] that meet the following: [FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 5]*].

Application Note: The ST Author should choose the algorithm implemented to perform digital signatures; if more than one algorithm is available, this requirement should be iterated to specify the functionality. For the algorithm chosen, the ST author should make the appropriate assignments/selections to specify the parameters that are implemented for that algorithm.

FCS_COP.1/UDE Cryptographic Operation (AES Data Encryption/Decryption)

FCS_COP.1.1/UDE

The TSF shall perform [*encryption and decryption*] in accordance with a specified cryptographic algorithm
[**selection**: *AES Key Wrap (KW) (as defined in NIST SP 800-38F), AES Key Wrap with Padding (KWP) (as defined in NIST SP 800-38F), AES-GCM (as defined in NIST SP 800-38D), AES-CCM (as defined in NIST SP 800-38C), AES-XTS (as defined in NIST SP 800-38E) mode, AES-CCMP-256 (as defined in NIST SP800-38C and IEEE 802.11ac-2013), AES-GCMP-256 (as defined in NIST SP800-38D and IEEE 802.11ac-2013), AES-CCMP (as defined in FIPS PUB 197, NIST SP 800-38C and IEEE 802.11-2012), AES-CBC (as defined in FIPS PUB 197, and NIST SP 800-38A) mode, AES-CTR (as defined in NIST SP 800-38A) mode*] and cryptographic key sizes [**selection**: *128-bit key sizes, 256-bit key sizes*].

Application Note: For the first selection of FCS_COP.1.1/UDE, the ST author should choose the mode or modes in which AES operates. For the second selection, the ST author should choose the key sizes that are supported by this functionality.

FCS_ENT_EXT.1 Entropy for Virtual Machines

FCS_ENT_EXT.1.1

The TSF shall provide a mechanism to make available to VMs entropy that meets FCS_RBG_EXT.1 through [**selection**: *Hypercall interface, virtual device interface, pass-through access to hardware entropy source*].

FCS_ENT_EXT.1.2

The TSF shall provide independent entropy across multiple VMs.

Application Note: This requirement ensures that sufficient entropy is available to any VM that requires it. The entropy need not provide high-quality entropy for every possible method that a VM might acquire it. The VMM must, however, provide some means for VMs to get sufficient entropy. For example, the VMM can provide an interface that returns entropy to a Guest VM. Alternatively, the VMM could provide pass-through access to entropy sources provided by the host platform.

This requirement allows for three general ways of providing entropy to guests: 1) The VS can provide a Hypercall accessible to VM-aware guests, 2) access to a virtualized device that provides entropy, or 3) pass-through access to a hardware entropy source (including a source of random numbers). In all cases, it is possible that the guest is made VM-aware through installation of software or drivers. For the second and third cases, it is possible that the guest could be VM-unaware. There is no requirement that the TOE provide entropy sources as expected by VM-unaware guests. That is, the TOE does not have to anticipate every way a guest might try to acquire entropy as long as it supplies a mechanism that can be used by VM-aware guests, or provides access to a standard mechanism that a VM-unaware guest would use.

The ST author should select “Hypercall interface” if the TSF provides an API function through which guest-resident software can obtain entropy or random numbers. The ST author should select “virtual device interface” if the TSF presents a virtual device interface to the Guest OS through which it can obtain entropy or random numbers. Such an interface could present a virtualized real device, such as a TPM, that can be accessed by VM-unaware guests, or a virtualized fictional device that would require the Guest OS to be VM-aware. The ST author should select “pass-through access to hardware entropy source” if the TSF permits Guest VMs to have direct access to hardware entropy or random number source on the platform. The ST author should select all items that are appropriate.

For FCS_ENT_EXT.1.2, the VMM must ensure that the provision of entropy to one VM cannot affect the quality of entropy provided to another VM on the same platform.

FCS_RBG_EXT.1 Cryptographic Operation (Random Bit Generation)

FCS_RBG_EXT.1.1

The TSF shall perform all deterministic random bit generation services in

accordance with NIST Special Publication 800-90A using [**selection**: *Hash_DRBG (any)*, *HMAC_DRBG (any)*, *CTR_DRBG (AES)*]

FCS_RBG_EXT.1.2

The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [**selection**: *a software-based noise source*, *a hardware-based noise source*] with a minimum of [**selection**: *128 bits*, *192 bits*, *256 bits*] of entropy at least equal to the greatest security strength according to NIST SP 800-57, of the keys and hashes that it will generate.
Application Note: NIST SP 800-90A contains three different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used, and include the specific underlying cryptographic primitives used in the requirement. While any of the identified hash functions (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed.
If the key length for the AES implementation used here is different than that used to encrypt the user data, then FCS_COP.1/UDE may have to be adjusted or iterated to reflect the different key length. For the selection in FCS_RBG_EXT.1.2, the ST author selects the minimum number of bits of entropy that is used to seed the RBG.

6.1.3 User Data Protection (FDP)

FDP_HBI_EXT.1 Hardware-Based Isolation Mechanisms

FDP_HBI_EXT.1.1

The TSF shall use [**selection**: *no mechanism*, *[**assignment**: list of platform-provided, hardware-based mechanisms]*] to constrain a Guest VM's direct access to the following physical devices: [**selection**: *no devices*, *[**assignment**: physical devices to which the VMM allows Guest VMs physical access]*].
Application Note: The TSF must use available hardware-based isolation mechanisms to constrain VMs when VMs have direct access to physical devices. “Direct access” in this context means that the VM can read or write device memory or access device I/O ports without the VMM being able to intercept and validate every transaction.

FDP_PPR_EXT.1 Physical Platform Resource Controls

FDP_PPR_EXT.1.1

The TSF shall allow an authorized administrator to control Guest VM access to the following physical platform resources: [**assignment**: *list of physical platform resources the VMM is able to control access to*].

FDP_PPR_EXT.1.2

The TSF shall explicitly deny all Guest VMs access to the following physical platform resources: [**selection**: *no physical platform resources*, *[**assignment**: list of physical platform resources to which access is explicitly denied]*].

FDP_PPR_EXT.1.3

The TSF shall explicitly allow all Guest VMs access to the following physical platform resources: [**selection**: *no physical platform resources*, *[**assignment**: list of physical platform resources to which access is always allowed]*].
Application Note: For purposes of this requirement, physical platform resources are divided into three categories:

1. those to which Guest OS access is configurable and moderated by the VMM
2. those to which the Guest OS is never allowed to have direct access, and
3. those to which the Guest OS is always allowed to have direct access.

For element 1, the ST author lists the physical platform resources that can be configured for Guest VM access by an administrator. For element 2, the ST author lists the physical platform resources to which Guest VMs may never be allowed direct access. If there are no such resources, the ST author selects "no physical platform resources." Likewise, any resources to which all Guest VMs automatically have access to are to be listed in the third element. If there are no such resources, then "no physical platform resources" is selected.

FDP_RIP_EXT.1 Residual Information in Memory

FDP_RIP_EXT.1.1

The TSF shall ensure that any previous information content of physical memory is cleared prior to allocation to a Guest VM.
Application Note: Physical memory must be zeroed before it is made accessible to a VM for general use by a Guest OS.
The purpose of this requirement is to ensure that a VM does not receive memory

containing data previously used by another VM or the host.

"For general use" means for use by the Guest OS in its page tables for running applications or system software.

This does not apply to pages shared by design or policy between VMs or between the VMMs and VMs, such as read-only OS pages or pages used for virtual device buffers.

FDP_RIP_EXT.2 Residual Information on Disk

FDP_RIP_EXT.2.1

The TSF shall ensure that any previous information content of physical disk storage is cleared to zeros upon allocation to a Guest VM.
Application Note: The purpose of this requirement is to ensure that a VM does not receive disk storage containing data previously used by another VM or by the host.

Clearing of disk storage only upon deallocation does not meet this requirement.

This does not apply to disk-resident files shared by design or policy between VMs or between the VMMs and VMs, such as read-only data files or files used for inter-VM data transfers permitted by policy.

FDP_VMS_EXT.1 VM Separation

FDP_VMS_EXT.1.1

The VS shall provide the following mechanisms for transferring data between Guest VMs: [**selection**: *no mechanism*, *virtual networking*, *[**assignment**: other inter-VM data sharing mechanisms]*].

FDP_VMS_EXT.1.2

The TSF shall by default enforce a policy prohibiting sharing of data between Guest VMs.

FDP_VMS_EXT.1.3

The TSF shall allow Administrators to configure the mechanisms selected in FDP_VMS_EXT.1.1 to enable and disable the transfer of data between Guest VMs.

FDP_VMS_EXT.1.4

The VS shall ensure that no Guest VM is able to read or transfer data to or from another Guest VM except through the mechanisms listed in FDP_VMS_EXT.1.1.
Application Note: The fundamental requirement of a Virtualization System is the ability to enforce separation between information domains implemented as Virtual Machines and Virtual Networks. The intent of this requirement is to ensure that VMs, VMMs, and the VS as a whole is implemented with this fundamental requirement in mind.

The ST author should select "no mechanism" in the unlikely event that the VS implements no mechanisms for transferring data between Guest VMs. Otherwise, the ST author should select "virtual networking" and identify all other mechanisms through which data can be transferred between Guest VMs.

Examples of non-network inter-VM sharing mechanisms are:

- User interface-based mechanisms, such as copy-paste and drag-and-drop
- Shared virtual or physical devices
- API-based mechanisms such as Hypercalls

For data transfer mechanisms implemented in terms of Hypercall functions, FDP_VMS_EXT.1.3 is met if FPT_HCL_EXT.1.1 is met for those Hypercall functions (Hypercall function parameters are checked).

For data transfer mechanisms that use shared physical devices, FDP_VMS_EXT.1.3 is met if the device is listed in and meets FDP_PPR_EXT.1.1 (VM access to the physical device is configurable).

For data transfer mechanisms that use virtual networking, FDP_VMS_EXT.1.3 is met if FDP_VNC_EXT.1.1 is met (VM access to virtual networks is configurable).

FDP_VNC_EXT.1 Virtual Networking Components

FDP_VNC_EXT.1.1

The TSF shall allow Administrators to configure virtual networking components to connect VMs to each other and to physical networks.

FDP_VNC_EXT.1.2

The TSF shall ensure that network traffic visible to a Guest VM on a virtual network--or virtual segment of a physical network--is visible only to Guest VMs configured to be on that virtual network or segment.
Application Note: Virtual networks must be separated from one another to provide isolation commensurate with that provided by physically separate

networks. It must not be possible for data to cross between properly configured virtual networks regardless of whether the traffic originated from a local Guest VM or a remote host.
Unprivileged users must not be able to connect VMs to each other or to external networks.

6.1.4 Identification and Authentication (FIA)

FIA_AFL_EXT.1 Authentication Failure Handling

FIA_AFL_EXT.1.1

The TSF shall detect when [**selection**: *[**assignment**: a positive integer number], an administrator configurable positive integer within a [**assignment**: range of acceptable values]*] unsuccessful authentication attempts occur related to Administrators attempting to authenticate remotely using [**selection**: *username and password*, *username and PIN*].

FIA_AFL_EXT.1.2

When the defined number of unsuccessful authentication attempts has been met, the TSF shall: [**selection**: *prevent the offending Administrator from successfully establishing a remote session using any authentication method that involves a password or PIN until [**assignment**: action to unlock] is taken by an Administrator, prevent the offending Administrator from successfully establishing a remote session using any authentication method that involves a password or PIN until an Administrator-defined time period has elapsed*]
Application Note: The action to be taken shall be populated in the selection of the ST and defined in the Administrator guidance.
This requirement applies to a defined number of successive unsuccessful remote password or PIN-based authentication attempts and does not apply to local Administrative access. Compliant TOEs may optionally include cryptographic and local authentication failures in the number of unsuccessful authentication attempts.

FIA_UAU.5 Multiple Authentication Mechanisms

FIA_UAU.5.1

The TSF shall provide the following authentication mechanisms: [**selection**: *[**selection**: local, directory-based] authentication based on username and password authentication based on username and a PIN that releases an asymmetric key stored in OE-protected storage [**selection**: local, directory-based] authentication based on X.509 certificates [**selection**: local, directory-based] authentication based on an SSH public key credential*] to support Administrator authentication.
Application Note: Selection of ‘authentication based on username and password’ requires that FIA_PMG_EXT.1 be included in the ST. This also requires that the ST include a management function for password management. If the ST author selects ‘authentication based on an SSH public-key credential’, the TSF shall be validated against the Functional Package for Secure Shell. The ST must include FIA_X509_EXT.1 and FIA_X509_EXT.2 if 'authentication based on X.509 certificates' is selected.

PINs used to access OE-protected storage are set and managed by the OE-protected storage mechanism. Thus requirements on PIN management are outside the scope of the TOE.

FIA_UAU.5.2

The TSF shall authenticate any **Administrator’s** claimed identity according to the [**assignment**: *rules describing how the multiple authentication mechanisms provide authentication*].

FIA_UIA_EXT.1 Administrator Identification and Authentication

FIA_UIA_EXT.1.1

The TSF shall require Administrators to be successfully identified and authenticated using one of the methods in FIA_UAU.5 before allowing any TSF-mediated management function to be performed by that Administrator.
Application Note: Users do not have to authenticate, only Administrators need to authenticate.

6.1.5 Security Management (FMT)

FMT_SMO_EXT.1 Separation of Management and Operational Networks

FMT_SMO_EXT.1.1

The TSF shall support the separation of management and operational network

traffic through [**selection**: *separate physical networks*, *separate logical networks*, *trusted channels as defined in FTP_ITC_EXT.1*, *data encryption using an algorithm specified in FCS_COP.1/UDE*].
Application Note: Management communications must be separate from user workload communications. Administrative network traffic—including communications between physical hosts concerning load balancing, audit data, VM startup and shutdown—must be isolated from guest operational networks. For purposes of this requirement, management traffic also includes VMs transmitted over management networks whether for backup, live migration, or deployment.
“Separate physical networks” refers to using separate physical interfaces and cables to isolate management and operational networks from each other.

“Separate logical networks” refers to using logical networking constructs, such as separate IP spaces or virtual networks to isolate traffic across general-purpose networking ports. Management and operational networks are kept separate within the hosts using separate virtualized networking components.

If the ST author selects “trusted channels...” then the protocols used for network separation must be selected in FTP_ITC_EXT.1.

The ST author selects "data encryption..." if, for example, the TOE encrypts VMs as data blobs for backup, storage, deployment, or live migration, and does not send the data through a tunnel. If the ST author selects "data encryption..." then the algorithms and key sizes must be selected in FCS_COP.1/UDE.

The ST author should select as many mechanisms as apply.

6.1.6 Protection of the TSF (FPT)

FPT_DDI_EXT.1 Device Driver Isolation

FPT_DDI_EXT.1.1

The TSF shall ensure that device drivers for physical devices are isolated from the VMM and all other domains.
Application Note: In order to function on physical hardware, the VMM must have access to the device drivers for the physical platform on which it runs. These drivers are often written by third parties, and yet are effectively a part of the VMM. Thus the integrity of the VMM in part depends on the quality of third party code that the virtualization vendor has no control over. By encapsulating these drivers within one or more dedicated driver domains (e.g., Service VM or VMs) the damage of a driver failure or vulnerability can be contained within the domain, and would not compromise the VMM. When driver domains have exclusive access to a physical device, hardware isolation mechanisms, such as Intel's VT-d, AMD's Input/Output Memory Management Unit (IOMMU), or ARM's System Memory Management Unit (MMU) should be used to ensure that operations performed by Direct Memory Access (DMA) hardware are properly constrained.

FPT_DVD_EXT.1 Non-Existence of Disconnected Virtual Devices

FPT_DVD_EXT.1.1

The TSF shall prevent Guest VMs from accessing virtual device interfaces that are not present in the VM’s current virtual hardware configuration.
Application Note: The virtualized hardware abstraction implemented by a particular VS might include the virtualized interfaces for many different devices. Sometimes these devices are not present in a particular instantiation of a VM. The interface for devices not present must not be accessible by the VM.
Such interfaces include memory buffers, PCI Bus interfaces, and processor I/O ports.

The purpose of this requirement is to reduce the attack surface of the VMM by blocking access to unused interfaces.

FPT_EEM_EXT.1 Execution Environment Mitigations

FPT_EEM_EXT.1.1

The TSF shall take advantage of execution environment-based vulnerability mitigation mechanisms supported by the Platform such as: [**selection**: *Address space randomization*, *Memory execution protection (e.g., DEP)*, *Stack buffer overflow protection*, *Heap corruption detection*, *[**assignment**: other mechanisms]*, *No mechanisms*]
Application Note: Processor manufacturers, compiler developers, and operating system vendors have developed execution environment-based mitigations that increase the cost to attackers by adding complexity to the task of compromising systems. Software can often take advantage of these mechanisms by using APIs provided by the operating system or by enabling the

mechanism through compiler or linker options.

This requirement does not mandate that these protections be enabled throughout the Virtualization System—only that they be enabled where they have likely impact. For example, code that receives and processes user input should take advantage of these mechanisms.

For the selection, the ST author selects the supported mechanisms and uses the assignment to include mechanisms not listed in the selection, if any.

FPT_GVI_EXT.1 Guest VM Integrity

FPT_GVI_EXT.1.1

The TSF shall verify the integrity of Guest VMs through the following mechanisms: [**assignment**: *list of Guest VM integrity mechanisms*].
Application Note: The primary purpose of this requirement is to identify and describe the mechanisms used to verify the integrity of Guest VMs that have been 'imported' in some fashion, though these mechanisms could also be applied to all Guest VMs, depending on the mechanism used. Importation for this requirement could include VM migration (live or otherwise), the importation of virtual disk files that were previously exported, VMs in shared storage, etc. It is possible that a trusted VM could have been modified during the migration or import/export process, or VMs could have been obtained from untrusted sources in the first place, so integrity checks on these VMs can be a prudent measure to take. These integrity checks could be as thorough as making sure the entire VM exactly matches a previously known VM (by hash for example), or by simply checking certain configuration settings to ensure that the VM's configuration will not violate the security model of the VS.

FPT_HAS_EXT.1 Hardware Assists

FPT_HAS_EXT.1.1

The VMM shall use [**assignment**: *list of hardware-based virtualization assists*] to reduce or eliminate the need for binary translation.

FPT_HAS_EXT.1.2

The VMM shall use [**assignment**: *list of hardware-based virtualization memory-handling assists*] to reduce or eliminate the need for shadow page tables.
Application Note: These hardware-assists help reduce the size and complexity of the VMM, and thus, of the trusted computing base, by eliminating or reducing the need for paravirtualization or binary translation. Paravirtualization involves modifying guest software so that instructions that cannot be properly virtualized are never executed on the physical processor.

For the assignment in FPT_HAS_EXT.1, the ST author lists the hardware-based virtualization assists on all platforms included in the ST that are used by the VMM to reduce or eliminate the need for software-based binary translation. Examples for the x86 platform are Intel VT-x and AMD-V. “None” is an acceptable assignment for platforms that do not require virtualization assists in order to eliminate the need for binary translation. This must be documented in the TSS.

For the assignment in FPT_HAS_EXT.1.2, the ST author lists the set of hardware-based virtualization memory-handling extensions for all platforms listed in the ST that are used by the VMM to reduce or eliminate the need for shadow page tables. Examples for the x86 platform are Intel EPT and AMD RVI. “None” is an acceptable assignment for platforms that do not require memory-handling assists in order to eliminate the need for shadow page tables. This must be documented in the TSS.

FPT_HCL_EXT.1 Hypercall Controls

FPT_HCL_EXT.1.1

The TSF shall validate the parameters passed to Hypercall interfaces prior to execution of the VMM functionality exposed by each interface.
Application Note: The purpose of this requirement is to help ensure the integrity of the VMM by protecting the attack surface exposed to untrusted Guest VMs through Hypercalls.

A Hypercall interface allows VMM functionality to be invoked by VM-aware guest software. For example, a hypercall interface could be used to get information about the real world, such as the time of day or the underlying hardware of the host system. A hypercall could also be used to transfer data between VMs through a copy-paste mechanism. Because hypercall interfaces expose the VMM to Guest software, these interfaces constitute attack surface.

There is no expectation that the evaluator will need to review source code in order to accomplish the evaluation activity.

FPT_IDV_EXT.1 Software Identification and Versions

FPT_IDV_EXT.1.1

The TSF shall include software identification (SWID) tags that contain a SoftwareIdentity element and an Entity element as defined in ISO/IEC 19770-2:2009.

FPT_IDV_EXT.1.2

The TSF shall store SWIDs in a .swidtag file as defined in ISO/IEC 19770-2:2009.
Application Note: SWID tags are XML files embedded within software that provide a standard method for IT departments to track and manage the software. The presence of SWIDs can greatly simplify the software management process and improve security by enhancing the ability of IT departments to manage updates.

FPT_INT_EXT.1 Support for Introspection

FPT_INT_EXT.1.1

The TSF shall support a mechanism for permitting the VMM or privileged VMs to access the internals of another VM for purposes of introspection.
Application Note: Introspection can be used to support malware and anomaly detection from outside of the guest environment. This not only helps protect the Guest OS, it also protects the VS by providing an opportunity for the VS to detect threats to itself that originate within VMs, and that may attempt to break out of the VM and compromise the VMM or other VMs.

The hosting of malware detection software outside of the guest VM helps protect the guest and helps ensure the integrity of the malware detection/antivirus software. This capability can be implemented in the VMM itself, but ideally it should be hosted by a Service VM so that it can be better contained and does not introduce bugs into the VMM.

FPT_ML_EXT.1 Measured Launch of Platform and VMM

FPT_ML_EXT.1.1

The TSF shall support a measured launch of the Virtualization System. Measured components of the VS shall include the static executable image of the Hypervisor and: [**selection**: *Static executable images of the Management Subsystem*, *[**assignment**: list of (static images of) Service VMs]*, *[**assignment**: list of configuration files]*, *no other components*]

FPT_ML_EXT.1.2

The TSF shall make the measurements selected in FPT_ML_EXT.1.1 available to the Management Subsystem.
Application Note: A measured launch of the platform and VS demonstrates that the proper TOE software was loaded. A measured launch process employs verifiable integrity measurement mechanisms. For example, a VS may hash components such as the hypervisor, service VMs, or the Management Subsystem. A measured launch process only allows components to be executed after the measurement has been recorded. An example process may add each component's hash before it is executed so that the final hash reflects the evidence of a component's state prior to execution. The measurement may be verified as the system boots, but this is not required.

The Platform is outside of the TOE. However, this requirement specifies that the VS must be capable of receiving Platform measurements if the Platform provides them. This requirement is requiring TOE support for Platform measurements if provided; it is not placing a requirement on the Platform to take such measurements.

If available, hardware should be used to store measurements in such a manner that they cannot be modified in any manner except to be extended. These measurements should be produced in a repeatable manner so that a third party can verify the measurements if given the inputs. Hardware devices, like Trusted Platform Modules (TPM), TrustZone, and MMU are some examples that may serve as foundations for storing and reporting measurements.

Platforms with a root of trust for measurement (RTM) should initiate the measured launch process. This may include core BIOS or the chipset. The chipset is the preferred RTM, but core BIOS or other firmware is acceptable. In a system without a traditional RTM, the first component that boots would be considered the RTM, this is not preferred.

FPT_RDM_EXT.1 Removable Devices and Media

FPT_RDM_EXT.1.1

The TSF shall implement controls for handling the transfer of virtual and physical removable media and virtual and physical removable media devices between information domains.

FPT_RDM_EXT.1.2

The TSF shall enforce the following rules when [**assignment**: *virtual or physical removable media and virtual or physical removable media devices*] are switched between information domains, then [**selection**: *the Administrator has granted explicit access for the media or device to be connected to the receiving domain, the media in a device that is being transferred is ejected prior to the receiving domain being allowed access to the device, the user of the receiving domain expressly authorizes the connection, the device or media that is being transferred is prevented from being accessed by the receiving domain*]
Application Note: The purpose of these requirements is to ensure that VMs are not given inadvertent access to information from different domains because of media or removable media devices left connected to physical machines.

Removable media is media that can be ejected from a device, such as a compact disc, floppy disk, SD, or compact flash memory card.

Removable media devices are removable devices that include media, such as USB flash drives and USB hard drives. Removable media devices can themselves contain removable media (e.g., USB CDROM drives).

For purposes of this requirement, an Information Domain is:

a. A VM or collection of VMs
b. The Virtualization System
c. Host OS
d. Management Subsystem

These requirements also apply to virtualized removable media—such as virtual CD drives that connect to ISO images—as well as physical media—such as CDROMs and USB flash drives. In the case of virtual CDROMs, virtual ejection of the virtual media is sufficient.

In the first assignment, the ST author lists all removable media and removable media devices (both virtual and real) that are supported by the TOE. The ST author then selects actions that are appropriate for all removable media and removable media devices (both virtual and real) that are being claimed in the assignment.

For clarity, the ST author may iterate this requirement so that like actions are grouped with the removable media or devices to which they apply (e.g., the first iteration could contain all devices for which media is ejected on a switch; the second iteration could contain all devices for which access is prevented on a switch, etc.).

FPT_TUD_EXT.1 Trusted Updates to the Virtualization System

FPT_TUD_EXT.1.1

The TSF shall provide administrators the ability to query the currently executed version of the TOE firmware/software as well as the most recently installed version of the TOE firmware/software.
Application Note: The version currently running (being executed) may not be the version most recently installed. For instance, maybe the update was installed but the system requires a reboot before this update will run. Therefore, it needs to be clear that the query should indicate both the most recently executed version as well as the most recently installed update.

FPT_TUD_EXT.1.2

The TSF shall provide administrators the ability to manually initiate updates to TOE firmware/software and [**selection**: *automatic updates*, *no other update mechanism*].

FPT_TUD_EXT.1.3

The TSF shall provide means to authenticate firmware/software updates to the TOE using a [**selection**: *digital signature mechanism using certificates*, *digital signature mechanism not using certificates*, *published hash*] prior to installing those updates.
Application Note: The digital signature mechanism referenced in FPT_TUD_EXT.1.3 is one of the algorithms specified in FCS_COP.1/SIG.

If certificates are used by the update verification mechanism, then FIA_X509_EXT.1 and FIA_X509_EXT.2 must be included in the ST. Certificates are validated in accordance with FIA_X509_EXT.1 and the appropriate selections should be made in FIA_X509_EXT.2.1. Additionally, FPT_TUD_EXT.2 must be included in the ST.

“Update” in the context of this SFR refers to the process of replacing a non-volatile, system resident software component with another. The former is referred to as the NV image, and the latter is the update image. While the update image is typically newer than the NV image, this is not a requirement. There are legitimate cases where the system owner may want to rollback a

component to an older version (e.g., when the component manufacturer releases a faulty update, or when the system relies on an undocumented feature no longer present in the update). Likewise, the owner may want to update with the same version as the NV image to recover from faulty storage.

All discrete software components (e.g., applications, drivers, kernel, firmware) of the TSF, should be digitally signed by the corresponding manufacturer and subsequently verified by the mechanism performing the update. Since it is recognized that components may be signed by different manufacturers, it is essential that the update process verify that both the update and NV images were produced by the same manufacturer (e.g., by comparing public keys) or signed by legitimate signing keys (e.g., successful verification of certificates when using X.509 certificates).

The Digital Signature option is the preferred mechanism for authenticating updates. The Published Hash option will be removed from a future version of this PP.

FPT_VDP_EXT.1 Virtual Device Parameters

FPT_VDP_EXT.1.1

The TSF shall provide interfaces for virtual devices implemented by the VMM as part of the virtual hardware abstraction.

FPT_VDP_EXT.1.2

The TSF shall validate the parameters passed to the virtual device interface prior to execution of the VMM functionality exposed by those interfaces.
Application Note: The purpose of this requirement is to ensure that the VMM is not vulnerable to compromise through the processing of malformed data passed to the virtual device interface from a Guest OS. The VMM cannot assume that any data coming from a VM is well-formed—even if the virtual device interface is unique to the VS and the data comes from a virtual device driver supplied by the Virtualization Vendor.

FPT_VIV_EXT.1 VMM Isolation from VMs

FPT_VIV_EXT.1.1

The TSF must ensure that software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform.

FPT_VIV_EXT.1.2

The TSF must ensure that a Guest VM is unable to invoke platform code that runs at a privilege level equal to or exceeding that of the VMM without involvement of the VMM.
Application Note: This requirement is intended to ensure that software running within a Guest VM cannot compromise other VMs, the VMM, or the platform. This requirement is not met if Guest VM software—whatever its privilege level—can crash the VS or the Platform, or breakout of its virtual hardware abstraction to gain execution on the platform, within or outside of the context of the VMM.
This requirement is not violated if software running within a VM can crash the Guest OS and there is no way for an attacker to gain execution in the VMM or outside of the virtualized domain.

FPT_VIV_EXT.1.2 addresses several specific mechanisms that must not be permitted to bypass the VMM and invoke privileged code on the Platform.

At a minimum, the TSF should enforce the following:

- On the x86 platform, a virtual System Management Interrupt (SMI) cannot invoke platform System Management Mode (SMM).
- An attempt to update virtual firmware or virtual BIOS cannot cause physical platform firmware or physical platform BIOS to be modified.
- An attempt to update virtual firmware or virtual BIOS cannot cause the VMM to be modified.

Of the above, the first bullet does not apply to platforms that do not support SMM. The rationale behind the third bullet is that a firmware update of a single VM must not affect other VMs. So if multiple VMs share the same firmware image as part of a common hardware abstraction, then the update of a single machine's BIOS must not be allowed to change the common abstraction. The virtual hardware abstraction is part of the VMM.

6.1.7 TOE Access Banner (FTA)

FTA_TAB.1 TOE Access Banner

FTA_TAB.1.1

Before establishing an administrative user session, the TSF shall display a

security Administrator-specified advisory notice and consent warning message regarding use of the TOE.

Application Note: This requirement is intended to apply to interactive sessions between a human user and a TOE. IT entities establishing connections or programmatic connections (e.g., remote procedure calls over a network) are not required to be covered by this requirement.

6.1.8 Trusted Path/Channel (FTP)

FTP_ITC_EXT.1 Trusted Channel Communications

FTP_ITC_EXT.1.1

The TSF shall use [**selection**: *TLS as conforming to the Functional Package for Transport Layer Security*, *TLS/HTTPS as conforming to FCS_HTTPS_EXT.1*, *IPsec as conforming to FCS_IPSEC_EXT.1*, *SSH as conforming to the Functional Package for Secure Shell*] and [**selection**: *certificate-based authentication of the remote peer*, *non-certificate-based authentication of the remote peer*, *no authentication of the remote peer*] to provide a trusted communication channel between itself, and

- audit servers (as required by FAU_STG_EXT.1), and

[**selection**: *remote administrators (as required by FTP_TRP.1.1 if selected in FMT_MOF_EXT.1.1 in the Client or Server PP-Module)*, *separation of management and operational networks (if selected in FMT_SMO_EXT.1)*, *[**assignment**: other capabilities]*, *no other capabilities*] that is logically distinct from other communication paths and provides assured identification of its endpoints and protection of the communicated data from disclosure and detection of modification of the communicated data.

Application Note: If the ST author selects either TLS or HTTPS, the TSF shall be validated against the Functional Package for TLS. This PP does not mandate that a product implement TLS with mutual authentication, but if the product includes the capability to perform TLS with mutual authentication, then mutual authentication must be included within the TOE boundary. The TLS Package requires that the X509 requirements be included by the PP, so selection of TLS or HTTPS causes FIA_X509_EXT.* to be selected.

If the ST author selects SSH, the TSF shall be validated against the Functional Package for Secure Shell.

If the ST author selects "certificate-based authentication of the remote peer," then FIA_X509_EXT.1 and FIA_X509_EXT.2 must be included in the ST. "No authentication of the remote peer" should be selected only if the TOE is acting as a server in a non-mutual authentication configuration.

The ST author must include the security functional requirements for the trusted channel protocol selected in FTP_ITC_EXT.1 in the main body of the ST.

FTP_UIF_EXT.1 User Interface: I/O Focus

FTP_UIF_EXT.1.1

The TSF shall indicate to users which VM, if any, has the current input focus.

Application Note: This requirement applies to all users—whether User or Administrator. In environments where multiple VMs run at the same time, the user must have a way of knowing which VM user input is directed to at any given moment. This is especially important in multiple-domain environments.

In the case of a human user, this is usually a visual indicator. In the case of headless VMs, the user is considered to be a program, but this program still needs to know which VM it is sending input to; this would typically be accomplished through programmatic means.

FTP_UIF_EXT.2 User Interface: Identification of VM

FTP_UIF_EXT.2.1

The TSF shall support the unique identification of a VM's output display to users.

Application Note: In environments where a user has access to more than one VM at the same time, the user must be able to determine the identity of each VM displayed in order to avoid inadvertent cross-domain data entry.

There must be a mechanism for associating an identifier with a VM so that an application or program displaying the VM can identify the VM to users. This is generally indicated visually for human users (e.g., VM identity in the window title bar) and programmatically for headless VMs (e.g., an API function). The identification must be unique to the VS, but does not need to be universally unique.

6.2 Security Assurance Requirements

7 Consistency Rationale

Appendix A - Optional SFRs

A.1 Strictly Optional Requirements

A.1.1 Security Audit (FAU)

FAU_ARP.1 Security Audit Automatic Response

FAU_ARP.1.1

The TSF shall take [**assignment**: *list of actions*] upon detection of a potential security violation.
Application Note: In certain cases, it may be useful for Virtualization Systems to perform automated responses to certain security events. An example may include halting a VM which has taken some action to violate a key system security policy. This may be especially useful with headless endpoints when there is no human user in the loop.
The potential security violation mentioned in FAU_ARP.1.1 refers to FAU_SAA.1.

FAU_SAA.1 Potential Violation Analysis

FAU_SAA.1.1

The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the enforcement of the SFRs.

FAU_SAA.1.2

The TSF shall enforce the following rules for monitoring audited events:

a. Accumulation or combination of [**assignment**: *subset of defined auditable events*] known to indicate a potential security violation;
b. [**assignment**: *any other rules*].

Application Note: The potential security violation described in FAU_SAA.1 can be used as a trigger for automated responses as defined in FAU_ARP.1.

A.1.2 Protection of the TSF (FPT)

FPT_GVI_EXT.1 Guest VM Integrity

FPT_GVI_EXT.1.1

The TSF shall verify the integrity of Guest VMs through the following mechanisms: [**assignment**: *list of Guest VM integrity mechanisms*].
Application Note: The primary purpose of this requirement is to identify and describe the mechanisms used to verify the integrity of Guest VMs that have been 'imported' in some fashion, though these mechanisms could also be applied to all Guest VMs, depending on the mechanism used. Importation for this requirement could include VM migration (live or otherwise), the importation of virtual disk files that were previously exported, VMs in shared storage, etc. It is possible that a trusted VM could have been modified during the migration or import/export process, or VMs could have been obtained from untrusted sources in the first place, so integrity checks on these VMs can be a prudent measure to take. These integrity checks could be as thorough as making sure the entire VM exactly matches a previously known VM (by hash for example), or by simply checking certain configuration settings to ensure that the VM's configuration will not violate the security model of the VS.

A.2 Objective Requirements

A.2.1 Protection of the TSF (FPT)

FPT_DDI_EXT.1 Device Driver Isolation

FPT_DDI_EXT.1.1

The TSF shall ensure that device drivers for physical devices are isolated from the VMM and all other domains.
Application Note: In order to function on physical hardware, the VMM must have access to the device drivers for the physical platform on which it runs. These drivers are often written by third parties, and yet are effectively a part of the VMM. Thus the integrity of the VMM in part depends on the quality of third party code that the virtualization vendor has no control over. By encapsulating these drivers within one or more dedicated driver domains (e.g., Service VM or

VMs) the damage of a driver failure or vulnerability can be contained within the domain, and would not compromise the VMM. When driver domains have exclusive access to a physical device, hardware isolation mechanisms, such as Intel's VT-d, AMD's Input/Output Memory Management Unit (IOMMU), or ARM's System Memory Management Unit (MMU) should be used to ensure that operations performed by Direct Memory Access (DMA) hardware are properly constrained.

FPT_IDV_EXT.1 Software Identification and Versions

FPT_IDV_EXT.1.1

The TSF shall include software identification (SWID) tags that contain a SoftwareIdentity element and an Entity element as defined in ISO/IEC 19770-2:2009.

FPT_IDV_EXT.1.2

The TSF shall store SWIDs in a .swidtag file as defined in ISO/IEC 19770-2:2009.
Application Note: SWID tags are XML files embedded within software that provide a standard method for IT departments to track and manage the software. The presence of SWIDs can greatly simplify the software management process and improve security by enhancing the ability of IT departments to manage updates.

FPT_INT_EXT.1 Support for Introspection

FPT_INT_EXT.1.1

The TSF shall support a mechanism for permitting the VMM or privileged VMs to access the internals of another VM for purposes of introspection.
Application Note: Introspection can be used to support malware and anomaly detection from outside of the guest environment. This not only helps protect the Guest OS, it also protects the VS by providing an opportunity for the VS to detect threats to itself that originate within VMs, and that may attempt to break out of the VM and compromise the VMM or other VMs.

The hosting of malware detection software outside of the guest VM helps protect the guest and helps ensure the integrity of the malware detection/antivirus software. This capability can be implemented in the VMM itself, but ideally it should be hosted by a Service VM so that it can be better contained and does not introduce bugs into the VMM.

FPT_ML_EXT.1 Measured Launch of Platform and VMM

FPT_ML_EXT.1.1

The TSF shall support a measured launch of the Virtualization System. Measured components of the VS shall include the static executable image of the Hypervisor and: [**selection**: *Static executable images of the Management Subsystem*, *[**assignment**: list of (static images of) Service VMs]*, *[**assignment**: list of configuration files]*, *no other components*]

FPT_ML_EXT.1.2

The TSF shall make the measurements selected in FPT_ML_EXT.1.1 available to the Management Subsystem.
Application Note: A measured launch of the platform and VS demonstrates that the proper TOE software was loaded. A measured launch process employs verifiable integrity measurement mechanisms. For example, a VS may hash components such as the hypervisor, service VMs, or the Management Subsystem. A measured launch process only allows components to be executed after the measurement has been recorded. An example process may add each component’s hash before it is executed so that the final hash reflects the evidence of a component’s state prior to execution. The measurement may be verified as the system boots, but this is not required.

The Platform is outside of the TOE. However, this requirement specifies that the VS must be capable of receiving Platform measurements if the Platform provides them. This requirement is requiring TOE support for Platform measurements if provided; it is not placing a requirement on the Platform to take such measurements.

If available, hardware should be used to store measurements in such a manner that they cannot be modified in any manner except to be extended. These measurements should be produced in a repeatable manner so that a third party can verify the measurements if given the inputs. Hardware devices, like Trusted Platform Modules (TPM), TrustZone, and MMU are some examples that may serve as foundations for storing and reporting measurements.

Platforms with a root of trust for measurement (RTM) should initiate the measured launch process. This may include core BIOS or the chipset. The chipset is the preferred RTM, but core BIOS or other firmware is acceptable. In a system without a traditional RTM, the first component that boots would be

considered the RTM, this is not preferred.

A.3 Implementation-based Requirements

This PP does not define any Implementation-based SFRs.

Appendix B - Selection-based Requirements

B.1 Cryptographic Support (FCS)

FCS_HTTPS_EXT.1 HTTPS Protocol

The inclusion of this selection-based component depends upon selection from: FTP_ITC_EXT.1.1 FIA_X509_EXT.2.1.

FCS_HTTPS_EXT.1.1

The TSF shall implement the HTTPS protocol that complies with RFC 2818.
Application Note: This SFR is included in the ST if the ST Author selects "TLS/HTTPS" in FTP_ITC_EXT.1.1.

The ST author must provide enough detail to determine how the implementation is complying with the standards identified; this can be done either by adding elements to this component, or by additional detail in the TSS.

FCS_HTTPS_EXT.1.2

The TSF shall implement HTTPS using TLS.

FCS_IPSEC_EXT.1 IPsec Protocol

The inclusion of this selection-based component depends upon selection from: FTP_ITC_EXT.1.1 FIA_X509_EXT.2.1.

FCS_IPSEC_EXT.1.1

The TSF shall implement the IPsec architecture as specified in RFC 4301.
Application Note: This SFR is included in the ST if the ST Author selected "IPsec" in FTP_ITC_EXT.1.1.

RFC 4301 calls for an IPsec implementation to protect IP traffic through the use of a Security Policy Database (SPD). The SPD is used to define how IP packets are to be handled: PROTECT the packet (e.g., encrypt the packet), BYPASS the IPsec services (e.g., no encryption), or DISCARD the packet (e.g., drop the packet). The SPD can be implemented in various ways, including router access control lists, firewall rule-sets, a "traditional" SPD, etc. Regardless of the implementation details, there is a notion of a "rule" that a packet is "matched" against and a resulting action that takes place.

While there must be a means to order the rules, a general approach to ordering is not mandated, as long as the TOE can distinguish the IP packets and apply the rules accordingly. There may be multiple SPDs (one for each network interface), but this is not required.

FCS_IPSEC_EXT.1.2

The TSF shall implement [**selection**: *transport mode*, *tunnel mode*].
Application Note: If the TOE is used to connect to a VPN gateway for the purposes of establishing a secure connection to a private network, the ST author shall select tunnel mode. If the TOE uses IPsec to establish an end-to-end connection to another IPsec VPN Client, the ST author shall select transport mode. If the TOE uses IPsec to establish a connection to a specific endpoint device for the purpose of secure remote administration, the ST author shall select transport mode.

FCS_IPSEC_EXT.1.3

The TSF shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

FCS_IPSEC_EXT.1.4

The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms [AES-GCM-128, AES-GCM-256 (as specified in RFC 4106), [**selection**: *AES-CBC-128 (specified in RFC 3602)*, *AES-CBC-256 (specified in RFC 3602)*, *no other algorithms*]] together with a Secure Hash Algorithm (SHA)-based HMAC.

FCS_IPSEC_EXT.1.5

The TSF shall implement the protocol:
[**selection**: *IKEv1, using Main Mode for Phase 1 exchanges, as defined in RFC 2407, RFC 2408, RFC 2409, RFC 4109, [**selection**: no other RFCs for extended*

*sequence numbers, RFC 4304 for extended sequence numbers], [**selection**: no other RFCs for hash functions, RFC 4868 for hash functions], and [**selection, choose one of**: support for XAUTH, no support for XAUTH] IKEv2 as defined in RFC 7296 (with mandatory support for NAT traversal as specified in section 2.23), RFC 8784, RFC 8247, and [**selection**: no other RFCs for hash functions, RFC 4868 for hash functions].*]

Application Note: If the TOE implements SHA-2 hash algorithms for IKEv1 or IKEv2, the ST author shall select RFC 4868.

FCS_IPSEC_EXT.1.6

The TSF shall ensure the encrypted payload in the [**selection**: *IKEv1*, *IKEv2*] protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and [**selection**: *AES-GCM-128 as specified in RFC 5282*, *AES-GCM-256 as specified in RFC 5282*, *no other algorithm*].

FCS_IPSEC_EXT.1.7

The TSF shall ensure that [**selection**: *IKEv2 SA lifetimes can be configured by [**selection**: an Administrator, a VPN Gateway] based on [**selection**: number of packets/number of bytes, length of time] IKEv1 SA lifetimes can be configured by [**selection**: an Administrator, a VPN Gateway] based on [**selection**: number of packets/number of bytes, length of time] IKEv1 SA lifetimes are fixed based on [**selection**: number of packets/number of bytes, length of time]. If length of time is used, it must include at least one option that is 24 hours or less for Phase 1 SAs and 8 hours or less for Phase 2 SAs.*]

Application Note: The ST author is afforded a selection based on the version of IKE in their implementation. There is a further selection within this selection that allows the ST author to specify which entity is responsible for "configuring" the life of the SA. An implementation that allows an administrator to configure the client or a VPN gateway that pushes the SA lifetime down to the client are both acceptable.

As far as SA lifetimes are concerned, the TOE can limit the lifetime based on the number of bytes transmitted, or the number of packets transmitted. Either packet-based or volume-based SA lifetimes are acceptable; the ST author makes the appropriate selection to indicate which type of lifetime limits are supported.

The ST author chooses either the IKEv1 requirements or IKEv2 requirements (or both, depending on the selection in FCS_IPSEC_EXT.1.5. The IKEv1 requirement can be accomplished either by providing Authorized Administrator-configurable lifetimes (with appropriate instructions in documents mandated by AGD_OPE), or by "hard coding" the limits in the implementation. For IKEv2, there are no hard-coded limits, but in this case it is required that an administrator be able to configure the values. In general, instructions for setting the parameters of the implementation, including lifetime of the SAs, should be included in the operational guidance generated for AGD_OPE. It is appropriate to refine the requirement in terms of number of MB/KB instead of number of packets, as long as the TOE is capable of setting a limit on the amount of traffic that is protected by the same key (the total volume of all IPsec traffic protected by that key).

FCS_IPSEC_EXT.1.8

The TSF shall ensure that all IKE protocols implement DH groups [19 (256-bit Random ECP), 20 (384-bit Random ECP), and [**selection**: *24 (2048-bit MODP with 256-bit POS)*, *15 (3072-bit MODP)*, *14 (2048-bit MODP)*, *no other DH groups*]].

Application Note: The selection is used to specify additional DH groups supported. This applies to IKEv1 and IKEv2 exchanges. It should be noted that if any additional DH groups are specified, they must comply with the requirements (in terms of the ephemeral keys that are established) listed in FCS_CKM.1.

Since the implementation may allow different Diffie-Hellman groups to be negotiated for use in forming the SAs, the assignments in FCS_IPSEC_EXT.1.9 and FCS_IPSEC_EXT.1.10 may contain multiple values. For each DH group supported, the ST author consults Table 2 in 800-57 to determine the "bits of security" associated with the DH group. Each unique value is then used to fill in the assignment (for 1.9 they are doubled; for 1.10 they are inserted directly into the assignment). For example, suppose the implementation supports DH group 14 (2048-bit MODP) and group 20 (ECDH using NIST curve P-384). From Table 2, the bits of security value for group 14 is 112, and for group 20 it is 192. For FCS_IPSEC_EXT.1.9, then, the assignment would read "[224, 384]" and for FCS_IPSEC_EXT.1.10 it would read "[112, 192]" (although in this case the requirement should probably be refined so that it makes sense mathematically).

FCS_IPSEC_EXT.1.9

The TSF shall generate the secret value x used in the IKE Diffie-Hellman key exchange ("x" in gx mod p) using the random bit generator specified in FCS_RBG_EXT.1, and having a length of at least [**assignment**: *(one or more) number of bits that is at least twice the "bits of security" value associated with*

the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General] bits.

FCS_IPSEC_EXT.1.10

The TSF shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^[**assignment**: *(one or more) "bits of security" value associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General*].

FCS_IPSEC_EXT.1.11

The TSF shall ensure that all IKE protocols perform peer authentication using a [**selection**: *RSA*, *ECDSA*] that use X.509v3 certificates that conform to RFC 4945 and [**selection**: *Pre-shared Keys*, *no other method*].
Application Note: At least one public-key-based Peer Authentication method is required in order to conform to this PP-Module; one or more of the public key schemes is chosen by the ST author to reflect what is implemented. The ST author also ensures that appropriate FCS requirements reflecting the algorithms used (and key generation capabilities, if provided) are listed to support those methods. Note that the TSS will elaborate on the way in which these algorithms are to be used (for example, 2409 specifies three authentication methods using public keys; each one supported will be described in the TSS).
If "pre-shared keys" is selected, the selection-based requirement FIA_PSK_EXT.1 must be claimed.

FCS_IPSEC_EXT.1.12

The TSF shall not establish an SA if the [[**selection**: *IP address*, *Fully Qualified Domain Name (FQDN)*, *user FQDN*, *Distinguished Name (DN)*] and [**selection**: *no other reference identifier type*, *[**assignment**: other supported reference identifier types]*]] contained in a certificate does not match the expected values for the entity attempting to establish a connection.
Application Note: The TOE must support at least one of the following identifier types: IP address, Fully Qualified Domain Name (FQDN), user FQDN, or Distinguished Name (DN). In the future, the TOE will be required to support all of these identifier types. The TOE is expected to support as many IP address formats (IPv4 and IPv6) as IP versions supported by the TOE in general. The ST author may assign additional supported identifier types in the second selection.

FCS_IPSEC_EXT.1.13

The TSF shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.
Application Note: At this time, only the comparison between the presented identifier in the peer's certificate and the peer's reference identifier is mandated by the testing below. However, in the future, this requirement will address two aspects of the peer certificate validation: 1) comparison of the peer's ID payload to the peer's certificate which are both presented identifiers, as required by RFC 4945 and 2) verification that the peer identified by the ID payload and the certificate is the peer expected by the TOE (per the reference identifier). At that time, the TOE will be required to demonstrate both aspects (i.e. that the TOE enforces that the peer's ID payload matches the peer's certificate which both match configured peer reference identifiers).
Excluding the DN identifier type (which is necessarily the Subject DN in the peer certificate), the TOE may support the identifier in either the Common Name or Subject Alternative Name (SAN) or both. If both are supported, the preferred logic is to compare the reference identifier to a presented SAN, and only if the peer's certificate does not contain a SAN, to fall back to a comparison against the Common Name. In the future, the TOE will be required to compare the reference identifier to the presented identifier in the SAN only, ignoring the Common Name.

The configuration of the peer reference identifier is addressed by FMT_SMF.1.1/VPN.

FCS_IPSEC_EXT.1.14

The [**selection**: *TSF*, *VPN Gateway*] shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [**selection**: *IKEv1 Phase 1*, *IKEv2 IKE_SA*] connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [selection: IKEv1 Phase 2, IKEv2 CHILD_SA] connection.
Application Note: If this functionality is configurable, the TSF may be configured by a VPN Gateway or by an Administrator of the TOE itself.
The ST author chooses either or both of the IKE selections based on what is implemented by the TOE. Obviously, the IKE versions chosen should be consistent not only in this element, but with other choices for other elements in

this component. While it is acceptable for this capability to be configurable, the default configuration in the evaluated configuration (either "out of the box" or by configuration guidance in the AGD documentation) must enable this functionality.

B.2 Identification and Authentication (FIA)

FIA_PMG_EXT.1 Password Management

The inclusion of this selection-based component depends upon selection from: FIA_UAU.5.1.

FIA_PMG_EXT.1.1

The TSF shall provide the following password management capabilities for administrative passwords:

a. Passwords shall be able to be composed of any combination of upper and lower case characters, digits, and the following special characters: [**selection**: *“!”, “@”, “#”, “$”, “%”, “^”, “& ”, “*”, “(“, “)”, [**assignment**: other characters]*]
b. Minimum password length shall be configurable
c. Passwords of at least 15 characters in length shall be supported

Application Note: This SFR is included in the ST if the ST Author selects ‘authentication based on username and password’ in FIA_UAU.5.1.

The ST author selects the special characters that are supported by the TOE; they may optionally list additional special characters supported using the assignment. “Administrative passwords” refers to passwords used by administrators to gain access to the Management Subsystem.

FIA_X509_EXT.1 X.509 Certificate Validation

The inclusion of this selection-based component depends upon selection from: FPT_TUD_EXT.1.3 FIA_UAU.5.1 FTP_ITC_EXT.1.1.

FIA_X509_EXT.1.1

The TSF shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation
- The certificate path must terminate with a trusted certificate
- The TOE shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The TSF shall validate that any CA certificate includes caSigning purpose in the key usage field
- The TSF shall validate revocation status of the certificate using [**selection**: *OCSP as specified in RFC 6960, a CRL as specified in RFC 5759, an OCSP TLS Status Request Extension (OCSP stapling) as specified in RFC 6066, OCSP TLS Multi-Certificate Status Request Extension (i.e., OCSP Multi-Stapling) as specified in RFC 6961*].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
  - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing Purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the EKU field.
  - OCSP certificates presented for OCSP responses shall have the OCSP Signing Purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field.

Application Note: This SFR must be included in the ST if the selection for FPT_TUD_EXT.1.3 is “digital signature mechanism,” if "certificate-based authentication of the remote peer" is selected in FTP_ITC_EXT.1.1, or if "authentication based on X.509 certificates" is selected in FIA_UAU.5.1.

FIA_X509_EXT.1.1 lists the rules for validating certificates. The ST author shall select whether revocation status is verified using OCSP or CRLs.

FIA_X509_EXT.2 requires that certificates are used for IPsec; this use requires that the extendedKeyUsage rules are verified. Certificates may optionally be used for SSH, TLS, and HTTPs and, if implemented, must be validated to contain the corresponding extendedKeyUsage.

OCSP stapling and OCSP multi-stapling support only TLS server certificate validation. If other certificate types are validated, either OCSP or CRL must be claimed. If OCSP is not supported the EKU provision for checking the OCSP Signing purpose is met by default.

Regardless of the selection of TSF or TOE platform, the validation must result in a trusted root CA certificate in a root store managed by the platform.
OCSP responses are signed using either the certificate's issuer's CA certificate or an OCSP certificate issued to an OCSP responder delegated by that issuer to sign OCSP responses. A compliant TOE is able to validate OCSP responses in either case, but the OCSP signing extended key usage purpose is only required to be checked in OCSP certificates.

FIA_X509_EXT.1.2

The TSF shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.
Application Note: This requirement applies to certificates that are used and processed by the TSF and restricts the certificates that may be added as trusted CA certificates.

FIA_X509_EXT.2 X.509 Certificate Authentication

The inclusion of this selection-based component depends upon selection from: FPT_TUD_EXT.1.3 FIA_UAU.5.1 FTP_ITC_EXT.1.1.

FIA_X509_EXT.2.1

The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [**selection**: *IPsec*, *TLS*, *HTTPS*, *SSH*, *code signing for system software updates*, *[**assignment**: other uses]*]
Application Note: This SFR must be included in the ST if the selection for FPT_TUD_EXT.1.3 is "digital signature mechanism," if "certificate-based authentication of the remote peer" is selected in FTP_ITC_EXT.1, or if "authentication based on X.509 certificates" is selected in FIA_UAU.5.1.
This SFR must also be included in the ST if X.509 certificate-based authentication is used for "other uses" as listed in the assignment in FIA_X509_EXT.2.1.

FIA_X509_EXT.2.2

When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall [**selection, choose one of**: *allow the administrator to choose whether to accept the certificate in these cases*, *accept the certificate*, *not accept the certificate*].
Application Note: Often a connection must be established to check the revocation status of a certificate - either to download a CRL or to perform a lookup using OCSP. The selection is used to describe the behavior in the event that such a connection cannot be established (for example, due to a network error). If the TOE has determined the certificate valid according to all other rules in FIA_X509_EXT.1, the behavior indicated in the selection shall determine the validity. The TOE must not accept the certificate if it fails any of the other validation rules in FIA_X509_EXT.1. If the administrator-configured option is selected by the ST Author, the ST Author must ensure that this is also defined as a management function that is provided by the TOE.

B.3 Protection of the TSF (FPT)

FPT_TUD_EXT.2 Trusted Update Based on Certificates

The inclusion of this selection-based component depends upon selection from: FPT_TUD_EXT.1.3 FIA_X509_EXT.2.1.

FPT_TUD_EXT.2.1

The TSF shall not install an update if the code signing certificate is deemed invalid.
Application Note: Certificates may optionally be used for code signing of system software updates (FPT_TUD_EXT.1.3). This element must be included in the ST if certificates are used for validating updates. If "code signing for system

software updates” is selected in FIA_X509_EXT.2.1, FPT_TUD_EXT.2 must be included in the ST.
Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with FIA_X509_EXT.1.

B.4 Trusted Path/Channel (FTP)

FTP_TRP.1 Trusted Path

The inclusion of this selection-based component depends upon selection from:.

FTP_TRP.1.1

The TSF shall **use a trusted channel as specified in FTP_ITC_EXT.1 to** provide a **trusted** communication path between itself and [*remote*] **administrators** that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from [*modification, disclosure*].

FTP_TRP.1.2

The TSF shall permit [*remote administrators*] to initiate communication via the trusted path.

FTP_TRP.1.3

The TSF shall require the use of the trusted path for [*[all remote administration actions]*].
Application Note: This SFR is included in the ST if "remote" is selected in FMT_MOF_EXT.1.1 of the client or server PP-Module.
Protocols used to implement the remote administration trusted channel must be selected in FTP_ITC_EXT.1.

This requirement ensures that authorized remote administrators initiate all communication with the TOE via a trusted path, and that all communications with the TOE by remote administrators is performed over this path. The data passed in this trusted communication channel are encrypted as defined the protocol chosen in the first selection in FTP_ITC_EXT.1. The ST author chooses the mechanism or mechanisms supported by the TOE, and then ensures that the detailed requirements in Appendix B corresponding to their selection are copied to the ST if not already present.

Appendix C - Extended Component Definitions

This appendix contains the definitions for all extended requirements specified in the Protection Profile.

C.1 Extended Components Table

All extended components specified in the Protection Profile are listed in this table:

Table 3: Extended Component Definitions

Functional Class	Functional Components
Cryptographic Support (FCS)	
	FCS_CKM_EXT - Cryptographic Key Management
	FCS_ENT_EXT - Entropy for Virtual Machines
	FCS_HTTPS_EXT - HTTPS Protocol
	FCS_IPSEC_EXT - IPsec Protocol
	FCS_RBG_EXT - Cryptographic Operation (Random Bit Generation)
Identification and Authentication (FIA)	
	FIA_AFL_EXT - Authentication Failure Handling
	FIA_PMG_EXT - Password Management
	FIA_UIA_EXT - Administrator Identification and Authentication
	FIA_X509_EXT - X.509 Certificate
Protection of the TSF (FPT)	
	FPT_DDI_EXT - Device Driver Isolation
	FPT_DVD_EXT - Non-Existence of Disconnected Virtual Devices
	FPT_EEM_EXT - Execution Environment Mitigations
	FPT_GVI_EXT - Guest VM Integrity
	FPT_HAS_EXT - Hardware Assists
	FPT_HCL_EXT - Hypercall Controls
	FPT_IDV_EXT - Software Identification and Versions
	FPT_INT_EXT - Support for Introspection
	FPT_ML_EXT - Measured Launch of Platform and VMM
	FPT_RDM_EXT - Removable Devices and Media
	FPT_TUD_EXT - Trusted Updates
	FPT_VDP_EXT - Virtual Device Parameters
	FPT_VIV_EXT - VMM Isolation from VMs
Security Audit (FAU)	
	FAU_STG_EXT - Off-Loading of Audit Data
Security Management (FMT)	
	FMT_SMO_EXT - Separation of Management and Operational Networks
Trusted Path/Channel (FTP)	
	FTP_ITC_EXT - Trusted Channel Communications
	FTP_UIF_EXT - User Interface

User Data Protection (FDP)	
	FDP_HBI_EXT - Hardware-Based Isolation Mechanisms
	FDP_PPR_EXT - Physical Platform Resource Controls
	FDP_RIP_EXT - Residual Information in Memory
	FDP_VMS_EXT - VM Separation
	FDP_VNC_EXT - Virtual Networking Components

C.2 Extended Component Definitions

C.2.1 Cryptographic Support (FCS)

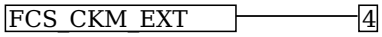
This Protection Profile defines the following extended components as part of the FCS class originally defined by CC Part 2:

C.2.1.1 FCS_CKM_EXT Cryptographic Key Management

Family Behavior

This family defines requirements for management of cryptographic keys.

Component Leveling



FCS_CKM_EXT.4, Cryptographic Key Destruction, requires the TSF to destroy or make unrecoverable empty keys in volatile and non-volatile memory. Note that component level 4 is used here because of this component's similarity to the CC Part 2 component FCS_CKM.4.

Management: FCS_CKM_EXT.4

No specific management functions are identified.

Audit: FCS_CKM_EXT.4

There are no auditable events foreseen.

FCS_CKM_EXT.4 Cryptographic Key Destruction

Hierarchical to: No other components.

Dependencies to: [FCS_CKM.1 Cryptographic Key Generation, or

FCS_CKM.2 Cryptographic Key Distribution]

FCS_CKM_EXT.4.1

The TSF shall cause disused cryptographic keys in volatile memory to be destroyed or rendered unrecoverable.

FCS_CKM_EXT.4.2

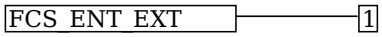
The TSF shall cause disused cryptographic keys in non-volatile storage to be destroyed or rendered unrecoverable.
This Protection Profile defines the following extended components as part of the FCS class originally defined by CC Part 2:

C.2.1.2 FCS_ENT_EXT Entropy for Virtual Machines

Family Behavior

This family defines requirements for availability of entropy data generated or collected by the TSF.

Component Leveling



FCS_ENT_EXT.1, Entropy for Virtual Machines, requires the TSF to provide entropy data to VMs in a specified manner.

Management: FCS_ENT_EXT.1

No specific management functions are identified.

Audit: FCS_ENT_EXT.1

There are no auditable events foreseen.

FCS_ENT_EXT.1 Entropy for Virtual Machines

Hierarchical to: No other components.

Dependencies to: FCS_RBG_EXT.1 Cryptographic Operation (Random Bit Generation)

FCS_ENT_EXT.1.1

The TSF shall provide a mechanism to make available to VMs entropy that meets FCS_RBG_EXT.1 through [**selection**: *Hypercall interface*, *virtual device interface*, *pass-through access to hardware entropy source*].

FCS_ENT_EXT.1.2

The TSF shall provide independent entropy across multiple VMs.
This Protection Profile defines the following extended components as part of the FCS class originally defined by CC Part 2:

C.2.1.3 FCS_HTTPS_EXT HTTPS Protocol

Family Behavior

This family defines requirements for protecting remote management sessions between the TOE and a Security Administrator. This family describes how HTTPS will be implemented.

Component Leveling

FCS_HTTPS_EXT.1, HTTPS Protocol, defines requirements for the implementation of the HTTPS protocol.

Management: FCS_HTTPS_EXT.1

No specific management functions are identified.

Audit: FCS_HTTPS_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Failure to establish an HTTPS session.
b. Establishment/termination of an HTTPS session.

FCS_HTTPS_EXT.1 HTTPS Protocol

Hierarchical to: No other components.

Dependencies to: [FCS_TLSC_EXT.1 TLS Client Protocol, or

FCS_TLSC_EXT.2 TLS Client Protocol with Mutual Authentication, or

FCS_TLSS_EXT.1 TLS Server Protocol, or

FCS_TLSS_EXT.2 TLS Server Protocol with Mutual Authentication]

FCS_HTTPS_EXT.1.1

The TSF shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2

The TSF shall implement HTTPS using TLS.
This Protection Profile defines the following extended components as part of the FCS class originally defined by CC Part 2:

C.2.1.4 FCS_IPSEC_EXT IPsec Protocol

Family Behavior

This family defines requirements for protecting communications using IPsec.

Component Leveling

FCS_IPSEC_EXT.1, IPsec Protocol, requires that IPsec be implemented as specified.

Management: FCS_IPSEC_EXT.1

No specific management functions are identified.

Audit: FCS_IPSEC_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Failure to establish an IPsec SA.
b. Establishment/Termination of an IPsec SA.

FCS_IPSEC_EXT.1 IPsec Protocol

Hierarchical to: No other components.

Dependencies to: FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.2 Cryptographic Key Establishment

FCS_COP.1 Cryptographic Operation

FCS_RBG_EXT.1 Cryptographic Operation (Random Bit Generation)

FIA_X509_EXT.1 X.509 Certificate Validation

FCS_IPSEC_EXT.1.1

The TSF shall implement the IPsec architecture as specified in RFC 4301.

FCS_IPSEC_EXT.1.2

The TSF shall implement [**selection**: *transport mode*, *tunnel mode*].

FCS_IPSEC_EXT.1.3

The TSF shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

FCS_IPSEC_EXT.1.4

The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms [AES-GCM-128, AES-GCM-256 (as specified in RFC 4106), [**selection**: *AES-CBC-128 (specified in RFC 3602)*, *AES-CBC-256 (specified in RFC 3602)*, *no other algorithms*]] together with a Secure Hash Algorithm (SHA)-based HMAC.

FCS_IPSEC_EXT.1.5

The TSF shall implement the protocol:
[**selection**: *IKEv1, using Main Mode for Phase 1 exchanges, as defined in RFC 2407, RFC 2408, RFC 2409, RFC 4109, [**selection**: no other RFCs for extended sequence numbers, RFC 4304 for extended sequence numbers], [**selection**: no other RFCs for hash functions, RFC 4868 for hash functions], and [**selection, choose one of**: support for XAUTH, no support for XAUTH] IKEv2 as defined in RFC 7296 (with mandatory support for NAT traversal as specified in section 2.23), RFC 8784, RFC 8247, and [**selection**: no other RFCs for hash functions, RFC 4868 for hash functions].*]

FCS_IPSEC_EXT.1.6

The TSF shall ensure the encrypted payload in the [**selection**: *IKEv1*, *IKEv2*] protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and [**selection**: *AES-GCM-128 as specified in RFC 5282*, *AES-GCM-256 as specified in RFC 5282*, *no other algorithm*].

FCS_IPSEC_EXT.1.7

The TSF shall ensure that [**selection**: *IKEv2 SA lifetimes can be configured by [**selection**: an Administrator, a VPN Gateway] based on [**selection**: number of packets/number of bytes, length of time] IKEv1 SA lifetimes can be configured by [**selection**: an Administrator, a VPN Gateway] based on [**selection**: number of packets/number of bytes, length of time] IKEv1 SA lifetimes are fixed based on [**selection**: number of packets/number of bytes, length of time]. If length of time is used, it must include at least one option that is 24 hours or less for Phase 1 SAs and 8 hours or less for Phase 2 SAs.*]

FCS_IPSEC_EXT.1.8

The TSF shall ensure that all IKE protocols implement DH groups [19 (256-bit Random ECP), 20 (384-bit Random ECP), and [**selection**: *24 (2048-bit MODP with 256-bit POS)*, *15 (3072-bit MODP)*, *14 (2048-bit MODP)*, *no other DH groups*]].

FCS_IPSEC_EXT.1.9

The TSF shall generate the secret value x used in the IKE Diffie-Hellman key exchange (“x” in gx mod p) using

the random bit generator specified in FCS_RBG_EXT.1, and having a length of at least [**assignment**: *(one or more) number of bits that is at least twice the "bits of security" value associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General*] bits.

FCS_IPSEC_EXT.1.10

The TSF shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^[**assignment**: *(one or more) "bits of security" value associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, Recommendation for Key Management – Part 1: General*].

FCS_IPSEC_EXT.1.11

The TSF shall ensure that all IKE protocols perform peer authentication using a [**selection**: *RSA*, *ECDSA*] that use X.509v3 certificates that conform to RFC 4945 and [**selection**: *Pre-shared Keys*, *no other method*].

FCS_IPSEC_EXT.1.12

The TSF shall not establish an SA if the [[**selection**: *IP address*, *Fully Qualified Domain Name (FQDN)*, *user FQDN*, *Distinguished Name (DN)*] and [**selection**: *no other reference identifier type*, *[**assignment**: other supported reference identifier types]*]] contained in a certificate does not match the expected values for the entity attempting to establish a connection.

FCS_IPSEC_EXT.1.13

The TSF shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.

FCS_IPSEC_EXT.1.14

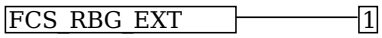
The [**selection**: *TSF*, *VPN Gateway*] shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [**selection**: *IKEv1 Phase 1*, *IKEv2 IKE_SA*] connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [selection: IKEv1 Phase 2, IKEv2 CHILD_SA] connection. This Protection Profile defines the following extended components as part of the FCS class originally defined by CC Part 2:

C.2.1.5 FCS_RBG_EXT Cryptographic Operation (Random Bit Generation)

Family Behavior

This family defines requirements for random bit/number generation.

Component Leveling



FCS_RBG_EXT.1, Cryptographic Operation (Random Bit Generation), requires random bit generation to be performed in accordance with selected standards and seeded by an entropy source.

Management: FCS_RBG_EXT.1

No specific management functions are identified.

Audit: FCS_RBG_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Failure of the randomization process.

FCS_RBG_EXT.1 Cryptographic Operation (Random Bit Generation)

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FCS_RBG_EXT.1.1

The TSF shall perform all deterministic random bit generation services in accordance with NIST Special Publication 800-90A using [**selection**: *Hash_DRBG (any)*, *HMAC_DRBG (any)*, *CTR_DRBG (AES)*]

FCS_RBG_EXT.1.2

The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [**selection**: *a software-based noise source*, *a hardware-based noise source*] with a minimum of [**selection**: *128 bits*, *192 bits*, *256 bits*] of entropy at least equal to the greatest security strength according to NIST SP 800-57, of the keys and hashes that it will generate.

C.2.2 Identification and Authentication (FIA)

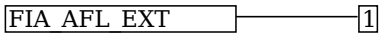
This Protection Profile defines the following extended components as part of the FIA class originally defined by CC Part 2:

C.2.2.1 FIA_AFL_EXT Authentication Failure Handling

Family Behavior

This family defines requirements for detection and prevention of brute force authentication attempts.

Component Leveling



FIA_AFL_EXT.1, Authentication Failure Handling, requires the TSF to lock an administrator account when an excessive number of failed authentication attempts have been observed until some restorative event occurs to enable the account.

Management: FIA_AFL_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure lockout policy through unsuccessful authentication attempts.

Audit: FIA_AFL_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Unsuccessful login attempts limit is met or exceeded.

FIA_AFL_EXT.1 Authentication Failure Handling

Hierarchical to: No other components.

Dependencies to: FIA_UIA_EXT.1 Administrator Identification and Authentication

FMT_SMR.1 Security Roles

FIA_AFL_EXT.1.1

The TSF shall detect when [**selection**: *[**assignment**: a positive integer number], an administrator configurable positive integer within a [**assignment**: range of acceptable values]*] unsuccessful authentication attempts occur related to Administrators attempting to authenticate remotely using [**selection**: *username and password*, *username and PIN*].

FIA_AFL_EXT.1.2

When the defined number of unsuccessful authentication attempts has been met, the TSF shall: [**selection**: *prevent the offending Administrator from successfully establishing a remote session using any authentication method that involves a password or PIN until [**assignment**: action to unlock] is taken by an Administrator, prevent the offending Administrator from successfully establishing a remote session using any authentication method that involves a password or PIN until an Administrator-defined time period has elapsed*]

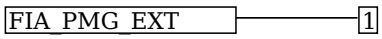
This Protection Profile defines the following extended components as part of the FIA class originally defined by CC Part 2:

C.2.2.2 FIA_PMG_EXT Password Management

Family Behavior

This family defines requirements for the composition of administrator passwords.

Component Leveling



FIA_PMG_EXT.1, Password Management, requires the TSF to ensure that administrator passwords meet a defined password policy.

Management: FIA_PMG_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure Administrator password policy, including the ability to change default authorization factors.

Audit: FIA_PMG_EXT.1

There are no auditable events foreseen.

FIA_PMG_EXT.1 Password Management

Hierarchical to: No other components.

Dependencies to: FIA_UIA_EXT.1 Administrator Identification and Authentication

FIA_PMG_EXT.1.1

The TSF shall provide the following password management capabilities for administrative passwords:

a. Passwords shall be able to be composed of any combination of upper and lower case characters, digits, and the following special characters: [**selection**: *"!", "@", "#", "$", "%", "^", "& ", "*", "(", ")", [**assignment**: other characters]*]
b. Minimum password length shall be configurable
c. Passwords of at least 15 characters in length shall be supported

This Protection Profile defines the following extended components as part of the FIA class originally defined by CC Part 2:

C.2.2.3 FIA_UIA_EXT Administrator Identification and Authentication

Family Behavior

This family defines requirements for ensuring that access to the TSF is not granted to unauthenticated subjects.

Component Leveling

FIA_UIA_EXT.1, Administrator Identification and Authentication, requires the TSF to ensure that all subjects attempting to perform TSF-mediated actions are identified and authenticated prior to authorizing these actions to be performed.

Management: FIA_UIA_EXT.1

No specific management functions are identified.

Audit: FIA_UIA_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Administrator authentication attempts.
b. All use of the identification and authentication mechanism.
c. Administrator session start time and end time.

FIA_UIA_EXT.1 Administrator Identification and Authentication

Hierarchical to: No other components.

Dependencies to: FIA_UAU.5 Multiple Authentication Mechanisms

FIA_UIA_EXT.1.1

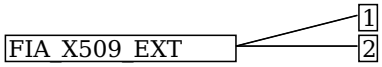
The TSF shall require Administrators to be successfully identified and authenticated using one of the methods in FIA_UAU.5 before allowing any TSF-mediated management function to be performed by that Administrator.
This Protection Profile defines the following extended components as part of the FIA class originally defined by CC Part 2:

C.2.2.4 FIA_X509_EXT X.509 Certificate

Family Behavior

This family defines requirements for the validation and use of X.509 certificates.

Component Leveling



FIA_X509_EXT.1, X.509 Certificate Validation, defines how the TSF must validate X.509 certificates that are presented to it.

FIA_X509_EXT.2, X.509 Certificate Authentication, requires the TSF to identify the functions for which it uses X.509 certificates for authentication

Management: FIA_X509_EXT.1

The following actions could be considered for the management functions in FMT:

a. Configuration of action to take if unable to determine the validity of a certificate.

Audit: FIA_X509_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Failure to validate a certificate.

FIA_X509_EXT.1 X.509 Certificate Validation

Hierarchical to: No other components.

Dependencies to: FPT_STM.1 Reliable Time Stamps

FIA_X509_EXT.1.1

The TSF shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation
- The certificate path must terminate with a trusted certificate
- The TOE shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The TSF shall validate that any CA certificate includes caSigning purpose in the key usage field
- The TSF shall validate revocation status of the certificate using [**selection**: *OCSP as specified in RFC 6960, a CRL as specified in RFC 5759, an OCSP TLS Status Request Extension (OCSP stapling) as specified in RFC 6066, OCSP TLS Multi-Certificate Status Request Extension (i.e., OCSP Multi-Stapling) as specified in RFC 6961*].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
  - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing Purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the EKU field.
  - OCSP certificates presented for OCSP responses shall have the OCSP Signing Purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field.

FIA_X509_EXT.1.2

The TSF shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

Management: FIA_X509_EXT.2

The following actions could be considered for the management functions in FMT:

a. Configuration of TSF behavior when certificate revocation status cannot be determined.

Audit: FIA_X509_EXT.2

There are no auditable events foreseen.

FIA_X509_EXT.2 X.509 Certificate Authentication

Hierarchical to: No other components.

Dependencies to: FIA_X509_EXT.1 X.509 Certificate Validation

FTP_ITC_EXT.1 Trusted Channel Communications

FIA_X509_EXT.2.1

[**assignment**: *secure transport protocols*][**assignment**: *other uses*]

FIA_X509_EXT.2.2

[**assignment**: *action to take*]

C.2.3 Protection of the TSF (FPT)

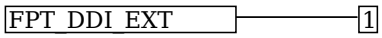
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.1 FPT_DDI_EXT Device Driver Isolation

Family Behavior

This family defines requirements for isolation of device drivers

Component Leveling



FPT_DDI_EXT.1, Device Driver Isolation, requires the TSF to isolate device drivers for physical devices from all virtual domains.

Management: FPT_DDI_EXT.1

No specific management functions are identified.

Audit: FPT_DDI_EXT.1

There are no auditable events foreseen.

FPT_DDI_EXT.1 Device Driver Isolation

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_DDI_EXT.1.1

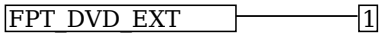
The TSF shall ensure that device drivers for physical devices are isolated from the VMM and all other domains.
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.2 FPT_DVD_EXT Non-Existence of Disconnected Virtual Devices

Family Behavior

This family defines requirements for ensuring that Guest VMs cannot access the virtual hardware interfaces disabled or disconnected virtual devices.

Component Leveling



FPT_DVD_EXT.1, Non-Existence of Disconnected Virtual Devices, requires the TSF to prevent Guest VMs from accessing virtual devices that it is not configured to have access to.

Management: FPT_DVD_EXT.1

No specific management functions are identified.

Audit: FPT_DVD_EXT.1

There are no auditable events foreseen.

FPT_DVD_EXT.1 Non-Existence of Disconnected Virtual Devices

Hierarchical to: No other components.

Dependencies to: FPT_VDP_EXT.1 Virtual Device Parameters

FPT_DVD_EXT.1.1

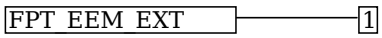
The TSF shall prevent Guest VMs from accessing virtual device interfaces that are not present in the VM's current virtual hardware configuration.
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.3 FPT_EEM_EXT Execution Environment Mitigations

Family Behavior

This family defines requirements for the TOE's compatibility with platform mechanisms that prevent vulnerabilities that allow for the execution of unauthorized code or bypass of access restrictions on memory or storage.

Component Leveling



FPT_EEM_EXT.1, Execution Environment Mitigations, requires the TSF to identify the execution environment-based protection mechanisms that it can use for self-protection.

Management: FPT_EEM_EXT.1

No specific management functions are identified.

Audit: FPT_EEM_EXT.1

There are no auditable events foreseen.

FPT_EEM_EXT.1 Execution Environment Mitigations

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_EEM_EXT.1.1

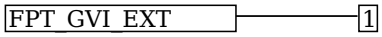
The TSF shall take advantage of execution environment-based vulnerability mitigation mechanisms supported by the Platform such as: [**selection**: *Address space randomization, Memory execution protection (e.g., DEP), Stack buffer overflow protection, Heap corruption detection, [**assignment**: other mechanisms], No mechanisms*]
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.4 FPT_GVI_EXT Guest VM Integrity

Family Behavior

This family defines requirements for the TOE to assert the integrity of Guest VMs.

Component Leveling



FPT_GVI_EXT.1, Guest VM Integrity, requires the TSF to specify the mechanisms it uses to verify the integrity of Guest VMs.

Management: FPT_GVI_EXT.1

No specific management functions are identified.

Audit: FPT_GVI_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Actions taken due to failed integrity check.

FPT_GVI_EXT.1 Guest VM Integrity

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_GVI_EXT.1.1

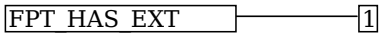
The TSF shall verify the integrity of Guest VMs through the following mechanisms: [**assignment**: *list of Guest VM integrity mechanisms*].
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.5 FPT_HAS_EXT Hardware Assists

Family Behavior

This family defines requirements for use of hardware-based virtualization assists as performance enhancements.

Component Leveling



FPT_HAS_EXT.1, Hardware Assists, requires the TSF to identify the hardware assists it uses to reduce TOE complexity.

Management: FPT_HAS_EXT.1

No specific management functions are identified.

Audit: FPT_HAS_EXT.1

There are no auditable events foreseen.

FPT_HAS_EXT.1 Hardware Assists

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_HAS_EXT.1.1

The VMM shall use [**assignment**: *list of hardware-based virtualization assists*] to reduce or eliminate the need for binary translation.

FPT_HAS_EXT.1.2

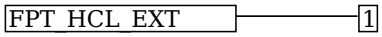
The VMM shall use [**assignment**: *list of hardware-based virtualization memory-handling assists*] to reduce or eliminate the need for shadow page tables.
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.6 FPT_HCL_EXT Hypercall Controls

Family Behavior

This family defines requirements for control of Hypercall interfaces.

Component Leveling



FPT_HCL_EXT.1, Hypercall Controls, requires the TSF to implement appropriate parameter validation to protect the VMM from unauthorized access through a hypercall interface.

Management: FPT_HCL_EXT.1

No specific management functions are identified.

Audit: FPT_HCL_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Invalid parameter to hypercall detected.
b. Hypercall interface invoked when documented preconditions are not met.

FPT_HCL_EXT.1 Hypercall Controls

Hierarchical to: No other components.

Dependencies to: FMT_SMR.1 Security Roles

FPT_HCL_EXT.1.1

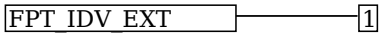
The TSF shall validate the parameters passed to Hypercall interfaces prior to execution of the VMM functionality exposed by each interface.
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.7 FPT_IDV_EXT Software Identification and Versions

Family Behavior

This family defines requirements for the use of SWID tags to identify the TOE.

Component Leveling



FPT_IDV_EXT.1, Software Identification and Versions, requires the TSF to identify itself using SWID tags.

Management: FPT_IDV_EXT.1

No specific management functions are identified.

Audit: FPT_IDV_EXT.1

There are no auditable events foreseen.

FPT_IDV_EXT.1 Software Identification and Versions

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_IDV_EXT.1.1

The TSF shall include software identification (SWID) tags that contain a SoftwareIdentity element and an Entity element as defined in ISO/IEC 19770-2:2009.

FPT_IDV_EXT.1.2

The TSF shall store SWIDs in a .swidtag file as defined in ISO/IEC 19770-2:2009.
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.8 FPT_INT_EXT Support for Introspection

Family Behavior

This family defines requirements for supporting VM introspection.

Component Leveling

FPT_INT_EXT.1, Support for Introspection, requires the TSF to support introspection.

Management: FPT_INT_EXT.1

No specific management functions are identified.

Audit: FPT_INT_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Introspection initiated/enabled.

FPT_INT_EXT.1 Support for Introspection

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_INT_EXT.1.1

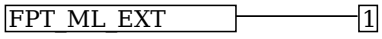
The TSF shall support a mechanism for permitting the VMM or privileged VMs to access the internals of another VM for purposes of introspection.
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.9 FPT_ML_EXT Measured Launch of Platform and VMM

Family Behavior

This family defines requirements for measured launch.

Component Leveling



FPT_ML_EXT.1, Measured Launch of Platform and VMM, requires the TSF to support a measured launch of itself.

Management: FPT_ML_EXT.1

No specific management functions are identified.

Audit: FPT_ML_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Integrity measurements collected.

FPT_ML_EXT.1 Measured Launch of Platform and VMM

Hierarchical to: No other components.

Dependencies to: No dependencies.

FPT_ML_EXT.1.1

The TSF shall support a measured launch of the Virtualization System. Measured components of the VS shall include the static executable image of the Hypervisor and: [**selection**: *Static executable images of the*

*Management Subsystem, [**assignment**: list of (static images of) Service VMs], [**assignment**: list of configuration files], no other components*]

FPT_ML_EXT.1.2

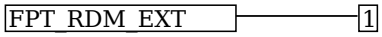
The TSF shall make the measurements selected in FPT_ML_EXT.1.1 available to the Management Subsystem.
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.10 FPT_RDM_EXT Removable Devices and Media

Family Behavior

This family defines requirements for enforcement of domain isolation when removable devices can be connected to a domain.

Component Leveling



FPT_RDM_EXT.1, Removable Devices and Media, requires the TSF to ensure that VMs are not inadvertently given access to information in different domains because removable media is simultaneously accessible from separate domains.

Management: FPT_RDM_EXT.1

The following actions could be considered for the management functions in FMT:

- Ability to configure removable media policy.
- Ability to connect/disconnect removable devices to/from a VM.

Audit: FPT_RDM_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Connection/disconnection of removable media or device to/from a VM.
b. Ejection/insertion of removable media or device from/to an already connected VM.

FPT_RDM_EXT.1 Removable Devices and Media

Hierarchical to: No other components.

Dependencies to: FDP_VMS_EXT.1 VM Separation

FPT_RDM_EXT.1.1

The TSF shall implement controls for handling the transfer of virtual and physical removable media and virtual and physical removable media devices between information domains.

FPT_RDM_EXT.1.2

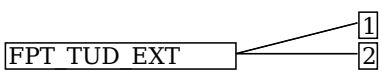
The TSF shall enforce the following rules when [**assignment**: *virtual or physical removable media and virtual or physical removable media devices*] are switched between information domains, then [**selection**: *the Administrator has granted explicit access for the media or device to be connected to the receiving domain, the media in a device that is being transferred is ejected prior to the receiving domain being allowed access to the device, the user of the receiving domain expressly authorizes the connection, the device or media that is being transferred is prevented from being accessed by the receiving domain*]
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.11 FPT_TUD_EXT Trusted Updates

Family Behavior

This family defines requirements for ensuring that updates to the TOE software and firmware are genuine.

Component Leveling



FPT_TUD_EXT.1, Trusted Updates to the Virtualization System, requires the TSF to define the mechanism for applying and verifying TOE updates.

FPT_TUD_EXT.2, Trusted Update Based on Certificates, requires the TSF to validate updates using a code signing certificate.

Management: FPT_TUD_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to update the Virtualization System.

Audit: FPT_TUD_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Initiation of update.
b. Failure of signature verification.

FPT_TUD_EXT.1 Trusted Updates to the Virtualization System

Hierarchical to: No other components.

Dependencies to: FCS_COP.1 Cryptographic Operation

FPT_TUD_EXT.1.1

The TSF shall provide administrators the ability to query the currently executed version of the TOE firmware/software as well as the most recently installed version of the TOE firmware/software.

FPT_TUD_EXT.1.2

The TSF shall provide administrators the ability to manually initiate updates to TOE firmware/software and [**selection**: *automatic updates*, *no other update mechanism*].

FPT_TUD_EXT.1.3

[**assignment**: *integrity action*]

Management: FPT_TUD_EXT.2

No specific management functions are identified.

Audit: FPT_TUD_EXT.2

There are no auditable events foreseen.

FPT_TUD_EXT.2 Trusted Update Based on Certificates

Hierarchical to: No other components.

Dependencies to: FPT_TUD_EXT.1 Trusted Updates to the Virtualization System

FIA_X509_EXT.1 X.509 Validation

FIA_X509_EXT.2 X.509 Authentication

FPT_TUD_EXT.2.1

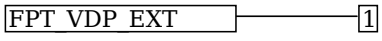
The TSF shall not install an update if the code signing certificate is deemed invalid.
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.12 FPT_VDP_EXT Virtual Device Parameters

Family Behavior

This family defines requirements for processing data transmitted to the TOE from a Guest VM.

Component Leveling



FPT_VDP_EXT.1, Virtual Device Parameters, requires the TSF to interface with Guest VMs through virtual hardware abstractions so that any data transmitted to the TOE from a Guest VM can be validated as well-formed.

Management: FPT_VDP_EXT.1

No specific management functions are identified.

Audit: FPT_VDP_EXT.1

There are no auditable events foreseen.

FPT_VDP_EXT.1 Virtual Device Parameters

Hierarchical to: No other components.

Dependencies to: FPT_VIV_EXT.1 VMM Isolation from VMs

FPT_VDP_EXT.1.1

The TSF shall provide interfaces for virtual devices implemented by the VMM as part of the virtual hardware abstraction.

FPT_VDP_EXT.1.2

The TSF shall validate the parameters passed to the virtual device interface prior to execution of the VMM functionality exposed by those interfaces.
This Protection Profile defines the following extended components as part of the FPT class originally defined by CC Part 2:

C.2.3.13 FPT_VIV_EXT VMM Isolation from VMs

Family Behavior

This family defines requirements for ensuring the TOE is logically isolated from its Guest VMs

Component Leveling

FPT_VIV_EXT.1, VMM Isolation from VMs, requires the TSF to ensure that there is no mechanism by which a Guest VM can interface with the TOE, other VMs, or the hardware platform without authorization.

Management: FPT_VIV_EXT.1

No specific management functions are identified.

Audit: FPT_VIV_EXT.1

There are no auditable events foreseen.

FPT_VIV_EXT.1 VMM Isolation from VMs

Hierarchical to: No other components.

Dependencies to: FDP_PPR_EXT.1 Physical Platform Resource Controls

FDP_VMS_EXT.1 VM Separation

FPT_VIV_EXT.1.1

The TSF must ensure that software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform.

FPT_VIV_EXT.1.2

The TSF must ensure that a Guest VM is unable to invoke platform code that runs at a privilege level equal to or exceeding that of the VMM without involvement of the VMM.

C.2.4 Security Audit (FAU)

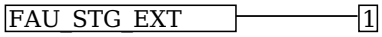
This Protection Profile defines the following extended components as part of the FAU class originally defined by CC Part 2:

C.2.4.1 FAU_STG_EXT Off-Loading of Audit Data

Family Behavior

This family defines requirements for the TSF to be able to securely transmit audit data between the TOE and an external IT entity.

Component Leveling



FAU_STG_EXT.1, Off-Loading of Audit Data, requires the TSF to transmit audit data using a trusted channel to an outside entity and to specify the action to be taken when local audit storage is full.

Management: FAU_STG_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure and manage the audit system and audit data, including the ability to configure name/address of audit/logging server to which to send audit/logging records.

Audit: FAU_STG_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Failure of audit data capture due to lack of disk space or pre-defined limit.
b. On failure of logging function, capture record of failure and record upon restart of logging function.

FAU_STG_EXT.1 Off-Loading of Audit Data

Hierarchical to: No other components.

Dependencies to: FAU_GEN.1 Audit Data Generation

FTP_ITC_EXT.1 Trusted Channel Communications

FAU_STG_EXT.1.1

The TSF shall be able to transmit the generated audit data to an external IT entity using a trusted channel as specified in FTP_ITC_EXT.1.

FAU_STG_EXT.1.2

The TSF shall [**selection**: *drop new audit data, overwrite previous audit records according to the following rule: [**assignment**: rule for overwriting previous audit records], [**assignment**: other action]*] when the local storage space for audit data is full.

C.2.5 Security Management (FMT)

This Protection Profile defines the following extended components as part of the FMT class originally defined by CC Part 2:

C.2.5.1 FMT_SMO_EXT Separation of Management and Operational Networks

Family Behavior

This family defines requirements for separation of management and operational networks.

Component Leveling

FMT_SMO_EXT.1, Separation of Management and Operational Networks, requires the TSF to separate its management and operational networks through a defined mechanism.

Management: FMT_SMO_EXT.1

No specific management functions are identified.

Audit: FMT_SMO_EXT.1

There are no auditable events foreseen.

FMT_SMO_EXT.1 Separation of Management and Operational Networks

Hierarchical to: No other components.

Dependencies to: No dependencies.

FMT_SMO_EXT.1.1

The TSF shall support the separation of management and operational network traffic through [**selection**: *separate physical networks, separate logical networks, trusted channels as defined in FTP_ITC_EXT.1, data encryption using an algorithm specified in FCS_COP.1/UDE*].

C.2.6 Trusted Path/Channel (FTP)

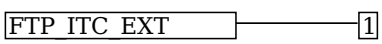
This Protection Profile defines the following extended components as part of the FTP class originally defined by CC Part 2:

C.2.6.1 FTP_ITC_EXT Trusted Channel Communications

Family Behavior

This family defines requirements for protection of data in transit between the TOE and its operational environment.

Component Leveling



FTP_ITC_EXT.1, Trusted Channel Communications, requires the TSF to implement one or more cryptographic protocols to secure connectivity between the TSF and various external entities.

Management: FTP_ITC_EXT.1

No specific management functions are identified.

Audit: FTP_ITC_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Initiation of the trusted channel.
b. Termination of the trusted channel.
c. Failures of the trusted path functions.

FTP_ITC_EXT.1 Trusted Channel Communications

Hierarchical to: No other components.

Dependencies to: FAU_STG_EXT.1 Off-Loading of Audit Data

FTP_ITC_EXT.1.1

[**assignment**: *transport mechanism*][**assignment**: *authentication mechanism*]

- audit servers (as required by FAU_STG_EXT.1), and

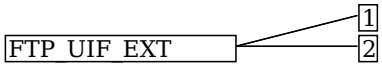
[**assignment**: *remote entities*]
This Protection Profile defines the following extended components as part of the FTP class originally defined by CC Part 2:

C.2.6.2 FTP_UIF_EXT User Interface

Family Behavior

This family defines requirements for unambiguously identifying the specific Guest VM that a TOE user is interacting with at any given point in time.

Component Leveling



FTP_UIF_EXT.1, User Interface: I/O Focus, requires the TSF to unambiguously identify the Guest VM that has the current input focus for input peripherals.

FTP_UIF_EXT.2, User Interface: Identification of VM, requires the TOE to perform power on self-tests to verify its functionality and the integrity of its stored executable code.

Management: FTP_UIF_EXT.1

No specific management functions are identified.

Audit: FTP_UIF_EXT.1

There are no auditable events foreseen.

FTP_UIF_EXT.1 User Interface: I/O Focus

Hierarchical to: No other components.

Dependencies to: No dependencies

FTP_UIF_EXT.1.1

The TSF shall indicate to users which VM, if any, has the current input focus.

Management: FTP_UIF_EXT.2

No specific management functions are identified.

Audit: FTP_UIF_EXT.2

There are no auditable events foreseen.

FTP_UIF_EXT.2 User Interface: Identification of VM

Hierarchical to: No other components.

Dependencies to: No dependencies

FTP_UIF_EXT.2.1

The TSF shall support the unique identification of a VM's output display to users.

C.2.7 User Data Protection (FDP)

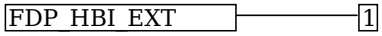
This Protection Profile defines the following extended components as part of the FDP class originally defined by CC Part 2:

C.2.7.1 FDP_HBI_EXT Hardware-Based Isolation Mechanisms

Family Behavior

This family defines requirements for isolation of Guest VMs from the hardware resources of the physical device on which the Guest VMs are deployed.

Component Leveling



FDP_HBI_EXT.1, Hardware-Based Isolation Mechanisms, requires the TSF to identify the mechanisms used to isolate Guest VMs from platform hardware resources.

Management: FDP_HBI_EXT.1

No specific management functions are identified.

Audit: FDP_HBI_EXT.1

There are no auditable events foreseen.

FDP_HBI_EXT.1 Hardware-Based Isolation Mechanisms

Hierarchical to: No other components.

Dependencies to: FDP_VMS_EXT.1 VM Separation

FDP_HBI_EXT.1.1

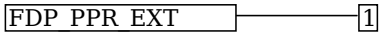
The TSF shall use [**selection**: *no mechanism*, *[**assignment**: list of platform-provided, hardware-based mechanisms]*] to constrain a Guest VM's direct access to the following physical devices: [**selection**: *no devices*, *[**assignment**: physical devices to which the VMM allows Guest VMs physical access]*].
This Protection Profile defines the following extended components as part of the FDP class originally defined by CC Part 2:

C.2.7.2 FDP_PPR_EXT Physical Platform Resource Controls

Family Behavior

This family defines requirements for the physical resources that the TOE will allow or prohibit Guest VMs to access.

Component Leveling



FDP_PPR_EXT.1, Physical Platform Resource Controls, requires the TSF to define the hardware resources that Guest VMs may always access, may never access, and may conditionally access based on administrative configuration.

Management: FDP_PPR_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure VM access to physical devices.

Audit: FDP_PPR_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Successful and failed VM connections to physical devices where connection is governed by configurable policy.
b. Security policy violations.

FDP_PPR_EXT.1 Physical Platform Resource Controls

Hierarchical to: No other components.

Dependencies to: FDP_HBI_EXT.1 Hardware-Based Isolation Mechanisms

FMT_SMR.1 Security Roles

FDP_PPR_EXT.1.1

The TSF shall allow an authorized administrator to control Guest VM access to the following physical platform resources: [**assignment**: *list of physical platform resources the VMM is able to control access to*].

FDP_PPR_EXT.1.2

The TSF shall explicitly deny all Guest VMs access to the following physical platform resources: [**selection**: *no physical platform resources, [**assignment**: list of physical platform resources to which access is explicitly denied]*].

FDP_PPR_EXT.1.3

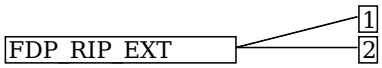
The TSF shall explicitly allow all Guest VMs access to the following physical platform resources: [**selection**: *no physical platform resources, [**assignment**: list of physical platform resources to which access is always allowed]*].
This Protection Profile defines the following extended components as part of the FDP class originally defined by CC Part 2:

C.2.7.3 FDP_RIP_EXT Residual Information in Memory

Family Behavior

This family defines requirements for ensuring that allocation of data to a Guest VM does not cause a disclosure of residual data from a previous VM.

Component Leveling



FDP_RIP_EXT.1, Residual Information in Memory, requires the TSF to ensure that physical memory is cleared to zeros prior to its allocation to a Guest VM.

FDP_RIP_EXT.2, Residual Information on Disk, requires the TSF to ensure that physical disk storage is cleared upon allocation to a Guest VM.

Management: FDP_RIP_EXT.1

No specific management functions are identified.

Audit: FDP_RIP_EXT.1

There are no auditable events foreseen.

FDP_RIP_EXT.1 Residual Information in Memory

Hierarchical to: No other components.

Dependencies to: No dependencies.

FDP_RIP_EXT.1.1

The TSF shall ensure that any previous information content of physical memory is cleared prior to allocation to a Guest VM.

Management: FDP_RIP_EXT.2

No specific management functions are identified.

Audit: FDP_RIP_EXT.2

There are no auditable events foreseen.

FDP_RIP_EXT.2 Residual Information on Disk

Hierarchical to: No other components.

Dependencies to: No dependencies.

FDP_RIP_EXT.2.1

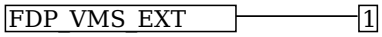
The TSF shall ensure that any previous information content of physical disk storage is cleared to zeros upon allocation to a Guest VM.
This Protection Profile defines the following extended components as part of the FDP class originally defined by CC Part 2:

C.2.7.4 FDP_VMS_EXT VM Separation

Family Behavior

This family defines requirements for the logical separation of multiple Guest VMs that are managed by the same Virtualization System.

Component Leveling



FDP_VMS_EXT.1, VM Separation, requires the TSF to maintain logical separation between Guest VMs except through the use of specific configurable methods.

Management: FDP_VMS_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure inter-VM data sharing.

Audit: FDP_VMS_EXT.1

There are no auditable events foreseen.

FDP_VMS_EXT.1 VM Separation

Hierarchical to: No other components.

Dependencies to: No dependencies.

FDP_VMS_EXT.1.1

The VS shall provide the following mechanisms for transferring data between Guest VMs: [**selection**: *no mechanism, virtual networking, [**assignment**: other inter-VM data sharing mechanisms]*].

FDP_VMS_EXT.1.2

The TSF shall by default enforce a policy prohibiting sharing of data between Guest VMs.

FDP_VMS_EXT.1.3

The TSF shall allow Administrators to configure the mechanisms selected in FDP_VMS_EXT.1.1 to enable and disable the transfer of data between Guest VMs.

FDP_VMS_EXT.1.4

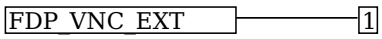
The VS shall ensure that no Guest VM is able to read or transfer data to or from another Guest VM except through the mechanisms listed in FDP_VMS_EXT.1.1.
This Protection Profile defines the following extended components as part of the FDP class originally defined by CC Part 2:

C.2.7.5 FDP_VNC_EXT Virtual Networking Components

Family Behavior

This family defines requirements for configuration of virtual networking between Guest VMs that are managed by the Virtualization System.

Component Leveling



FDP_VNC_EXT.1, Virtual Networking Components, requires the TSF to support the configuration of virtual networking between Guest VMs.

Management: FDP_VNC_EXT.1

The following actions could be considered for the management functions in FMT:

a. Ability to configure virtual networks including VM.

Audit: FDP_VNC_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a. Successful and failed attempts to connect VMs to virtual and physical networking components.
b. Security policy violations.
c. Administrator configuration of inter-VM communications channels between VMs.

FDP_VNC_EXT.1 Virtual Networking Components

Hierarchical to: No other components.

Dependencies to: FDP_VMS_EXT.1 VM Separation

FMT_SMR.1 Security Roles

FDP_VNC_EXT.1.1

The TSF shall allow Administrators to configure virtual networking components to connect VMs to each other and to physical networks.

FDP_VNC_EXT.1.2

The TSF shall ensure that network traffic visible to a Guest VM on a virtual network--or virtual segment of a physical network--is visible only to Guest VMs configured to be on that virtual network or segment.

Appendix D - Optional Requirements

Appendix E - Selection-Based Requirements

Appendix F - Implicitly Satisfied Requirements

Table 4: : Implicitly Satisfied Requirements

Requirement	Rationale for Satisfaction
FCS_CKM.4 – Cryptographic Key Destruction	FCS_CKM.1 has a dependency on FCS_CKM.4. The extended SFR FCS_CKM_EXT.4 addresses this dependency by defining an alternate requirement for key destruction.
FCS_CKM.4 – Cryptographic Key Destruction	FCS_CKM.2 has a dependency on FCS_CKM.4. The extended SFR FCS_CKM_EXT.4 addresses this dependency by defining an alternate requirement for key destruction.
FCS_CKM.4 – Cryptographic Key Destruction	Each iteration of FCS_COP.1 has a dependency on FCS_CKM.4. The extended SFR FCS_CKM_EXT.4 addresses this dependency by defining an alternate requirement for key destruction.
FIA_UID.1 – Timing of Identification	FMT_SMR.2 has a dependency on FIA_UID.1. The extended SFR FIA_UID_EXT.1 expresses this dependency by also requiring user identification for use of the TOE.
FPT_STM.1 – Reliable Time Stamps	FAU_GEN.1 has a dependency on FPT_STM.1. While not explicitly stated in the PP, it is assumed that this will be provided by the underlying hardware platform on which the TOE is installed. This is because the TOE is installed as a software or firmware product that runs on general-purpose computing hardware so a hardware clock is assumed to be available.
FPT_STM.1 – Reliable Time Stamps	FIA_X509_EXT.1 has a dependency on FPT_STM.1. While not explicitly stated in the PP, it is assumed that this will be provided by the underlying hardware platform on which the TOE is installed. This is because the TOE is installed as a software or firmware product that runs on general-purpose computing hardware so a hardware clock is assumed to be available.

Appendix G - Entropy Documentation and Assessment

G.1 Design Description

Documentation shall include the design of the entropy source as a whole, including the interaction of all entropy source components. It will describe the operation of the entropy source to include how it works, how entropy is produced, and how unprocessed (raw) data can be obtained from within the entropy source for testing purposes. The documentation should walk through the entropy source design indicating where the random comes from, where it is passed next, any post-processing of the raw outputs (hash, XOR, etc.), if/where it is stored, and finally, how it is output from the entropy source. Any conditions placed on the process (e.g., blocking) should also be described in the entropy source design. Diagrams and examples are encouraged.

This design must also include a description of the content of the security boundary of the entropy source and a description of how the security boundary ensures that an adversary outside the boundary cannot affect the entropy rate.

G.2 Entropy Justification

There should be a technical argument for where the unpredictability in the source comes from and why there is confidence in the entropy source exhibiting probabilistic behavior (an explanation of the probability distribution and justification for that distribution given the particular source is one way to describe this). This argument will include a description of the expected entropy rate and explain how you ensure that sufficient entropy is going into the TOE randomizer seeding process. This discussion will be part of a justification for why the entropy source can be relied upon to produce bits with entropy.

G.3 Operating Conditions

Documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. It will clearly describe the measures that have been taken in the system design to ensure the entropy source continues to operate under those conditions. Similarly, documentation shall describe the conditions under which the entropy source is known to malfunction or become inconsistent. Methods used to detect failure or degradation of the source shall be included.

G.4 Health Testing

More specifically, all entropy source health tests and their rationale will be documented. This will include a description of the health tests, the rate and conditions under which each health test is performed (e.g., at startup, continuously, or on-demand), the expected results for each health test, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source.

Appendix H - Equivalency Guidelines

H.1 Introduction

The purpose of equivalence in PP-based evaluations is to find a balance between evaluation rigor and commercial practicability--to ensure that evaluations meet customer expectations while recognizing that there is little to be gained from requiring that every variation in a product or platform be fully tested. If a product is found to be compliant with a PP on one platform, then all equivalent products on equivalent platforms are also considered to be compliant with the PP.

A Vendor can make a claim of equivalence if the Vendor believes that a particular instance of their Product implements PP-specified security functionality in a way equivalent to the implementation of the same functionality on another instance of their Product on which the functionality was tested. The Product instances can differ in version number or feature level (model), or the instances may run on different platforms. Equivalency can be used to reduce the testing required across claimed evaluated configurations. It can also be used during Assurance Maintenance to reduce testing needed to add more evaluated configurations to a certification.

These equivalency guidelines do not replace Assurance Maintenance requirements or NIAP Policy #5 requirements for CAVP certificates. Nor may equivalency be used to leverage evaluations with expired certifications.

This document provides guidance for determining whether Products and Platforms are equivalent for purposes of evaluation against the Protection Profile for Virtualization (VPP) when instantiated with either the Client or Server PP-Module.

Equivalence has two aspects:

1. ***Product Equivalence:*** Products may be considered equivalent if there are no differences between Product Models and Product Versions with respect to PP-specified security functionality.
2. ***Platform Equivalence:*** Platforms may be considered equivalent if there are no significant differences in the services they provide to the Product--or in the way the platforms provide those services--with respect to PP-specified security functionality.

The equivalency determination is made in accordance with these guidelines by the Validator and Scheme using information provided by the Evaluator/Vendor.

H.2 Approach to Equivalency Analysis

There are two scenarios for performing equivalency analysis. One is when a product has been certified and the vendor wants to show that a later product should be considered certified due to equivalence with the earlier product. The other is when multiple product variants are going though evaluation together and the vendor would like to reduce the amount of testing that must be done. The basic rules for determining equivalence are the same in both cases. But there is one additional consideration that applies to equivalence with previously certified products. That is, the product with which equivalence is being claimed must have a valid certification in accordance with scheme rules and the Assurance Maintenance process must be followed. If a product’s certification has expired, then equivalence cannot be claimed with that product.

When performing equivalency analysis, the Evaluator/Vendor should first use the factors and guidelines for Product Model equivalence to determine the set of Product Models to be evaluated. In general, Product Models that do not differ in PP-specified security functionality are considered equivalent for purposes of evaluation against the VPP.

If multiple revision levels of Product Models are to be evaluated--or to determine whether a revision of an evaluated product needs re-evaluation--the Evaluator/Vendor and Validator should use the factors and guidelines for Product Version equivalence to determine whether Product Versions are equivalent.

Having determined the set of Product Models and Versions to be evaluated, the next step is to determine the set of Platforms that the Products must be tested on.

Each non-equivalent Product for which compliance is claimed must be fully tested on each non-equivalent platform for which compliance is claimed. For non-equivalent Products on equivalent platforms, only the differences that affect PP-specified security functionality must be tested for each product.

If the set of equivalent Products includes only bare-metal installations, then the equivalency analysis is complete. But if any members of the set include hosted installations or installations that integrate with an existing host operating system or control domain, then software platform equivalence must be taken into consideration. The Evaluator/Vendor and Validator should use the factors and guidance for software platform equivalence to determine whether different models or versions of host or control domain operating systems require separate testing.

"Differences in PP-Specified Security Functionality" Defined
If PP-specified security functionality is implemented by the TOE, then differences in the actual implementation between versions or product models break equivalence for that feature. Likewise, if the TOE implements the functionality in one version or model and the functionality is implemented by the platform in another version or model, then equivalence is broken. If the functionality is implemented by the platform in multiple models or versions on equivalent platforms, then the functionality is considered different if the product invokes the platform differently to perform the function.

H.3 Specific Guidance for Determining Product Model Equivalence

Product Model equivalence attempts to determine whether different feature levels of the same product across a product line are equivalent for purposes of PP testing. For example, if a product has a "basic" edition and an "enterprise" edition, is it necessary to test both models? Or does testing one model provide sufficient confidence that both models are compliant?

figure Table , below, lists the factors for determining Product Model equivalence.

Table 5: : Factors for Determining Product Model Equivalence

Factor	Same/Different	Guidance
Target Platform	Different	Product Models that virtualize different instruction sets (e.g., x86, ARM, POWER, SPARC, MIPS) are not equivalent.
Installation Types	Different	If a Product can be installed either on bare metal or onto an operating system and the vendor wants to claim that both installation types constitute a single Model, then see the guidance for "PP-Specified Functionality," below.
Software Platform	Different	Product Models that run on substantially different software environments, such as different host operating systems, are not equivalent. Models that install on different versions of the same software environment may be equivalent depending on the below factors.
PP-Specified Functionality	Same	If the differences between Models affect only non-PP-specified functionality, then the Models are equivalent.
	Different	If PP-specified security functionality is affected by the differences between Models, then the Models are not equivalent and must be tested separately. It is necessary to test only the functionality affected by the software differences. If only differences are tested, then the differences must be enumerated, and for each difference the Vendor must provide an explanation of why each difference does or does not affect PP-specified functionality. If the Product Models are fully tested separately, then there is no need to document the differences.

H.4 Specific Guidance for Determining Product Version Equivalence

In cases of version equivalence, differences are expressed in terms of changes implemented in revisions of an evaluated Product. In general, versions are equivalent if the changes have no effect on any security-relevant claims about the TOE or evaluation evidence. Non-security-relevant changes to TOE functionality or the addition of non-security-relevant functionality does not affect equivalence.

Table 6: : Factors for Determining Product Version Equivalence

Factor	Same/Different	Guidance
Product Models	Different	Versions of different Product Models are not equivalent unless the Models are equivalent as defined in Section 3.
PP-Specified Functionality	Same	If the differences affect only non-PP-specified functionality, then the Versions are equivalent.
	Different	If PP-specified security functionality is affected by the differences, then the Versions are considered to be not equivalent and must be tested separately. It is necessary only to test the functionality affected by the changes. If only the differences are tested, then for each difference the Vendor must provide an explanation of why the difference does or does not affect PP-specified functionality. If the Product Versions are fully

		tested separately, then there is no need to document the differences.

H.5 Specific Guidance for Determining Platform Equivalence

Platform equivalence is used to determine the platforms that a product must be tested on. These guidelines are divided into sections for determining hardware equivalence and software (host OS/control domain) equivalence. If the Product is installed onto bare metal, then only hardware equivalence is relevant. If the Product is installed onto an OS—or is integrated into an OS—then both hardware and software equivalence are required. Likewise, if the Product can be installed either on bare metal or on an operating system, both hardware and software equivalence are relevant.

H.5.1 Hardware Platform Equivalence

If a Virtualization Solution runs directly on hardware without an operating system, then platform equivalence is based primarily on processor architecture and instruction sets.

Platforms with different processor architectures and instruction sets are not equivalent. This is probably not an issue because there is likely to be a different product model for different hardware environments.

Equivalency analysis becomes important when comparing platforms with the same processor architecture. Processors with the same architecture that have instruction sets that are subsets or supersets of each other are not disqualified from being equivalent for purposes of a VPP evaluation. If the VS takes the same code paths when executing PP-specified security functionality on different processors of the same family, then the processors can be considered equivalent with respect to that application.

For example, if a VS follows one code path on platforms that support the AES-NI instruction and another on platforms that do not, then those two platforms are not equivalent with respect to that VS functionality. But if the VS follows the same code path whether or not the platform supports AES-NI, then the platforms are equivalent with respect to that functionality.

The platforms are equivalent with respect to the VS if the platforms are equivalent with respect to all PP-specified security functionality.

Table 7: : Factors for Determining Hardware Platform Equivalence

Factor	Same/Different/None	Guidance
Platform Architectures	Different	Hardware platforms that implement different processor architectures and instruction sets are not equivalent.
PP-Specified Functionality	Same	For platforms with the same processor architecture, the platforms are equivalent with respect to the application if execution of all PP-specified security functionality follows the same code path on both platforms.

H.5.2 Software Platform Equivalence

If the Product installs onto or integrates with an operating system that is not installed with the product--and thus is not part of the TOE--then the Product must be tested on all non-equivalent Software Platforms.

The guidance for Product Model (Section 3) specifies that Products intended for use on substantially different operating systems (e.g., Windows vs. Linux vs. SunOS) are different Models. Therefore, platforms running substantially different operating systems are not equivalent. Likewise, operating systems with different major version numbers are not equivalent for purposes of this PP.

As a result, Software Platform equivalence is largely concerned with revisions and variations of operating systems that are substantially the same (e.g., different versions and revision levels of Windows or Linux).

Table 8: : Factors for Determining Software Platform Equivalence

Factor	Same/Different/None	Guidance
Platform Type/Vendor	Different	Operating systems that are substantially different or come from different vendors are not equivalent.
Platform Versions	Different	Operating systems are not equivalent if they have different major version numbers.

PP-Specified Functionality	Same	If the differences between software platform models or versions affect only non-PP-specified functionality, then the software platforms are equivalent.
	Different	If PP-specified security functionality is affected by the differences between software platform versions or models, then the software platforms are not considered equivalent and must be tested separately. It is necessary only to test the functionality affected by the changes. If only the differences are tested, then for each difference the Vendor must provide an explanation of why the difference does or does not affect PP-specified functionality. If the Products are fully tested on each platform, then there is no need to document the differences.

H.6 Level of Specificity for Tested and Claimed Equivalent Configurations

In order to make equivalency determinations, the vendor and evaluator must agree on the equivalency claims. They must then provide the scheme with sufficient information about the TOE instances and platforms that were evaluated, and the TOE instances and platforms that are claimed to be equivalent.

The ST must describe all configurations evaluated down to processor manufacturer, model number, and microarchitecture version.

The information regarding claimed equivalent configurations depends on the platform that the VS was developed for and runs on.

Bare-Metal VS

For VSes that run without an operating system on bare-metal or virtual bare-metal, the claimed configuration must describe the platform down to the specific processor manufacturer, model number, and microarchitecture version. The Vendor must describe the differences in the TOE with respect to PP-specified security functionality and how the TOE operates differently to leverage platform differences (e.g., instruction set extensions) in the tested configuration versus the claimed equivalent configuration.

VS with OS Support

For VSes that run on an OS host or with the assistance of an OS, then the claimed configuration must describe the OS down to its specific model and version number. The Vendor must describe the differences in the TOE with respect to PP-specified security functionality and how the TOE functions differently to leverage platform differences in the tested configuration versus the claimed equivalent configuration.

Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
Base-PP	Base Protection Profile
CPU	Central Processing Unit
cPP	Collaborative Protection Profile
CC	Common Criteria
CEM	Common Evaluation Methodology
DEP	Data Execution Prevention
DKM	Derived Keying Material
DSS	Digital Signature Standard
ECC	Elliptic Curve Cryptography
EP	Extended Package
FIPS	Federal Information Processing Standard
FFC	Finite-Field Cryptography
FP	Functional Package
IT	Information Technology
ITSEF	Information Technology Security Evaluation Facility
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
IP	Internet Protocol
KDF	Key Derivation Function
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
OS	Operating System
OE	Operational Environment
PP	Protection Profile
PP-Configuration	Protection Profile Configuration
PP-Module	Protection Profile Module
PKV	Public Key Verification
RSA	Rivest, Shamir, Adleman
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
SPD	Security Policy Database
ST	Security Target
SWID	Software Identification
SP	Special Publication
SSP	System Security Policy

TOE	Target of Evaluation
TSF	TOE Security Functionality
TSS	TOE Summary Specification
TPM	Trusted Platform Module
TSFI	TSF Interface
VM	Virtual Machine
VMM	Virtual Machine Manager
VS	Virtualization System

Bibliography

Identifier	Title
[CC]	Common Criteria for Information Technology Security Evaluation - • Part 1: Introduction and General Model, CCMB-2017-04-001, Version 3.1 Revision 5, April 2017. • Part 2: Security Functional Components, CCMB-2017-04-002, Version 3.1 Revision 5, April 2017. • Part 3: Security Assurance Components, CCMB-2017-04-003, Version 3.1 Revision 5, April 2017.
[CEM]	Common Evaluation Methodology for Information Technology Security - Evaluation Methodology, CCMB-2017-04-004, Version 3.1, Revision 5, April 2017.