Protection Profile for Virtualization

Version: 1.1
2021-02-16
National Information Assurance Partnership

Revision History

Version	Date	Comment
1.0	2016-11-17	Initial Publication
1.1	2020-11-17	Incorporate TDs, Reference TLS Package, Add Equivalency Guidelines

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 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.
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.
Distributed TOE	A TOE composed of multiple components operating as a logical whole.
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 Protection Profiles.
Security Assurance Requirement (SAR)	A requirement to assure the security of the TOE.
Security Functional Requirement (SFR)	A requirement for security enforcement by the TOE.
Security Target (ST)	A set of implementation-dependent security requirements for a specific product.

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.
Target of Evaluation (TOE)	The product under evaluation.

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.
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 for purposes of VM transfer or coordinate, and backend storage networks 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 a physical devices, 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 other. 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 other. 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, 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.

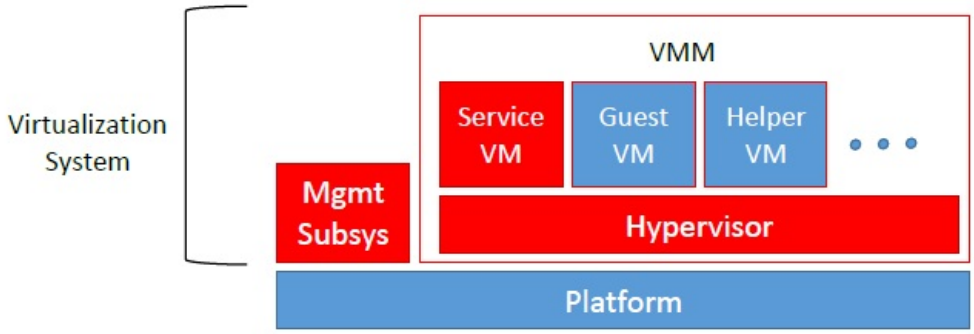

Figure 1: Virtualization System and Platform

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 assurance 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 the assurance level does not change 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).

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 Conformance Claims

Conformance Statement
An ST must claim exact conformance to this PP, as defined in the CC and CEM addenda for Exact Conformance, Selection-Based SFRs, and Optional SFRs (dated May 2017).

CC Conformance Claims
This PP is conformant to Parts 2 (extended) and 3 (conformant) of Common Criteria Version 3.1, Revision 5.

PP Claim
This PP does not claim conformance to any Protection Profile.

Package Claim
This PP is Network Encryption (TLS) Package, Version 1.1 Conformant and Secure Shell (SSH) Package, Version 1.0 Conformant .

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).

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

PP Claims
This PP does not claim conformance to any other PP.

Module Claim
One or more of the following modules must be specified in a PP-Configuration with this PP.

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

Package Claims
This PP is Functional Package for TLS-conformant.

3 Security Problem Description

3.1 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 secuity 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.

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. 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 anauthorized 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.

3.2 Assumptions

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.COVERT_CHANNELS
There is sufficient assurance relative to the value of the VM’s IT assets to address the risk of covert storage and timing channels to the VMs executing on the TOE.

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.

3.3 Organizational Security Policies

There are no organizational security policies defined for this PP.

4 Security Objectives

4.1 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 sharing of particular resources 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.

Devices, whether virtual or physical, are resources requiring access control. The VMM must enforce access control in accordance to 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 is capable of undermining 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 certain authorized system users (administrators) are allowed to 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, 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 might be 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 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.

4.2 Security Objectives for the Operational Environment

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.COVERT_CHANNELS
If the TOE has covert storage or timing channels, then for all VMs executing on that TOE, it is assumed that those VMs will have sufficient assurance relative to the IT assets to which they have access, to outweigh the risk that they will violate the security policy of the TOE by using those covert channels.

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

4.3 Security Objectives Rationale

This section describes how the assumptions, threats, and organization 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.COVERT_CHANNELS	OE.COVERT_CHANNELS	It is expected that the value of any data contained within VMs is commensurate with the security provided by the TOE, which includes any vulnerabilities due to the potential presence of covert storage or timing channels.
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.

5 Security Requirements

This chapter describes the security requirements which have to be fulfilled by the product under evaluation. Those requirements comprise functional components from Part 2 and assurance components from Part 3 of [CC]. The following conventions are used for the completion of operations:

- **Refinement** operation (denoted by **bold text** or ~~strikethrough text~~): is used to add details to a requirement (including replacing an assignment with a more restrictive selection) or to remove part of the requirement that is made irrelevant through the completion of another operation, and thus further restricts a requirement.
- **Selection** (denoted by *italicized text*): is used to select one or more options provided by the [CC] in stating a requirement.
- **Assignment** operation (denoted by *italicized text*): is used to assign a specific value to an unspecified parameter, such as the length of a password. Showing the value in square brackets indicates assignment.
- **Iteration** operation: is indicated by appending the SFR name with a slash and unique identifier suggesting the purpose of the operation, e.g. "/EXAMPLE1."

5.1 Security Functional Requirements

5.1.1 Auditable Events for Mandatory SFRs

Table 2: Auditable Events for Mandatory SFRs

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	No events specified	
FAU_SAR.1	No events specified	
FAU_STG.1	No events specified	
FAU_STG_EXT.1	Failure of audit data capture due to lack of disk space or pre-defined limit.	
FAU_STG_EXT.1	On failure of logging function, capture record of failure and record upon restart of logging function.	
FCS_CKM.1	No events specified	
FCS_CKM.2	No events specified	
FCS_CKM_EXT.4	No events specified	
FCS_COP.1/UDE	No events specified	
FCS_COP.1/HASH	No events specified	
FCS_COP.1/Sig	No events specified	
FCS_COP.1/KeyHash	No events specified	
FCS_ENT_EXT.1	No events specified	
FCS_RBG_EXT.1	Failure of the randomization process	
FDP_HBI_EXT.1	No events specified	
FDP_PPR_EXT.1	Successful and failed VM connections to physical devices where connection is governed by configurable policy.	VM and physical device identifiers.
FDP_PPR_EXT.1	Security policy violations.	Identifier for the security policy that was violated.
FDP_RIP_EXT.1	No events specified	
FDP_RIP_EXT.2	No events specified	
FDP_VMS_EXT.1	No events specified	
FDP_VNC_EXT.1	Successful and failed attempts to connect VMs to virtual and physical networking components.	VM and virtual or physical networking component identifiers.

FDP_VNC_EXT.1	Security policy violations.	Identifier for the security policy that was violated. VM and virtual or physical networking component identifiers.
FDP_VNC_EXT.1	Administrator configuration of inter-VM communications channels between VMs.	VM and virtual or physical networking component identifiers.
FIA_AFL_EXT.1	Unsuccessful login attempts limit is met or exceeded.	Origin of attempt (e.g., IP address).
FIA_UAU.5	No events specified	
FIA_UIA_EXT.1	Administrator authentication attempts.	Provided user identity, origin of the attempt (e.g. console, remote IP address).
FIA_UIA_EXT.1	All use of the identification and authentication mechanism.	Provided user identity, origin of the attempt (e.g. console, remote IP address).
FIA_UIA_EXT.1	**[selection:** *Start and end of administrator session., None***]**	Start time and end time of administrator session.
FMT_MSA_EXT.1	No events specified	
FMT_SMO_EXT.1	No events specified	
FPT_DVD_EXT.1	No events specified	
FPT_EEM_EXT.1	No events specified	
FPT_HAS_EXT.1	No events specified	
FPT_HCL_EXT.1	Attempts to access disabled hypercall interfaces	Interface for which access was attempted.
FPT_HCL_EXT.1	Security policy violations.	Identifier for the security policy that was violated.
FPT_RDM_EXT.1	Connection/disconnection of removable media or device to/from a VM.	VM Identifier, Removable media/device identifier, event description or identifier (connect/disconnect, ejection/insertion, etc.)
FPT_RDM_EXT.1	Ejection/insertion of removable media or device from/to an already connected VM.	VM Identifier, Removable media/device identifier, event description or identifier (connect/disconnect, ejection/insertion, etc.)
FPT_TUD_EXT.1	Initiation of update.	
FPT_TUD_EXT.1	Failure of signature verification.	
FPT_VDP_EXT.1	No events specified	
FPT_VIV_EXT.1	No events specified	
FTA_TAB.1	No events specified	
FTP_ITC_EXT.1	Initiation of the trusted channel.	User ID and remote source (IP Address) if feasible.
FTP_ITC_EXT.1	Termination of the trusted channel.	User ID and remote source (IP Address) if feasible.
FTP_ITC_EXT.1	Failures of the trusted path functions.	User ID and remote source (IP Address) if feasible.
FTP_UIF_EXT.1	No events specified	
FTP_UIF_EXT.2	No events specified	

5.1.2 Security Audit (FAU)

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 Table 2**]
d. **[selection: *Auditable events defined in Table atref-optional-depfor Strictly Optional SFRs, Auditable events defined in Table atref-objective-depfor Objective SFRs, Auditable events defined in Table atref-sel-based-depfor Selection-Based SFRs, Auditable events defined in the audit table for the TLS Functional Package (see Table3), Auditable events defined in the audit table for the SSH Functional Package (see Table3), 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 Table 2**]
f. **[selection: *Additional information defined in Table atref-optional-depfor Strictly Optional SFRs, Additional information defined in Table atref-objective-depfor Objective SFRs, Additional information defined in Table atref-sel-based-depfor Selection-Based SFRs, Additional information defined in the audit table for the TLS Functional Package (see Table3), Additional information defined in the audit table for the SSH Functional Package (see Table3), no other information*]**

Application Note: The ST author can include other auditable events directly in Table 2; 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 Table atref-optional-dep, Table atref-objective-dep, and Table atref-sel-based-dep should be included in the ST if the associated SFRs and selections are included.

The Table 2 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 tables for the SSH and TLS Functional Packages. 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 FP audit table. This list is included here for reference.

Table3: Auditable Events for Functional Packages

Requirement	Auditable Events	Additional Audit Record Contents
FCS_SSHC_EXT.1	Failure to establish a session.	Reason for failure, Non-TOE endpoint of attempted connection (IP Address).
FCS_SSHC_EXT.1	Establishment of a session.	Non-TOE endpoint of connection (IP Address).

FCS_SSHC_EXT.1	Termination of a session.	Non-TOE endpoint of connection (IP Address).
FCS_SSHS_EXT.1	Failure to establish a session.	Reason for failure, Non-TOE endpoint of attempted connection (IP Address).
FCS_SSHS_EXT.1	Establishment of a session.	Non-TOE endpoint of connection (IP Address).
FCS_SSHS_EXT.1	Termination of a session.	Non-TOE endpoint of connection (IP Address).
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.

Evaluation Activities

FAU_GEN.1:
TSS
The evaluator shall check the TSS and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type shall be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the PP-Configuration is described in the TSS.

Guidance
The evaluator shall also make a determination of the administrative actions that are relevant in the context of this PP-Configuration. The evaluator shall examine the administrative guide and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the PP and PP-Module(s). The evaluator shall document the methodology or approach taken while determining which actions in the administrative guide are security-relevant with respect to this PP-Configuration.

Tests
The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed and administrative actions. For administrative actions, the evaluator shall test that each action determined by the evaluator above to be security relevant in the context of this PP is auditable. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.

Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly.

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.

Evaluation Activities

FAU_SAR.1:
Guidance
The evaluator shall review the operational guidance for the procedure on how to review the audit records.
Tests
The evaluator shall verify that the audit records provide all of the information specified in FAU_GEN.1 and that this information is suitable for human interpretation. The assurance activity for this requirement is performed in conjunction with the assurance activity for FAU_GEN.1.

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 assurance 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.

Evaluation Activities

FAU_STG.1:
TSS
The evaluator shall ensure that the TSS describes how the audit records are protected from unauthorized modification or deletion. The evaluator shall ensure that the TSS describes the conditions that must be met for authorized deletion of audit records.
Guidance
Tests
The evaluator shall perform the following tests:

- ***Test 1:*** *The evaluator shall access the audit trail as an unauthorized Administrator and attempt to modify and delete the audit records. The evaluator shall verify that these attempts fail.*
- ***Test 2:*** *The evaluator shall access the audit trail as an authorized Administrator and attempt to delete the audit records. The evaluator shall verify that these attempts succeed. The evaluator shall verify that only the records authorized for deletion are deleted.*

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'.

Evaluation Activities

FAU_STG_EXT.1:
Protocols used for implementing the trusted channel must be selected in FTP_ITC_EXT.1.
TSS
The evaluator shall examine the TSS to ensure it describes the means by which the audit data are transferred to the external audit server, and how the trusted channel is provided.
Guidance
The evaluator shall examine the operational guidance to ensure it describes how to establish the trusted channel to the audit server, as well as describe any requirements on the audit server (particular audit server protocol, version of the protocol required, etc.), as well as configuration of the TOE needed to communicate with the audit server.

Tests
Testing of the trusted channel mechanism is to be performed as specified in the assurance activities for FTP_ITC_EXT.1.

The evaluator shall perform the following test for this requirement:

- ***Test 1:*** *The evaluator shall establish a session between the TOE and the audit server according to the configuration guidance provided. The evaluator shall then examine the traffic that passes between the audit server and the TOE during several activities of the evaluator's choice designed to generate audit data to be transferred to the audit server. The evaluator shall observe that these data are not able to be viewed in the clear during this transfer, and that they are successfully received by the audit server. The evaluator shall record the particular software (name, version) used on the audit server during testing.*

TSS
The evaluator shall examine the TSS to ensure it describes what happens when the local audit data store is full.
Guidance
The evaluator shall also examine the operational guidance to determine that it describes the relationship between the local audit data and the audit data that are sent to the audit log server. For example, when an audit event is generated, is it simultaneously sent to the external server and the local store, or is the local store used as a buffer and "cleared" periodically by sending the data to the audit server.
Tests
The evaluator shall perform operations that generate audit data and verify that this data is stored locally. The evaluator shall perform operations that generate audit data until the local storage space is exceeded and verifies that the TOE complies with the behavior defined in the ST for FAU_STG_EXT.1.2.

5.1.3 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]].*

].

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.

Evaluation Activities

FCS_CKM.1:
TSS
The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.
Guidance
The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all uses defined in this PP.
Tests
Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Key Generation for FIPS PUB 186-4 RSA Schemes
The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d.

Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

- *Random Primes:*
  - *Provable primes*
  - *Probable primes*
- *Primes with Conditions:*
  - *Primes p1, p2, q1,q2, p and q shall all be provable primes*
  - *Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes*
  - *Primes p1, p2, q1,q2, p and q shall all be probable primes*

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator shall seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.
FIPS 186-4 Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.

The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

- *Primes q and p shall both be provable primes*
- *Primes q and field prime p shall both be probable primes*

and two ways to generate the cryptographic group generator g:

- *Generator g constructed through a verifiable process*
- *Generator g constructed through an unverifiable process.*

The Key generation specifies 2 ways to generate the private key x:

- *len(q) bit output of RBG where 1 □ x □ q-1*
- *len(q) + 64 bit output of RBG, followed by a mod q-1 operation where 1 □ x □ q-1*

The security strength of the RBG shall be at least that of the security offered by the FFC parameter set.

To test the cryptographic and field prime generation method for the provable primes method and the group generator g for a verifiable process, the evaluator shall seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.

For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification shall also confirm

- *g != 0,1*
- *q divides p-1*
- *g^q mod p = 1*
- *g^x mod p = y*

for each FFC parameter set and key pair.

FCS_CKM.2 Cryptographic Key Establishment

FCS_CKM.2.1

The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method: [**selection**:

- *RSA-based key establishment schemes that meets the following: NIST Special Publication 800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography",*
- *Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A, "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, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"]*

]

Evaluation Activities

FCS_CKM.2:
TSS
The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

Guidance
The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

Tests
The evaluator shall verify the implementation of the key establishment schemes of the supported by the TOE using the applicable tests below.

Key Establishment Schemes

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the to implement the key agreement schemes correctly. To conduct this test, the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding TOE's public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information field

OI and TOE id fields.

If the TOE does not use a KDF defined in SP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

SP800-56B Key Establishment Schemes

The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

If the TOE acts as a sender, the following assurance activity shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

- *To conduct this test, the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.*

If the TOE acts as a receiver, the following assurance activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

To conduct this test, the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE shall not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM-KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

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 to 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 3rd party TOE modules. The assurance activities are designed with flexibility to address cases where the TOE manufacturer has limited insight into the behavior of 3rd 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 pseudo-random 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.

Evaluation Activities

FCS_CKM_EXT.4:
TSS
The evaluator shall check to ensure the TSS lists each type of key and its origin and location in memory or storage. The evaluator shall verify that the TSS describes when each type of key is cleared.
Tests
For each key clearing situation the evaluator shall perform one of the following activities:

- *The evaluator shall use appropriate combinations of specialized operational or development environments, development tools (debuggers, emulators, simulators, etc.), or instrumented builds (developmental, debug, or release) to demonstrate that keys are cleared correctly, including all intermediate copies of the key that may have been created internally by the TOE during normal cryptographic processing.*
- *In cases where testing reveals that 3rd-party software modules or programming language run-time environments do not properly overwrite keys, this fact must be documented. Likewise, it must be documented if there is no practical way to determine whether such modules or environments destroy keys properly.*
- *In cases where it is impossible or impracticable to perform the above tests, the evaluator shall describe how keys are destroyed in such cases, to include:*
  - *Which keys are affected*
  - *The reasons why testing is impossible or impracticable*
  - *Evidence that keys are destroyed appropriately (e.g., citations to component documentation, component developer/vendor attestation, component vendor test results)*
  - *Aggravating and mitigating factors that may affect the timeliness or execution of key destruction (e.g., caching, garbage collection, operating system memory management)*

Use of debug or instrumented builds of the TOE and TOE components is permitted in order to demonstrate that the TOE takes appropriate action to destroy keys. These builds should be based on the same source code as are release builds (of course, with instrumentation and debug-specific code added).

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.

Evaluation Activities

FCS_COP.1/UDE:
Assurance Activity Note:
The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Tests
AES-CBC Tests

AES-CBC Known Answer Tests

There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

KAT-1.*To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.*

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

KAT-2.*To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.*

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

KAT-3.*To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N].*

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

KAT-4.*To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].*

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i □ 10 . The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i □ 10 . The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with

the same key and IV using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
    if i == 1:
        CT[1] = AES-CBC-Encrypt(Key, IV, PT)
        PT = IV
    else:
        CT[i] = AES-CBC-Encrypt(Key, PT)
        PT = CT[i-1]
```

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-CCM Tests

The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:

128 bit and 256 bit keys

Two payload lengths.*One payload length shall be the shortest supported payload length, greater than or equal to zero bytes. The other payload length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits).*

Two or three associated data lengths.*One associated data length shall be 0, if supported. One associated data length shall be the shortest supported payload length, greater than or equal to zero bytes. One associated data length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of 2^{16} bytes, an associated data length of 2^{16} bytes shall be tested.*

Nonce lengths.*All supported nonce lengths between 7 and 13 bytes, inclusive, shall be tested.*

Tag lengths.*All supported tag lengths of 4, 6, 8, 10, 12, 14 and 16 bytes shall be tested.*

To test the generation-encryption functionality of AES-CCM, the evaluator shall perform the following four tests:

- ***Test 1:*** *For EACH supported key and associated data length and ANY supported payload, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*
- ***Test 2:*** *For EACH supported key and payload length and ANY supported associated data, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*
- ***Test 3:*** *For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator shall supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.*
- ***Test 4:*** *For EACH supported key and tag length and ANY supported associated data, payload and nonce length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*

To determine correctness in each of the above tests, the evaluator shall compare the ciphertext with the result of generation-encryption of the same inputs with a known good implementation.

To test the decryption-verification functionality of AES-CCM, for EACH combination of supported associated data length, payload length, nonce length and tag length, the evaluator shall supply a key value and 15 nonce, associated data and ciphertext 3-tuples and obtain either a FAIL result or a PASS result with the decrypted payload. The evaluator shall supply 10 tuples that should FAIL and 5 that should PASS per set of 15.

Additionally, the evaluator shall use tests from the IEEE 802.11-02/362r6 document "Proposed Test vectors for IEEE 802.11 TGi", dated September 10, 2002, Section 2.1 AES-CCMP Encapsulation Example and Section 2.2 Additional AES CCMP Test Vectors to further verify the IEEE 802.11-2007 implementation of AES-CCMP.

AES-GCM Test

The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

Two plaintext lengths.*One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.*

Three AAD lengths.*One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.*

Two IV lengths.*If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.*

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

XTS-AES Test

The evaluator shall test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:

- *256 bit (for AES-128) and 512 bit (for AES-256) keys*
- ***Three data unit (i.e., plaintext) lengths.*** *One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 2^{16} bits, whichever is smaller.*

using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.

AES Key Wrap (AES-KW) and Key Wrap with Padding (AES-KWP) Test

The evaluator shall test the authenticated encryption functionality of AES-KW for EACH combination of the following input parameter lengths:

- *128 and 256 bit key encryption keys (KEKs)*
- ***Three plaintext lengths.*** *One of the plaintext lengths shall be two semi-blocks (128 bits). One of the plaintext lengths shall be three semi-blocks (192 bits). The third data unit length shall be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).*

using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator shall use the AES-KW authenticated-encryption function of a known good implementation.

The evaluator shall test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption.

The evaluator shall test the authenticated-encryption functionality of AES-KWP using the same test as for AES-KW authenticated-encryption with the following change in the three plaintext lengths:

One plaintext length shall be one octet. One plaintext length shall be 20 octets (160 bits).

One plaintext length shall be the longest supported plaintext length less than or equal to 512 octets (4096 bits).

The evaluator shall test the authenticated-decryption functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption.

AES-CTR Test

- ***Test 1:*** *Known Answer Tests (KATs)*

  There are four Known Answer Tests (KATs) described below. For all KATs, the plaintext, initialization vector (IV), and ciphertext values shall be 128-bit blocks. The results from each test may either be obtained by the validator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

  Test 1a: *To test the encrypt functionality, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all zeros key, and the other five shall be encrypted with a 256-bit all zeros key. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input.*

  Test 1b: *To test the encrypt functionality, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value and an IV of all zeros. Five of the key values shall be 128-bit keys, and the other five shall be 256-bit keys. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using an all zero ciphertext value as input.*

  Test 1c: *To test the encrypt functionality, the evaluator shall supply the two sets of key values described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second shall have 256 256-bit keys. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1, N]. To test the decrypt functionality, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from decryption of the given ciphertext using the given key values and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit pairs. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros for i in [1, N]. The ciphertext value in each pair shall be the value that results in an all zeros plaintext when decrypted with its corresponding key.*

  Test 1d: *To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from encryption of the given plaintext using a 128-bit key value of all zeros and using a 256 bit key value of all zeros, respectively, and an IV of all zeros. Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 128-i bits be zeros, for i in [1, 128]. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input.*

- ***Test 2:*** *Multi-Block Message Test*

  The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key, IV, and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality by decrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key using a known good implementation.

- ***Test 3:*** *Monte-Carlo Test*

  For AES-CTR mode perform the Monte Carlo Test for ECB Mode on the encryption engine of the counter mode implementation. There is no need to test the decryption engine.

  The evaluator shall test the encrypt functionality using 200 plaintext/key pairs. 100 of these shall use 128 bit keys, and 100 of these shall use 256 bit keys. The plaintext values shall be

128-bit blocks. For each pair, 1000 iterations shall be run as follows:

For AES-ECB mode
 # Input: PT, Key
for i = 1 to 1000:
CT[i] = AES-ECB-Encrypt(Key, PT)
PT = CT[i]

The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

If "invoke platform-provided" is selected, the evaluator confirms that SSH connections are only successful if appropriate algorithms and appropriate key sizes are configured. To do this, the evaluator shall perform the following tests:

- ***Test 1:** [Conditional: TOE is an SSH server] The evaluator shall configure an SSH client to connect with an invalid cryptographic algorithm and key size for each listening SSH socket connection on the TOE. The evaluator initiates SSH client connections to each listening SSH socket connection on the TOE and observes that the connection fails in each attempt.*
- ***Test 2:** [Conditional: TOE is an SSH client] The evaluator shall configure a listening SSH socket on a remote SSH server that accepts only invalid cryptographic algorithms and keys. The evaluator uses the TOE to attempt an SSH connection to this server and observes that the connection fails.*

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).

Evaluation Activities

FCS_COP.1/HASH:
TSS
The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.
Guidance
The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required hash sizes is present.
Tests
SHA-1 and SHA-2 Tests *The TSF hashing functions can be implemented in one of two modes. The first mode is the byteoriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bitoriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bitoriented vs. the byteoriented testmacs.*

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

Assurance Activity Note:
The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Short Messages Test Bit-oriented Mode

The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The

message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test Byte-oriented Mode

The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test Bit-oriented Mode

*The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where 1 □ i □ m . The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*

Selected Long Messages Test Byte-oriented Mode

*The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 □ i □ m/8 . The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*

Pseudo-randomly Generated Messages Test

This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

SHA-3 Tests *The tests below are derived from the The Secure Hash Algorithm-3 Validation System (SHA3VS), Updated: April 7, 2016, from the National Institute of Standards and Technology.*

For each SHA-3-XXX implementation, XXX represents d, the digest length in bits. The capacity, c, is equal to 2d bits. The rate is equal to 1600-c bits.

The TSF hashing functions can be implemented with one of two orientations. The first is a bit-oriented mode that hashes messages of arbitrary length. The second is a byte-oriented mode that hashes messages that are an integral number of bytes in length (i.e., the length (in bits) of the message to be hashed is divisible by 8). Separate tests for each orientation are given below.

The evaluator shall perform all of the following tests for each hash algorithm and orientation implemented by the TSF and used to satisfy the requirements of this PP. The evaluator shall compare digest values produced by a known-good SHA-3 implementation against those generated by running the same values through the TSF.

Short Messages Test, Bit-oriented Mode

The evaluators devise an input set consisting of rate+1 short messages. The length of the messages ranges sequentially from 0 to rate bits. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF. The message of length 0 is omitted if the TOE does not support zero-length messages.

Short Messages Test, Byte-oriented Mode

The evaluators devise an input set consisting of rate/8+1 short messages. The length of the messages ranges sequentially from 0 to rate/8 bytes, with each message being an integral number of bytes. The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF. The message of length 0 is omitted if the TOE does not support zero-length messages.

Selected Long Messages Test, Bit-oriented Mode

The evaluators devise an input set consisting of 100 long messages ranging in size from rate+ (rate+1) to rate+(100(rate+1)), incrementing by rate+1. (For example, SHA-3-256 has a rate of 1088 bits. Therefore, 100 messages will be generated with lengths 2177, 3266, ..., 109988 bits.) The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*

Selected Long Messages Test, Byte-oriented Mode

The evaluators devise an input set consisting of 100 messages ranging in size from (rate+ (rate+8)) to (rate+100(rate+8)), incrementing by rate+8. (For example, SHA-3-256 has a rate of 1088 bits. Therefore 100 messages will be generated of lengths 2184, 3280, 4376, ..., 110688 bits.) The message text shall be pseudo-randomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*

Pseudo-randomly Generated Messages Monte Carlo) Test, Byte-oriented Mode

The evaluators supply a seed of d bits (where d is the length of the message digest produced by the hash function to be tested. This seed is used by a pseudorandom function to generate 100,000 message digests. One hundred of the digests (every 1000th digest) are recorded as checkpoints. The TOE then uses the same procedure to generate the same 100,000 message digests and 100 checkpoint values. The evaluators then compare the results generated ensure that the correct result is produced when the messages are generated by the TSF.

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.

Evaluation Activities

FCS_COP.1/Sig:
Tests
Assurance Activity Note:
The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

ECDSA FIPS 186-4 Signature Verification Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

Signature Generation Test

The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test, the evaluator shall generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages.

The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

Signature Verification Test

The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

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

FCS_COP.1.1/KeyHash

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*, *256*, *384*, *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.

Evaluation Activities

FCS_COP.1/KeyHash:
TSS
The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.
Tests
Assurance Activity Note:
The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and IV using a known good implementation.

FCS_ENT_EXT.1 Extended: 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*, *passthrough 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 “passthrough 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.

Evaluation Activities

FCS_ENT_EXT.1:
TSS
The evaluator shall verify that the TSS describes how the TOE provides entropy to Guest VM"s, and how to access the interface to acquire entropy or random numbers. The evaluator shall verify that the TSS describes the mechanisms for ensuring that one VM does not affect the entropy acquired by another.
Tests
The evaluator shall perform the following tests:

- ***Test 1:*** *The evaluator shall invoke entropy from each Guest VM. The evaluator shall verify that each VM acquires values from the interface.*
- ***Test 2:*** *The evaluator shall invoke entropy from multiple VMs as nearly simultaneously as practicable. The evaluator shall verify that the entropy used in one VM is not identical to that invoked from the other VMs.*

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-44 512) 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.

Evaluation Activities

FCS_RBG_EXT.1:
Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix D, Entropy Documentation and Assessment.

Tests
The evaluator shall also perform the following tests, depending on the standard to which the RBG conforms.

The evaluator shall perform 15 trials for the RBG implementation. If the RBG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RBG functionality.

If the RBG has prediction resistance enabled, each trial consists of (1) instantiate drbg, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. "generate one block of random bits" means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90A).

If the RBG does not have prediction resistance, each trial consists of (1) instantiate drbg, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to re-seed. The final value is additional input to the second generate call.

The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

- *Entropy input: the length of the entropy input value must equal the seed length*
- *Nonce: If a nonce is supported (CTR_DRBG with no df does not use a nonce), the nonce bit length is one-half the seed length. Personalization string: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.*
- *Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.*

5.1.4 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.

Evaluation Activities

FDP_HBI_EXT.1:
TSS
The evaluator shall ensure that the TSS provides evidence that hardware-based isolation mechanisms are used to constrain VMs when VMs have direct access to physical devices, including an explanation of the conditions under which the TSF invokes these protections.

Guidance
The evaluator shall verify that the operational guidance contains instructions on how to ensure that the platform-provided, hardware-based mechanisms are enabled.

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: This requirement specifies that the VMM controls access to physical platform resources, and indicates that it must be configurable, but does not specify the means by which that is done. The ST author should list the physical platform resources that can be configured for Guest VM access by the administrator. Guest VMs may not be allowed direct access to certain physical resources; those resources are listed in the second element. 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 listed in the third element; if there are no such resources, then "no physical platform resources" is selected.

Evaluation Activities

FDP_PPR_EXT.1:
TSS
The evaluator shall examine the TSS to determine that it describes the mechanism by which the VMM controls a Guest VM's access to physical platform resources is described. This description shall cover all of the physical platforms allowed in the evaluated configuration by the ST. This description shall include how the VMM distinguishes among Guest VMs, and how each physical platform resource that is controllable (that is, listed in the assignment statement in the first element) is identified. The evaluator shall ensure that the TSS describes how the Guest VM is associated with each physical resources, and how other Guest VMs cannot access a physical resource without being granted explicit access. For TOEs that implement a robust interface (other than just "allow access" or "deny access"), the evaluator shall ensure that the TSS describes the possible operations or modes of access between a Guest VMs and physical platform resources.

If physical resources are listed in the second element, the evaluator shall examine the TSS and operational guidance to determine that there appears to be no way to configure those resources for access by a Guest VM. The evaluator shall document in the evaluation report their analysis of why the controls offered to configure access to physical resources can't be used to specify access to the resources identified in the second element (for example, if the interface offers a drop-down list of resources to assign, and the denied resources are not included on that list, that would be sufficient justification in the evaluation report).
Guidance
The evaluator shall examine the operational guidance to determine that it describes how an administrator is able to configure access to physical platform resources for Guest VMs for each platform allowed in the evaluated configuration according to the ST. The evaluator shall also determine that the operational guidance identifies those resources listed in the second and third elements of the component and notes that access to these resources is explicitly denied/allowed, respectively.
Tests
Using the operational guidance, the evaluator shall perform the following tests for each physical platform identified in the ST:

- ***Test 1:*** *For each physical platform resource identified in the first element, the evaluator shall configure a Guest VM to have access to that resource and show that the Guest VM is able to successfully access that resource.*
- ***Test 2:*** *For each physical platform resource identified in the first element, the evaluator shall configure the system such that a Guest VM does not have access to that resource and show that the Guest VM is unable to successfully access that resource.*
- ***Test 3:*** *[conditional]: For TOEs that have a robust control interface, the evaluator shall*

exercise each element of the interface as described in the TSS and the operational guidance to ensure that the behavior described in the operational guidance is exhibited.

- ***Test 4:*** *[conditional]: If the TOE explicitly denies access to certain physical resources, the evaluator shall attempt to access each listed (in FDP_PPR_EXT.1.2) physical resource from a Guest VM and observe that access is denied.*
- ***Test 5:*** *[conditional]: If the TOE explicitly allows access to certain physical resources, the evaluator shall attempt to access each listed (in FDP_PPR_EXT.1.3) physical resource from a Guest VM and observe that the access is allowed. If the operational guidance specifies that access is allowed simultaneously by more than one Guest VM, the evaluator shall attempt to access each resource listed from more than one Guest VM and show that access is allowed.*

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.

Evaluation Activities

FDP_RIP_EXT.1:
TSS
The evaluator shall ensure that the TSS documents the process used for clearing physical memory prior to allocation to a Guest VM, providing details on when and how this is performed. Additionally, the evaluator shall ensure that the TSS documents the conditions under which physical memory is not cleared prior to allocation to a Guest VM, and describes when and how the memory is cleared.

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 prior to allocation to a Guest VM.

Application Note: Disk storage must be zeroed before it is made accessible to a VM for use by a Guest OS.

The purpose of this requirement is to ensure that a VM does not receive disk storage containing data previously used by another VM or the host.

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.

Evaluation Activities

FDP_RIP_EXT.2:
TSS
The evaluator shall ensure that the TSS documents the conditions under which physical disk storage is not cleared prior to allocation to a Guest VM. The evaluator shall also ensure that the TSS documents the metadata used in its virtual disk files.
Tests
The evaluator shall perform the following test:

- ***Test 1:*** *On the host, the evaluator creates a file that is more than half the size of a connected physical storage device (or multiple files whose individual sizes add up to more than half the size of the storage media). This file (or files) shall be filled entirely with a non-*

zero value. Then, the file (or files) shall be released (freed for use but not cleared). Next, the evaluator (as a VS Administrator) creates a virtual disk at least that large on the same physical storage device and connects it to a powered-off VM. Then, from outside the Guest VM, scan through and check that all the non-metadata (as documented in the TSS) in the file corresponding to that virtual disk is set to zero.

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 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.1.3

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. This should be the same list of mechanisms supplied for FMT_MSA_EXT.1.

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.1.2 is met if FPT_HCL_EXT.1.2 is met for those Hypercall functions (VM access to Hypercall functions is configurable).

For data transfer mechanisms that use shared physical devices, FDP_VMS_EXT.1.2 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.1.2 is met if FDP_VNC_EXT.1.1 is met (VM access to virtual networks is configurable).

Evaluation Activities

FDP_VMS_EXT.1:
TSS
The evaluator shall examine the TSS to verify that it documents all inter-VM communications mechanisms (as defined above), and explains how the TSF prevents the transfer of data between VMs outside of the mechanisms listed in FDP_VMS_EXT.1.1.
Guidance
The evaluator shall examine the operational guidance to ensure that it documents how to configure all inter-VM communications mechanisms, including how they are invoked and how they are disabled.
Tests
The evaluator shall perform the following tests for each documented inter-VM communications channel:

- ***Test 1:***
  a. *Create two VMs without specifying any communications mechanism or overriding the default configuration.*

b. *Test that the two VMs cannot communicate through the mechanisms selected in FMT_MSA_EXT.1.1.*
c. *Create two new VMs, overriding the default configuration to allow communications through a channel selected in FMT_MSA_EXT.1.1.*
d. *Test that communications can be passed between the VMs through the channel.*
e. *Create two new VMs, the first with the inter-VM communications channel currently being tested enabled, and the second with the inter-VM communications channel currently being tested disabled.*
f. *Test that communications cannot be passed between the VMs through the channel.*
g. *As an Administrator, enable inter-VM communications between the VMs on the second VM.*
h. *Test that communications can be passed through the inter-VM channel.*
i. *As an Administrator again, disable inter-VM communications between the two VMs.*
j. *Test that communications can no longer be passed through the channel.*

FDP_VMS_EXT.1.2 is met if communication is successful in step (d) and unsuccessful in step (f).

FMT_MSA_EXT.1.1 is met if communication is unsuccessful in step (b). FMT_MSA_EXT.1.2 is met if communication is successful in step (d). Additionally, FMT_MSA_EXT.1 requires that the evaluator verifies that the TSS documents the inter-VM communications mechanisms as described above.

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 isolated from one another to provide assurance 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.

Evaluation Activities

FDP_VNC_EXT.1:
TSS
The evaluator shall examine the TSS (or a proprietary annex) to verify that it describes the mechanism by which virtual network traffic is ensured to be visible only to Guest VMs configured to be on that virtual network.
Guidance
The evaluator must ensure that the Operational Guidance describes how to create virtualized networks and connect VMs to each other and to physical networks.
Tests

- ***Test 1:*** *The evaluator shall assume the role of the Administrator and attempt to configure a VM to connect to a network component. The evaluator shall verify that the attempt is successful. The evaluator shall then assume the role of an unprivileged user and attempt the same connection. If the attempt fails, or there is no way for an unprivileged user to configure VM network connections, the requirement is met.*
- ***Test 2:*** *The evaluator shall assume the role of the Administrator and attempt to configure a VM to connect to a physical network. The evaluator shall verify that the attempt is successful. The evaluator shall then assume the role of an unprivileged user and make the same attempt. If the attempt fails, or there is no way for an unprivileged user to configure VM network connections, the requirement is met.*

5.1.5 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 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 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 authentication failures and local authentication failures in the number of unsuccessful authentication attempts.

Evaluation Activities

FIA_AFL_EXT.1:
Tests
The evaluator shall perform the following tests for each credential selected in FIA_AFL_EXT.1.1:

- ***Test 1:** The evaluator will set an Administrator-configurable threshold n for failed attempts, or note the ST-specified assignment.*
  - ***Test 1:** The evaluator will attempt to authenticate remotely with the credential n-1 times. The evaluator will then attempt to authenticate using a good credential and verify that authentication is successful.*
  - ***Test 2:** The evaluator will make n attempts to authenticate using a bad credential. The evaluator will then attempt to authenticate using a good credential and verify that the attempt is unsuccessful. Note that the authentication attempts and lockouts must also be logged as specified in FAU_GEN.1.*
  - ***Test 3:** After reaching the limit for unsuccessful authentication attempts the evaluator will proceed as follows:*
    - ***Test 1:** If the Administrator action selection in FIA_AFL_EXT.1.2 is selected, then the evaluator will confirm by testing that following the operational guidance and performing each action specified in the ST to re-enable the remote Administrator's access results in successful access (when using valid credentials for that Administrator).*
    - ***Test 2:** If the time period selection in FIA_AFL_EXT.1.2 is selected, the evaluator will wait for just less than the time period configured and show that an authentication attempt using valid credentials does not result in successful access. The evaluator will then wait until just after the time period configured and show that an authentication attempt using valid credentials results in successful access.*

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 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*].

Evaluation Activities

FIA_UAU.5:
Tests
If ‘username and password authentication‘ is selected, the evaluator will configure the VS with a known username and password and conduct the following tests:

- ***Test 1:*** *The evaluator will attempt to authenticate to the VS using the known username and password. The evaluator will ensure that the authentication attempt is successful.*
- ***Test 2:*** *The evaluator will attempt to authenticate to the VS using the known username but an incorrect password. The evaluator will ensure that the authentication attempt is unsuccessful.*

If ‘username and PIN that releases an asymmetric key‘ is selected, the evaluator will examine the TSS for guidance on supported protected storage and will then configure the TOE or OE to establish a PIN which enables release of the asymmetric key from the protected storage (such as a TPM, a hardware token, or isolated execution environment) with which the VS can interface. The evaluator will then conduct the following tests:

- ***Test 1:*** *The evaluator will attempt to authenticate to the VS using the known user name and PIN. The evaluator will ensure that the authentication attempt is successful.*
- ***Test 2:*** *The evaluator will attempt to authenticate to the VS using the known user name but an incorrect PIN. The evaluator will ensure that the authentication attempt is unsuccessful.*

If ‘X.509 certificate authentication‘ is selected, the evaluator will generate an X.509v3 certificate for an Administrator user with the Client Authentication Enhanced Key Usage field set. The evaluator will provision the VS for authentication with the X.509v3 certificate. The evaluator will ensure that the certificates are validated by the VS as per FIA_X509_EXT.1.1 and then conduct the following tests:

- ***Test 1:*** *The evaluator will attempt to authenticate to the VS using the X.509v3 certificate. The evaluator will ensure that the authentication attempt is successful.*
- ***Test 2:*** *The evaluator will generate a second certificate identical to the first except for the public key and any values derived from the public key. The evaluator will attempt to authenticate to the VS with this certificate. The evaluator will ensure that the authentication attempt is unsuccessful.*

If ‘SSH public-key credential authentication‘ is selected, the evaluator shall generate a public-private host key pair on the TOE using RSA or ECDSA, and a second public-private key pair on a remote client. The evaluator shall provision the VS with the client public key for authentication over SSH, and conduct the following tests:

- ***Test 1:*** *The evaluator will attempt to authenticate to the VS using a message signed by the client private key that corresponds to provisioned client public key. The evaluator will ensure that the authentication attempt is successful.*
- ***Test 2:*** *The evaluator will generate a second client key pair and will attempt to authenticate to the VS with the private key over SSH without first provisioning the VS to support the new key pair. The evaluator will ensure that the authentication attempt is unsuccessful.*

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.

Evaluation Activities

FIA_UIA_EXT.1:
TSS
The evaluator shall examine the TSS to determine that it describes the logon process for each logon method (local, remote (HTTPS, SSH, etc.)) supported for the product. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a "successful logon". The evaluator shall examine the operational guidance to determine that any necessary preparatory steps (e.g., establishing credential material such as pre-shared keys, tunnels, certificates, etc.) to logging in are described. For each supported the login method, the evaluator shall ensure the operational guidance provides clear instructions for successfully logging on. If configuration is necessary to ensure the services provided before login are limited, the evaluator shall determine that the operational guidance provides sufficient instruction on limiting the allowed services.

5.1.6 Security Management (FMT)

FMT_MSA_EXT.1 Default Data Sharing Configuration

FMT_MSA_EXT.1.1

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

FMT_MSA_EXT.1.2

The TSF shall allow Administrators to specify alternative initial configuration values to override the default values when a Guest VM is created.

Application Note: By default, the VMM must enforce a policy prohibiting sharing of data between VMs. The default policy applies to all mechanisms for sharing data between VMs, including inter-VM communication channels, shared physical devices, shared virtual devices, and virtual networks. The default policy does not apply to covert channels and architectural side-channels.

The list of data sharing mechanisms implemented by the TOE is contained in the selection in FDP_VMS_EXT.1.1.
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

.

Evaluation Activities

FMT_MSA_EXT.1:
Tests
This requirement is met if FDP_VMS_EXT.1 is met.

FMT_SMO_EXT.1 Separation of Management and Operational Networks

FMT_SMO_EXT.1.1

The TSF shall support the configuration of separate management and operational networks through [**selection**: *physical means*, *logical means*, *trusted channel*].

Application Note: Management communications must be separate from user workloads. Administrative communications—including communications between physical hosts concerning load balancing, audit data, VM startup and shutdown —must be separate from guest operational networks.

"Physical means" refers to using separate physical networks for management and operational networks. For example, the machines in the management network are connected by separate cables plugged into separate and dedicated physical ports on each physical host.

"Logical means" refers to using separate network cables to connect physical hosts together using general-purpose networking ports. The management and operational networks are kept separate within the hosts using separate virtualized networking components.

If the ST author selects "trusted channel", then the protocols used for network separation must be selected in FTP_ITC_EXT.1.

Evaluation Activities

FMT_SMO_EXT.1:
TSS
The evaluator shall examine the TSS to verify that it describes how management and operational networks may be separated.
Guidance
The evaluator shall examine the operational guidance to verify that it details how to configure the VS with separate Management and Operational Networks.
Tests
The evaluator shall configure the management network as documented. If separation is cryptographic or logical, then the evaluator shall capture packets on the management network. If Guest network traffic is detected, the requirement is not met.

5.1.7 Protection of the TSF (FPT)

FPT_DVD_EXT.1 Non-Existence of Disconnected Virtual Devices

FPT_DVD_EXT.1.1

The TSF shall limit a Guest VM's access to virtual devices to those that are 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 accessible by the VM.

Such interfaces include memory buffers and processor I/O ports.

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

Evaluation Activities

FPT_DVD_EXT.1:
Tests
The evaluator shall connect a device to a VM, then using a device driver running in the guest, scan the VM's processor I/O ports to ensure that the device's ports are present. (The device's interface should be documented in the TSS under FPT_VDP_EXT.1.) The evaluator shall remove the device from the VM and run the scan again. This requirement is met if the device's I/O ports are no longer present.

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 mechanism(s) and uses the assignment to include mechanisms not listed in the selection, if any.

Evaluation Activities

FPT_EEM_EXT.1:
TSS
The evaluator shall examine the TSS to ensure that it states, for each platform listed in the ST, the execution environment-based vulnerability mitigation mechanisms used by the TOE on that platform. The evaluator shall ensure that the lists correspond to what is specified in FPT_EEM_EXT.1.1.

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.

Evaluation Activities

FPT_HAS_EXT.1:
TSS
The evaluator shall examine the TSS to ensure that it states, for each platform listed in the ST, the hardware assists and memory-handling extensions used by the TOE on that platform. The evaluator shall ensure that these lists correspond to what is specified in the applicable FPT_HAS_EXT component.

FPT_HCL_EXT.1 Hypercall Controls

FPT_HCL_EXT.1.1

The TSF shall provide a Hypercall interface for Guest VMs to use to invoke functionality provided by the VMM.

FPT_HCL_EXT.1.2

The TSF shall allow administrators to configure any VM’s Hypercall interface to disable access to individual functions, all functions, or groups of functions.

FPT_HCL_EXT.1.3

The TSF shall permit exceptions to the configuration of the following Hypercall interface functions: [**assignment**: *list of functions that are not subject to the configuration controls in FPT_HCL_EXT.1.2*].

FPT_HCL_EXT.1.4

The TSF shall validate the parameters passed to the hypercall interface prior to execution of the VMM functionality exposed by that interface.

Application Note: The purpose of this requirement is to help ensure the integrity of the VMM by defining the attack surface exposed to Guest VMs through Hypercalls, testing the mechanisms for reducing that attack surface by

disabling Hypercalls, and ensuring that Hypercall parameters are properly validated prior to use by the VMM.

A Hypercall interface allows a set of VMM functions to be invoked by software running within a VM. Hypercall interfaces are used by virtualization-aware VMs to communicate and exchange data with the VMM. For example, a VM could use a hypercall interface 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 VMs, these interfaces constitute attack surface. In order to minimize attack surface, these interfaces must be limited to the minimum needed to support VM functionality.

For the selection in FPT_HCL_EXT.1.2, the ST author selects the applicable actions that administrators can perform to configure functions supported by the interface.

For the assignment in FPT_HCL_EXT.1.3, the ST author lists the interface functions that cannot be configured per FPT_HCL_EXT.1.2.

A vendor-provided test harness may reduce evaluation time.

There is no expectation that the evaluator will need to review source code in order to accomplish the Assurance Activity. The evaluator documentation review should ensure that there are documented Hypercall functions in the TSS, that each documented Hypercall function contains the specified information, and that there are not obvious or publicly known Hypercall functions missing.

Evaluation Activities

FPT_HCL_EXT.1:
TSS
The evaluator shall examine the TSS or operational guidance to ensure it documents all Hypercall functions at the level necessary for the evaluator to disable the functions and run tests 1 and 2, below. Documentation must include, for each function, how to call the function, function parameters and legal values, configuration settings for enabling/disabling the function, and conditions under which the function can be disabled. The TSS must also specify those functions that cannot be disabled. While there is no expectation that the evaluator will need to examine source code in order to accomplish this Assurance Activity, the evaluator must ensure that there are no obvious or publicly known Hypercall functions missing from the TSS.
Guidance
The evaluator shall examine the operational guidance to ensure it contains instructions for how to configure interface functions per FPT_HCL_EXT.1.2.
Tests
The evaluator shall perform the following tests:

- ***Test 1:*** *For each configurable function that meets FPT_HCL_EXT.1.2, the evaluator shall follow the operational guidance to enable the function. The evaluator shall then attempt to call each function from within the VM. If the call is allowed, then the test succeeds.*
- ***Test 2:*** *For each configurable function that meets FPT_HCL_EXT.1.2, the evaluator shall configure the TSF to disable the function. The evaluator shall then attempt to call the function from within the VM. If the call is blocked, then the test succeeds.*

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 switch, etc.).

Evaluation Activities

FPT_RDM_EXT.1:
TSS
The evaluator shall examine the TSS to ensure it describes the association between the media or devices supported by the TOE and the actions that can occur when switching information domains. The evaluator shall examine the operational guidance to ensure it documents how an administrator or user configures the behavior of each media or device.
Tests
The evaluator shall perform the following test for each listed media or device:

- ***Test 1:*** *The evaluator shall configure two VMs that are members of different information domains, with the media or device connected to one of the VMs. The evaluator shall disconnect the media or device from the VM and connect it to the other VM. The evaluator shall verify that the action performed is consistent with the action assigned in the TSS.*

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, certificates are validated in accordance with FIA_X509_EXT.1 and should be selected 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.

Evaluation Activities

FPT_TUD_EXT.1:
TSS
The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system software. Updates to the TOE either have a hash associated with them, or are signed by an authorized source. The evaluator shall verify that the description includes either a digital signature or published hash verification of the software before installation and that installation fails if the verification fails. The evaluator shall verify that the TSS describes the method by which the digital signature or published hash is verified to include how the candidate updates are obtained, the processing associated with verifying the update, and the actions that take place for both successful and unsuccessful verification. If digital signatures are used, the evaluator shall also ensure the definition of an authorized source is contained in the TSS.

If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the TSS contains a description of how the certificates are contained on the device. The evaluator also ensures that the TSS (or administrator guidance) describes how the certificates are installed/updated/selected, if necessary.
Tests
The evaluator shall perform the following tests:

- ***Test 1:*** *The evaluator performs the version verification activity to determine the current version of the product. The evaluator obtains a legitimate update using procedures described in the operational guidance and verifies that it is successfully installed on the TOE. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update.*
- ***Test 2:*** *The evaluator performs the version verification activity to determine the current version of the product. The evaluator obtains or produces illegitimate updates as defined below, and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:*
  1. *A modified version (e.g., using a hex editor) of a legitimately signed or hashed update*
  2. *An image that has not been signed/hashed*
  3. *An image signed with an invalid hash or invalid signature (e.g., by using a different key as expected for creating the signature or by manual modification of a legitimate hash/signature)*

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.

Evaluation Activities

FPT_VDP_EXT.1:
TSS
The evaluator shall examine the TSS to ensure it lists all virtual devices accessible by the guest OS. The TSS, or a separate proprietary document, must also document all virtual device interfaces at the level of I/O ports - including port number(s) (absolute or relative to a base), port name, and a description of legal input values.

The TSS must also describe the expected behavior of the interface when presented with illegal input values. This behavior must be deterministic and indicative of parameter checking by the TSF.

The evaluator must ensure that there are no obvious or publicly known virtual I/O ports missing from the TSS.

There is no expectation that evaluators will examine source code to verify the "all" part of the Assurance Activity.
Tests
For each virtual device interface, the evaluator shall attempt to access the interface using at least one parameter value that is out of range or illegal. The test is passed if the interface behaves in the manner documented in the TSS. Interfaces that do not have input parameters need not be tested. This test can be performed in conjunction with the tests for FPT_DVD_EXT.1.

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:

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

Of the above, (a) does not apply to platforms that do not support SMM. The rationale behind activity (c) 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.

Evaluation Activities

FPT_VIV_EXT.1:
TSS
The evaluator shall verify that the TSS (or a proprietary annex to the TSS) describes how the TSF ensures that guest software cannot degrade or disrupt the functioning of other VMs, the VMM or the platform. And how the TSF prevents guests from invoking higher-privilege platform code, such as the examples in the note.
Guidance
There is no operational guidance for this requirement.
Tests
There are no tests for this requirement.

5.1.8 TOE Access (FTA)

FTA_TAB.1 TOE Access Banner

FTA_TAB.1.1

Before establishing an administrative user session, the TSF shall display a security Administrator-specifiedad 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.

Evaluation Activities

FTA_TAB.1:
Tests
The evaluator shall configure the TOE to display the advisory warning message "TEST TEST Warning Message TEST TEST". The evaluator shall then log out and confirm that the advisory message is displayed before logging can occur.

5.1.9 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 Extended Package for Secure Shell*

] 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 Extended Package for Secure Shell. The SSH package imports the X509 requirements if necessary, so selecting SSH here does not automatically cause their inclusion.

Selection of IPSec also automatically causes inclusion of the X509 requirements.

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.

Evaluation Activities

FTP_ITC_EXT.1:
TSS
The evaluator will review the TSS to determine that it lists all trusted channels the TOE uses for remote communications, including both the external entities and/or remote users used for the channel as well as the protocol that is used for each.
Tests
The evaluator will configure the TOE to communicate with each external IT entity and/or type of remote user identified in the TSS. The evaluator will monitor network traffic while the VS performs communication with each of these destinations. The evaluator will ensure that for each session a trusted channel was established in conformance with the protocols identified in the selection.

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.

Evaluation Activities

FTP_UIF_EXT.1:
TSS
The evaluator shall ensure that the TSS lists the supported user input devices.
Guidance
The evaluator shall ensure that the operational guidance specifies how the current input focus is indicated to the user.
Tests
For each supported input device, the evaluator shall demonstrate that the input from each device listed in the TSS is directed to the VM that is indicated to have the input focus.

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.

Evaluation Activities

FTP_UIF_EXT.2:
TSS
The evaluator shall ensure that the TSS describes the mechanism for identifying VMs to the user, how identities are assigned to VMs, and how conflicts are prevented.
Tests
The evaluator shall perform the following test:

The evaluator shall attempt to create and start at least three Guest VMs on a single display device where the evaluator attempts to assign two of the VMs the same identifier. If the user interface displays different identifiers for each VM, then the requirement is met. Likewise, the requirement is met if the system refuses to create or start a VM when there is already a VM with the same identifier.

5.1.10 TOE Security Functional Requirements Rationale

The following rationale provides justification for each security objective for the TOE, showing that the SFRs are suitable to meet and achieve the security objectives:

Table 4: SFR Rationale

OBJECTIVE	ADDRESSED BY	RATIONALE
O.VM_ISOLATION	FAU_GEN.1	Audit events can report attempts to breach isolation.
	FCS_CKM_EXT.4	Requires cryptographic key destruction to protect domain data in shared storage.
	FDP_PPR_EXT.1	Requires support for reducing attack surface through disabling access to unneeded physical platform resources.
	FDP_RIP_EXT.1	Ensures that domain data is cleared from memory before memory is re-allocated.
	FDP_RIP_EXT.2	Ensures that domain data is cleared from storage before the storage is re-allocated.
	FDP_VMS_EXT.1	Ensures that authorized data transfers between VMs are done securely.
	FDP_VNC_EXT.1	Ensures that network traffic is visible only to VMs configured to be that network.
	FPT_DVD_EXT.1	Ensures that VMs can access only those virtual devices that they are configured to access.
	FPT_EEM_EXT.1	Requires that the TOE use security mechanisms supported by the physical platform.
	FPT_HAS_EXT.1	Requires that the TOE use platform-supported virtualization assists to reduce attack surface.
	FPT_HCL_EXT.1	Requires that Administrators can disable unneccessary hypercall interfaces to reduce attack surface.
	FPT_VDP_EXT.1	Requires validation of parameter data passed to the hardware abstraction by untrusted VMs.
	FPT_VIV_EXT.1	Ensures that untrusted VMs cannot invoke priviledged code without proper hypervisor mediation.
O.VMM_INTEGRITY	FAU_GEN.1	Audit events can report potential

		integrity breaches and attempts.
	FCS_CKM.1	Requires generation of asymmetric keys for protection of integrity measures.
	FCS_COP.1	Ensures proper functioning of cryptographic algorithms used to protect data integrity.
	FCS_RBG_EXT.1	Requires that the TOE has access to high-quality entropy for cryptographic purposes.
	FDP_PPR_EXT.1	Requires support for reducing attack surface through disabling access to unneeded physical platform resources.
	FDP_VMS_EXT.1	Ensures that authorized data transfers between VMs are done securely.
	FDP_VNC_EXT.1	Ensures that network traffic is visible only to VMs configured to be that network.
	FPT_EEM_EXT.1	Requires that the TOE use security mechanisms supported by the physical platform.
	FPT_HAS_EXT.1	Requires that the TOE use platform-supported virtualization assists to reduce attack surface.
	FPT_VDP_EXT.1	Requires validation of parameter data passed to the hardware abstraction by untrusted VMs.
	FPT_VIV_EXT.1	Ensures that untrusted VMs cannot invoke priviledged code without proper hypervisor mediation.
O.PLATFORM_INTEGRITY	FDP_HBI_EXT.1	Requires that the TOE use platform-supported mechanisms for access to physical devices.
	FDP_PPR_EXT.1	Requires support for reducing attack surface through disabling access to unneeded physical platform resources.
	FDP_VMS_EXT.1	Ensures that authorized data transfers between VMs are done securely.
	FDP_VNC_EXT.1	Ensures that network traffic is visible only to VMs configured to be that network.
	FPT_DVD_EXT.1	Ensures that VMs cannot access virtual devices that they are not onfigured to access.
	FPT_EEM_EXT.1	Requires that the TOE use security mechanisms supported by the physical platform.
	FPT_HAS_EXT.1	Requires that the TOE use platform-supported virtualization assists to reduce attack surface.
	FPT_HCL_EXT.1	Requires that Administrators can disable unneccessary hypercall interfaces to reduce attack surface.
	FPT_VDP_EXT.1	Requires validation of parameter data passed to the hardware abstraction by untrusted VMs.
	FPT_VIV_EXT.1	Ensures that untrusted VMs cannot

		invoke priviledged code without proper hypervisor mediation.
O.DOMAIN_INTEGRITY	FCS_CKM_EXT.4	Requires cryptographic key destruction to protect domain data in shared storage.
	FCS_ENT_EXT.1	Requires that domains have access to high-quality entropy for cryptographic purposes.
	FCS_RBG_EXT.1	Requires that the TOE has access to high-quality entropy for cryptographic purposes.
	FDP_RIP_EXT.1	Ensures that domain data is cleared from memory before memory is re-allocated to another domain.
	FDP_RIP_EXT.2	Ensures that domain data is cleared from storage before the storage is re-allocated to another domain.
	FDP_VMS_EXT.1	Ensures that authorized data transfers between domains are done securely.
	FDP_VNC_EXT.1	Ensures that network traffic is visible only to VMs configured to be that network.
	FPT_EEM_EXT.1	Requires that the TOE use security mechanisms supported by the physical platform.
	FPT_HAS_EXT.1	Requires that the TOE use platform-supported virtualization assists to reduce attack surface.
	FPT_RDM_EXT.1	Requires support for rules for switching removeable media between domains to reduce the chance of data spillage.
	FPT_VDP_EXT.1	Requires validation of parameter data passed to the hardware abstraction by untrusted VMs.
	FTP_UIF_EXT.1	Ensures that users are able to determine the domain with the current input focus.
	FTP_UIF_EXT.2	Ensures that users can know the identity of any VM that they can access.
O.MANAGEMENT_ACCESS	FAU_GEN.1	Audit events report attempts to access the management subsystem.
	FCS_CKM.1	Requires generation of asymmetric keys for trusted communications channels.
	FCS_CKM.2	Requires establishment of cryptographic keys for trusted communications channels.
	FCS_COP.1	Ensures proper functioning of cryptographic algorithms used to implement access controls.
	FCS_RBG_EXT.1	Requires that the TOE has access to high-quality entropy for cryptographic purposes.
	FIA_AFL_EXT.1	Requires that the TOE detect failed authentication attempts for Administrator access.
	FIA_UAU.5	Ensures that strong mechanisms are used for Administrator authentication.
	FIA_UIA_EXT.1	Requires that Administrators be successfully authenticated before

		performing management functions.
	FMT_SMO_EXT.1	Requires that the TOE support having separate management and operational networks.
	FTP_ITC_EXT.1	Ensures that trusted communications channels are implemented using good cryptography.
O.PATCHED_SOFTWARE	FPT_TUD_EXT.1	Requires support for product updates.
O.VM_ENTROPY	FCS_ENT_EXT.1	Requires that domains have access to high-quality entropy for cryptographic purposes.
	FCS_RBG_EXT.1	Requires that the TOE has access to high-quality entropy for cryptographic purposes.
O.AUDIT	FAU_GEN.1	Requires reporting of audit events.
	FAU_SAR.1	Requires support for Administrator review of audit records.
	FAU_STG.1	Requires protection of stored audit records.
	FAU_STG_EXT.1	Requires support for protected transmission of audit records off the TOE.
O.CORRECTLY_APPLIED_CONFIGURATION	FMT_MSA_EXT.1	Ensures that data sharing between VMs is turned off by default.
O.RESOURCE_ALLOCATION	FCS_CKM_EXT.4	Requires cryptographic key destruction to ensure residual data in shared storage is unrecoverable.
	FDP_RIP_EXT.1	Ensures that domain data is cleared from memory before memory is re-allocated.
	FDP_RIP_EXT.2	Ensures that domain data is cleared from storage before the storage is re-allocated.

5.2 Security Assurance Requirements

The Security Objectives for the TOE in Section 4 were constructed to address threats identified in Section 3.1. The Security Functional Requirements (SFRs) in Section 5.1 are a formal instantiation of the Security Objectives. The PP identifies the Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

This section lists the set of Security Assurance Requirements (SARs) from Part 3 of the Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 4 that are required in evaluations against this PP. Individual assurance activities to be performed are specified in both Section 5.1 as well as in this section.

After the ST has been approved for evaluation, the Information Technology Security Evaluation Facility (ITSEF) will obtain the TOE, supporting environmental IT, and the administrative/user guides for the TOE. The ITSEF is expected to perform actions mandated by the CEM for the ASE and ALC SARs. The ITSEF also performs the assurance activities contained within Section 5, which are intended to be an interpretation of the other CEM assurance requirements as they apply to the specific technology instantiated in the TOE. The assurance activities that are captured in Section 5 also provide clarification as to what the developer needs to provide to demonstrate the TOE is compliant with the PP.

5.2.1 Class ASE: Security Target Evaluation

As per ASE activities defined in [CEM] plus the TSS assurance activities defined for any SFRs claimed by the TOE.

5.2.2 Class ADV: Development

The information about the TOE is contained in the guidance documentation available to the end user as well as the TOE Summary Specification (TSS) portion of the ST. The TOE developer must concur with the

description of the product that is contained in the TSS as it relates to the functional requirements. The Assurance Activities contained in Section 5.2 should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

ADV_FSP.1 Basic functional specification

Developer action elements:

ADV_FSP.1.1D
The developer shall provide a functional specification.

ADV_FSP.1.2D
The developer shall provide a tracing from the functional specification to the SFRs.

Developer Note: As indicated in the introduction to this section, the functional specification is composed of the information contained in the AGD_OPR and AGD_PRE documentation, coupled with the information provided in the TSS of the ST. The assurance activities in the functional requirements point to evidence that should exist in the documentation and TSS section; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.

Content and presentation elements:

ADV_FSP.1.3C
The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.4C
The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.5C
The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.

ADV_FSP.1.6C
The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

Evaluator action elements:

ADV_FSP.1.7E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.8E
The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

Application Note: There are no specific assurance activities associated with these SARs. The functional specification documentation is provided to support the evaluation activities described in Section 5.2, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other assurance activities being performed; if the evaluator is unable to perform an activity because the there is insufficient interface information, then an adequate functional specification has not been provided.

Evaluation Activities

5.2.3 Class AGD: Guidance Documents

The guidance documents will be provided with the developer’s security target. Guidance must include a description of how the authorized user verifies that the Operational Environment can fulfill its role for the security functionality. The documentation should be in an informal style and readable by an authorized user.

Guidance must be provided for every operational environment that the product supports as claimed in the ST. This guidance includes

- instructions to successfully install the TOE in that environment; and
- instructions to manage the security of the TOE as a product and as a component of the larger operational environment.

Guidance pertaining to particular security functionality is also provided; specific requirements on such

guidance are contained in the assurance activities specified with individual SFRs where applicable.

AGD_OPE.1 Operational User Guidance

Developer action elements:

AGD_OPE.1.1D

The developer shall provide operational user guidance.

Developer Note: Rather than repeat information here, the developer should review the assurance activities for this component to ascertain the specifics of the guidance that the evaluators will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

Content and presentation elements:

AGD_OPE.1.2C

The operational user guidance shall describe what **the authorized user-**accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.3C

The operational user guidance shall describe, for **the authorized user**, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.4C

The operational user guidance shall describe, for **the authorized user**, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

AGD_OPE.1.5C

The operational user guidance shall, for **the authorized user**, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.6C

The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AGD_OPE.1.7C

The operational user guidance shall, for **the authorized user**, describe the security measures to be followed in order to fulfill the security objectives for the operational environment as described in the ST.

AGD_OPE.1.8C

The operational user guidance shall be clear and reasonable.

Evaluator action elements:

AGD_OPE.1.9E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

AGD_OPE.1:
Some of the contents of the operational guidance will be verified by the assurance activities in Section 5.2 and evaluation of the TOE according to the CEM. The following additional information is also required.

The operational guidance shall contain instructions for configuring the password characteristics, number of allowed authentication attempt failures, the lockout period times for inactivity, and the notice and consent warning that is to be provided when authenticating.

The operational guidance shall contain step-by-step instructions suitable for use by an end-user of the VS to configure a new, out-of-the-box system into the configuration evaluated under this Protection Profile.

The documentation shall describe the process for verifying updates to the TOE, either by checking the hash or by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

- *Instructions for querying the current version of the TOE software.*
- *For hashes, a description of where the hash for a given update can be obtained. For digital*

signatures, instructions for obtaining the certificate that will be used by the FCS_COP.1/SIG mechanism to ensure that a signed update has been received from the certificate owner. This may be supplied with the product initially, or may be obtained by some other means.
- *Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).*
- *Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature.*

AGD_PRE.1 Preparative procedures

Developer action elements:

AGD_PRE.1.1D

The developer shall provide the TOE including its preparative procedures.

Developer Note: As with the operational guidance, the developer should look to the assurance activities to determine the required content with respect to preparative procedures.

Content and presentation elements:

AGD_PRE.1.2C

The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_PRE.1.3C

The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

Evaluator action elements:

AGD_PRE.1.4E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_PRE.1.5E

The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

Evaluation Activities

AGD_PRE.1:
As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support TOE functional requirements. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms (that is, combination of hardware and operating system) claimed for the TOE in the ST.

The operational guidance shall contain step-by-step instructions suitable for use by an end-user of the VS to configure a new, out-of-the-box system into the configuration evaluated under this Protection Profile.

5.2.4 Class ALC: Life-Cycle Support

At the assurance level specified for TOEs conformant to this PP, life-cycle support is limited to an examination of the TOE vendor's development and configuration management process in order to provide a baseline level of assurance that the TOE itself is developed in a secure manner and that the developer has a well-defined process in place to deliver updates to mitigate known security flaws. This is a result of the critical role that a developer's practices play in contributing to the overall trustworthiness of a product.

ALC_CMC.1 Labeling of the TOE

Developer action elements:

ALC_CMC.1.1D

The developer shall provide the TOE and a reference for the TOE.

Content and presentation elements:

ALC_CMC.1.2C

The TOE shall be labeled with its unique reference.

Evaluator action elements:

ALC_CMC.1.3E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_CMC.1:
The evaluator shall check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST.

The evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST.

If the vendor maintains a web site advertising the TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

ALC_CMS.1 TOE CM coverage

Developer action elements:

ALC_CMS.1.1D

The developer shall provide a configuration list for the TOE.

Content and presentation elements:

ALC_CMS.1.2C

The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.1.3C

The configuration list shall uniquely identify the configuration items.

Evaluator action elements:

ALC_CMS.1.4E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_CMS.1:
The evaluator shall ensure that the developer has identified (in public-facing development guidance for their platform) one or more development environments appropriate for use in developing applications for the developer's platform. For each of these development environments, the developer shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler and linker flags). The evaluator shall ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator shall ensure that the TSF is uniquely identified (with respect to other products from the TSF vendor), and that documentation provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

ALC_TSU_EXT.1 Timely Security Updates

This component requires the TOE developer, in conjunction with any other necessary parties, to provide information as to how the VS is updated to address security issues in a timely manner. The documentation describes the process of providing updates to the public from the time a security flaw is reported/discovered, to the time an update is released. This description includes the parties involved (e.g., the developer, hardware vendors) and the steps that are performed (e.g., developer testing), including worst case time periods, before an update is made available to the public.

Developer action elements:

ALC_TSU_EXT.1.1D

The developer shall provide a description in the TSS of how timely security updates are made to the TOE.

Content and presentation elements:

ALC_TSU_EXT.1.2C

The description shall include the process for creating and deploying security updates for the TOE software/firmware.

ALC_TSU_EXT.1.3C

The description shall express the time window as the length of time, in days, between public disclosure of a vulnerability and the public availability of security updates to the TOE.

Application Note: The total length of time may be presented as a summation of the periods of time that each party (e.g., TOE developer, hardware vendor) on the critical path consumes. The time period until public availability per deployment mechanism may differ; each is described.

ALC_TSU_EXT.1.4C

The description shall include the mechanisms publicly available for reporting security issues pertaining to the TOE.

Application Note: The reporting mechanism could include web sites, email addresses, and a means to protect the sensitive nature of the report (e.g., public keys that could be used to encrypt the details of a proof-of-concept exploit).

Evaluator action elements:

ALC_TSU_EXT.1.5E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

5.2.5 Class ATE: Tests

Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through ATE_IND family, while the latter is through the AVA_VAN family. At the assurance level specified in this PP, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirements.

ATE_IND.1 Independent Testing - Conformance

Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operation) documentation provided. The focus of the testing is to confirm that the requirements specified in Section 5.1 are being met, although some additional testing is specified for SARs in Section 5.2. The Assurance Activities identify the additional testing activities associated with these components. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this PP.

Developer action elements:

ATE_IND.1.1D

The developer shall provide the TOE for testing.

Content and presentation elements:

ATE_IND.1.2C

The TOE shall be suitable for testing.

Evaluator action elements:

ATE_IND.1.3E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.1.4E

The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

Evaluation Activities

ATE_IND.1:
The evaluator shall prepare a test plan and report documenting the testing aspects of the

system. While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluators must document in the test plan that each applicable testing requirement in the ST is covered.

The Test Plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluators are expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) is provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform. This also includes the configuration of cryptographic engines to be used. The cryptographic algorithms implemented by these engines are those specified by this PP and used by the cryptographic protocols being evaluated (IPsec, TLS/HTTPS, SSH).

The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results. The test report (which could just be an annotated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a "fail" and "pass" result (and the supporting details), and not just the "pass" result.

5.2.6 Class AVA: Vulnerability Assessment

For the first generation of this Protection Profile, the evaluation lab is expected to survey open sources to learn what vulnerabilities have been discovered in these types of products. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. Until penetration tools are created and uniformly distributed to the evaluation labs, evaluators will not be expected to test for these vulnerabilities in the TOE. The labs will be expected to comment on the likelihood of these vulnerabilities given the documentation provided by the vendor. This information will be used in the development of penetration testing tools and for the development of future PPs.

AVA_VAN.1 Vulnerability survey

Developer action elements:

AVA_VAN.1.1D
The developer shall provide the TOE for testing.

Content and presentation elements:

AVA_VAN.1.2C
The TOE shall be suitable for testing.

Evaluator action elements:

AVA_VAN.1.3E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VAN.1.4E
The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_VAN.1.5E
The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

Evaluation Activities

AVA_VAN.1:
As with ATE_IND the evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to determine the vulnerabilities that have been found in virtualization in general, as well as those that pertain to the particular TOE. The evaluator documents the sources consulted and the

vulnerabilities found in the report. For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. For example, if the vulnerability can be detected by pressing a key combination on boot-up, a test would be suitable at the assurance level of this PP. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

Appendix A - Optional Requirements

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE) are contained in the body of this PP. This appendix contains three other types of optional requirements that may be included in the ST, but are not required in order to conform to this PP. However, applied modules, packages and/or use cases may refine specific requirements as mandatory.

The first type (A.1 Strictly Optional Requirements) are strictly optional requirements that are independent of the TOE implementing any function. If the TOE fulfills any of these requirements or supports a certain functionality, the vendor is encouraged to included the SFRs in the ST, but are not required in order to conform to this PP.

The second type ( ) are objective requirements that describe security functionality not yet widely available in commercial technology. The requirements are not currently mandated in the body of this PP, but will be included in the baseline requirements in future versions of this PP. Adoption by vendors is encouraged and expected as soon as possible.

The third type ( ) are dependent on the TOE implementing a particular function. If the TOE fulfills any of these requirements, the vendor must either add the related SFR or disable the functionality for the evaluated configuration.

A.1 Strictly Optional Requirements

A.1.1 Auditable Events for Strictly Optional Requirements