Microsoft Windows

Common Criteria Evaluation

Microsoft Windows Server Microsoft Windows 10 version 1909 (November 2019 Update)

Microsoft Windows Server 2019 version 1809

Hyper-V Security Target

Document Information	
Version Number	0.02
Updated On	January 8, 2021

Version History

Version	Date	Summary of changes
0.01	December 15, 2019	Initial draft
0.02	January 8, 2021	Final draft

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

The example companies, organizations, products, people and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

TABLE OF CONTENTS

LIST OF TABLES

1 Security Target Introduction

This section presents the following information required for a Common Criteria (CC) evaluation:

- Identifies the Security Target (ST) and the Target of Evaluation (TOE)
- Specifies the security target conventions,
- Describes the organization of the security target

1.1 ST Reference

ST Title: Microsoft Windows Server, Microsoft Windows 10 version 1909 (November 2019 Update), Microsoft Windows Server 2019 version 1809 Hyper-V Security Target

ST Version: version 0.02, January 8, 2021

1.2 TOE Reference

TOE Software Identification: The following Windows Operating Systems (OS):

- Microsoft Windows Server Standard edition, version 1909
- Microsoft Windows Server Datacenter edition, version 1909
- Microsoft Windows Server 2019 Standard edition
- Microsoft Windows Server 2019 Datacenter edition
- Microsoft Windows 10 Enterprise edition, version 1909 (64-bit version)

TOE Versions:

- Windows Server: build 10.0.18363 (also known as version 1909)
- Windows Server 2019: build 10.0.17763 (also known as version 1809) with KB4586819
- Windows 10: build 10.0.18363 (also known as version 1909) with KB4586819

The following security updates must be applied for:

- Windows Server and Windows 10: all critical updates as of December 31, 2020
- Windows Server 2019: all critical updates as of December 31, 2020

1.3 TOE Overview

The TOE includes the hypervisor and virtualization subsystem, known as “Hyper-V” in the Windows Server operating system, Windows Server 2019 operating system, the Windows 10 operating system, and those applications necessary to manage, support and configure the operating system. All three operating systems can be delivered preinstalled on a new computer or downloaded from the Microsoft website.

1.3.1 TOE Types

All Windows Server, Windows Server 2019, and Windows 10 and editions, collectively called “Windows”, are preemptive multitasking, multiprocessor, and multi-user operating systems. In general, operating

systems provide users with a convenient interface to manage underlying hardware. They control the allocation and manage computing resources such as processors, memory, and Input/Output (I/O) devices. Windows expands these basic operating system capabilities to controlling the allocation and managing higher level IT resources such as security principals (user or machine accounts), files, printing objects, services, window station, desktops, cryptographic keys, network ports traffic, directory objects, and web content. Multi-user operating systems such as Windows keep track of which user is using which resource, grant resource requests, account for resource usage, and mediate conflicting requests from different programs and users.

1.3.2 TOE Usage

Windows 10 is suited for business desktops, notebook, and convertible computers. It is the workstation product and while it can be used by itself, it is designed to serve as a client within Windows domains.

Built for workloads ranging from the department to the enterprise to the cloud, Windows Server delivers intelligent file and printer sharing; secure connectivity based on Internet technologies, and centralized desktop policy management. It provides the necessary scalable and reliable foundation to support mission-critical solutions for databases, enterprise resource planning software, high-volume, real-time transaction processing, server consolidation, public key infrastructure, virtualization, and additional server roles.

Windows provides an interactive User Interface (UI), as well as a network interface. The TOE includes a set of computer systems that can be connected via their network interfaces and organized into domains and forests. A domain is a logical collection of Windows systems that allows the administration and application of a common security policy and the use of a common accounts database. One or more domains combine to comprise a forest. Windows supports single-domain and multiple-domain (i.e., forest) configurations as well as federation between forests and external authentication services.

Each domain must include at least one designated server known as a Domain Controller (DC) to manage the domain. The TOE allows for multiple DCs that replicate TOE user and machine account as well as group policy management data among themselves to provide for higher availability.

Each Windows system, whether it is a DC server, non-DC server, or workstation, provides a subset of the TSFs. The TSF subset for Windows can consist of the security functions from a single system, for a stand-alone system, or the collection of security functions from an entire network of systems, for a domain configuration.

1.3.3 TOE Security Services

This section summarizes the security services provided by the TOE:

- **Security Audit:** Windows has the ability to collect audit data, review audit logs, protect audit logs from overflow, and restrict access to audit logs. Audit information generated by the system includes the date and time of the event, the user identity that caused the event to be generated, and other event specific data. Authorized administrators can review audit logs and have the ability to search and sort audit records. Authorized Administrators can also configure the audit system to include or exclude potentially auditable events to be audited based on a wide range of characteristics. In the context of this evaluation, the protection profile requirements cover

generating audit events, selecting which events should be audited, and providing secure storage for audit event entries.

- **Cryptographic Support:** Windows provides FIPS 140-2 CAVP validated cryptographic functions that support encryption/decryption, cryptographic signatures, cryptographic hashing, cryptographic key agreement, and random number generation. The TOE additionally provides support for public keys, credential management and certificate validation functions and provides support for the National Security Agency's Suite B cryptographic algorithms. Windows also provides extensive auditing support of cryptographic operations, the ability to replace cryptographic functions and random number generators with alternative implementations,[1] and a key isolation service designed to limit the potential exposure of secret and private keys. In addition to using cryptography for its own security functions, Windows offers access to the cryptographic support functions for user-mode and kernel-mode programs. Public key certificates generated and used by Windows authenticate users and machines as well as protect both user and system data in transit.
    - **TLS:** Windows implements Transport Layer Security to provide protected, authenticated, confidential, and tamper-proof networking between two peer computers
    - **IPsec:** Windows implements IPsec to provide protected, authenticated, confidential, and tamper-proof networking between two peer computers.
- **User Data Protection**: In the context of this evaluation Windows protects computer virtualization capabilities, user data and provides secure networking capabilities.
- **Identification and Authentication** Each Windows user must be identified and authenticated based on administrator-defined policy prior to performing any TSF-mediated functions. An interactive user invokes a trusted path in order to protect his I&A information. Windows maintains databases of accounts including their identities, authentication information, group associations, and privilege and logon rights associations. Windows account policy functions include the ability to define the minimum password length, the number of failed logon attempts, the duration of lockout, and password age. Windows provides the ability to use, store, and protect X.509 certificates that are used for IPsec VPN sessions.
- **Protection of the TOE Security Functions**: Windows provides a number of features to ensure the protection of TOE security functions. Windows protects against unauthorized data disclosure and modification by using a suite of Internet standard protocols including IPsec, IKE, and TLS. Windows ensures process isolation security for all processes through private virtual address spaces, execution context, and security context. The Windows data structures defining process address space, execution context, memory protection, and security context are stored in protected kernel-mode memory. Windows includes self-testing features that ensure the integrity of executable program images and its cryptographic functions. Finally, Windows provides a trusted update mechanism to update Windows binaries itself.
- **TOE Access:** Windows allows an authorized administrator to configure the system to display a logon banner before the logon dialog.

[1] This option is not included in the Windows Common Criteria evaluation.

- **Trusted Path for Communications**: Windows uses the IPsec suite of protocols to provide a Virtual Private Network Connection (VPN) between itself, acting as a VPN client, and a VPN gateway in addition to providing protected communications for HTTPS and TLS.
- **Security Management:** Windows includes several functions to manage security policies. Policy management is controlled through a combination of access control, membership in administrator groups, and privileges.

1.3.4 Non-TOE Hardware, Software, Firmware in the Evaluation

Non-TOE Hardware Identification: The following real and virtualized hardware platforms, corresponding firmware, and components are included in the evaluated configuration:

- Dell PowerEdge R640
- Dell PowerEdge R7425
- Microsoft Surface Book 2

1.4 TOE Description

The TOE includes the hypervisor and virtualization subsystem, known as “Hyper-V” in the Windows Server operating system, Windows Server 2019 operating system, the Windows 10 operating system, and those applications necessary to manage, support and configure the operating system.

Hyper-V enables the computer administrator to specify “partitions” that have separate address spaces where they can load an operating system and applications operating in parallel of the (host) operating system that executes in the root partition of the computer. An operating system executing in a partition has access to virtualized peripheral devices that is controlled by Hyper-V. An operating system may either access devices using the same I/O related instructions as on a real system or it may use a specific interface offered by Hyper-V, called the VMBus, to communicate with Hyper-V for access to peripheral devices. In the first case the operating system can only access the devices virtualized by Hyper-V. When using the VMBus interface, an operating system in a guest partition must have “enlightenments” that establish the VMBus communication and then use those “synthetic” devices accessible via VMBus. Note that the “enlightenments” within a guest operating system is part of the TOE, but not part of the TSF.

1.4.1 Evaluated Configurations

The TOE includes five product variants of Windows:

- Microsoft Windows Server Standard edition, version 1909
- Microsoft Windows Server Datacenter edition, version 1909
- Microsoft Windows Server 2019 Standard edition
- Microsoft Windows Server 2019 Datacenter edition
- Microsoft Windows 10 Enterprise edition, version 1909 (64-bit version)

1.4.2 Security Environment and TOE Boundary

The TOE includes both physical and logical boundaries. Its operational environment is a networked environment.

1.4.2.1 Logical Boundaries

Conceptually the Target of Evaluation can be thought of as a collection of the following security services which the security target describes with increasing detail:

- Security Audit
- Cryptographic Support
- User Data Protection
- Identification and Authentication
- Security Management
- Protection of the TOE Security Functions
- Access to the TOE
- Trusted Path and Channels

These services are primarily provided by Windows components:

- The **Boot Manager**, which is invoked by the computer's bootstrapping code.
- The **Windows Loader** which loads the operating system into the computer's memory.
- **Windows OS Resume** which reloads an image of the executing operating system from a hibernation file as part of resuming from a hibernated state.
- The **Windows Kernel** which contains device drivers for the Windows NT File System, full volume encryption, the crash dump filter, and the kernel-mode cryptographic library.
- The **IPv4 / IPv6 network stack** in the kernel.
- The **IPsec** module in user-mode.
- The **IKE and AuthIP Keying Modules** service which hosts the IKE and Authenticated Internet Protocol (AuthIP) keying modules. These keying modules are used for authentication and key exchange in Internet Protocol security (IPsec).
- The **Remote Access Service** device driver in the kernel, which is used primarily for ad hoc or user-defined VPN connections; known as the "RAS IPsec VPN" or "RAS VPN".
- The **IPsec Policy Agent** service which enforces IPsec policies.
- The **Key Isolation Service** which protects secret and private keys.
- The **Local Security Authority Subsystem** which identifies and authenticates users prior to log on and generates events for the security audit log.
- FIPS-Approved **cryptographic algorithms** to protect user and system data.
- **Local and remote administrative interfaces** for security management.
- **Windows Explorer** which can be used to manage the OS and check the integrity of Windows files and updates.
- The **Windows Trusted Installer** which installs updates to the Windows operating system.

- The **Hypervisor** which regulates physical access to the computer's processors and memory.
- The **Virtual Machine Manager** which is the service in the root partition that manages virtual machines in child partitions.

- The **Hyper-V Data Exchange Service** which provides a mechanism to exchange data between the virtual machine and the operating system running on the physical computer.
- The **Hyper-V Guest Service Interface** which provides an interface for the Hyper-V host to interact with specific services running inside the virtual machine.
- The **Hyper-V Guest Shutdown Service** which provides a mechanism to shut down the operating system of this virtual machine from the management interfaces on the physical computer.
- The **Hyper-V Heartbeat Service** which monitors the state of this virtual machine by reporting a heartbeat at regular intervals.
- The **Hyper-V Remote Desktop Virtualization Service** provides a platform for communication between the virtual machine and the operating system running on the physical computer.
- The **Hyper-V Time Synchronization Service** synchronizes the system time of this virtual machine with the system time of the physical computer.
- The **Hyper-V Volume Shadow Copy Requestor** coordinates the communications that are required to use Volume Shadow Copy Service to back up applications and data on this virtual machine from the operating system on the physical computer.

1.4.2.2 Physical Boundaries

The TOE executes on processors from Intel (x64) or AMD (x64) along with peripherals for input/output (keyboard, mouse, display, and network).

The TOE was tested on the following physical and virtual computer platforms:

- Dell PowerEdge R640
- Dell PowerEdge R7425
- Microsoft Surface Book 2

The Assurance Activity Report describes the relationship between the different hardware platforms and the operating systems examined during the evaluation.

The TOE does not include any hardware or network infrastructure components between the computers that comprise the distributed TOE. The security target assumes that any network connections, equipment, peripherals and cables are appropriately protected in the TOE security environment.

The Windows operating system must be pre-installed on a computer by an OEM, installed by the end-user, by an organization's IT administrator, or updated from a previous Windows 10 version downloaded from Windows Update. Consumers can download Windows 10 from https://www.microsoft.com/en-us/software-download/windows10 and IT professionals can obtain a copy of Windows Server from https://www.microsoft.com/Licensing/servicecenter/default.aspx. The obtained file is in .iso format. Enterprises typically obtain Windows using volume licensing programs and subscriptions such as these for Windows 10.

TOE Guidance Identification: The following administrator, user, and configuration guides were evaluated as part of the TOE and delivered in .docx format:

- *Windows Server, Windows Server 2019, and Microsoft Windows 10 Hyper-V Common Criteria Operational and Administrative Guidance (version 1909)* along with all the documents referenced therein.

The administrator and user must follow the instructions in the *Windows Server, Windows Server 2019, and Microsoft Windows 10 Hyper-V Common Criteria Operational and Administrative Guidance (version 1909)* to configure and remain in the evaluated configuration, which is deemed the secure state.

1.5 Conventions, Terminology, Acronyms

This section specifies the formatting information used in the security target.

1.5.1 Conventions

The following conventions have been applied in this document:

- Security Functional Requirements (SFRs): Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
  - Iteration: allows a component to be used more than once with varying operations.
  - Assignment: allows the specification of an identified parameter.
  - Selection: allows the specification of one or more elements from a list.
  - Refinement: allows the addition of details.

  The conventions for the assignment, selection, refinement, and iteration operations are described in Section 5.

- Other sections of the security target use a bold font to highlight text of special interest, such as captions.

1.5.2 Terminology

The following terminology is used in the security target:

Term	Definition
Access	Interaction between an entity and an object that results in the flow or modification of data.
Access control	Security service that controls the use of resources[2] and the disclosure and modification of data[3].
Accountability	Tracing each activity in an IT system to the entity responsible for the activity.
Active Directory	Active Directory manages enterprise identities, credentials, information protection, system and application settings through AD Domain Services,

[2] Hardware and software
[3] Stored or communicated

	Federation Services, Certificate Services and Lightweight Directory Services.
Administrator	An authorized user who has been specifically granted the authority to manage some portion or the entire TOE and thus whose actions may affect the TOE Security Policy (TSP). Administrators may possess special privileges that provide capabilities to override portions of the TSP.
Assurance	A measure of confidence that the security features of an IT system are sufficient to enforce the IT system's security policy.
Attack	An intentional act attempting to violate the security policy of an IT system.
Authentication	A security measure that verifies a claimed identity.
Authentication data	The information used to verify a claimed identity.
Authorization	Permission, granted by an entity authorized to do so, to perform functions and access data.
Authorized user	An authenticated user who may, in accordance with the TOE Security Policy, perform an operation.
Availability	Timely[4], reliable access to IT resources.
Boot Configuration Database	Boot Configuration Data (BCD) is a firmware-independent database for boot-time configuration data for the computer.
Compromise	Violation of a security policy.
Confidentiality	A security policy pertaining to disclosure of data.
Critical cryptographic security parameters	Security-related information appearing in plaintext or otherwise unprotected form and whose disclosure or modification can compromise the security of a cryptographic module or the security of the information protected by the module.
Cryptographic boundary	An explicitly defined contiguous perimeter that establishes the physical bounds (for hardware) or logical bounds (for software) of a cryptographic module.
Cryptographic key (key)	A parameter used in conjunction with a cryptographic algorithm that determines: • the transformation of plaintext data into ciphertext data • the transformation of ciphertext data into plaintext data • a digital signature computed from data • the verification of a digital signature computed from data • a data authentication code computed from data
Cryptographic module	The set of hardware, software, and/or firmware that implements approved security functions, including cryptographic algorithms and key generation, which is contained within the cryptographic boundary.
Defense-in-depth	A security design strategy whereby layers of protection are utilized to establish an adequate security posture for an IT system.
Discretionary Access Control (DAC)	A means of restricting access to objects based on the identity of subjects and groups to which the objects belong. The controls are discretionary meaning that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject.
Edition	A distinct variation of a Windows OS version. Examples of editions are Windows 10 Pro and Windows 10 Enterprise.

[4] According to a defined metric

Enclave	A collection of entities under the control of a single authority and having a homogeneous security policy. They may be logical or based on physical location and proximity.
Entity	A subject, object, user or external IT device.
General-Purpose Operating System	A general-purpose operating system is designed to meet a variety of goals, including protection between users and applications, fast response time for interactive applications, high throughput for server applications, and high overall resource utilization.
Identity	A means of uniquely identifying an authorized user of the TOE.
Integrated Windows authentication	An authentication protocol formerly known as NTLM or Windows NT Challenge/Response.
Named object	• An object that exhibits all of the following characteristics: • The object may be used to transfer information between subjects of differing user identities within the TOE Security Function (TSF). • Subjects in the TOE must be able to request a specific instance of the object. • The name used to refer to a specific instance of the object must exist in a context that potentially allows subjects with different user identities to request the same instance of the object.
Object	An entity under the control of the TOE that contains or receives information and upon which subjects perform operations.
Operating environment	The total environment in which a TOE operates. It includes the physical facility and any physical, procedural, administrative and personnel controls.
Persistent storage	All types of data storage media that maintain data across system boots (e.g., hard disk, removable media).
Public object	An object for which the TSF unconditionally permits all entities “read” access under the Discretionary Access Control SFP. Only the TSF or authorized administrators may create, delete, or modify the public objects.
Resource	A fundamental element in an IT system (e.g., processing time, disk space, and memory) that may be used to create the abstractions of subjects and objects.
SChannel	A security package (SSP) that provides network authentication between clients and servers.
Secure State	Condition in which all TOE security policies are enforced.
Security attributes	TSF data associated with subjects, objects and users that is used for the enforcement of the TSP.
Security context	The security attributes or rules that are currently in effect. For SSPI, a security context is an opaque data structure that contains security data relevant to a connection, such as a session key or an indication of the duration of the session.
Security package	The software implementation of a security protocol. Security packages are contained in security support provider libraries or security support provider/authentication package libraries.
Security principal	An entity recognized by the security system. Principals can include human users as well as autonomous processes.

Security Support Provider (SSP)	A dynamic-link library that implements the SSPI by making one or more security packages available to applications. Each security package provides mappings between an application's SSPI function calls and an actual security model's function. Security packages support security protocols such as Kerberos authentication and Integrated Windows Authentication.
Security Support Provider Interface (SSPI)	A common interface between transport-level applications. SSPI allows a transport application to call one of several security providers to obtain an authenticated connection. These calls do not require extensive knowledge of the security protocol's details.
Security Target (ST)	A set of security requirements and specifications to be used as the basis for evaluation of an identified TOE.
Subject	An active entity within the TOE Scope of Control (TSC) that causes operations to be performed. Subjects can come in two forms: trusted and untrusted. Trusted subjects are exempt from part or all of the TOE security policies. Untrusted subjects are bound by all TOE security policies.
Target of Evaluation (TOE)	An IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation.
Threat	Capabilities, intentions and attack methods of adversaries, or any circumstance or event, with the potential to violate the TOE security policy.
Unauthorized individual	A type of threat agent in which individuals who have not been granted access to the TOE attempt to gain access to information or functions provided by the TOE.
Unauthorized user	A type of threat agent in which individuals who are registered and have been explicitly granted access to the TOE may attempt to access information or functions that they are not permitted to access.
Universal Unique Identifier (UUID)	UUID is an identifier that is unique across both space and time, with respect to the space of all UUIDs. A UUID can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects across a network.
User	Any person who interacts with the TOE.
User Principal Name (UPN)	An identifier used by Microsoft Active Directory that provides a user name and the Internet domain with which that username is associated in an e-mail address format. The format is [*AD username*]@[associated *domain*]; an example would be *john.smith@microsoft.com.*
Uniform Resource Locator (URL)	The address that is used to locate a Web site. URLs are text strings that must conform to the guidelines in RFC 2396.
Version	A Version refers to a release level of the Windows operating system. Windows 7 and Windows 8 are different versions.
(VMX) machine instructions	Virtual Machine instructions that support virtualization of processor hardware for the Intel 64 and IA-32 computer architectures
Vulnerability	A weakness that can be exploited to violate the TOE security policy.

1.5.3 Acronyms

The acronyms used in this security target are specified **in Appendix A: List of Abbreviations**.

1.6 ST Overview and Organization

This security target contains the following additional sections:

- CC Conformance Claims (Section 2): Formal conformance claims which are examined during the evaluation.
- Security Problem Definition (Section 3): Describes the threats, organizational security policies and assumptions that pertain to the TOE.
- Security Objectives (Section 4): Identifies the security objectives that are satisfied by the TOE and the TOE operational environment.
- Security Requirements (Section 5): Presents the security functional and assurance requirements met by the TOE.
- TOE Summary Specification (TSS) (Section 6): Describes the security functions provided by the TOE to satisfy the security requirements and objectives.
- Protection Profile Conformance Claim (Section 7): Presents the rationale concerning compliance of the ST with the ***Protection Profile for Virtualization***.
- Rationale for Modifications to the Security Requirements (Section 8): Presents the rationale for the security objectives, requirements, and TOE Summary Specification as to their consistency, completeness and suitability.

2 CC Conformance Claims

This ST and the Windows 10 editions (TOEs) are consistent with the following specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional requirements, Version 3.1, Revision 4, September 2012, extended (Part 2 extended)
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance requirements Version 3.1, Revision 4 September 2012, (Part 3 extended)
- Protection Profile for Virtualization, version 1.0, November 17, 2016 (Virtualization PP)
- Protection Profile for Virtualization: Extended Package Server Virtualization, version 1.0, November 17, 2016 (“Server Virtualization EP”)

The security functional requirements and assurance activities have been modified with the following NIAP Technical Decisions:

- NIAP Technical Decision 526 for FIA_X509_EXT.1
- NIAP Technical Decision 443 for FPT_VDP_EXT.1
- NIAP Technical Decision 433 for FIA_X509_EXT.1 is superseded by TD #526
- NIAP Technical Decision 432 for FIA_AFL_EXT.1
- NIAP Technical Decision 431 for FAU_GEN.1, FCS_IPSEC_EXT.1.15, FCS_TLSC_EXT.2, FCS_TLSC_EXT.2, FCS_TLSS_EXT.1, FCS_TLSS_EXT.2
- NIAP Technical Decision 363 for FTA_TAB.1
- NIAP Technical Decision 360 for FMT_MOF_EXT.1.2, FIA_UAU.5.1
- NIAP Technical Decision 264 for FPT_RDM_EXT.1
- NIAP Technical Decision 250 for FPT_HCL_EXT.1
- NIAP Technical Decision 249 for FTP_ITC_EXT.1.1
- NIAP Technical Decision 247 for FPT_VDP_EXT.1 is superseded by TD #443
- NIAP Technical Decision 230 for ALC_CMC.1 and ALC_CMS.1
- NIAP Technical Decision 206 for FPT_DVD_EXT.1
- NIAP Technical Decision 139 for FDP_RIP_EXT.2

Evaluation Assurance: As specified in section 5.2.1 and specific Assurance Activities associated with the security functional requirements from section 5.2.2.

CC Identification: CC for Information Technology (IT) Security Evaluation, Version 3.1, Revision 5, April 2017.

3 Security Problem Definition

The security problem definition consists of the threats to security, organizational security policies, and usage assumptions as they relate to Windows. The assumptions, threats, and policies are copied from the Protection Profile for Virtualization, version 1.0, November 17, 2016 ("Virtualization PP"), and the Protection Profile for Virtualization Extended Package Server Virtualization ("Server Virtualization EP").

3.1 Threats to Security

Table 1 presents known or presumed threats to protected resources that are addressed by Windows based on conformance to the Protection Profile for Virtualization.

Table 1 Virtualization PP Threats Addressed by Windows

Threat	Description
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 Virtualization System 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 Virtualization System 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, critical functions are by necessity performed by software not produced by the virtualization vendor. Such software may include Host Operating Systems and physical device drivers. Vulnerabilities in this software can be exploited by an adversary and result in VMM compromise. Where possible, the VS should mitigate the results of potential vulnerabilities or malicious content in third-party code.
T.VMM_COMPROMISE	The Virtualization System 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. This must be prevented to avoid compromising the Virtualization System.
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 Virtualization System 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 Virtualization System, 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.
T.UNPATCHED_SOFTWARE	Vulnerabilities in outdated or unpatched software can be exploited by adversaries to compromise the Virtualization System or platform.
T.MISCONFIGURATION	The Virtualization System 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.

The Server Virtualization EP does not define any additional threats to the virtualization system.

3.2 Organizational Security Policies

An organizational security policy is a set of rules or procedures imposed by an organization upon its operations to protect its sensitive data and IT assets. **Table 2** describes organizational security policies which are necessary for conformance to the protection profile.

Table 2 Organizational Security Policies

Security Policy	Description
[None]	There are no Organizational Security Policies for the protection profile or extended package.

3.3 Secure Usage Assumptions

Table 3 describes the core security aspects of the environment in which Windows is intended to be used. It includes information about the physical, personnel, procedural, and connectivity aspects of the environment.

The following specific conditions are assumed to exist in an environment where the TOE is employed in order to conform to the protection profile:

Table 3 Virtualization PP Secure Usage Assumptions

Assumption	Description
A.PLATFORM_INTEGRITY	The platform has not been compromised prior to installation of the Virtualization System.
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	If the TOE has covert storage or timing channels, then for all VMs executing on that TOE, it is assumed that relative to the IT assets to which they have access, those VMs will have assurance sufficient to outweigh the risk that they will violate the security policy of the TOE by using those covert channels.
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.

The Server Virtualization EP does not make any additional usage assumptions.

4 Security Objectives

This section defines the security objectives for Windows and its supporting environment. Security objectives, categorized as either TOE security objectives or objectives by the supporting environment, reflect the stated intent to counter identified threats, comply with any organizational security policies identified, or address identified assumptions. All of the identified threats, organizational policies, and assumptions are addressed under one of the categories below.

4.1 TOE Security Objectives

Table 4 describes the security objectives for Windows which are needed to comply with the protection profile.

Table 4 Virtualization PP Security Objectives for the TOE

Security Objective	Source
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 the use case, a VM may require 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 Virtualization System—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 Virtualization System. 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 virtualization system, 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 Virtualization System 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	The purpose of audit is to capture and protect data about what happens on a system so that it can later be examined to determine what has happened in the past.
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 VMs, 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.

The Server Virtualization EP does not define any additional security objectives for the virtualization system.

4.2 Security Objectives for the Operational Environment

The TOE is assumed to be complete and self-contained and, as such, is not dependent upon any other products to perform properly. However, certain objectives with respect to the general operating environment must be met. **Table 5** describes the security objectives for the operational environment as specified in the protection profile.

Table 5 Virtualization PP Security Objectives for the Operational Environment

Environment Objective	Description
OE.CONFIG	TOE administrators will configure the Virtualization System 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.

The Server Virtualization EP does not define any additional security objectives for the operational environment.

5 Security Requirements

The section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) for the TOE. The requirements in this section have been drawn from the Protection Profile for Virtualization, Version 1.0, November 17, 2016 (Virtualization PP), the Protection Profile for Virtualization Extended Package Server Virtualization, version 1.0, November 17, 2016 (Server Virtualization EP), the Common Criteria, or are defined in the following section.

Conventions:

Where requirements are drawn from the protection profile, the requirements are copied verbatim, except for some changes to required identifiers to match the iteration convention of this document, from that protection profile and only operations performed in this security target are identified.

The extended requirements, extended component definitions and extended requirement conventions in this security target are drawn from the protection profile; the security target reuses the conventions from the protection profile which include the use of the word “Extended” and the “_EXT” identifier to denote extended functional requirements. The security target assumes that the protection profile correctly defines the extended components and so they are not reproduced in the security target.

Where applicable the following conventions are used to identify operations:

- **Iteration**: Iterated requirements (components and elements) are identified with letter following the base component identifier. For example, iterations of FMT_MOF.1 are identified in a manner similar to FMT_MOF.1(Audit) (for the component) and FCS_COP.1.1(Audit) (for the elements).
- **Assignment**: Assignments are identified in brackets and bold (e.g., [**assigned value**]).
- **Selection**: Selections are identified in brackets, bold, and italics (e.g., [***selected value***]).
    - Assignments within selections are identified using the previous conventions, except that the assigned value would also be italicized and extra brackets would occur (e.g., [***selected value [assigned value]***]).

a) **Refinement**: Refinements are identified using bold text (e.g., **added text**) for additions and strike-through text (e.g., ~~deleted text~~) for deletions.

5.1 TOE Security Functional Requirements

This section specifies the SFRs for the TOE.

Table 6 TOE Security Functional Requirements for Virtualization PP

Requirement Class	Requirement Component
Security Audit (FAU)	Audit Data Generation (FAU_GEN.1)
	Audit Review (FAU_SAR.1)
	Protected Audit Trail Storage (FAU_STG.1)
	Off-Loading of Audit Data (FAU_STG_EXT.1)
Cryptographic Support (FCS)	Cryptographic Key Generation (FCS_CKM.1)
	Cryptographic Key Establishment (FCS_CKM.2)

	Cryptographic Key Destruction (FCS_CKM_EXT.4)
	Cryptographic Operation for AES Data Encryption/Decryption (FCS_COP.1(SYM))
	Cryptographic Operation for Hashing (FCS_COP.1(HASH))
	Cryptographic Operation for Signature Algorithms (FCS_COP.1(SIGN))
	Cryptographic Operation for Keyed Hash Algorithms (FCS_COP.1(HMAC))
	Cryptographic Operation for Random Bit Generation (FCS_RBG_EXT.1)
	Entropy for Virtual Machines (FCS_ENT_EXT.1)
	IPsec Protocol (FCS_IPSEC_EXT.1)
	TLS Client Protocol with Mutual Authentication (FCS_TLSC_EXT.2)
	TLS Server Protocol with Mutual Authentication (FCS_TLSS_EXT.2)
	HTTPS Protocol (FCS_HTTPS_EXT.1)
User Data Protection (FDP)	Hardware-Based Isolation Mechanisms (FDP_HBI_EXT.1)
	Physical Platform Resource Controls (FDP_PPR_EXT.1)
	Residual Information in Memory (FDP_RIP_EXT.1)
	Residual Information on Disk (FDP_RIP_EXT.2)
	VM Separation (FDP_VMS_EXT.1)
	Virtual Networking Components (FDP_VNC_EXT.1)
Identification & Authentication (FIA)	Authentication Failure Handling (FIA_AFL_EXT.1)
	Password Management (FIA_PMG_EXT.1)
	Multiple Authentication Mechanisms (FIA_UAU.5)
	Administrator Identification and Authentication (FIA_UIA_EXT.1)
	X.509 Certificate Validation (FIA_X509_EXT.1)
	X.509 Certificate Authentication (FIA_X509_EXT.2(TLS))
	X.509 Certificate Authentication (FIA_X509_EXT.2(IPSEC))
Security Management (FMT)	Default Data Sharing Configuration (FMT_MSA_EXT.1)
	Separation of Management and Operational Networks (FMT_SMO_EXT.1)
	Non-Existence of Disconnected Virtual Devices (FPT_DVD_EXT.1)
	Execution Environment Mitigations (FPT_EEM_EXT.1)
	Guest VM Integrity (FPT_GVI_EXT.1)
	Hardware Assists (FPT_HAS_EXT.1)
	Hypercall Controls (FPT_HCL_EXT.1)
	Measured Launch of Platform and VMM (FPT_ML_EXT.1)
	Removable Devices and Media (FPT_RDM_EXT.1)
	Trusted Updates to the Virtualization System (FPT_TUD_EXT.1)
	Trusted Update Based on Certificates (FPT_TUD_EXT.2)
	Virtual Device Parameters (FPT_VDP_EXT.1)
	VMM Isolation from VMs (FPT_VIV_EXT.1)
TOE Access (FTA)	TOE Access Banner (FTA_TAB.1)
Trusted Path/Channels (FTP)	Trusted Channel Communications (FTP_ITC_EXT.1)
	Trusted Path (FTP_TRP.1)
	User Interface: I/O Focus (FTP_UIF_EXT.1)
	User Interface: Identification of VM (FTP_UIF_EXT.2)

Table 7 TOE Security Functional Requirements for Server Virtualization EP

Requirement Class	Requirement Component
Security Management (FMT)	Management of Security Functions Behavior (FMT_MOF_EXT.1)

5.1.1 Security Audit (FAU)

5.1.1.1 Audit Data Generation (FAU_GEN.1)

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;
c. Specifically defined auditable events in Table **8** ~~1~~,
d. [***additional information defined in Table 9*** ~~***2***~~***, additional information defined in Table 10*** ~~***3***~~***, additional information defined in Table 11 4, additional information defined in in Table 12*** ~~***5***~~].

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 **8** ~~1~~; and
f. [***additional information defined in Table 9*** ~~***2***~~***, additional information defined in Table 10*** ~~***3***~~***, additional information defined in Table 11 4, additional information defined in in Table 12*** ~~***5***~~].

Table 8 TOE Virtualization PP Mandatory Audit Events[5]

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	None.	None.
FAU_SAR.1	None.	None.
FAU_STG.1	None.	None.
FAU_STG_EXT.1	Failure of audit data capture due to lack of disk space or pre-defined limit. On failure of logging function, capture record of failure and record upon restart of logging function.	None.
FCS_CKM.1	None.	None.
FCS_CKM.2	None.	None.
FCS_CKM_EXT.4	None.	None.
FCS_COP.1(SYM)	None.	None.
FCS_COP.1(HASH)	None.	None.

[5] This protection profile functional requirement was modified as part of NIAP Technical Decision 431.

FCS_COP.1(SIGN)	None.	None.
FCS_COP.1(HMAC)	None.	None.
FCS_RBG_EXT.1	Failure of the randomization process.	No additional information.
FCS_ENT_EXT.1	None.	None.
FDP_HBI_EXT.1	None.	None.
FDP_PPR_EXT.1	Successful and failed VM connections to physical devices where connection is governed by configurable policy. Security policy violations.	VM and physical device identifiers. Identifier for the security policy that was violated
FDP_RIP_EXT.1	None.	None.
FDP_RIP_EXT.2	None.	None.
FDP_VMS_EXT.1	None.	None.
FDP_VNC_EXT.1	Successful and failed attempts to connect VMs to virtual and physical networking components. Security policy violations. Administrator configuration of inter-VM communications channels between VMs.	VM and virtual or physical networking component identifiers. Identifier for the security policy that was violated.
FIA_UAU.5	None.	None.
FIA_UIA_EXT.1	Administrator authentication attempts All use of the identification and authentication mechanism.	Provided user identity, origin of the attempt (e.g., console, remote IP address).
FMT_MSA_EXT.1	None.	None.
FMT_SMO_EXT.1	None.	None.
FPT_DVD_EXT.1	None.	None.
FPT_EEM_EXT.1	None.	None.
FPT_HAS_EXT.1	None.	None.
FPT_HCL_EXT.1	Attempts to access disabled hypercall interfaces. Security policy violations.	Interface for which access was attempted. Identifier for the security policy that was violated.
FPT_RDM_EXT.1[6]	Connection/disconnection of removable media or device to/from a VM. 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. Failure of signature verification.	No additional information.
FPT_VDP_EXT.1	None.	None.

[6] This protection profile functional requirement was modified as part of NIAP Technical Decision 264.

FPT_VIV_EXT.1	None.	None.
FTA_TAB.1	None.	None.
FTP_ITC_EXT.1	Initiation of the trusted channel. Termination of the trusted channel. Failures of the trusted path functions	User ID and remote source (IP Address) if feasible.
FTP_UIF_EXT.1	None.	None.
FTP_UIF_EXT.2	None.	None.

Table 9 TOE Virtualization PP Optional Audit Events

Requirement	Auditable Events	Additional Audit Record Contents
FPT_GVI_EXT.1	Actions taken due to failed integrity check.	None.

Table 10 TOE Virtualization PP Selection-based Audit Events for Authentication[7]

Requirement	Auditable Events	Additional Audit Record Contents
FCS_TLSC_EXT.2	Failure to establish a TLS Session. Establishment/Termination of a TLS session.	Reason for failure. Non-TOE endpoint of connection (IP address).
FCS_TLSS_EXT.2	Failure to establish a TLS Session. Establishment/Termination of a TLS session.	Reason for failure. Non-TOE endpoint of connection (IP address).
FCS_HTTPS_EXT.1	Failure to establish a HTTPS Session. Establishment/Termination of a HTTPS session.	Reason for failure. Non-TOE endpoint of connection (IP address) for both successes and failures.
FIA_UIA_EXT.1	Administrator session start time and end time	None.

Table 11 TOE Virtualization PP Selection-based Audit Events[8]

Requirement	Auditable Events	Additional Audit Record Contents
FCS_IPSEC_EXT.1	Failure to establish an IPsec SA. Establishment/Termination of an IPsec SA.	Reason for failure.

[7] This protection profile functional requirement was modified as part of NIAP Technical Decision 431.
[8] This protection profile functional requirement was modified as part of NIAP Technical Decision 431.

		Non-TOE endpoint of connection (IP address) for both successes and failures.
FIA_PMG_EXT.1	None.	None.
FIA_X509_EXT.1	Failure to validate a certificate.	Reason for failure.
FIA_X509_EXT.2	None.	None.
FPT_TUD_EXT.2	None.	None.
FTP_TRP.1	Initiation of the trusted channel. Termination of the trusted channel. Failure of the trusted channel functions.	User ID and remote source (IP address) if feasible.

Table 12 TOE Virtualization PP Objective Audit Events

Requirement	Auditable Events	Additional Audit Record Contents
FPT_ML_EXT.1 1	Integrity measurements collected	Integrity measurement values

5.1.1.2 Audit Review (FAU_SAR.1)

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.

5.1.1.3 Protected Audit Trail Storage (FAU_STG.1)

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 modifications to the stored audit records in the audit trail.

5.1.1.4 Off-Loading of Audit Data (FAU_STG_EXT.1)

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 [***overwrite previous audit records according to the following rule: [overwrite the oldest stored audit event]***[9]***, [notify the administrator when the security audit log is full]***] when the local storage space for audit data is full.

5.1.2 Cryptographic Support (FCS)

5.1.2.1 Cryptographic Key Generation (FCS_CKM.1)

FCS_CKM.1.1 The TSF shall generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [

[9] The phrase “overwrite the oldest stored audit event” is the same phrase as used in other Windows evaluations and is equivalent to the phrase “overwrite the oldest audit records in a first-in-first-out manner” in the DoD Annex.

- ***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 [P-521] 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***].

5.1.2.2 Cryptographic Key Establishment (FCS_CKM.2)

FCS_CKM.2.1 The TSF shall perform cryptographic key establishment in accordance with a specified cryptographic key establishment method: [

- ***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"***].

5.1.2.3 Cryptographic Key Destruction (FCS_CKM_EXT.4)

FCS_CKM_EXT.4.1 The TSF shall cause disused cryptographic keys in volatile memory to be destroyed or rendered unrecoverable.

FCS_CKM_EXT.4.2 The TSF shall cause disused cryptographic keys in non-volatile storage to be destroyed or rendered unrecoverable.

5.1.2.4 Cryptographic Operation for AES Data Encryption/Decryption (FCS_COP.1(SYM))

Application Note: FCS_COP.1(SYM) corresponds to FCS_COP.1(1) in the Virtualization PP.

FCS_COP.1.1(SYM) The TSF shall perform encryption and decryption in accordance with a specified cryptographic algorithm [

- ***AES Key Wrap (KW) (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 (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 [***128-bit, 256-bit***].

5.1.2.5 Cryptographic Operation for Hashing (FCS_COP.1(HASH))

Application Note: FCS_COP.1(HASH) corresponds to FCS_COP.1(2) in the Virtualization PP.

FCS_COP.1.1(HASH) The TSF shall perform cryptographic hashing in accordance with a specified cryptographic algorithm [***SHA-1, SHA-256, SHA-384, SHA-512***] and message digest sizes [***160, 256, 384, 512 bits***] that meet the following: FIPS PUB 180-4, "Secure Hash Standard".

5.1.2.6 Cryptographic Operation for Signature Algorithms (FCS_COP.1(SIGN))

Application Note: FCS_COP.1(SIGN) corresponds to FCS_COP.1(3) in the Virtualization PP.

FCS_COP.1.1(SIGN) The TSF shall perform cryptographic signature services (generation and verification) in accordance with a specified cryptographic algorithm [

- ***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 [P-521] that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5***

].

5.1.2.7 Cryptographic Operation for Keyed Hash Algorithms (FCS_COP.1(HMAC))

Application Note: FCS_COP.1(HMAC) corresponds to FCS_COP.1(4) in the Virtualization PP.

FCS_COP.1.1(HMAC) The TSF shall perform [keyed-hash message authentication] in accordance with a specified cryptographic algorithm [***HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512***] and cryptographic key sizes [**128 and 256 bits**] and message digest sizes [***160, 256, 384, 512 bits***] that meet the following: FIPS PUB 198-1, "The Keyed-Hash Message Authentication Code", FIPS PUB 180-4, "Secure Hash Standard".

5.1.2.8 Cryptographic Operation for Random Bit Generation (FCS_RBG_EXT.1)

FCS_RBG_EXT.1.1 The TSF shall perform all deterministic random bit generation services in accordance with NIST Special Publication 800-90A using [***CTR_DRBG (AES)***].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [***a software-based noise source, a hardware-based noise source***] with a minimum of [***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.

5.1.2.9 Entropy for Virtual Machines (FCS_ENT_EXT.1)

FCS_ENT_EXT.1.1 The TSF shall provide a mechanism to make available to VMs entropy that meets FCS_RBG_EXT.1 through [***Hypercall interface, virtual device interface***].

FCS_ENT_EXT.1.2 The TSF shall provide independent entropy across multiple VMs.

5.1.2.10 IPsec Protocol (FCS_IPSEC_EXT.1)

FCS_IPSEC_EXT.1.1 The TSF shall implement the IPsec architecture as specified in RFC 4301.

FCS_IPSEC_EXT.1.2 The TSF shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

FCS_IPSEC_EXT.1.3 The TSF shall implement transport mode and [***tunnel mode, transport mode***].

FCS_IPSEC_EXT.1.4 The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms AES-CBC-128, AES-CBC-256 (both specified by RFC 3602) and [***AES-GCM-128 (specified in RFC 4106), AES-GCM-256 (specified in RFC 4106)***] together with a Secure Hash Algorithm (SHA)-based HMAC.

FCS_IPSEC_EXT.1.5 The TSF shall implement the protocol: [

- ***IKEv1, using Main Mode for Phase 1 exchanges, as defined in RFCs 2407, 2408, 2409, RFC 4109, [RFC 4304 for extended sequence numbers], and [RFC 4868 for hash functions];***
- ***IKEv2 as defined in RFC 5996 and [with mandatory support for NAT traversal as specified in RFC 5996, section 2.23)], and [RFC 4868 for hash functions]***

].

FCS_IPSEC_EXT.1.6 The TSF shall ensure the encrypted payload in the [***IKEv1, IKEv2***] protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 3602 and [***no other algorithm***].

FCS_IPSEC_EXT.1.7 The TSF shall ensure that [

- ***IKEv1 Phase 1 SA lifetimes can be configured by an Administrator based on [***
  - ***number of packets/bytes;***
  - ***Length of time, where the time values can be configured within [any time up to 24] hours***

  ];
- ***IKEv2 SA lifetimes can be configured by an Administrator based on [selection:***
  - ***number of packets/bytes;***
  - ***length of time, where the time values can be configured within [any time up to 24] hours***

  ]

].

FCS_IPSEC_EXT.1.8 The TSF shall ensure that [

- ***IKEv1 Phase 2 SA lifetimes can be configured by an Administrator based on [***
  - ***number of packets/bytes;***
  - ***length of time, where the time values can be configured within [integer range up to 8] hours***

  ];
- ***IKEv2 Child SA lifetimes can be configured by an Administrator based on [***
  - ***number of packets/bytes;***
  - ***length of time, where the time values can be configured within [integer range up to 8] hours***

  ]

].

FCS_IPSEC_EXT.1.9 The TSF shall generate the secret value x used in the IKE Diffie-Hellman key exchange ("x" in g^x mod p) using the random bit generator specified in FCS_RBG_EXT.1, and having a length of at least [**224, 256, 384**] bits.

FCS_IPSEC_EXT.1.10 The TSF shall generate nonces used in [***IKEv1, IKEv2***] exchanges of length [

- ***[256 bits]***

]

FCS_IPSEC_EXT.1.11 The TSF shall ensure that all IKE protocols implement DH Groups 14 (2048-bit MODP), and [***19 (256-bit Random ECP), 24 (2048bit MODP with 256-bit POS), 20 (384-bit Random ECP)***].

FCS_IPSEC_EXT.1.12 The TSF shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [***IKEv1 Phase 1, IKEv2 IKE_SA***] connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [***IKEv1 Phase 2, IKEv2 CHILD_SA***] connection.

FCS_IPSEC_EXT.1.13 The TSF shall ensure that all IKE protocols perform peer authentication using a [***RSA, ECDSA***] that use X.509v3 certificates that conform to RFC 4945 and [***Pre-shared Keys***].

FCS_IPSEC_EXT.1.14 The TSF shall support peer identifiers of the following types: [***IP address, Fully Qualified Domain Name (FQDN), user FQDN, Distinguished Name (DN)***] and [***no other reference identifier type***].

FCS_IPSEC_EXT.1.15 The TSF shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.

5.1.2.11 TLS Client Protocol with Mutual Authentication (FCS_TLSC_EXT.2)[10]

FCS_TLSC_EXT.2.1 The TSF shall implement [***TLS 1.2 (RFC 5246), TLS 1.1 (RFC 4346)***] supporting the following ciphersuites:

- [***TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268***
- ***TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268***
- ***TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492***
- ***TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492***
- ***TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246***
- ***TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246***
- ***TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288***
- ***TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***
- ***TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***
- ***TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289***
- ***TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289***].

FCS_TLSC_EXT.2.2 The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125.

FCS_TLSC_EXT.2.3 The TSF shall establish a trusted channel only if the peer certificate is valid.

FCS_TLSC_EXT.2.4 The TSF shall support mutual authentication using X.509v3 certificates.

[10] This protection profile functional requirement was modified as part of NIAP Technical Decision 431.

FCS_TLSC_EXT.2.5 The TSF shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [***secp256r1, secp384r1, secp521r1***] and no other curves.

5.1.2.12 TLS Server Protocol with Mutual Authentication (FCS_TLSS_EXT.2)[11]

FCS_TLSS_EXT.2.1 The TSF shall implement [***TLS 1.2 (RFC 5246), TLS 1.1 (RFC 4346)***] supporting the following ciphersuites:

- [***TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268***
- ***TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268***
- ***TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492***
- ***TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492***
- ***TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246***
- ***TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246***
- ***TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288***
- ***TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289***
- ***TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289***
- ***TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***
- ***TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289***
- ***TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289***
- ***TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289***
- ***TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289***].

FCS_TLSS_EXT.2.2 The TSF shall deny connections from clients requesting SSL 1.0, SSL 2.0, SSL 3.0, TLS 1.0, and [***none***].

FCS_TLSS_EXT.2.3 The TSF shall [***perform RSA key establishment with key size [2048 bits, 3072 bits]; generate EC Diffie-Hellman parameters over NIST curves [secp256r1, secp384r1, secp521r1] and no other curves; generate Diffie-Hellman parameters of size [2048 bits, 3072 bits]***].

FCS_TLSS_EXT.2.4 The TSF shall support mutual authentication of TLS clients using X.509v3 certificates.

FCS_TLSS_EXT.2.5 The TSF shall not establish a trusted channel if the peer certificate is invalid.

FCS_TLSS_EXT.2.6 The TSF shall not establish a trusted channel if the distinguished name (DN) or Subject Alternative Name (SAN) contained in a certificate does not match the expected identifier for the peer.

5.1.2.13 HTTPS Protocol (FCS_HTTPS_EXT.1)

FCS_HTTPS_EXT.1.1 The TSF shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2 The TSF shall implement HTTPS using TLS.

[11] This protection profile functional requirement was modified as part of NIAP Technical Decision 431.

5.1.3 User Data Protection (FDP)

5.1.3.1 Hardware-Based Isolation Mechanisms (FDP_HBI_EXT.1)

FDP_HBI_EXT.1.1 The TSF shall use [***[Hyper-V Virtual Machine Manager]***] to constrain a Guest VM's direct access to the following physical devices: [***[removable media, storage volume]***].

5.1.3.2 Physical Platform Resource Controls (FDP_PPR_EXT.1)

FDP_PPR_EXT.1.1 The TSF shall allow an authorized administrator to control Guest VM access to the following physical platform resources: [**removable media, storage volume, network adapter, processor, memory**].

FDP_PPR_EXT.1.2 The TSF shall explicitly deny all Guest VMs access to the following physical platform resources: [***[all other physical platform resources]***].

FDP_PPR_EXT.1.3 The TSF shall explicitly allow all Guest VMs access to the following physical platform resources: [***[no physical platform resources]***].

5.1.3.3 Residual Information in Memory (FDP_RIP_EXT.1)

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.

5.1.3.4 Residual Information on Disk (FDP_RIP_EXT.2)

FDP_RIP_EXT.2.1 The TSF shall ensure that any previous information content of physical disk storage is cleared prior to allocation to a Guest VM.

5.1.3.5 VM Separation (FDP_VMS_EXT.1)

FDP_VMS_EXT.1.1 The VS shall provide the following mechanisms for transferring data between Guest VMs: [***virtual networking, [cut-and-paste, hypervisor call, inter-partition communication including HV socket and VM Bus, allocating a storage volume to one partition at a time]***].

FDP_VMS_EXT.1.2 The TSF shall allow Administrators to configure these mechanisms to [***enable***] 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.

5.1.3.6 Virtual Networking Components (FDP_VNC_EXT.1)

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.

5.1.4 Identification and Authentication (FIA)

5.1.4.1 Authentication Failure Handling (FIA_AFL_EXT.1)[12]

FIA_AFL_EXT.1.1 The TSF shall detect when [

[12] This protection profile functional requirement was modified as part of NIAP Technical Decision 432.

- ***an administrator configurable positive integer within a [range of 1 - 999]***]

unsuccessful authentication attempts occur related to Administrators attempting to authenticate remotely using a [***password, PIN***].

FIA_AFL_EXT.1.2 When the defined number of unsuccessful authentication attempts have been met, the TSF shall: [***prevent the offending Administrator from successfully establishing remote session using any authentication method that involves a password or PIN until [the account has been re-enabled] 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***].

5.1.4.2 Password Management (FIA_PMG_EXT.1)

FIA_PMG_EXT.1.1 The TSF shall provide the following password management capabilities for administrative passwords:

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

5.1.4.3 Multiple Authentication Mechanisms (FIA_UAU.5)

FIA_UAU.5.1[13] The TSF shall provide the following authentication mechanisms: [

- ***[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,***
- ***[local, directory-based] authentication based on X.509 certificates,***

]

to support Administrator authentication.

FIA_UAU.5.2 The TSF shall authenticate any Administrator's claimed identity according to the [**authentication based on username and password is performed for TOE-originated requests and with credentials stored by the OS for Windows Hello, smart card and virtual smart card**].

5.1.4.4 Administrator Identification and Authentication (FIA_UIA_EXT.1)

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.

5.1.4.5 X.509 Certificate Validation (FIA_X509_EXT.1)[14]

FIA_X509_EXT.1.1 The TSF shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted certificate.

[13] This protection profile functional requirement was modified as part of NIAP Technical Decision 360.
[14] This protection profile functional requirement was modified as part of NIAP Technical Decision 526.

- The TOE shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The TSF shall validate that any CA certificate includes caSigning purpose in the key usage field.
- The TSF shall validate revocation status of the certificate using [***OCSP as specified in RFC6960, a CRL as specified in RFC5759, an OCSP TLS Status Request Extension (OCSP stapling) as specified in RFC 6066***].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
    - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
    - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage **(EKU)** field.
    - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the EKU field.
    - OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field.

5.1.4.6 X.509 Certificate Authentication (FIA_X509_EXT.2(TLS))

FIA_X509_EXT.2.1 (TLS)	The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [***TLS, HTTPS***], and [***[the uses described in FIA_X509_EXT.2(IPSEC)]***].
FIA_X509_EXT.2.2 (TLS)	When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall [***allow the administrator to choose whether to accept the certificate in these cases***].

5.1.4.7 X.509 Certificate Authentication (FIA_X509_EXT.2(IPSEC))

FIA_X509_EXT.2.1(IPSEC)	The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [***IPsec***], and [***code signing for system software updates, code signing for integrity verification***].
FIA_X509_EXT.2.2(IPSEC)	When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall [***not accept the certificate***].

5.1.5 Security Management (FMT)

5.1.5.1 Default Data Sharing Configuration (FMT_MSA_EXT.1)

FMT_MSA_EXT.1.1	The TSF shall by default enforce a policy prohibiting sharing of data between Guest VMs using [***virtual networking***].
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.

5.1.5.2 Separation of Management and Operational Networks (FMT_SMO_EXT.1)

FMT_SMO_EXT.1.1 The TSF shall support the configuration of separate management and operational networks through [***physical means, logical means, trusted channel***].

5.1.5.3 Management of Security Functions Behavior (FMT_MOF_EXT.1)

FMT_MOF_EXT.1.1 The TSF shall be capable of supporting [***local, remote***] administration.

FMT_MOF_EXT.1.2 The TSF shall be capable of performing the following management functions, controlled by an Administrator or User as shown in **Table 13** ~~1~~, based on the following key:

X = Mandatory (TOE must provide that function to that role)
O = Optional (TOE may or may not provide that function to that role)
N = Not Permitted (TOE must not provide that function to that role)
S = Selection-Based (TOE must provide that function to that role if the TOE claims a particular selection-based SFR)

Table 13 TOE Server Virtualization EP Management Functions

Number	Function	Administrator	User	Notes
1.	Ability to update the Virtualization System	X	N	See FPT_TUD_EXT.1
2.	Ability to configure Administrator password policy as defined in FIA_PMG_EXT.1	S	N	Must be selected if ST includes FIA_PMG_EXT.1.
3.	Ability to create, configure and delete VMs	X	O	
4.	Ability to set default initial VM configurations	X	N	
5.	Ability to configure virtual networks including VM	X	O	See FDP_VNC_EXT.1
6.	Ability to configure and manage the audit system and audit data	X	N	
7.	Ability to configure VM access to physical devices	X	O	
8.	Ability to configure inter-VM data sharing	X	O	See FDP_VMS_EXT.1 and FMT_MSA_EXT.1
9.	Ability to enable/disable VM access to Hypercall functions	X	O	See FPT_HCL_EXT.1
10.	Ability to configure removable media policy	X	N	See FPT_RDM_EXT.1
11.	Ability to configure the cryptographic functionality	X	N	
12.	Ability to change default authorization factors	X	N	
13.	Ability to enable/disable screen lock	O	O	

Number	Function	Administrator	User	Notes
14.	Ability to configure screen lock inactivity timeout	O	O	
15.	Ability to configure remote connection inactivity timeout	X	N	
16.	Ability to configure lockout policy for unsuccessful authentication attempts through [***timeouts between attempts, limiting number of attempts during a time period***]	X	N	See FIA_AFL_EXT.1
17. [15]	Ability to configure name/address of directory server to bind with	S	O	Must be selected if "directory-based" is selected anywhere in FIA_UAU.5.1 in the Base Virtualization PP.
18.	Ability to configure name/address of audit/logging server to which to send audit/logging records	X	N	
19.	Ability to configure name/address of network time server	X	O	
20.	Ability to configure banner	X	N	See FTA_TAB.1
21.	Ability to connect/disconnect removable devices to/from a VM	O	O	
22.	Ability to start a VM	O	O	
23.	Ability to stop/halt a VM	O	O	
24.	Ability to checkpoint a VM	O	O	
25.	Ability to suspend a VM	O	O	
26.	Ability to resume a VM	O	O	

5.1.6 Protection of the TSF (FPT)

5.1.6.1 Non-Existence of Disconnected Virtual Devices (FPT_DVD_EXT.1)

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.

5.1.6.2 Execution Environment Mitigations (FPT_EEM_EXT.1)

FPT_EEM_EXT.1.1 The TSF shall take advantage of execution environment-based vulnerability mitigation mechanisms supported by the Platform such as: [

- ***a. Address space randomization,***
- ***b. Memory execution protection (e.g., DEP),***
- ***c. Stack buffer overflow protection,***

[15] This extended package functional requirement was modified as part of NIAP Technical Decision 360.

d. Heap corruption detection,
e. [Control Flow Guard],
].

5.1.6.3 Guest VM Integrity (FPT_GVI_EXT.1)

FPT_GVI_EXT.1.1 The TSF shall verify the integrity of Guest VMs through the following mechanisms: [**Shielded VM**].[16]

5.1.6.4 Hardware Assists (FPT_HAS_EXT.1)

FPT_HAS_EXT.1.1 The VMM shall use [**virtualized (ex. VMX) machine instructions, Second Level Address Translation (SLAT)**] to reduce or eliminate the need for binary translation.

FPT_HAS_EXT.1.2 The VMM shall use [**Second Level Address Translation (SLAT)**] to reduce or eliminate the need for shadow page tables.

5.1.6.5 Hypercall Controls (FPT_HCL_EXT.1)[17]

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: [**no exceptions**].

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.

5.1.6.6 Measured Launch of Platform and VMM (FPT_ML_EXT.1)

FPT_ML_EXT.1.1 The TSF shall support a measured launch of the Virtualization System. Measured components of the Virtualization system shall include the static executable image of the Hypervisor and:
[
a) Static executable images of the Management Subsystem,
b) [Boot Configuration Database]
].

FPT_ML_EXT.1.2 The TSF shall make these measurements available to the Management Subsystem.

5.1.6.7 Removable Devices and Media (FPT_RDM_EXT.1)

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 [**virtual removable media (.ISO image) and physical removable media devices (USB flash drive and DVD/CD drive**] are switched between information domains, then [
a. the Administrator has granted explicit access for the media or device to be connected to the receiving domain
].

[16] Shielded Virtual Machines is a feature of Windows Server 2019 and Windows Server version 1909.
[17] This protection profile functional requirement was modified as part of NIAP Technical Decision 250.

5.1.6.8 Trusted Updates to the Virtualization System (FPT_TUD_EXT.1)

FPT_TUD_EXT.1.1 The TSF shall provide administrators the ability to query the currently executed version of the TOE firmware/software as well as the most recently installed version of the TOE firmware/software.

FPT_TUD_EXT.1.2 The TSF shall provide administrators the ability to manually initiate updates to TOE firmware/software and [***automatic updates***].

FPT_TUD_EXT.1.3 The TSF shall provide means to authenticate firmware/software updates to the TOE using a [***digital signature mechanism***] prior to installing those updates.

5.1.6.9 Trusted Update Based on Certificates (FPT_TUD_EXT.2)

FPT_TUD_EXT.2.1 The TSF shall not install an update if the code signing certificate is deemed invalid.

5.1.6.10 Virtual Device Parameters (FPT_VDP_EXT.1)

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.

5.1.6.11 VMM Isolation from VMs (FPT_VIV_EXT.1)

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.

5.1.7 TOE Access (FTA)

5.1.7.1 TOE Access Banner (FTA_TAB.1)[18]

FTA_TAB.1.1 Before establishing an administrative user session the TSF shall display a Security Administrator-specified advisory notice and consent warning message regarding use of the TOE.

5.1.8 Trusted Path / Channels (FTP)

5.1.8.1 Trusted Channel Communications (FTP_ITC_EXT.1)[19]

FTP_ITC_EXT.1.1 The TSF shall use [

- ***TLS as conforming to [FCS_TLSC_EXT.2, FCS_TLSS_EXT.2],***
- ***TLS/HTTPS as conforming to FCS_HTTPS_EXT.1,***
- ***IPsec as conforming to FCS_IPSEC_EXT.1***

]

to provide a trusted communication channel between itself and:

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

[18] This protection profile functional requirement was modified as part of NIAP Technical Decision 363.
[19] This protection profile functional requirement was modified as part of NIAP Technical Decision 249.

- ***remote administrators (as required by FTP_TRP.1.1 if selected in FMT_MOF_EXT.1.1 in the selected EP),***
- ***separation of management and operational networks (if selected in FMT_SMO_EXT.1)***

]
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.

5.1.8.2 Trusted Path (FTP_TRP.1)

FTP_TRP.1.1	The TSF shall use a trusted channel as specified in FTP_ITC_EXT.1 to provide a trusted communication path between itself and remote administrators that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from modification, disclosure.
FTP_TRP.1.2	The TSF shall permit remote administrators to initiate communication via the trusted path.
FTP_TRP.1.3	The TSF shall require the use of the trusted path for all remote administration actions.

5.1.8.3 User Interface: I/O Focus (FTP_UIF_EXT.1)

FTP_UIF_EXT.1.1	The TSF shall indicate to users which VM, if any, has the current input focus.

5.1.8.4 User Interface: Identification of VM (FTP_UIF_EXT.2)

FTP_UIF_EXT.2.1	The TSF shall support the unique identification of a VM's output display to users.

5.2 TOE Security Assurance Requirements

5.2.1 CC Part 3 Assurance Requirements

The following table is the collection of CC Part 3 assurance requirements from the Protection Profile for Virtualization.

Table 14 TOE Security Assurance Requirements

Requirement Class	Requirement Component
Design (ADV)	Basic Functional Specification (ADV_FSP.1)
Guidance (AGD)	Operational User Guidance (AGD_OPE.1)
	Preparative Procedures (AGD_PRE.1)
Lifecycle (ALC)	Labeling of the TOE (ALC_CMC.1)
	TOE CM Coverage (ALC_CMS.1)
	Timely Security Updates (ALC_TSU_EXT.1)
Testing (ATE)	Independent Testing – Conformance (ATE_IND.1)
Vulnerability Assessment (AVA)	Vulnerability Survey (AVA_VAN.1)

5.2.1.1 Timely Security Updates (ALC_TSU_EXT.1)

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.1.1C	The description shall include the process for creating and deploying security updates for the TOE software/firmware.
ALC_TSU_EXT.1.2C	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.
ALC_TSU_EXT.1.3C	The description shall include the mechanisms publicly available for reporting security issues pertaining to the TOE.

Evaluator action elements:

ALC_TSU_EXT.1.1E	The evaluator shall confirm that the information provided meets all the requirements for content and presentation of evidence.

5.2.2 Virtualization PP Assurance Activities

This section copies the assurance activities from the protection profile in order to ease reading and comparisons between the protection profile and the security target.

5.2.2.1 Security Audit (FAU)

5.2.2.1.1 Audit Data Generation (FAU_GEN.1)

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 is described in the TSS.

The evaluator shall also make a determination of the administrative actions that are relevant in the context of this PP. 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. 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.

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.

5.2.2.1.2 Audit Review (FAU_SAR.1)

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 evaluator shall review the operational guidance for the procedure on how to review the audit records. The assurance activity for this requirement is performed in conjunction with the assurance activity for FAU_GEN.1.

5.2.2.1.3 Protected Audit Trail Storage (FAU_STG.1)

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. 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.

5.2.2.1.4 Off-Loading of Audit Data (FAU_STG_EXT.1)

FAU_STG_EXT.1.1

Protocols used for implementing the trusted channel must be selected in FTP_ITC_EXT.1.

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. Testing of the trusted channel mechanism is to be performed as specified in the assurance activities for FTP_ITC_EXT.1. The evaluator shall also 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.

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.

FAU_STG_EXT.1.2

The evaluator shall examine the TSS to ensure it describes what happens when the local audit data store is full. 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.

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.

5.2.2.2 Cryptographic Support (FCS)

5.2.2.2.1 Cryptographic Key Generation (FCS_CKM.1)

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.

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.

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/or 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.

5.2.2.2.2 Cryptographic Key Establishment (FCS_CKM.2)

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.

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

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 TOE 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 80056B 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.

5.2.2.2.3 Cryptographic Key Destruction (FCS_CKM_EXT.4)

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. 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).

Note: using 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. It is expected that these builds are based on the same source code as are release builds (of course, with instrumentation and debug-specific code added).

5.2.2.2.4 Cryptographic Operation for Encryption / Decryption (FCS_COP.1(SYM))

Application Note: FCS_COP.1(SYM) corresponds to FCS_COP.1(1) in the Virtualization PP.

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. 90 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 AESCBC 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. 92 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 AESCBC 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 128bit 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: 102

Input: PT, IV, Key

for i = 1 to 1000:

if i == 1: 105 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. 111 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: 114 128 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 216 bytes, an associated data length of 216 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. 133 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 XTSAES 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 AESKWP 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 AESKWP 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.

5.2.2.2.5 Cryptographic Operation for Hashing (FCS_COP.1(HASH))

Application Note: FCS_COP.1(HASH) corresponds to FCS_COP.1(2) in the Virtualization PP.

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. 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.

The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented 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 bit-oriented 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 bit-oriented vs. the byte-oriented 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.

5.2.2.2.6 Cryptographic Operation for Signing (FCS_COP.1(SIGN))

Application Note: FCS_COP.1(SIGN) corresponds to FCS_COP.1(3) in the Virtualization 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.

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.

5.2.2.2.7 Cryptographic Operation for Keyed Hash Algorithms (FCS_COP.1(HMAC))

Application Note: FCS_COP.1(HMAC) corresponds to FCS_COP.1(4) in the Virtualization PP.

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.

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.

5.2.2.2.8 Random Bit Generation (FCS_RBG_EXT.1)

Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Annex D, Entropy Documentation and Assessment.

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.

5.2.2.2.9 Entropy for Virtual Machines (FCS_ENT_EXT.1)

The evaluator shall verify that the TSS describes how the TOE provides entropy to Guest VMs, 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 VM. 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.

5.2.2.2.10 IPsec Protocol (FCS_IPSEC_EXT.1)

FCS_IPSEC_EXT.1.1

The evaluator shall examine the TSS and determine that it describes what takes place when a packet is processed by the TOE, e.g., the algorithm used to process the packet. The TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how those rules and actions form the SPD in terms of the BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in RFC 4301.

As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.

Operational Guidance

The evaluator shall examine the operational guidance to verify it instructs the Administrator how to construct entries into the SPD that specify a rule for processing a packet. The description includes all three cases – a rule that ensures packets are encrypted/decrypted, dropped, and flow through the TOE without being encrypted. The evaluator shall determine that the description in the operational guidance is consistent with the description in the TSS, and that the level of detail in the operational guidance is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

Tests

The evaluator uses the operational guidance to configure the TOE to carry out the following tests:

- Test 1: The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the gateway with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule (e.g., a packet that matches the rule and another that does not match the rule). The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behavior: appropriate packets were dropped, allowed to flow without modification, encrypted by the IPsec implementation.
- Test 2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. As with Test 1, the evaluator ensures both positive and negative test cases are constructed. These scenarios shall exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and operational guidance. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets,

and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the operational guidance.

FCS_IPSEC_EXT.1.2

The assurance activity for this element is performed in conjunction with the activities for FCS_IPSEC_EXT.1.1.

Tests

The evaluator uses the operational guidance to configure the TOE to carry out the following tests:

- Test 1: The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches the rule to allow the packet to flow in plaintext and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a "TOE/platform created" final entry that discards packets that do not match any previous entries). The evaluator sends the packet, and observes that the packet was dropped.

FCS_IPSEC_EXT.1.3

The evaluator checks the TSS to ensure it states that the VPN can be established to operate in tunnel mode and/or transport mode (as identified in FCS_IPSEC_EXT.1.3).

Operational Guidance

The evaluator shall confirm that the operational guidance contains instructions on how to configure the connection in each mode selected.

Tests

The evaluator shall perform the following test(s) based on the selections chosen:

- Test 1 (conditional): If tunnel mode is selected, the evaluator uses the operational guidance to configure the TOE/platform to operate in tunnel mode and also configures a VPN peer to operate in tunnel mode. The evaluator configures the TOE/platform and the VPN peer to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the TOE/Platform to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.
- Test 2: The evaluator uses the operational guidance to configure the TOE/platform to operate in transport mode and also configures a VPN peer to operate in transport mode. The evaluator configures the TOE/platform and the VPN peer to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE/platform to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.

FCS_IPSEC_EXT.1.4

The evaluator shall examine the TSS to verify that the algorithms AES-CBC-128 and AES-CBC-256 are implemented. If the ST author has selected either AESGCM-128 or AES-GCM-256 in the requirement, then the evaluator verifies the TSS describes these as well. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1(4) Cryptographic Operations (for keyed-hash message authentication).

Operational Guidance

The evaluator checks the operational guidance to ensure it provides instructions on how to configure the TOE/platform to use the algorithms, and if either AES-GCM-128 or AES-GCM-256 have been selected the guidance instructs how to use these as well.

Tests

The evaluator shall configure the TOE/platform as indicated in the operational guidance configuring the TOE/platform to use each of the supported algorithms, attempt to establish a connection using ESP, and verify that the attempt succeeds.

FCS_IPSEC_EXT.1.5

TSS

The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented. If IKEv1 is claimed, the evaluator shall examine the TSS to ensure that, in the description of the IPsec protocol, it states that aggressive mode is not used for IKEv1 Phase 1 exchanges, and that only main mode is used. It may be that this is a configurable option.

Operational Guidance

The evaluator shall check the operational guidance to ensure it instructs the administrator how to configure the TOE/platform to use IKEv1 and/or IKEv2 (as selected), and uses the guidance to configure the TOE/platform to perform NAT traversal for the following test (if selected). If IKEv1 is claimed and the use of main mode requires configuration of the TOE/platform prior to its operation, the evaluator shall check the operational guidance to ensure that instructions for this configuration are contained within that guidance.

Tests

Tests are performed in conjunction with the other IPsec evaluation activities with the exception of the activities below:

- (conditional): If the TOE claims IKEv1, the evaluator shall configure the TOE/platform as indicated in the operational guidance (if applicable) and attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator should then show that main mode exchanges are supported.
- (conditional): The evaluator shall configure the TOE/platform so that it will perform NAT traversal processing as described in the TSS and RFC 5996, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

FCS_IPSEC_EXT.1.6

The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms AES-CBC-128, AESCBC-256 are specified, and if others are chosen in the selection of the requirement, those are included in the TSS discussion.

Operational Guidance

The evaluator ensures that the operational guidance describes the configuration of the mandated algorithms, as well as any additional algorithms selected in the requirement. The guidance is then used to configure the TOE/platform to perform the following test for each ciphersuite selected.

Tests

The evaluator shall configure the TOE/platform to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation.

FCS_IPSEC_EXT.1.7

Operational Guidance

The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the operational guidance. If time-based limits are supported, the evaluator ensures that the Administrator is able to configure Phase 1 SA values for 24 hours. Currently there are no values mandated for the number of packets or number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

Tests

When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC “A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered.”

Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

- Test 1 (Conditional): The evaluator shall configure a maximum lifetime in terms of the number of packets (or bytes) allowed following the operational guidance. The evaluator shall configure a test peer with a packet/byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, and determine that once the allowed number of packets (or bytes) through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.
- Test 2 (Conditional): The evaluator shall configure a maximum lifetime of 24 hours for the Phase 1 SA following the operational guidance. The evaluator shall configure a test peer with a lifetime

that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, maintain the Phase 1 SA for 24 hours, and determine that once 24 hours has elapsed, a new Phase 1 SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

FCS_IPSEC_EXT.1.8

Operational Guidance

The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the operational guidance. If time-based limits are supported, the evaluator ensures that the Administrator is able to configure Phase 2 SA values for 8 hours. Currently there are no values mandated for the number of packets or number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

Tests

When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC “A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered.”

Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

- Test 1 (Conditional): The evaluator shall configure a maximum lifetime in terms of the number of packets (or bytes) allowed following the operational guidance. The evaluator shall configure a test peer with a packet/byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, and determine that once the allowed number of packets (or bytes) through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.
- Test 2 (Conditional): The evaluator shall configure a maximum lifetime of 8 hours for the Phase 2 SA following the operational guidance. The evaluator shall configure a test peer with a lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, maintain the Phase 1 SA for 8 hours, and determine that once 8 hours has elapsed, a new Phase 2 SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

FCS_IPSEC_EXT.1.9

The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating "x" (as defined in FCS_IPSEC_EXT.1.). The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of "x" meets the stipulations in the requirement.

FCS_IPSEC_EXT.1.10

Tests

- (conditional) If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.
- (conditional) If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

FCS_IPSEC_EXT.1.11

The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

Tests

For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols can be successfully completed using that particular DH group.

FCS_IPSEC_EXT.1.12

The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.

Tests

The evaluator simply follows the guidance to configure the TOE/platform to perform the following tests.

- Test 1: This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.
- Test 2: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.
- Test 3: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.
- Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.

FCS_IPSEC_EXT.1.13

The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. The description shall be consistent with the algorithms as specified in FCS_COP.1(2) Cryptographic Operations (for cryptographic signature).

If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The evaluator shall check that the operational guidance describes how pre-shared keys are to be generated and established. The description in the TSS and the operational guidance shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

Operational Guidance

The evaluator ensures the operational guidance describes how to set up the TOE to use certificates with RSA and/or ECDSA signatures and public keys.

In order to construct the environment and configure the TOE for the following tests, the evaluator will ensure that the operational guidance describes how to configure the TOE to connect to a trusted CA, and ensure a valid certificate for that CA is loaded into the TOE and marked “trusted”.

Tests

For efficiency sake, the testing that is performed may be combined with the testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), and FCS_IPSEC_EXT.1.1. The following tests shall be repeated for each peer authentication selected in the FCS_IPSEC_EXT.1.1 selection above:

- Test 1: The evaluator shall configure the TOE to use a private key and associated certificate signed by a trusted CA and shall establish an IPsec connection with the peer.
- Test 2 [conditional]: The evaluator shall generate a pre-shared key off-TOE and use it, as indicated in the operational guidance, to establish an IPsec connection with the peer.

FCS_IPSEC_EXT.1.14

The assurance activities for this element are performed in conjunction with the assurance activities for the next element

FCS_IPSEC_EXT.1.15

TSS

The evaluator shall ensure that the TSS describes how the TOE compares the peer’s presented identifier to the reference identifier. This description shall include whether the certificate presented identifier is compared to the ID payload presented identifier, which field(s) of the certificate are used as the presented identifier (DN, Common Name, or SAN), and, if multiple fields are supported, the logical order comparison. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer’s presented certificate.

Guidance

The evaluator shall ensure that the operational guidance includes the configuration of the reference identifier(s) for the peer.

Tests

For each supported identifier type (excluding DNs), the evaluator shall repeat the following tests:

Test 1: For each field of the certificate supported for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds.

Test 2: For each field of the certificate support for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to not match the field in the peer's presented certificate and shall verify that the IKE authentication fails.

The following tests are conditional:

Test 3: (conditional) If, according to the TSS, the TOE supports both Common Name and SAN certificate fields and uses the preferred logic outlined in the Application Note, the tests above with the Common Name field shall be performed using peer certificates with no SAN extension. Additionally, the evaluator shall configure the peer's reference identifier on the TOE to not match the SAN in the peer's presented certificate but to match the Common Name in the peer's presented certificate, and verify that the IKE authentication fails.

Test 4: (conditional) If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKE authentication succeeds. To demonstrate a bit-wise comparison of the DN, the evaluator shall change a single bit in the DN (preferably, in an Object Identifier (OID) in the DN) and verify that the IKE authentication fails.

Test 5: (conditional) If the TOE supports both IPv4 and IPv6 and supports IP address identifier types, the evaluator must repeat test 1 and 2 with both IPv4 address identifiers and IPv6 identifiers. Additionally, the evaluator shall verify that the TOE verifies that the IP header matches the identifiers by setting the presented identifiers and the reference identifier with the same IP address that differs from the actual IP address of the peer in the IP headers and verifying that the IKE authentication fails.

Test 6: (conditional) If, according to the TSS, the TOE performs comparisons between the peer's ID payload and the peer's certificate, the evaluator shall repeat the following test for each combination of supported identifier types and supported certificate fields (as above). The evaluator shall configure the peer to present a different ID payload than the field in the peer's presented certificate and verify that the TOE fails to authenticate the IKE peer.

5.2.2.2.11 TLS Client Protocol **with Mutual Authentication** (FCS_TLSC_EXT.2)[20]

FCS_TLSC_EXT.2.1

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The evaluator shall check the TSS to ensure that the cipher suites specified include those listed for this component.

Tests

Test 1: The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.

Test 3: The evaluator shall send a server certificate in the TLS connection that the does not match the server-selected cipher suite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites.) The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.

Test 5: The evaluator shall perform the following modifications to the traffic:

- Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.
- Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE cipher suite) or that the server denies the client's Finished handshake message.
- Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.
- (conditional): If an ECDHE or DHE cipher suite is selected, modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.

[20] This protection profile assurance activity was modified as part of NIAP Technical Decision 431.

- Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.
- Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

FCS_TLSC_EXT.2.2

The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of reference identifiers are supported (e.g., Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the TOE.

Tests

The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection:

Test 1: The evaluator shall present a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. The evaluator shall verify that the connection fails.

Test 2: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.

Test 3: The evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds.

Test 4: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.

Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier:

1. The evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g., foo.*.example.com) and verify that the connection fails.
2. The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g., *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g., foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g., example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g., bar.foo.example.come) and verify that the connection fails.

Test 6: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

Test 7: [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

FCS_TLSC_EXT.2.3

Tests

Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

FCS_TLSC_EXT.2.4

The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

Tests

Test 1: The evaluator shall perform the following modification to the traffic:

- Configure the server to require mutual authentication and then modify a byte in a CA field in the Server's Certificate Request handshake message. The modified CA field shall not be the CA used to sign the client's certificate. The evaluator shall verify the connection is unsuccessful.

FCS_TLSC_EXT.2.5

The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behavior is performed by default or may be configured.

Tests

Test 1: The evaluator shall configure the server to perform an ECDHE key exchange in the TLS connection using a non-supported curve (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

5.2.2.2.12 TLS Server Protocol with Mutual Authentication (FCS_TLSS_EXT.2)

FCS_TLSS_EXT.2.1

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified are identical to those listed for this component.

Operational Guidance

The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

Tests

Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

Test 3: The evaluator shall use a client to send a key exchange message in the TLS connection that the does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after the receiving the key exchange message.

Test 4: The evaluator shall perform the following modifications to the traffic:

- Modify at a byte in the client's nonce in the Client Hello handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.
- (conditional): If an ECDHE or DHE cipher suite is selected, modify the signature block in the Client's Key Exchange handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.[21]
- Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.
- [conditional] After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection. This test is not required for applications with a TLS implementation that does not support session IDs.[22]
- Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

[21] This protection profile assurance activity was modified as part of NIAP Technical Decision 431
[22] This protection profile assurance activity was modified as part of NIAP Technical Decision 431.

FCS_TLSS_EXT.2.2[23]

The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions.

Operational Guidance

The evaluator shall verify that any configuration necessary to meet the requirement are contained in the AGD guidance.

Tests

The evaluator shall send a Client Hello requesting a connection for all mandatory and selected protocol versions in the SFR (e.g. by enumeration of protocol versions in a test client) and verify that the server denies the connection for each attempt.

~~The evaluator shall send a Client Hello requesting a connection with version SSL 1.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 2.0, SSL 3.0, TLS 1.0, and any selected TLS versions.~~

FCS_TLSS_EXT.2.3

The evaluator shall verify that the TSS describes the key agreement parameters of the server key exchange message.

Operational Guidance

The evaluator shall verify that any configuration necessary to meet the requirement is contained in the AGD guidance.

Tests

If the second selection includes any choice other than “no other”, the evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve and, using a packet analyzer, verify that the key agreement parameters in the Key Exchange message are the ones configured. (Determining that the size matches the expected size for the configured curve is sufficient.) The evaluator shall repeat this test for each supported NIST Elliptic Curve and each supported Diffie-Hellman key size.

FCS_TLSS_EXT.2.4

The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

Operational Guidance

The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

Tests

Test 1: The evaluator shall configure the server to send a certificate request to the client and shall attempt a connection without sending a certificate from the client. The evaluator shall verify that the connection is denied.

[23] This protection profile assurance activity was modified as part of NIAP Technical Decision 431.

Test 2: The evaluator shall configure the server to send a certificate request to the client without the supported_signature_algorithm used by the client's certificate. The evaluator shall attempt a connection using the client certificate and verify that the connection is denied.

Test 3: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

Test 4: If the TOE supports sending a non-empty Certificate Authorities list in its Certificate Request message, the evaluator shall configure the client to send a certificate that does not chain to one of the Certificate Authorities (either a Root or Intermediate CA) in the server's Certificate Request message. The evaluator shall verify that the attempted connection is denied. If the TOE doesn't support sending a non-empty Certificate Authorities list in its Certificate Request message, this test shall be omitted.[24]

Test 5: The evaluator shall configure the client to send a certificate with the Client Authentication purpose in the extendedKeyUsage field and verify that the server accepts the attempted connection. The evaluator shall repeat this test without the Client Authentication purpose and shall verify that the server denies the connection. Ideally, the two certificates should be identical except for the Client Authentication purpose.

Test 6: The evaluator shall perform the following modifications to the traffic:

- Configure the server to require mutual authentication and then modify a byte in the client's certificate. The evaluator shall verify that the server rejects the connection.
- Configure the server to require mutual authentication and then modify a byte in the client's Certificate Verify handshake message. The evaluator shall verify that the server rejects the connection.

FCS_TLSS_EXT.2.5

The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

Operational Guidance

The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

Tests

Test 1: The evaluator shall configure the server to send a certificate request to the client and shall attempt a connection without sending a certificate from the client. The evaluator shall verify that the connection is denied.

[24] This protection profile assurance activity was modified as part of NIAP Technical Decision 431.

Test 2: The evaluator shall configure the server to send a certificate request to the client without the supported_signature_algorithm used by the client's certificate. The evaluator shall attempt a connection using the client certificate and verify that the connection is denied.

Test 3: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

Test 4: The evaluator shall configure the client to send a certificate that does not chain to one of the Certificate Authorities (either a Root or Intermediate CA) in the server's Certificate Request message. The evaluator shall verify that the attempted connection is denied.

Test 5: The evaluator shall configure the client to send a certificate with the Client Authentication purpose in the extendedKeyUsage field and verify that the server accepts the attempted connection. The evaluator shall repeat this test without the Client Authentication purpose and shall verify that the server denies the connection. Ideally, the two certificates should be identical except for the Client Authentication purpose.

Test 6: The evaluator shall perform the following modifications to the traffic:

- Configure the server to require mutual authentication and then modify a byte in the client's certificate. The evaluator shall verify that the server rejects the connection.
- Configure the server to require mutual authentication and then modify a byte in the client's Certificate Verify handshake message. The evaluator shall verify that the server rejects the connection.

FCS_TLSS_EXT.2.6

The evaluator shall verify that the TSS describes how the DN or SAN in the certificate is compared to the expected identifier.

Operational Guidance

If the TOE implements mutual authentication such that the DN is not compared automatically to the Domain Name or IP address, username, or email address, then the evaluator shall ensure that the AGD guidance includes configuration of the expected DN or the directory server for the connection.

Tests

The evaluator shall send a client certificate with an identifier that does not match an expected identifier and verify that the server denies the connection.

5.2.2.2.13 HTTPS Protocol (FCS_HTTPS_EXT.1)

The evaluator shall check the TSS to ensure that it is clear on how HTTPS uses TLS to establish an administrative session, focusing on any client authentication required by the TLS protocol vs. security administrator authentication which may be done at a different level of the processing stack. Testing for this activity is done as part of the TLS testing; this may result in additional testing if the TLS tests are done at the TLS protocol level.

5.2.2.3 User Data Protection (FDP)

5.2.2.3.1 Hardware-Based Isolation Mechanisms (FDP_HBI_EXT.1)

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

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.

5.2.2.3.2 Physical Platform Resource Controls (FDP_PPR_EXT.1)

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).

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.

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.

5.2.2.3.3 Residual Information in Memory (FDP_RIP_EXT.1)

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.

5.2.2.3.4 Residual Information on Disk (FDP_RIP_EXT.2)[25]

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.

The evaluator shall perform the following test:

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.

5.2.2.3.5 VM Separation (FDP_VMS_EXT.1)

The evaluator shall examine the TSS to verify that it documents all inter-VM communications mechanisms (as defined above), including how the mechanisms are configured, how they are invoked, and how they are disabled.

The evaluator shall perform the following tests for each documented inter-VM communications channel:

a. Create two 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.
b. Test that communications cannot be passed between the VMs through the channel.

[25] This protection profile assurance activity was modified as part of NIAP Technical Decision 139.

c. As an Administrator, enable inter-VM communications between the VMs on the second VM.

5.2.2.3.6 Virtual Networking Components (FDP_VNC_EXT.1)

The evaluator must ensure that the TSS and Operational Guidance describes how to create virtualized networks and connect VMs to each other and to physical networks.

- 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.

The evaluator must ensure that the ST includes the following statement attesting that virtual network traffic is visible only to VMs configured to be on that virtual network:

"Traffic traversing a virtual network is visible only to Guest VMs that are configured by an Administrator to be members of that virtual network. There are no design or implementation flaws that permit the virtual networking configuration to be bypassed or defeated, or for data to be transferred through undocumented mechanisms. This claim does not apply to covert channels or architectural side-channels."

5.2.2.4 Identification and Authentication (FIA)

5.2.2.4.1 Authentication Failure Handling (FIA_AFL_EXT.1)[26]

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

1. The evaluator will set an Administrator-configurable threshold n for failed attempts, or note the ST-specified assignment.
    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.
    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.
    3. After reaching the limit for unsuccessful authentication attempts the evaluator will proceed as follows:
        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

[26] This protection profile assurance activity was modified as part of NIAP Technical Decision 432.

Administrator's access results in successful access (when using valid credentials for that Administrator).

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.

5.2.2.4.2 Password Management (FIA_PMG_EXT.1)

The evaluator shall examine the operational guidelines to determine that it provides guidance to security administrators in the composition of strong passwords, and that it provides instructions on setting the minimum password length. The evaluator shall also perform the following tests. Note that one or more of these tests may be performed with a single test case.

- Test 1: The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible combinations of passwords, the evaluator shall ensure that all characters, rule characteristics, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

5.2.2.4.3 Multiple Authentication Mechanisms (FIA_UAU.5)

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.

5.2.2.4.4 Administrator Identification and Authentication (FIA_UIA_EXT.1)

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.

5.2.2.4.5 X.509 Certificate Validation (FIA_X509_EXT.1)[27]

The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

The tests described must be performed in conjunction with the other Certificate Services assurance activities, including the use cases in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules.

[27] This protection profile assurance activity was modified as part of NIAP Technical Decision 526.

- Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn:
    - by establishing a certificate path in which one of the issuing certificates is not a CA certificate,
    - by omitting the basicConstraints field in one of the issuing certificates,
    - by setting the basicConstraints field in an issuing certificate to have CA=False,
    - by omitting the CA signing bit of the key usage field in an issuing certificate, and
    - by setting the path length field of a valid CA field to a value strictly less than the certificate path.
- The evaluator shall then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator shall then remove trust in one of the CA certificates, and show that the function fails.
- Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.
- Test 3: The evaluator shall test that the TOE can properly handle revoked certificates – conditional on whether CRL, OCSP, OCSP stapling, or OCSP multi-stapling is selected; if multiple methods are selected, and then a test is performed for each method. The evaluator has to only test one up in the trust chain (future revisions may require to ensure the validation is done up the entire chain). The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator shall then attempt the test with a certificate that will be revoked (for each method chosen in the selection) and verify that the validation function fails.
- Test 4: If any OCSP option is selected, the evaluator shall configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set and verify that validation of the CRL fails.
- Test 5a: (Conditional on support for EC certificates as indicated in FCS_COP.1(**SIGN** ~~3~~)). The evaluator shall establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.
- Test 5b: (Conditional on support for EC certificates as indicated in FCS_COP.1(**SIGN** ~~3~~)). The evaluator shall replace the intermediate certificate in the certificate chain for Test 5a with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 5a, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

5.2.2.4.6 X.509 Certificate Authentication (FIA_X509_EXT.2)

The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

The evaluator shall perform Test 1 for each function listed in FIA_X509_EXT.2.1 that requires the use of certificates:

- Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.
- Test 2: The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the operational guidance to determine that all supported administrator-configurable options behave in their documented manner.

5.2.2.5 Security Management (FMT)

5.2.2.5.1 Default Data Sharing Configuration (FMT_MSA_EXT.1)

This requirement is met if FDP_VMS_EXT.1 is met.

5.2.2.5.2 Separation of Management and Operational Networks (FMT_SMO_EXT.1)

The evaluator shall examine the TSS to verify that it describes how management and operational networks may be separated.

The evaluator shall examine the operational guidance to verify that it details how to configure the VS with separate Management and Operational Networks.

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.

5.2.2.6 Protection of the TSF (FPT)

5.2.2.6.1 Non-Existence of Disconnected Virtual Devices (FPT_DVD_EXT.1)[28]

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.

[28] This protection profile assurance activity was modified as part of NIAP Technical Decision 206.

5.2.2.6.2 Execution Environment Mitigations (FPT_EEM_EXT.1)

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.

5.2.2.6.3 Guest VM Integrity (FPT_GVI_EXT.1)

For each mechanism listed in the assignment, the evaluator shall ensure that the TSS documents the mechanism, including how it verifies VM integrity, which set of Guest VMs it will check (all Guest VMs, only migrated VMs, etc.), when such checks occur (before VM startup, immediately following importation/migration, on demand, etc.), and which actions are taken if a VM fails the integrity check (or which range of actions are possible if the action is configurable).

5.2.2.6.4 Hardware Assists (FPT_HAS_EXT.1)

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.

5.2.2.6.5 Hypercall Controls (FPT_HCL_EXT.1)[29]

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.

The evaluator shall examine the operational guidance to ensure it contains instructions for how to configure interface functions per FPT_HCL_EXT.1.2.

The evaluator shall perform the following tests:

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.
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.

5.2.2.6.6 Measured Launch of Platform and VMM (FPT_ML_EXT.1)

The evaluator shall verify that the TSS or Operational Guidance describes how integrity measurements are performed and made available to the Management Subsystem. The evaluator shall examine the operational guidance to verify that it documents how to access the measurements in the Management Subsystem.

[29] This protection profile assurance activity was modified as part of NIAP Technical Decision 250.

The evaluator shall perform the following tests:

Test 1: The evaluator shall start the VS, login as an Administrator, and verify that the measurements for the specified components are viewable in the Management Subsystem.

5.2.2.6.7 Removable Devices and Media (FPT_RDM_EXT.1)

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.

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.

5.2.2.6.8 Trusted Updates to the Virtualization System (FPT_TUD_EXT.1)

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.

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)

5.2.2.6.9 Trusted Update Based on Certificates (FPT_TUD_EXT.2)

The assurance activity for this requirement is performed in conjunction with the assurance activity for FIA_X509_EXT.1 and FIA_X509_EXT.2.

5.2.2.6.10 Virtual Device Parameters (FPT_VDP_EXT.1)[30], [31]

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 documentation must be sufficient to enable the evaluator to effectively run the tests in FPT_DVD_EXT.1. The evaluator must ensure that there are no obvious or publicly known virtual I/O ports missing from the TSS.

Assurance Activity Note: There is no expectation that evaluators will examine source code to verify the "all" part of the Assurance Activity.

The evaluator ensures that the ST includes the following statement attesting that parameters passed from a Guest VM to virtual device interfaces are thoroughly validated, that all values outside the legal values specified in the TSS are rejected, and that any data passed to the virtual device interfaces is unable to degrade or disrupt the functioning of other VMs, the VMM, or the Platform:

> "Parameters passed from Guest VMs to virtual device interfaces are thoroughly validated and all illegal values (as specified in the TSS) are rejected. Additionally, parameters passed from Guest VMs to virtual device interfaces are not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform. Thorough testing and architectural design reviews have been conducted to ensure the accuracy of these claims, and there are no known design or implementation flaws that bypass or defeat the security of the virtual device interfaces."

~~The evaluator shall examine the TSS to ensure it documents all virtual device interfaces at the virtual I/O port level, to specify port number (absolute or relative to a base), port name, and a description of legal input values. The documentation must be sufficient to enable the evaluator to effectively run the tests in FPT_DVD_EXT.1. The evaluator must ensure that there are no obvious or publicly known virtual I/O ports missing from the TSS.~~

~~The evaluator ensures that the ST includes the following statement attesting that parameters passed from a Guest VM to virtual device interfaces are thoroughly validated, that all values outside the legal values specified in the TSS are rejected, and that any data passed to the virtual device interfaces is unable to degrade or disrupt the functioning of other VMs, the VMM, or the Platform:~~

> ~~"Parameters passed from Guest VMs to virtual device interfaces are thoroughly validated and all illegal values (as specified in the TSS) are rejected. Additionally, parameters passed from Guest~~

[30] This protection profile assurance activity was modified as part of NIAP Technical Decision 247.
[31] This protection profile assurance activity was modified as part of NIAP Technical Decision 443.

~~VMs to virtual device interfaces are not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform. Thorough testing and architectural design reviews have been conducted to ensure the accuracy of these claims, and there are no known design or implementation flaws that bypass or defeat the security of the virtual device interfaces.”~~

5.2.2.6.11 VMM Isolation from VMs (FPT_VIV_EXT.1)

The evaluator ensures that the ST includes the following statement attesting that software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform:

“Software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform. There are no design or implementation flaws that bypass or defeat VM isolation.”

5.2.2.7 TOE Access (FTA)

5.2.2.7.1 TOE Access Banner (FTA_TAB.1)

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.

5.2.2.8 Trusted Path / Channels (FTP)

5.2.2.8.1 Trusted Channel Communications (FTP_ITC_EXT.1)

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.

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.

5.2.2.8.2 Trusted Path (FTP_TRP.1)

The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST. The evaluator shall confirm that the operational guidance contains instructions for establishing the remote administrative sessions for each supported method. The evaluator shall also perform the following tests:

- Test 1: The evaluators shall ensure that communications using each specified (in the operational guidance) remote administration method is tested during the course of the evaluation, setting up the connections as described in the operational guidance and ensuring that communication is successful. Test 2: For each method of remote administration supported, the evaluator shall follow the operational guidance to ensure that there is no available interface that can be used by a remote user to establish remote administrative sessions without invoking the trusted path.
- Test 3: The evaluator shall ensure, for each method of remote administration, the channel data is not sent in plaintext.

- Test 4: The evaluator shall ensure, for each method of remote administration, modification of the channel data is detected by the TOE.

Further assurance activities are associated with the specific protocols.

5.2.2.8.3 User Interface: I/O Focus (FTP_UIF_EXT.1)

1. The evaluator shall ensure that the TSS lists the supported user input devices.
2. The evaluator shall ensure that the operational guidance specifies how the current input focus is indicated to the user.
3. 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.

5.2.2.8.4 User Interface: Identification of VM (FTP_UIF_EXT.2)

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.

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.

5.2.3 Server Virtualization EP Assurance Activities

This section copies the assurance activities from the protection profile in order to ease reading and comparisons between the extended package and the security target.

5.2.3.1 Security Management (FMT)

5.2.3.1.1 Management of Security Functions Behavior (FMT_MOF_EXT.1)

The evaluator shall examine the TSS and Operational Guidance to ensure that it describes which security management functions require Administrator privilege and the actions associated with each management function. The evaluator shall verify that for each management function and role specified in Table 1, the defined role is able to perform all mandatory functions as well as all optional or selection-based functions claimed in the ST.

The evaluator shall examine the Operational Guidance to ensure that it describes how the Administrator and/or User are able to perform each management function that the ST claims the TOE supports.

The evaluator shall verify for each claimed management function that the Operational Guidance is sufficiently detailed to allow the function to be performed and that the function can be performed by the role(s) that are authorized to do so.

The evaluator shall also verify for each claimed management function that if the TOE claims not to provide a particular role with access to the function, then it is not possible to access the TOE as that role and perform that function.

6 TOE Summary Specification (TSS)

This chapter describes the Windows security functions that satisfy the security functional requirements of the protection profile. The TOE also includes additional relevant security functions which are also described in the following sections, as well as a mapping to the security functional requirements satisfied by the TOE.

This section presents the TOE Security Functions (TSFs) and a mapping of security functions to Security Functional Requirements (SFRs). The TOE performs the following security functions:

- Audit
- Cryptographic Support
- User Data Protection
- Identification and Authentication
- Security Management
- Protection of the TSF
- TOE Access
- Trusted Channels

6.1 Audit

The TOE Audit security function performs:

- Audit Collection
- Selective Audit
- Audit Log Overflow Protection
- Audit Log Restricted Access Protection

6.1.1 Audit Collection

The Windows Event Log service creates the security event log, which contains security relevant audit records collected on a system, along with other event logs which are also registered by other audit entry providers. The Local Security Authority (LSA) server collects audit events from all other parts of the TSF and forwards them to the Windows Event Log service which will place the event into the log for the appropriate provider. While there is no size limit for a single audit record, the authorized administrator can specify a limit for the size of each event log. For each audit event, the Windows Event Log service stores the following data in each audit entry:

Table 15 Standard Fields in a Windows Audit Entry

Field in Audit Entry	Description
Date	The date the event occurred.
Time	The time the event occurred.
User	The security identifier (SID) of that represents the user on whose behalf the event occurred that represents the user.
Event ID	A unique number within the audit category that identifies the specific audit event.

Source	The Windows component that generated the audit event.
Outcome	Indicates whether the security audit event recorded is the result of a successful or failed attempt to perform the action.
Category	The type of the event defined by the event source.

The LSA service defines the following categories for audit events in the security log:

- System,
- Logon / Logoff
- Object Access
- Directory Service Access
- Privilege Use
- Detailed Process Tracking
- Policy Change
- Account Management
- Account Logon

Each audit entry may also contain category-specific data that is contained in the body of the entry as described below:

- For the System Category, the audit entry includes information relating to the system such as the time the audit trail was cleared, start or shutdown of the audit function, and startup and shutdown of Windows. Furthermore, the specific cryptographic operation is identified when such operations are audited.
- For the Logon and Account Logon Category, the audit entry includes the reason the attempted logon failed.
- For the Object Access and the Directory Service Access Category, the audit entry includes the object name and the desired access requested.
- For the Privilege Use Category, the audit entry identifies the privilege.
- For the Detailed Process Tracking Category, the audit event includes the process identifier.
- For the Policy Change and Account Management Category, the audit event includes the new values of the policy or account attributes.
- For the Account Logon Category, the audit event includes the logon type that indicates the source of the logon attempt as one of the following types in the audit record:
    - Interactive (local logon)
    - Network (logon from the network)
    - Service (logon as a service)
    - Batch (logon as a batch job)
    - Unlock (for Unlock screen saver)
    - Network_ClearText (for anonymous authentication to IIS)

There are two places within the TSF where security audit events are collected. Inside the kernel, the Security Reference Monitor (SRM), a part of the NT Executive, is responsible for generation of all audit

entries for the object access, privilege use, and detailed process tracking event categories. Windows components can request the SRM to generate an audit record and supply all of the elements in the audit record except for the system time, which the Executive provides. With one exception, audit events for the other event categories are generated by various services that either co-exist in the LSA server or call, with the SeAuditPrivilege privilege, the Authz Report Audit interfaces implemented in the LSA Policy subcomponent. The exception is that the Event Log Service itself records an event record when the security log is cleared and when the security log exceeds the warning level configured by the authorized administrator.

The LSA server maintains an audit policy in its database that determines which categories of events are actually collected. Defining and modifying the audit policy is restricted to the authorized administrator. The authorized administrator can select events to be audited by selecting the category or categories to be audited. An authorized administrator can individually select each category. Those services in the security process determine the current audit policy via direct local function calls. The only other TSF component that uses the audit policy is the SRM in order to record object access, privilege use, and detailed tracking audit. LSA and the SRM share a private local connection port, which is used to pass the audit policy to the SRM. When an authorized administrator changes the audit policy, the LSA updates its database and notifies the SRM. The SRM receives a control flag indicating if auditing is enabled and a data structure indicating that the events in particular categories to audit.

In addition to the system-wide audit policy configuration, it is possible to define a per-user audit policy using auditpol.exe. This allows individual audit categories (of success or failure) to be enabled or disabled on a per user basis.[32] The per-user audit policy refines the system-wide audit policy with a more precise definition of the audit policy for which events will be audited for a specific user.

Within each category, auditing can be performed based on success, failure, or both. For object access events, auditing can be further controlled based on user/group identify and access rights using System Access Control Lists (SACLs). SACLs are associated with objects and indicate whether or not auditing for a specific object, or object attribute, is enabled.

6.1.2 Audit Log Overflow Protection

The TSF protects against the loss of events through a combination of controls associated with audit queuing and event logging. As configured in the TOE, audit data is appended to the audit log until it is full. The TOE protects against lost audit data by allowing the authorized administrator to configure the system to generate an audit event when the security audit log reaches a specified capacity percentage (e.g., 90%). Additionally, the authorized administrator can configure the system not to overwrite events – overwriting the oldest stored audit records if the audit trail is full – and instead will shut down when the security audit log is full. When so configured, after the system shutdowns due to audit overflow, only the authorized administrator can restart the system to log on and manage the security log. When the security log is full, a message is written to the display indicating the audit log has overflowed.

As described above, the TSF collects security audit data in two ways, via the SRM and via the LSA server. Both components maintain audit in-memory event queues. The SRM puts audit records on an internal

[32] Windows will prevent a local administrator from disabling auditing for local administrator accounts. If an administrator can bypass auditing, they can avoid accountability for such actions as exfiltrating files without authorization.

queue to be sent to the LSA server. The LSA maintains a second queue where it holds the audit data from SRM and the other services in the security process. Both audit queues detect when an audit event loss has occurred. The SRM service maintains a high water mark and a low water mark on its audit queue to determine when full. The LSA also maintains marks in its queue to indicate when it is full.

Windows also provides an eventing infrastructure that other system components can use to log events which are not managed by the SRM or the LSA. The maximum size for these administrative and operational event logs can either be limited to the maximum size for the log file (and then prevent generation of new audit events for that particular log) or overwrite the oldest audit event in the log file when the log file reaches its maximum size. The security target selects the second option.

6.1.3 Audit Log Restricted Access Protection

The Windows Event Log service controls and protects the security audit log. Note that the underlying files are configured so that only the TSF can open the files and the Event Log service opens those files exclusively when it starts and keeps them open while it is running. To view the contents of the security audit log, the user must be an authorized administrator. The security audit log is a system resource, created during system startup. No interfaces exist to create, destroy, or modify an event within the event log. The LSA subsystem is the only service registered to enter events into the security log. The TOE only offers user interfaces to read and clear the security event log. In order to read the event log, the user must have a read ACE in the access control list for the Event Log service.

6.1.4 SFR Summary

- **FAU_GEN.1**: The TOE audit collection is capable of generating audit events for items identified in section **5.1.1.1**. For each audit event the TSF records the date, time, user Security Identifier (SID) or name, logon type (for logon audit records), event ID, source, type, and category. **Table 25 Audit Events for Virtualization PP Requirements**, **Table 26 Audit Events for Extended Package Server Virtualization Management Requirements**, and **Table 27 Format for Audit Event Records** list the audit events.
- **FAU_SAR.1:** The PowerShell get-eventlog cmdlet and Windows Event Viewer provide authorized administrators with the ability to review audit data in a readable format.
- **FAU_STG.1:** The interface to the logs are restricted to authorized administrators, including clearing the audit log, and does not allow for the modification of audit data within the audit log. The TOE can be configured such that when any event logs are full the system will overwrite the oldest events in each log type, based on a system-defined value which can be modified by the administrator. The operational logs listed above also restrict authorized administrators to only read-only access.
- **FAU_STG_EXT.1:** Windows event and operational logs can be stored on a remote computer using an IPsec trusted network path as described in sections **6.2.3.3** and **6.8**.

6.2 Cryptographic Support

6.2.1 Cryptographic Algorithms and Operations

The Cryptography API: Next Generation (CNG) API is designed to be extensible at many levels and agnostic to cryptographic algorithm suites. Windows uses CNG exclusively for its own encryption needs

and provides public APIs for external developers. An important feature of CNG is its native implementation of the Suite B algorithms, including algorithms for AES (128, 192, 256 key sizes)[33], the SHA-1 and SHA-2 family (SHA-256, SHA-384 and SHA-512) of hashing algorithms, elliptic curve Diffie Hellman (ECDH), and elliptical curve DSA (ECDSA) over the NIST-standard prime curves P-256, P-384, and P-521.

Protocols such as the Internet Key Exchange (IKE), and Transport Layer Security (TLS), make use of elliptic curve Diffie-Hellman (ECDH) included in Suite B as well as hashing functions.

Deterministic random bit generation (DRBG) is implemented in accordance with NIST Special Publication 800-90. Windows generates random bits by taking the output of a cascade of two SP800-90 AES-256 counter mode based DRBGs in kernel-mode and four cascaded SP800-90 AES-256 DRBGs in user-mode; programmatic callers can choose to obtain either 128 or 256 bits from the RBG which is seeded from the Windows entropy pool. Windows has different entropy sources (deterministic and nondeterministic) which produce entropy data that is used for random numbers generation. In particular, this entropy data together with other data (such as the nonce) seed the DRBG algorithm. The entropy pool is populated using the following values:

- An initial entropy value from a seed file provided to the Windows OS Loader at boot time (512 bits of entropy). [34]
- A calculated value based on the high-resolution CPU cycle counter which fires after every 1024 interrupts (a continuous source providing 16384 bits of entropy).
- Random values gathered periodically from the Trusted Platform Module (TPM), (320 bits of entropy on boot, 384 bits thereafter on demand based on an OS timer).
- Random values gathered periodically by calling the RDRAND CPU instruction, (256 bits of entropy).

The entropy data is obtained from the entropy sources in a raw format and is health-tested before using it as input for the DRBG. The main source of entropy in the system is the CPU cycle counter which continuously tracks hardware interrupts. This serves as a sufficient health test; if the computer were not accumulating hardware and software interrupts it would not be running and therefore there would be no need for any entropy to seed, or reseed, the random bit generator. In the same manner, a failure of the TPM chip or the RDRAND instruction for the processor would be a critical error that halts the computer, effectively serving as an on-demand self-test.[35] In addition, when the user chooses to follow the CC administrative guidance, which includes operating Windows in the FIPS validated mode, it will run FIPS 140 AES-256 Counter Mode DBRG Known Answer Tests (instantiate, generate) on start-up. Windows always runs the SP 800-90-mandated self-tests for AES-CTR-DRBG during a reseed when the user chooses to operate Windows in the FIPS validated mode.[36]

[33] Note that the 192-bit key size is not used by Windows but is available to developers.
[34] The Windows OS Loader implements a SP 800-90 AES-CTR-DRBG and passes along 384 bits of entropy to the kernel for CNG to be use during initialization. This DBRG uses the same algorithms to obtain entropy from the CPU cycle counter, TPM, and RDRAND as described above.
[35] In other words, the expected result from the CPU cycle counter, the RDRAND instruction, and the TPM RBG is an apparently random value which will be used as an input to seed the RBG.
[36] Running Windows in FIPS validated mode is required according to the administrative guidance.

Each entropy source is independent of the other sources and does not depend on time. The CPU cycle counter inputs vary by environmental conditions such as data received on a network interface card, key presses on a keyboard, mouse movement and clicks, and touch input.

The root partition can provide entropy to operating systems executing in virtual partitions by means of a software-based TPM executing in the root partition, which called the “virtual TPM” (vTPM), and by a guest OS calling the RDRAND instruction. Isolation between guest VMs issuing RDRAND instructions is ensured by the hypervisor, and by the VMBus for access to the vTPM.

The TSF defends against tampering of the random number generation (RNG) / pseudorandom number generation (PRNG) sources by encapsulating its use in Kernel Security Device Driver. The interface for the Windows random number generator is BCryptGenRandom.

The CNG provider for random number generation is the AES_CTR_DRBG, when Windows requires the use of a salt it uses the Windows RBG.

The encryption and decryption operations are performed by independent modules, known as Cryptographic Service Providers (CSPs). Windows generates symmetric keys (AES keys) using the FIPS Approved random number generator.

In addition to encryption and decryption services, the TSF provides other cryptographic operations such as hashing and digital signatures. Hashing is used by other FIPS Approved algorithms implemented in Windows (the hashed message authentication code, RSA, DSA, and EC DSA signature services, Diffie-Hellman and elliptic curve Diffie-Hellman key agreement, and random bit generation). When Windows needs to establish an RSA-based shared secret key it can act both as a sender or recipient, any decryption errors which occur during key establishment are presented to the user at a highly abstracted level, such as a failure to connect.

The hash-based message authentication code functions (HMAC) are based on SHA-1, SHA-256, SHA-384, and SHA-512, have the following characteristics:

Table 16 HMAC Characteristics

HMAC Algorithm	Hash Function	Block Size	Output MAC Length	Key Length / Key Size
HMAC-SHA-1	SHA-1	512 bits	20 bytes	The key size is 10-63 bytes when the key size is less than the block size and the key size is 65 to 1024 bytes when the key size is greater than the block size. The key size may also equal the block size. The key size is variable.
HMAC-SHA2-256	SHA2-256	512 bits	32 bytes	Same as HMAC-SHA-1.
HMAC-SHA2-383	SHA2-383	1024 bits	48 bytes	The key size is 24-127 bytes when the key size is less than the block size and the key size is 129-1024 bytes when the key size is greater than the block size. The key size may also equal the block size.

HMAC Algorithm	Hash Function	Block Size	Output MAC Length	Key Length / Key Size
				The key size is variable.
HMAC-SHA2-512	SHA2-512	1024 bits	64 bytes	The key size is 32-127 bytes when the key size is less than the block size and the key size is 129-1024 bytes when the key size is greater than the block size. The key size may also equal the block size. The key size is variable.

6.2.2 Cryptographic Algorithm Validation

Table 17 Windows Server, Windows 10 version 1909 Cryptographic Algorithm Standards and Evaluation Methods

Cryptographic Operation	Standard	Requirement	Evaluation Method
Encryption/Decryption	FIPS 197 AES	FCS_COP.1(SYM)	NIST CAVP # C1363, # C1364, # C1365, # C1368
	NIST SP 800-38A CBC mode		# C1363, # C1368
	NIST SP 800-38C CCM mode		# C1363, # C1364
	NIST SP 800-38E XTS mode		# C1363
	NIST SP 800-38F KW mode		# C1365
	NIST SP 800-38D GCM mode		# C1363
	NIST SP 800-38A CTR mode		# C1363, # C1368
Digital signature (key generation)	FIPS 186-4 RSA	FCS_CKM.1	NIST CAVP # C1363, # C1366
Digital signature (generation)	FIPS 186-4 RSA	FCS_COP.1(SIGN)	NIST CAVP # C1363, # C1366, # C1368
Digital signature (verification)	FIPS 186-4 RSA	FCS_COP.1(SIGN)	NIST CAVP # C1363, # C1366, C1367, # C1368
Digital signature (key generation)	FIPS 186-4 DSA	FCS_CKM.1	NIST CAVP # C1363
Digital signature (generation and verification)	FIPS 186-4 DSA	Added as a prerequisite of NIST CAVP KAS # C1363	NIST CAVP # C1363

Digital signature (key generation)	FIPS 186-4 ECDSA	FCS_CKM.1	NIST CAVP C1363, # C1366, # C1368
Digital signature (signature generation and verification)	FIPS 186-4 ECDSA	FCS_COP.1(SIGN)	NIST CAVP # C1363, # C1366
Hashing	FIPS 180-4 SHA-1 and SHA-256, SHA-384, SHA-512	FCS_COP.1(HASH)	NIST CAVP # C1363
Keyed-Hash Message Authentication Code	FIPS 198-2 (HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512)	FCS_COP.1(HMAC)	NIST CAVP #C1363
Keyed-Hash Message Authentication Code	FIPS 198-2 (HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-384)	FCS_COP.1(HMAC)	NIST CAVP # C1368
Random number generation	NIST SP 800-90 CTR_DRBG	FCS_RBG_EXT.1	NIST CAVP # C1363, # C1368
Key agreement	NIST SP 800-56A ECDH	FCS_CKM.2	NIST CAVP # C1363, # C1368
Key establishment	NIST SP 800-56B RSA	FCS_CKM.2	NIST # C1363, # C1366, Tested by the CC evaluation lab[37]
Key establishment	NIST SP 800-56A FFC	FCS_CKM.2	NIST # C1363
Key-based key derivation	SP800-108		NIST CAVP # C1363, # C1365, # C1368
IKEv1	SP800-135	FCS_IPESEC_EXT.1	NIST CAVP # C1363
IKEv2	SP800-135	FCS_IPESEC_EXT.1	NIST CAVP # C1363
TLS	SP800-135	FCS_TLSC_EXT.2, FCS_TLSS_EXT.2, FCS_HTTPS_EXT.1	NIST CAVP # C1363

Table 18 Windows Server 2019 Cryptographic Algorithm Standards and Evaluation Methods

Cryptographic Operation	Standard	Requirement	Evaluation Method
Encryption/Decryption	FIPS 197 AES	FCS_COP.1(SYM)	NIST CAVP # C211, # C346, # C347, # C350
	NIST SP 800-38A CBC mode		# C211, # C350
	NIST SP 800-38C CCM mode		# C211, # C346

[37] The test results are described in the evaluation and Assurance Activity Report.

	NIST SP 800-38E XTS mode		# C211
	NIST SP 800-38F KW mode		# C347
	NIST SP 800-38D GCM mode		# C211
	NIST SP 800-38A CTR mode		# C211, # C350
Digital signature (key generation)	FIPS 186-4 RSA	FCS_CKM.1	NIST CAVP # C211, # C348
Digital signature (generation)	FIPS 186-4 RSA	FCS_COP.1(SIGN)	NIST CAVP # C211, # C348, # C350
Digital signature (verification)	FIPS 186-4 RSA	FCS_COP.1(SIGN)	NIST CAVP # C211, # C348, # C349, # C350
Digital signature (key generation)	FIPS 186-4 DSA	FCS_CKM.1	NIST CAVP # C211
Digital signature (generation and verification)	FIPS 186-4 DSA	Added as a prerequisite of NIST CAVP KAS # C 211	NIST CAVP # C211
Digital signature (key generation)	FIPS 186-4 ECDSA	FCS_CKM.1	NIST CAVP C 211, # C348, # C350
Digital signature (signature generation and verification)	FIPS 186-4 ECDSA	FCS_COP.1(SIGN)	NIST CAVP # C211, # C348
Hashing	FIPS 180-4 SHA-1 and SHA-256, SHA-384, SHA-512	FCS_COP.1(HASH)	NIST CAVP # C211
Keyed-Hash Message Authentication Code	FIPS 198-2 HMAC (SHA-1, SHA2-256, SHA2-384, SHA2-512)	FCS_COP.1(HMAC)	NIST CAVP # C211
Keyed-Hash Message Authentication Code	FIPS 198-2 HMAC (SHA-1, SHA2-256, SHA2-384)	FCS_COP.1(HMAC)	NIST CAVP # C350
Random number generation	NIST SP 800-90 CTR_DRBG	FCS_RBG_EXT.1	NIST CAVP # C211, # C350
Key agreement	NIST SP 800-56A ECDH	FCS_CKM.2	NIST CAVP # C211, # C350
Key establishment	NIST SP 800-56B RSA	FCS_CKM.2	NIST # C211, # C348, Tested by the CC evaluation lab[38]
Key establishment	NIST SP 800-56A FFC	FCS_CKM.2	NIST # C211
Key-based key derivation	SP800-108		NIST CAVP # C347, # C350
IKEv1	SP800-135	FCS_IPESEC_EXT.1	NIST CAVP # C211

[38] The test results are described in the evaluation and Assurance Activity Report.

IKEv2	SP800-135	FCS_IPESEC_EXT.1	NIST CAVP # C211
TLS	SP800-135	FCS_TLSC_EXT.2, FCS_TLSS_EXT.2, FCS_HTTPS_EXT.1	NIST CAVP # C211

CNG includes a user-mode key isolation service designed specifically to host secret and private keys in a protected process to mitigate tampering or access to sensitive key materials for user-mode processes. CNG performs a key error detection check on each transfer of key (internal and intermediate transfers). CNG prevents archiving of expired (private) signature keys and destroys non-persistent cryptographic keys. Windows overwrites each intermediate storage area for plaintext key/critical cryptographic security parameter (i.e., any storage, such as memory buffers for the key or plaintext password which was typed by the user that is included in the path of such data). This overwriting is performed as follows:

- For volatile memory, the overwrite is a single direct overwrite consisting of zeros using the RtlSecureZeroMemory function.

The following table describes the keys and secrets used for networking data protection, and TPM-based attestation; all keys originate within Windows unless stated otherwise. when these non-persistent keys or secrets are no longer needed, due to either normal end of the session or abnormal termination, or after protecting sensitive data using DPAPI, they are deleted as described above and in section 5.1.2.3; keys within the TPM are destroyed when the TPM is reset.

Table 19 Types of Non-persistent Keys Used by Windows

Key	Description
Symmetric encryption/decryption keys	Keys used for AES (FIPS 197) encryption/decryption for IPsec ESP, TLS, Wi-Fi.
HMAC keys	Keys used for HMAC-SHA1, HMAC-SHA256, HMAC-SHA384, and HMAC-SHA512 (FIPS 198-1) as part of IPsec
Asymmetric ECDSA Public Keys	Keys used for the verification of ECDSA digital signatures (FIPS 186-4) for IPsec traffic and peer authentication.
Asymmetric ECDSA Private Keys	Keys used for the calculation of ECDSA digital signatures (FIPS 186-4) for IPsec traffic and peer authentication.
Asymmetric RSA Public Keys	Keys used for the verification of RSA digital signatures (FIPS 186-4) for IPsec and TLS.
Asymmetric RSA Private Keys	Keys used for the calculation of RSA digital signatures (FIPS 186-4) for IPsec, TLS as well as TPM-based health attestations. The key size can be 2048 or 3072 bits.
Asymmetric DSA Private Keys	Keys used for the calculation of DSA digital signatures (FIPS 186-4) for IPsec and TLS. The key size can be 2048 or 3072 bits.
Asymmetric DSA Public Keys	Keys used for the verification of DSA digital signatures (FIPS 186-4) for IPsec and TLS. The key size can be 2048 or 3072 bits.

DH Private and Public values	Private and public values used for Diffie-Hellman key establishment for TLS.
ECDH Private and Public values	Private and public values used for EC Diffie-Hellman key establishment for TLS.
DPAPI master secret	512-bit random value used by DPAPI
DPAPI master AES key	256-bit encryption key that protects the DPAPI master secret
DPAPI AES key	256-bit encryption key used by DPAPI
DRBG seed	Seed for the main DRBG, zeroized during reseeding

Windows also uses persisted asymmetric RSA public keys to verify signed product updates.

6.2.3 Networking

6.2.3.1 TLS, HTTPS

The TOE implements TLS to enable a trusted network path that is used for server authentication and client (mutual) authentication, as well as HTTPS.

The following table summarizes the TLS RFCs implemented in Windows:

Table 20 TLS RFCs Implemented by Windows

RFC #	Name	How Used
3268	Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)	Specifies additional ciphersuites implemented by Windows.
4346	The Transport Layer Security (TLS) Protocol Version 1.1	Specifies requirements for TLS 1.1.
4366	Transport Layer Security (TLS) Extensions	Obsoletes RFC 3546 Requirements for TLS 1.1 extensions implemented by Windows.
4492	Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)	Specifies additional ciphersuites implemented by Windows.
4681	TLS User Mapping Extension	Extends TLS to include a User Principal Name during the TLS handshake.
5246	The Transport Layer Security (TLS) Protocol Version 1.2	Obsoletes RFCs 3268, 4346, and 4366. Specifies requirements for TLS 1.2.
5289	TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)	Specifies additional ciphersuites implemented by Windows.

The https://docs.microsoft.com/en-us/windows/win32/secauthn/cipher-suites-in-schannel article describes the complete set of TLS cipher suites implemented in Windows, of which the following are used in the evaluated configuration:

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
- TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268

- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492
- TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
- TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
- TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288
- TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289

In this evaluation, Windows is configured to request and accept only TLS 1.1 and TLS 1.2 protocol sessions, it will reject older and newer TLS versions. When negotiating a TLS 1.2 elliptic curve cipher suite, Windows will include automatically as part of the Client and Server Hello messages its supported elliptic curves extension, i.e., secp256r1, secp384r1, and secp521r1 and signature algorithm, i.e., SHA256, SHA384, and SHA512 based on the ciphersuites selected by the administrator**.** By default, the curve secp521r1 is disabled. This curve can be enabled adding its name in the ECC Curve Order file. In addition, the curve priority can be edited in this file**.**

On the other hand, by default the signature algorithms in the Client Hello message are: SHA1, SHA256, SHA384 and SHA512. The signature algorithm extension is configurable by editing a registry key to meet with the TLS client and server requirements. Each Windows component that uses TLS checks that the identifying information in the certificate matches what is expected, the component should reject the connection, these checks include checking the expected Distinguished Name (DN), Subject Name (SN), or Subject Alternative Name (SAN) attributes along with any applicable extended key usage identifiers. The DN, and any Subject Alternative Name, in the certificate is checked against the identity of the remote computer's DNS entry or IP address to ensure that it matches as described at http://technet.microsoft.com/en-us/library/cc783349(v=WS.10).aspx, and in particular the "Server Certificate Message" section. The reference identifier in Windows for TLS is the DNS name or IP address of the remote server, which is compared against the DNS name as presented identifier in the Subject Alternative Name (SAN) or the Subject Name of the certificate. There is no configuration of the reference identifier.

A certificate that uses a wildcard in the leftmost portion of the resource identifier (i.e., *.contoso.com) can be accepted for authentication, otherwise the certificate will be deemed invalid. Windows does not provide a general-purpose capability to "pin" TLS certificates.

Windows implements HTTPS as described in RFC 2818 so that Windows Store and system applications executing on the TOE can securely connect to external servers using HTTPS.

6.2.3.2 Wireless Networking

Windows has native implementations of IEEE 802.11-2012 and IEEE 802.11ac-2013 to provide secure wireless local area networking (Wi-Fi). Windows can use PRF-384 in WPA2 Wi-Fi sessions and generate AES 128-bit keys or use PRF-704 to generate AES 256-bit keys, both utilize the Windows RBG. Windows complies with the IEEE 802.11-2012 and IEEE 802.11ac-2013 standards and interoperates with other devices that implement the standard. Computers running a Windows OS typically have Wi-Fi CERTIFIED Interoperability Certificates from the Wi-Fi Alliance.

Windows implements key wrapping and unwrapping according to the NIST SP 800-38F specification (the "KW" mode) and so unwraps the Wi-Fi Group Temporal Key (GTK) which was sent by the access point. Because the GTK was protected by AES Key Wrap when it was delivered in an EAPOL-Key frame, the GTK is not exposed to the network.

6.2.3.3 IPsec

The Windows IPsec implementation is an integral part of the Windows operating system; it conforms to RFC 4301, Security Architecture for the Internet Protocol. This is documented publicly in the Windows protocol documentation at section 7.5.1 IPsec Overview.[39]

Windows implements both RFCS 2409, Internet Key Exchange (IKEv1), and RFC 4306, Internet Key Exchange version 2, (IKEv2).[40] Windows IPsec supports both tunnel mode and transport mode and provides an option for NAT traversal (reference: section 7.5.5, IPsec Encapsulations).[41] The RAS VPN interface uses tunnel mode only.

The Windows IPsec implementation includes a security policy database (SPD), which states how Windows should process network packets. The SPD uses the traffic source, destination and transport protocol to determine if a packet should be transmitted or received, blocked, or protected with IPsec, (reference: 7.5.3, Security Policy Database Structure), based on firewall processing rules.[42] These rules are described in the Windows Filtering Platform and section 6.5 of the *Windows Server, Windows Server 2019, and Microsoft Windows 10 Hyper-V Common Criteria Operational and Administrative Guidance (version 1909)*. In order to prevent unsolicited inbound traffic, an authorized administrator does not need to define a final catch-all rule which will discard a network packet when no other rules in the SPD apply because Windows will discard the packet. The security policy database also includes configuration settings to limit the time and number of sessions before a new key needs to be generated.

Windows 10 implements AES-GCM-128, AES-GCM-256, AES-CBC-128, and AES-CBC-256 as encryption algorithms for the encapsulating security payload (ESP) (reference: section 6, Appendix A, Product

[39] An offline copy is available in the Windows Protocols .zip file downloaded from https://docs.microsoft.com/en-us/openspecs/protocols/ms-protocolslp/9a3ae8a2-02e5-4d05-874a-b3551405d8f9, *Windows Protocol Overview,* [MS-WPO], starting from section 7.5.1.

[40] An offline copy is available in the Windows Protocols .zip file downloaded from https://docs.microsoft.com/en-us/openspecs/protocols/ms-protocolslp/9a3ae8a2-02e5-4d05-874a-b3551405d8f9, *Internet Key Exchange Protocol Extensions*, [MS-IKEE].

[41] [MS-WPO], starting from section 7.5.5.

[42] [MS-WPO], section 7.5.3.

Behavior).[43] However only AES-CBC-128 and AES-CBC-256 can be used for IKEv1 and IKEv2 to protect the encrypted payload. The resulting potential strength of the symmetric key will be 128 or 256 bits of security depending on whether the IPsec VPN client and IPsec VPN server agreed to use a 128 or 256 AES symmetric key to protect the network traffic. Windows implements HMAC-SHA1, HMAC-SHA-256 and HMAC-SHA-384[44] as authentication algorithms for key exchange as well as Diffie-Hellman Groups 14, 19, 20, and 24 (reference: section 6, Appendix A, Product Behavior); the IT administrator specifies the ciphersuite, integrity method, and DH Group as part of specifying the parameters for an IPsec association to a remote host.[45] The IPsec VPN client will propose a cryptosuite to the IPsec VPN server; if the server responds with a cryptosuite that the client supports, the client will use the server's proposed cryptosuite; in other words if multiple ciphersuites, including DH groups, are proposed by the IPsec VPN client, the IPsec VPN server selects the ciphersuite, including the DH group, for the IPsec security association. If the IPsec VPN client and server cannot agree on a cryptosuite, either side may terminate the connection attempt.

In order to prevent security being reduced while transitioning from IKE Phase 1 / IKEv2 SA, an authorized administrator must configure the IPsec VPN client such that algorithms with same strength are used for both IKE Phase 1 and Phase 2 as well as for IKEv2 SA and IKEv2 Child SA.

Windows constructs nonces, which are 32-byte random values, as specified in RFC 2408, Internet Security Association and Key Management Protocol (ISAKMP) section 3.13.[46] When a random number is needed for either a nonce or for key agreement, Windows uses a FIPS-validated random bit generator. When requested, the Windows random bit generator can generate 256 or 512 bits for the caller, the probability of guessing a 256-bit value is 1 in 2^{256} and a 512 bit value is 1 in 2^{512}. When generating the security value *x* used in the IKE Diffie-Hellman key exchange, g^x mod p, Windows uses a FIPS validated random number generator to generate 'x' with length 224, 256, 384, or 2048 bits for DH groups 14, 19, 20, and 24 respectively. [47] See **Table 17** and **Table 18** for the NIST CAVP validation numbers.

Windows operating systems do not implement the IKEv1 aggressive mode option during a Phase 1 key exchange. The administrator can specify the duration of IPsec Security Associations based on the amount of data sent during the SA or elapsed time since the SA began.

Windows implements peer authentication using 2048-bit RSA certificates,[48] or ECDSA certificates using the P-256 and P-384 curves for both IKEv1 and IKEv2.[49]

While Windows supports pre-shared IPsec keys, it is not recommended due to the potential use of weak pre-shared keys. Windows simply uses the pre-shared key that was entered by the authorized administrator, there is no additional processing on the input data.

[43] [MS-IKEE], Appendix A.
[44] Windows truncates the HMAC output as described in RFC 4868 for HMAC-SHA-256 and HMAC-SHA-384 and for HMAC-SHA1-96 as described in RFC 2404.
[45] [MS-IKEE], Appendix A.
[46] RFC 2408.
[47] http://technet.microsoft.com/en-us/library/cc962035.aspx describes the Diffie-Hellman key agreement process.
[48] [MS-IKEE], page 73.
[49] http://technet.microsoft.com/en-us/library/905aa96a-4af7-44b0-8e8f-d2b6854a91e6.

Windows will validate certificates as described in section 6.4.1 by comparing the distinguished name (DN) in the certificate to the expected distinguished name in the X.509v3 certificate presented by the VPN gateway and does not require additional configuration. This comparison occurs in the encrypted and authenticated IKE identification payload. The reference identifiers of the remote computer is compared against the presented identifier in either the Subject Alternative Name (SAN) or the Subject Name of the certificate. The reference identifier may be any of the IP address, Distinguished Name (DN) or Fully Qualified Domain Name (FQDN) of the VPN gateway or the user FQDN.

Table 21 Windows 10 Implementation of IPsec RFCs

RFC #	Name	How Used
2407	The Internet IP Security Domain of Interpretation for ISAKMP	Integral part of the Windows Internet Key Exchange (IKE) implementation.
2408	Internet Security Association and Key Management Protocol (ISAKMP)	Integral part of the Windows Internet Key Exchange (IKE) implementation.
2409	The Internet Key Exchange (IKE)	Integral part of the Windows Internet Key Exchange (IKE) implementation.
2986	PKCS #10: Certification Request Syntax Specification; Version 1.7	Public key certification requests issued by Windows.
4106	The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)	Certain IPsec cryptosuites implemented by Windows.
4109	Algorithms for Internet Key Exchange version 1 (IKEv1)	Certain IPsec cryptosuites implemented by Windows.
4301	Security Architecture for the Internet Protocol	Description of the general security architecture for IPsec.
4303	IP Encapsulating Security Payload (ESP)	Specifies the IP Encapsulating Security Payload (ESP) implemented by Windows.
4304	Extended Sequence Number (ESN) Addendum to IPsec Domain of Interpretation (DOI) for Internet Security Association and Key Management Protocol (ISAKMP)	Specifies a sequence number high-order extension that is implemented by Windows.
4306	Internet Key Exchange (IKEv2) Protocol	Integral part of the Windows Internet Key Exchange (IKE) implementation.
4307	Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)	Certain IPsec cryptosuites implemented by Windows.
4868	Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec	Certain IPsec cryptosuites implemented by Windows.
4945	The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX	Integral part of the Windows Internet Key Exchange (IKE) implementation.
5280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	Specifies PKI support implemented by Windows.
5282	Using Authenticated Encryption Algorithms with the Encrypted Payload	Certain IPsec cryptosuites implemented by Windows.

	of the Internet Key Exchange version 2 (IKEv2) Protocol	
5996	Internet Key Exchange Protocol Version 2 (IKEv2)	Integral part of the Windows Internet Key Exchange (IKE) implementation.
6379	Suite B Cryptographic Suites for IPsec	Certain IPsec cryptosuites implemented by Windows.

6.2.4 Protecting Data with DPAPI

Windows provides the Data Protection API, DPAPI, which Windows components, first-party and third-party applications can use to protect any persisted data which the developer deems to be sensitive. DPAPI will use AES CBC encryption with a key that is based in part on the user's password to protect the user data. When storing private keys and secrets associated with the user account, the encrypted data is stored on the file system in a directory which is part of the user's profile.

6.2.5 SFR Summary

- **FCS_CKM.1,[50] FCS_CKM.2,[51] FCS_COP.1(SYM), FCS_COP.1(HASH), FCS_COP.1(SIGN), FCS_COP.1(HMAC), FCS_RBG_EXT.1**: See **Table 17 Windows Server, Windows 10 version 1909 Cryptographic Algorithm Standards and Evaluation Methods** and **Table 18 Windows Server 2019 Cryptographic Algorithm Standards and Evaluation Methods**.
- **FCS_CKM_EXT.4**: Windows overwrites critical cryptographic parameters immediately after that data is no longer needed, this includes the secure key storage for private (asymmetric) keys and other data deemed by an authorized subject, such as the pre-shared key, to require secure storage using DPAPI and the NTFS discretionary access control policy.[52]
- **FCS_ENT_EXT.1:** The root partition provides entropy to virtual machines executing in child partitions.
- **FCS_TLSC_EXT.2, FCS_TLSS_EXT.2, FCS_HTTPS_EXT.1**: Windows implements TLS 1.1 and 1.2 to provide both server authentication and mutual authentication using X.509v3 certificates, confidentiality and integrity to upper-layer protocols such as Extensible Authentication Protocol and HTTP.
- **FCS_IPSEC_EXT.1**: Windows provides an IPsec implementation as described above in section **6.2.3.3**.

6.3 Virtualization (User Data Protection)

Hyper-V, which is the Microsoft Windows technology for computer virtualization, requires an Intel architecture computer with a 64-bit processor (x64), a minimum of 4 GB physical memory, support for

[50] In the context of this evaluation, Windows will generate RSA and ECC key pairs as part of establishing a TLS session.
[51] In the context of this evaluation, Windows will generate RSA and ECC key pairs as part of establishing a TLS session.
[52] See https://www.niap-ccevs.org/st/st_vid10677-st.pdf and http://www.commoncriteriaportal.org/files/epfiles/st_windows10.pdf.

Second-Level Address Translation (SLAT), hardware-enforced support for Data Execution Protection (DEP) and virtualization support from the underlying UEFI firmware.[53]

Other Windows features which leverage the core virtualization requirements described in section 5.1 may have additional hardware requirements; however those features extend beyond the scope of this evaluation.

6.3.1 Windows Hypervisor

The Windows Hypervisor provides the isolation services for the partitions executing on the physical processors; the Hypervisor Top-Level Functional Specification published at https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/reference/tlfs describes the externally visible behavior of the hypervisor. To ensure protection between different execution domains the Windows Hypervisor checks the input parameters for API calls into the hypervisor and returns an error if a non-permissible value is requested.

6.3.2 Virtual Machine Manager

The Virtual Machine Manager (VMM), which is a Windows user-mode win32 service, executes in the root partition and mediates all data flows between the root partition and each guest partition as well as between the OS executing in the guest partition and the peripheral resources attached to the physical computer such as storage and networking, and logical resources in the root partition such as the virtualized firmware (including firmware interfaces like ACPI) and the Virtual TPM. The VMM maintains a list of which virtualized resources are assigned to each partition and the access control mapping between the virtualized resource and the underlying physical resource.

By default, a partition is authorized access only to the virtualized processor and the virtualized view of physical memory. The Hyper-V administrator must explicitly provide access to fixed storage volumes, removable storage, networking, and the virtualized TPM. Furthermore, the Hyper-V administrator can grant “direct” access to a physical hard disk and removable storage.

The local administrator can manage the VMM, and by extension, the guest VMs controlled by the VMM, by using the Hyper-V PowerShell Cmdlets. The communication between the VMM and individual VMs is through the VMBus interface.

The virtualized devices accessible to the guest OS are:

- Storage
  - Hard Disk Drive
  - DVD Drive
  - Floppy Disk
  - Fiber Channel
  - IDE Controller
  - SCSI Controller
- Networking
  - Storage Area Network

[53] Intel Virtualization Technology (Intel VT) and AMD Virtualization (AMD-V) technology both provide these capabilities although only Intel processors were used in this evaluation.

  - Network Adapter
  - COM Port
- Virtual TPM

"A Guest VM cannot access the data of another Guest VM, or transfer data to another Guest VM other than through the mechanisms described in FDP_VMS_EXT.1.1 when expressly enabled by an authorized Administrator. There are no ~~known~~[54] design or implementation flaws that permit the above mechanisms to be bypassed or defeated, or for data to be transferred through undocumented mechanisms. This claim does not apply to covert channels or architectural side-channels." [55] This includes the virtualization removable media, such as a DVD (e.g. .ISO image) and USB storage (e.g., VHDX image).

"Software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform. There are no known design or implementation flaws that bypass or defeat VM isolation."[56]

6.3.2.1 Shielded Virtual Machines (VMs)

A shielded VM is a generation 2[57] virtual machine that has a virtual TPM, is encrypted, and can run only on healthy and approved host operating systems in the computing environment. The Hyper-V administrator can designate a virtual machine to become a Shielded VM. A Shielded VM is protected using BitLocker, to encrypt the VM configuration and virtual storage volume. When the VMM detects an integrity check error for either resource, it will not start or resume the VM and will generate an audit event.

The host operating system is deemed to be healthy if it passes the health attestation check. The combination of a shielded VM and attestation is the starting point for enclave-based computing.

6.3.3 Hardware Support

When the local administrator has enabled Hyper-V, the Windows hypervisor leverages virtualized machine instructions, such as the VMX instructions from Intel, to provide a virtual execution mode that emulates ring-0 runtime environment.

The hypervisor, also relies on processor support for second level address memory page translation tables to provide each partition with its own isolated view of physical memory allocated to the partition.

Guest operating systems that are aware they are executing in a virtualized environment are deemed to be "enlightened" and can leverage "enlightenments" to improve the performance of virtualized operations between partitions using VM Bus and HV Socket interfaces between the enlightened guest VM and the Hyper-V VM Manager.

6.3.4 Device Virtualization

"Parameters passed from Guest VMs to virtual device interfaces are thoroughly validated and all illegal values (as specified in the TSS) are rejected. Additionally, parameters passed from Guest VMs to virtual device interfaces are not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform. Thorough testing and architectural design reviews have been conducted to ensure the

[54] This protection profile assurance activity was modified as part of NIAP Technical Decision 109.
[55] Protection Profile for Virtualization, page 50
[56] Protection Profile for Virtualization, page 63.
[57] A feature for Windows operating systems beginning with Windows Server 2012.

accuracy of these claims, and there are no known design or implementation flaws that bypass or defeat the security of the virtual device interfaces."[58]

These virtualized devices are also known as "synthetic devices" in Windows computing.

6.3.5 Network Virtualization and Communications between VMs

Data can be shared between guest operating systems running in different partitions using virtual network adapters for VM to VM communications and by cut-and-paste to copy user-generated text and images between virtual machines. Both capabilities are disabled by default but the Hyper-V administrator can enable virtual networking and cut-and-paste using the Hyper-V Manager tool or PowerShell Cmdlets to configure networking.

"Traffic traversing a virtual network is visible only to Guest VMs that are configured by an Administrator to be members of that virtual network. There are no known design or implementation flaws that permit the virtual networking configuration to be bypassed or defeated, or for data to be transferred through undocumented mechanisms. This claim does not apply to covert channels or architectural side-channels."[59]

Hypervisor and OS developers also have ability to add support in their products for communications between a child partition and the host partition using hypercalls and between partitions using the inter-partition communication channel described in the Hypervisor TLFS.

A basic way to share data between two virtual machines is for a VM to acquire exclusive access to a physical or virtual storage volume, relinquish access to the volume, which is then acquired by the second virtual machine. The virtual storage volume is also known as a Virtual Hard Disk (VHD) which is a publicly-available image format specification that allows encapsulation of the hard disk into an individual file for use by the operating system as a virtual disk in all the same ways physical hard disks are used.[60]

6.3.6 VPN Client

The Windows IPsec VPN client can be configured by the device local administrator. The administrator can configure the IPsec VPN client that all IP traffic is routed through the IPsec tunnel except for:

- IKE traffic used to establish the VPN tunnel
- IPv4 ARP traffic for resolution of local network layer addresses and to establish a local address
- IPv6 NDP traffic for resolution of local network layer addresses and to establish a local address

The IPsec VPN is an end-to-end internetworking technology and so VPN sessions can be established over physical network protocols such as wireless LAN (Wi-Fi) or local area network.

The components responsible for routing IP traffic through the VPN client:

- The **IPv4 / IPv6 network stack** in the kernel processes ingoing and outgoing network traffic.

[58] Protection Profile for Virtualization, page 63.
[59] Protection Profile for Virtualization, page 51.
[60] The VHD file format is documented at https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-vhdx/83e061f8-f6e2-4de1-91bd-5d518a43d477.

- The **IPsec** and **IKE and AuthIP Keying Modules** service which hosts the IKE and Authenticated Internet Protocol (AuthIP) keying modules. These keying modules are used for authentication and key exchange in Internet Protocol security (IPsec).
- The **Remote Access Service** device driver in the kernel, which is used primarily for VPN connections; known as the "RAS IPsec VPN" or "RAS VPN".
- The **IPsec Policy Agent** service which enforces IPsec policies.

Universal Windows App developers can implement their own VPN client if authorized by Microsoft to use the **networkingVpnProvider** capability, which includes setting the policy to lockdown networking traffic as described above.[61]

6.3.7 Memory Management and Object Reuse

Windows ensures that any previous information content is unavailable upon allocation to subjects and objects. The TSF ensures that resources processed by the kernel or are exported to user-mode processes do not have residual information in the following ways:

- All objects are based on memory and disk storage. Memory allocated for objects, which includes memory allocated for network packets, is either overwritten with all zeros or overwritten with the provided data before being assigned to an object. Read/write pointers prevent reading beyond the space used by the object. Only the exact value of what is most recently written can be read and no more. For varying length objects, subsequent reads only return the exact value that was set, even though the actual allocated size of the object may be greater than this. Objects stored on disk are restricted to only disk space used for that object.
- Subject processes have associated memory and an execution context. The TSF ensures that the memory associated with subjects is either overwritten with all zeros or overwritten with user data before allocation as described in the previous point for memory allocated to objects. In addition, the execution context (processor registers) is initialized when new threads within a process are created and restored when a thread context switch occurs.
- Network packets processed by IPsec are encrypted in place. In other words, the data to be encrypted is not copied to a separate buffer and then encrypted. The encrypted network packet is encrypted into the same buffer and overwrites the plaintext network packet. The buffers allocated to hold network packets are allocated with enough space to accommodate padding required for encryption. Each network packet is held in its own buffer. There is a list of buffers, one for each packet. A buffer that holds a network packet is not reused for another network packet. After a buffer holding a network packet is no longer in use the memory allocated for the buffer is freed and released back to the TSF.

The above, in combination, will ensure that the memory used for inbound and outbound network packets does not contain data from previous use.

6.3.8 SFR Summary

- **FDP_HBI_EXT.1:** Aside from the physical hard disk and removable storage media, guest virtual machines do not have unmediated access to physical resources.

[61] See https://docs.microsoft.com/en-us/uwp/api/Windows.Networking.Vpn.

- **FDP_PPR_EXT.1:** The physical hard disk and removable media is the only non-virtualized resource that a guest virtual machine can access.
- **FDP_VMS_EXT.1:** Virtual networking and exclusive access to a storage volume can be used to transfer data between guest virtual machines on the computer.
- **FDP_VNC_EXT.1:** The administrator can enable or disable virtual network access for a guest virtual machine.
- **FDP_RIP_EXT.1:** The Hypervisor ensures that physical memory is cleared prior to use in child partition.
- **FDP_RIP_EXT.2**: The Hypervisor ensures that physical disk storage is cleared prior to use by a guest machine in a child partition.

6.4 Identification and Authentication

All logons are treated essentially in the same manner regardless of their source (e.g., interactive logon, network interface, internally initiated service logon) and start with an account name, domain name (which may be NULL; indicating the local system), and credentials, i.e., authentication data, that must be provided to the TSF to prevent an unauthorized individual from gaining access to the operating system.

Windows 10 and Windows Server can authenticate users based on username and password as well as using a Windows Hello PIN which is backed by a TPM. Windows 10 and Windows Server can also use physical or virtual smart card thus supporting multiple user authentication.

Password-based authentication to Windows succeeds when the credential provided by the user matches the stored protected representation of the password; Windows Hello and smart cards both use PIN-based authentication to unlock a protected resource, a private key, the stored representation of the PIN is protected by the kernel.

Password authentication can be used for interactive, service, and network logons and to initiate the “change password” screen; the Windows Hello PIN, physical and virtual smart cards can be used for interactive logons; and the Windows Hello PIN is used to re-authenticate the user when the user chooses to change their PIN.

When the authentication succeeds, the user will be logged onto their desktop, their screen unlocked, or their authentication factors changed depending whether the user logged onto the computer, the display was locked, or the PIN or password was to be changed.

The Local Security Authority component within Windows maintains a count of the consecutive failed logon attempts by security principals from their last successful authentication. When the number of consecutive failed logon attempts is larger than the policy for failed logon attempts, which ranges from 0 (never lockout the account) to 999, Windows will lockout the user account until the account has been re-enabled by an administrator or a period of time specified by the administrator has elapsed. Windows persists the number of consecutive failed logons on for the user and so rebooting the computer does not reset the failed logon counter. Interactive logons are done on the secure desktop, which does not allow other programs to run, and therefore prevents automated password guessing. In addition, the Windows logon component enforces a one second delay between every failed logon with an increased delay after several consecutive logon failures.

6.4.1 X.509 Certificate Validation and Generation

Every Windows component that uses X.509 certificates is responsible for performing certificate validation, however all components use a common system subcomponent,[62] which validates certificates as described in RFC 5280, and particular, the specific validation listed in section 5.1.4.5 including all applicable usage constraints such as Server Authentication for networking sessions and Code Signing when installing product updates. Every component that uses X.509 certificates will have a repository for public certificates and will select a certificate based on criteria such as entity name for the communication partner, any extended key usage constraints, and cryptographic algorithms associated with the certificate. The Windows component will use the same kinds of information along with a certification path and certificate trust lists as part of deciding to accept the certificate.

If certificate validation fails, or if Windows is not able to check the validation status for a certificate, Windows will not establish a trusted network channel, e.g. IPsec, however it will inform the user and seek their consent before establishing a HTTPS web browsing session. Certification validation for updates to Windows, mobile applications, and integrity verification is mandatory, neither the administrator nor the user have the option to bypass the results of a failed certificate validation; software installation and updates is further described in Windows and Application Updates.

When Windows needs to generate a certificate enrollment request it will include a distinguished name, information about the cryptographic algorithms used for the request, any certification extensions, and information about the client requesting the certificate.

6.4.2 Certificate Storage

In a Windows OS, stored certificates known as *trusted root certificates* are contained in certificate stores. Each user has their own certificate store and there is a certificate store for the computer account; access to a certificate store is managed by the discretionary access control policy in Windows such that only the authorized administrator, i.e., the user or the local administrator, can add or remove entries. Certificates which are used by applications, for example, TLS, are also placed in certificate stores for the user.

In addition to the standard certificate revocation processes, application certificates can be loaded by either using administrative tools such as **certutil.exe**, changes to the trusted root certificates can be made using Certificate Trust Lists.

6.4.3 IPsec and Pre-shared Keys

IPsec is the only protocol in this evaluation which supports the use of pre-shared keys. These keys can range from a-z, A-Z, the numbers 0 – 9, and any special character entered from the keyboard. The length of the pre-shared key can range from 1 to 9,999 characters, and so the specific length of 22 characters which the protection profile requires is supported.

The IPsec pre-shared key is used as-is without modification by Windows and so the pre-shared key does not use the Windows random number generator. The reasoning for this is that if the user needs to supply a particular key, that specific key should be used. If the user desires a randomized bit string, then

[62] See https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptography-functions for the win32 interface description for this component.

the solution is to use a X.509 certificate which will contain a bit string of suitable length and randomness.

6.4.4 SFR Summary

- **FIA_AFL_EXT.1**: After the number of consecutive failed authentication attempts for a user account has been surpassed, Windows can be configured to lockout the user account.
- **FIA_PMG_EXT.1:** Windows devices support logon passwords at least 14 characters in length to as long as 127 characters, with a minimum length specified by the administrator. Windows logon passwords can be composed from uppercase characters, lowercase characters, digits, and special characters to be used in passwords.
- **FIA_UAU.5**: A user can log on to Windows running on a physical or virtualized computer with a password; a user can also log on to a Windows physical computer with physical or virtual (TPM-based) smart card or PIN, a user can also establish network logons via TLS or IPsec with X.509 certificate.
- **FIA_UIA_EXT.1:** In the context of this evaluation, Windows will display a logon notice prior to interactive logon to Windows as described in section 6.7.
- **FIA_X509_EXT.1**: Windows validates X.509 certificates according to RFC 5280 and provides OCSP and CRL services, per RFC 2560 and 5759 respectively, for applications to check certificate revocation status.
- **FIA_X509_EXT.2(TLS), FIA_X509_EXT.2(IPSEC)**: Windows uses X.509 certificates for TLS, HTTPS, IPsec, code signing for system software updates, code signing for mobile applications, and code signing for integrity verification.

6.5 Security Management

The complete set of management functions are described in X.509 Certificate Authentication (FIA_X509_EXT.2(IPSEC))

FIA_X509_EXT.2.1(IPSEC)	The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [***IPsec***], and [***code signing for system software updates, code signing for integrity verification***].
FIA_X509_EXT.2.2(IPSEC)	When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall [***not accept the certificate***].

Security Management (FMT), the following table maps which activities can be done by a standard Windows user or a local administrator. A checkmark indicates which entity can invoke the management function on the host operating system. Standard users, or programs running on their behalf, are not able to modify policy or configuration that is set by the administrator, the result is that the user cannot override the configuration specified by the administrator.

The following two tables document which management activities can be done by the Hyper-V administrator to the host operating system, i.e., not the guest operating systems which execute on different partitions of the computer.

Table 22 Hyper-V Security Management Functions for Server Virtualization Scenarios

Number	Function	Administrator	User
1.	Ability to update the Virtualization System	√	No
2.	Ability to configure Administrator password policy as defined in FIA_PMG_EXT.1	√	No
3.	Ability to create, configure and delete VMs	√	No
4.	Ability to set default initial VM configurations	√	No
5.	Ability to configure virtual networks including VM	√	No
6.	Ability to configure and manage the audit system and audit data	√	No
7.	Ability to configure VM access to physical devices	√	√
8.	Ability to configure inter-VM data sharing	√	√
9.	Ability to enable/disable VM access to Hypercall functions	√	No
10.	Ability to configure removable media policy	√	No
11.	Ability to configure the cryptographic functionality	√	No
12.	Ability to change default authorization factors	√	No
13.	Ability to enable/disable screen lock	√	No
14.	Ability to configure screen lock inactivity timeout	√	No
15.	Ability to configure remote connection inactivity timeout	√	No
16.	Ability to configure lockout policy for unsuccessful authentication attempts through [***timeouts between attempts, limiting number of attempts during a time period***]	√	No
17. [63]	Ability to configure name/address of directory server to bind with	√	No
18.	Ability to configure name/address of audit/logging server to which to send audit/logging records	√	No
19.	Ability to configure name/address of network time server	√	No
20.	Ability to configure banner	√	No
21.	Ability to connect/disconnect removable devices to/from a VM	√	No
22.	Ability to start a VM	√	No
23.	Ability to stop/halt a VM	√	No
24.	Ability to checkpoint a VM	√	No
25.	Ability to suspend a VM	√	No
26.	Ability to resume a VM	√	No

6.5.1 SFR Summary

- **FMT_MOF_EXT.1**: Windows provides the user with the capability to administer the security functions described in the security target. The mappings to specific functions are described in each applicable section of the TOE Summary Specification.

[63] This extended package functional requirement was modified as part of NIAP Technical Decision 360.

- **FMT_MSA_EXT.1:** Data can be shared between guest virtual machines only through the mechanisms described in section 6.3.5 after the VM administrator enables the mechanism for the virtual machine.
- **FMT_SMO_EXT.1:** Windows provides the administrator flexibility to separate administrative and operational network traffic onto separate networks using physical means (separate network adapters with distinct IP addresses and subnets), logical means (logical subnets bound to the same physical adapter), TLS, TLS/HTTPS, and IPsec as referenced in the administrative guide.

6.6 Protection of the TSF

6.6.1 Separation and Domain Isolation

The TSF provides a security domain for its own protection and provides process isolation. The security domains used within and by the TSF consists of the following:

- Hardware
- Virtualization Partitions
- Kernel-mode software
- Trusted user-mode processes
- User-mode Administrative tools process
- Application Containers

The TSF hardware is managed by the TSF kernel-mode software and is not modifiable by untrusted subjects. The TSF kernel-mode software is protected from modification by hardware execution state and protection for both physical memory and memory allocated to a partition; an operating system image runs within a partition. The TSF hardware provides a software interrupt instruction that causes a state change from user mode to kernel mode within a partition. The TSF kernel-mode software is responsible for processing all interrupts and determines whether or not a valid kernel-mode call is being made. In addition, the TSF memory protection features ensure that attempts to access kernel-mode memory from user mode results in a hardware exception, ensuring that kernel-mode memory cannot be directly accessed by software not executing in the kernel mode.

The TSF provides process isolation for all user-mode processes through private virtual address spaces (private per process page tables), execution context (registers, program counters), and security context (handle table and token). The data structures defining process address space, execution context and security context are all stored in protected kernel-mode memory. All security relevant privileges are considered to enforce TSF Protection.

User-mode administrator tools execute with the security context of the process running on behalf of the authorized administrator. Administrator processes are protected like other user-mode processes, by process isolation.

Application Containers (“App Containers”) provide an execution environment for Universal Windows Applications which prevents Universal Windows Applications from accessing data created by other Universal Windows Applications except through brokered operating system services such as the File Picker dialog.

Like TSF processes, user processes also are provided a private address space and process context, and therefore are protected from each other. Additionally, the TSF has the added ability to protect memory pages using Data Execution Prevention (DEP) which marks memory pages in a process as non-executable unless the location explicitly contains executable code. When the processor is asked to execute instructions from a page marked as data, the processor will raise an exception for the OS to handle.

The TSF implements cryptographic mechanisms within a distinct user-mode process, where its services can be accessed by both kernel- and user-mode components, in order to isolate those functions from the rest of the TSF to limit exposure to possible errors while protecting those functions from potential tampering attempts.

Furthermore, the TSF includes a Code Integrity Verification feature, also known as Kernel-mode code signing (KMCS), whereby device drivers will be loaded only if they are digitally signed by either Microsoft or from a trusted root certificate authority recognized by Microsoft. KMCS uses public-key cryptography technology to verify the digital signature of each driver as it is loaded. When a driver tries to load, the TSF decrypts the hash included with the driver using the public key stored in the certificate. It then verifies that the hash matches the one that it computes based on the driver code using the FIPS - certified cryptographic libraries in the TSF. The authenticity of the certificate is also checked in the same way, but using the certificate authority's public key, which must be configured in and trusted by the TOE.

6.6.2 Protection of OS Binaries, Audit and Configuration Data

By default, a Windows operating system is installed into the \Windows\ directory of the first bootable storage partition for the computer. The logical name for this directory is %systemRoot%. The kernel, device drivers (.sys files), system executables (.exe files) and dynamically loadable libraries (.dll files) are stored in the \%systemRoot%\system32 directory and subdirectories below system32. Standard users have permissions to read and execute these files, however modify and write permissions are limited to the local administrator and system service accounts.

The root directory for audit logs is %systemRoot%\system32\winevt. The local administrator, Event Log service, and the system account have full control over the audit files; standard users are not authorized to access the logs.

The primary configuration data store for Windows is the registry, and there are separate registry hives for the computer itself and each user authorized to use the computer. The registry hives for operating system configuration data is located at %systemRoot%\system32\config; the registry hive for the user is located in the user’s profile home directory. Registry files use the same protection scheme as event log files.

6.6.3 Protection from Implementation Weaknesses

Windows runs on processors that provide support for virtual memory and enforce restrictions to read, write, and execute pages of virtual and physical memory. Collectively, this is known as Data Execution Prevention (DEP). On Intel platforms, DEP is called NX (no execute).

Windows provides a default heap allocator for use by user-mode processes; Windows applications can use the default heap or implement their own allocator. The heap is managed with a collection of metadata (which isn’t pre-allocated to a specific address), with integrity protection provided by internal

checksums and encoding the metadata. If the heap detects corruption due to a heap overrun (e.g. integrity checks fail), and heap termination on corruption is enabled for the process, then the process is immediately terminated.

The Windows kernel, user-mode applications, and all Windows Store Applications implement Address Space Layout Randomization (ASLR) in order to load executable code at unpredictable base addresses.[64] The base address is generated using a pseudo-random number generator that is seeded by high quality entropy sources when available which provides at least 8 random bits for memory mapping. [65]

The Windows runtime also provides stack buffer overrun protection capability that will terminate a process after Windows detects a potential buffer overrun on the thread's stack by checking canary values in the function prolog and epilog as well as reordering the stack. All Windows binaries and Windows Store Applications implement stack buffer overrun protection by being complied with the /GS option,[66] and checking that all Windows Store Applications are compiled with buffer overrun protection before ingesting the Windows Store Application into the Windows Store.

To enable these protections using the Microsoft Visual Studio development environment, programs are complied with /DYNAMICBASE option for ASLR, and optionally with /HIGHENTROPYVA for 64-bit ASLR, or /NXCOMPAT:NO to opt out of software-based DEP, and /GS (switched on by default) for stack buffer overrun protection.

Windows Store Applications are compiled with the /APPCONTAINER option which builds the executable to run in a Windows appcontainer, to run with the user-mode protections described in this section.

Control Flow Guard prevents exploitation of memory corruption vulnerabilities by checking before each indirect function call that the call target in the running OS was the same call target as specified in the source program text.

6.6.4 Windows Platform Integrity and Code Integrity

A Windows operating system verifies the integrity of Windows program code using the combination of Secure Boot and Code Integrity capabilities in Windows. On computers with a TPM, such as those used in this evaluation, before Windows will boot, the computer will read the Boot Configuration Database to select a boot application, and then verify the integrity of the early boot components, which includes the Boot Loader, the OS Loader, and the OS Resume binaries.

This capability, known as Secure Boot, checks that the file integrity of early boot components has not been compromised, mitigating the risk of rootkits and viruses, and additionally checks that critical boot-time data have not been modified. Secure Boot collects these file and configuration measurements and seals them to the TPM. When Secure Boot starts in the preboot environment, it will compare the sealed values from the TPM to the measured values from the current boot cycle and if those values do not match the sealed values, Secure Boot will lock the system (which prevents booting) and display a warning on the computer display. While the TPM is part of the external IT environment in this

[64] The 64-bit version of the Windows microkernel, ntoskrnl.exe, implements Kernel Patch Protection to prevent the modification of kernel data structures which could be exploited by stack-based vulnerabilities.
[65] The PRNG is seeded by the TPM RBG, the RDRAND instruction and other sources.
[66] Winload.exe, winresume.exe, tcblaunch.exe, tcbloader.dll, and hvloader.exe are loaded before the stack buffer overrun protection mechanism is operational and therefore are not compiled with this option.

evaluation, the hardware-protected hashes serve as the first step of the chain that provides integrity from the hardware, through the bootchain into the kernel and required device drivers.

When the measurements match, the UEFI firmware will load the OS Boot Manager, which is an Authenticode-signed image file, based on the Portable Executable (PE) image file format. A SHA-256 hash-based signature and a public key certificate chain are embedded in the boot manager Authenticode signed image file under the "Certificate" IMAGE_DATA_DIRECTORY of the IMAGE_OPTIONAL_HEADER of the file. This public key certificate chain ends in a root public key. The boot manager uses the embedded SHA-256 hash-based signature and public key certificate chain to validate its own integrity. A SHA-256 hash of the boot manager image file is calculated for the whole file, with the exception of the following three elements which are excluded from the hash calculation: the CheckSum field in the IMAGE_OPTIONAL_HEADER, the IMAGE_DIRECTORY_ENTRY_SECURITY IMAGE_DATA_DIRECTORY, and the public key certificate table, which always resides at the end of the image file.

If the boot manager is validated, then the root public key of the embedded public key certificate chain must match one of the Microsoft root public keys which indicate that Microsoft is the publisher of the boot manager. These root public keys are necessarily hardcoded in the boot manager. If the boot manager cannot validate its own integrity, then the boot manager does not continue to load other modules and displays an error message.

After the boot manager determines its integrity, it attempts to load one application from the following list of boot applications:

- OS Loader: (Winload.exe or Winload.efi): the boot application started by the boot manager load the Windows kernel to start the boot process
- OS Resume (winresume.exe or winresume.efi): the boot application started by the boot manager to resume the instance of the executing OS which is persisted in the hibernation file "hiberfil.sys"[67]
- A physical memory testing application (memtest.exe) to check the physical memory ICs for the machine are working correctly.[68]

These boot applications are also Authenticode signed image files and so, the Boot Manager uses the embedded trusted SHA-256 hash based signature and public key certificate chain within the boot application's IMAGE_OPTIONAL_HEADER to validate the integrity of the boot application before attempting to load it. Except for three elements which are excluded from the hash calculation (these are the same three elements mentioned above in the Boot Manager description), a hash of a boot application image file is calculated in the same manner as for the Boot Manager.[69]

If the boot application is validated, then the root public key of the embedded public key certificate chain must match one of the hardcoded Microsoft's root public keys. If the boot manager cannot validate the integrity of the boot application, then the boot manager will not load the boot application and instead

[67] The evaluated configuration precludes suspending/resuming Windows and so this boot application will not be used when operating Windows per the administrative guidance.
[68] This is considered to be a non-operational mode for the evaluation.
[69] Note that this is an additional integrity check in addition to the TPM measurements check.

displays an error message below along with the full name of the boot application that failed the integrity check.

After the boot application's integrity has been determined, the boot manager attempts to load the boot application. If the boot application is successfully loaded, the boot manager then transfers execution to the loaded application.

After the Winload boot application is loaded, it receives the transfer of execution from the boot manager. During its execution, Winload attempts to load the Windows kernel (ntoskrnl.exe) together with a number of early-launch drivers. Among the modules that Winload must validate in the Portable Executable (PE) image file format, are the cryptography-related modules listed below

- The Windows kernel (ntoskrnl.exe)
- The BitLocker drive encryption filter driver (fvevol.sys)
- The Windows kernel cryptography device driver (cng.sys)
- The Windows code integrity library module (ci.dll)

The four image files above have their trusted SHA hashes stored in catalog files that reside in the local machine catalog directory.

Because they are PKCS #7 SignedData messages, catalog files are signed. The root public key of the certificate chain used to verify the signature of a Microsoft's catalog file must match one of the Microsoft's root public keys indicating that Microsoft is the publisher of the Windows image files. These Microsoft's root public keys are hardcoded in the Winload boot application.

If the image files are validated, their SHA-256 hashes, as calculated by the Winload boot application, must match their trusted SHA-256 hashes in a Microsoft's catalog file, which has been verified by the Winload boot application. A hash of an image file is calculated for the whole file, with the exception of the following three elements which are excluded from the hash calculation: the CheckSum field in the IMAGE_OPTIONAL_HEADER, the IMAGE_DIRECTORY_ENTRY_SECURITY IMAGE_DATA_DIRECTORY, and the public key certificate table, which always resides at the end of the image file.

Should the Winload boot application be unable to validate the integrity of one of the Windows image files, the Winload boot application does not continue to load other Windows image files. Rather it displays an error message and fails into a non-operational mode. In limited circumstances the pre-boot environment will attempt to repair the boot environment, such as copying files from a repair partition to repair files with integrity errors. When repair is not possible, the boot manager will ask the user to reinstall Windows.

After the initial device drivers have been loaded, the Windows kernel will continue to boot the rest of the operating system using the Code Integrity capability (ci.dll) to measure code integrity for (1) the remaining kernel-mode and user-mode programs which need to be loaded for the OS to complete its boot and (2) after booting, CI also verifies the integrity of applications launched by the user (applications from Microsoft are always signed by Microsoft, and third-party applications which may be signed by the developer) by checking the RSA signature for the binary and SHA-256 hashes of the binary which are compared to the catalog files described above.

Kernel-mode code signing (KMCS), also managed by CI, prevents kernel-mode device drivers, such as the TCIP/IP network driver (tcpip.sys), from loading unless they are published and digitally signed by developers who have been vetted by one of a handful of trusted certificate authorities (CAs). KMCS, using public-key cryptography technologies, requires that kernel-mode code include a digital signature generated by one of the trusted certificate authorities. When a kernel device driver tries to load, Windows decrypts the hash included with the driver using the public key stored in the certificate, then verifies that the hash matches the one computed with the code. The authenticity of the certificate is checked in the same way, but using the certificate authority's public key, which is trusted by Windows. The root public key of the certificate chain that verifies the signature must match one of the Microsoft's root public keys indicating that Microsoft is the publisher of the Windows image files. These Microsoft's root public keys are hardcoded in the Windows boot loader.[70]

In addition, Windows File Protection maintains a set of protected files that are stored in a cache along with cryptographic hashes of each of those files. Once the system is initialized, Windows File Protection is loaded and will scan the protected files to ensure they have valid cryptographic hashes. Windows File Protection also registers itself to be notified should any of the protected files be modified so that it can recheck the cryptographic checksum at any point while the system is operational. Should the any of the cryptographic hash checks fail, the applicable file will be restored from the cache.

The description above describes the measured launch capability for the local Management System as defined in the Protection Profile for Virtualization; in data center environments the measured launch capability also is the basis for attestation for the computer booting into a known good state.

6.6.5 Windows and Application Updates

Updates to Windows are delivered as Microsoft Update Standalone Package files (.msu files) which are signed by Microsoft with two digital signatures, a RSA SHA1 signature for legacy applications and a RSA SHA-256 signature for modern applications. The digital signature is signed by *Microsoft Corporation*, with a certification path through a Microsoft Code Signing certificate and ultimately the Microsoft Root Certification Authority. These certificates are checked by the Windows Trusted Installer prior to installing the update.

The Windows operating system will check that the certificate is valid and has not been revoked using a standard PKI CRL. Once the Trusted Installer determines that the package is valid, it will update Windows; otherwise the installation will abort and there will be an error message in the event log. Note that the Windows installer will not install an update if the files in the package have lower version numbers than the installed files.

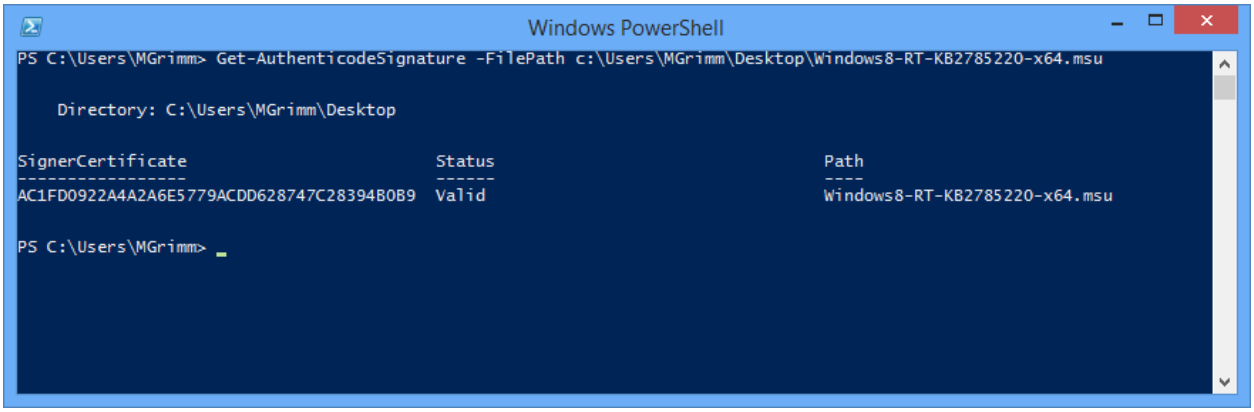
The integrity of the Microsoft Code Signing certificate on the computer is protected by the storage root key within the TPM, and the validated integrity of the Windows binaries as a result of Secure Boot and Code Integrity.

Updates to the Windows operating system, Windows applications, and Microsoft desktop applications are delivered through the Windows Update capability (for Windows) and Microsoft Update (for

[70] Enforcing the Kernel Mode Code Signing policy is mandatory for the x64 version of Windows. For the x86 version of Windows, Windows will check the signatures for all kernel executable code and will halt OS if it detects an integrity error in ntoskrnl.exe, bootvid.dll, hall.dll, kdcom.dll, ci.dll, clfs.dll, ksecdd.sys, pshed.dll, or tpm.sys.

Microsoft desktop applications), which is enabled by default, or the user can go to http://catalog.update.microsoft.com to search and obtain security updates on their own volition.

A user can then check that the signature is valid either by viewing the digital signature details of the file from Windows Explorer or by using the Get-`AuthenticodeSignature` PowerShell Cmdlet. The following is an example of using PowerShell:



If the Get-`AuthenticodeSignature` PowerShell Cmdlet or Windows Explorer could not verify the signature, the status will be marked as invalid. This verification check uses the same functionality described above.

6.6.5.1 Distributing updates

There are several distribution channels for updates to Windows and Windows applications:

- Windows Update: Windows Update is the web service for delivering Windows updates to directly to consumers.
- Windows Server Update Services (WSUS): WSUS is a server role in Windows Server which IT administrators can use to distribute application updates to users within their enterprise.
- Windows Store: The Windows Store is a web service for delivering updates to Universal Windows Platform apps which were originally installed from the Windows Store.

6.6.6 SFR Summary

- **FPT_DVD_EXT.1:** The Virtual Machine Manager will prevent a guest virtual machine from accessing a physical device that is not authorized for that VM.
- **FPT_EEM_EXT.1:** Windows provides multiple defense-in-depth protections by randomizing user-mode process address spaces and kernel-mode address space (ASLR), preventing overflows or corruption in the program and heaps, memory execution protection, and Control Flow Guard. Windows binaries are compiled with stack overflow protection (compiled using the **/Gs** option for native applications).
- **FPT_GVI_EXT.1:** Windows provides a Shielded VM capability through the combination of data protection as described in 6.3.2.1 and health attestation as described in FPT_ML_EXT.1.
- **FPT_HAS_EXT.1:** The Hypervisor will leverage VMX or VMT machine instructions to minimize the need for binary translation and Second Level Address Translation (SLAT) hardware support to

provide isolation between partitions and reduce the need for software-based shadow page tables.

- **FPT_HCL_EXT.1:** The Hypervisor provides a documented interface for virtual machines to invoke. The administrator enables VMs to use the hypercall interface by installing an enlightened operating system.
- **FPT_ML_EXT.1:** Windows and Hyper-V provide a measured launch service that provides assurance for the integrity of boot-stage components that comprise the hypervisor, the OS loader program, the kernel, kernel and the Boot Configuration Database.
- **FPT_RDM_EXT.1:** The Virtual Machine Manager implements an access control policy restricting access to any physical or virtualized removable media, such as DVD (e.g. .ISO image) and USB (e.g., .VHDX image) storage, that may be attached to the physical computer by first, ensuring that the Hyper-V administrator has granted access to the device for the VM, and then the Hyper-V VM Manager component ensures that the media device is allocated to only one partition (VM) at a time by providing a view of the actual media device to only the assigned partition.
- **FPT_VDP_EXT.1:** The Hypervisor and Virtual Machine Manager provide virtualized devices for guest virtual machines to use; data passed through the Hypervisor and Virtual Machine Manager interfaces are checked before it is used.
- **FPT_VIV_EXT.1:** The Hypervisor ensures that virtual machines executing in child partitions cannot interfere with each other nor the operating system in the root partition.
- **FPT_TUD_EXT.1, FPT_TUD_EXT.2**: Windows provides a means to identify the current version of the Windows software, the hardware model, and installed applications. Windows has update mechanisms to deliver updated operating system and application binaries and a means for a user to confirm that the digital signatures, which ensure the integrity of the update, are valid for both the operating system, applications, and Windows Store Applications.

6.7 TOE Access

As part of establishing the interactive logon session, Windows can be configured to display a logon banner, which is specified by the administrator, that the user must accept prior to establishing the session.

6.7.1 SFR Summary

- **FTA_TAB.1**: An authorized administrator can define and modify a banner that will be displayed prior to allowing a user to logon.

6.8 Trusted Channels

Windows provides trusted network channels to communicate with supporting IT infrastructure or applications:

- Using TLS (HTTPS) for certificate enrollment; CRL checking; authentication to network resources such as web (HTTPS) and directory (LDAP-S) servers.
- Using IPsec for remote management of Windows, including transferring audit events to another computer.

In order to establish a trusted channel, these communications are protected as described above in section **0**.

The remote access can be performed through the following methods:

- Remote Desktop Services Overview: https://technet.microsoft.com/en-us/library/hh831447.aspx
- Connect to another computer using Remote Desktop Connection: http://windows.microsoft.com/en-us/windows/connect-using-remote-desktop-connection#connect-using-remote-desktop-connection=windows-7
- PowerShell Remoting: https://docs.microsoft.com/en-US/powershell/scripting/setup/winrmsecurity?view=powershell-6

Both methods use TLS (1.1 or 1.2) protocol for establishing the remote connection.

The specific details for each protocol are described in section Network Protocols.

Hyper-V provides the human end-user with an interface to associate the keyboard and mouse from the user's local computer to a virtualized OS. Virtual machines are uniquely identified by their fully-qualified domain name in the organization's network.

6.8.1 SFR Summary

- **FTP_ITC_EXT.1**: Windows provides several trusted network channels that protect data in transit from disclosure, provide data integrity, and endpoint identification that is used by TLS, HTTPS, and IPsec. TLS and HTTPS are used as part of network-based authentication and certification validation, HTTPS is used for web-browsing and by other connection-based and datagram-based application protocols
- **FTP_TRP**.1: Windows provide a local trusted path service as described in **TOE Access** and a network-based trusted channel built on the network protocols described in this section.
- **FTP_UIF_EXT.1, FTP_UIF_EXT.2:** The Hyper-V client user interfaces indicates which virtual machine is connected to the physical keyboard and mouse. Windows ensures that the output display for a VM is identified uniquely.

6.9 Security Response Process

Microsoft utilizes industry standard practices to address reported product vulnerabilities. This includes a central email address (secure@microsoft.com) to report issues (as described at https://www.microsoft.com/en-us/msrc/faqs-report-an-issue?rtc=1), timely triage and root cause analysis, and responsible resolution of the report which may result in the release of a binary update. If a binary update is required, it is made available through automated channels to all customers following the process described at https://docs.microsoft.com/en-us/security-updates/. If the sender wishes to send secure email, there is a public PGP key for S/MIME at https://www.microsoft.com/en-us/msrc/pgp-key-msrc?rtc=1. Security updates for Microsoft products – operating system, firmware, and applications – are delivered as described in section **6.6.4** and **6.6.5**.

7 Protection Profile Conformance Claim

This section provides the protection profile conformance claim and supporting justifications and rationale.

This Security Target is in compliance with the Protection Profile for Virtualization, Version 1.0, November 17, 2016 (Virtualization PP), and the Extended Package Server Virtualization, version 1.0, November 17, 2016 (“Server Virtualization EP”).

For all of the content incorporated from the protection profile, the corresponding rationale in that protection profile remains applicable to demonstrate the correspondence between the TOE security functional requirements and TOE security objectives. Moreover, as demonstrated in this security target Windows runs on a wide variety of hardware ranging from tablets, convertibles, notebooks, desktop, and server computers and so it is a general-purpose operating system that offers virtualization capabilities.

The requirements in the protection profile are assumed to represent a complete set of requirements that serve to address any interdependencies. All the functional requirements in this security target have been copied from the protection profile or extended package so that all dependencies between SFRs are satisfied by the inclusion of the relevant component.

8 Rationale for Modifications to the Security Requirements

This section provides a rationale that describes how the Security Target reproduced the security functional requirements and security assurance requirements from the protection profile.

8.1 Functional Requirements

This Security Target includes security functional requirements (SFRs) that can be mapped to SFRs found in the protection profile along with SFRs that describe additional features and capabilities. The mapping from protection profile SFRs to security target SFRs along with rationale for operations is presented in **Table 23 Rationale for Operations**. SFR operations left incomplete in the protection profile have been completed in this security and are identified within each SFR in section 5.1 TOE Security Functional Requirements.

Table 23 Rationale for Operations

PP or EP	Requirement	ST Requirement	Operation & Rationale
PP	FAU_GEN.1	FAU_GEN.1	Two selections and refinements which are allowed by the PP.
PP	FAU_SAR.1	FAU_SAR.1	Copied from the PP without changes.
PP	FAU_STG.1	FAU_STG.1	Copied from the PP without changes.
PP	FAU_STG_EXT.1	FAU_STG_EXT.1	A selection and two assignments which are allowed by the PP.
PP	FCS_CKM.1	FCS_CKM.1	Two selections which are allowed by the PP.
PP	FCS_CKM.2	FCS_CKM.2	A selection which is allowed by the PP.
PP	FCS_CKM_EXT.4	FCS_CKM_EXT.4	Copied from the PP without changes.
PP	FCS_COP.1(1)	FCS_COP.1(SYM)	Two selections which are allowed by the PP.
PP	FCS_COP.1(2)	FCS_COP.1(HASH)	Two selections which are allowed by the PP.
PP	FCS_COP.1(3)	FCS_COP.1(SIGN)	Two selections which are allowed by the PP.
PP	FCS_COP.1(4)	FCS_COP.1(HMAC)	Two selections and an assignment which are allowed by the PP.
PP	FCS_RBG_EXT.1	FCS_RBG_EXT.1	Two selections which are allowed by the PP.
PP	FCS_ENT_EXT.1	FCS_ENT_EXT.1	A selection which is allowed by the PP.
PP	FCS_IPSEC_EXT.1	FCS_IPSEC_EXT.1	Multiple selections and assignments which are allowed by the PP.
PP	FCS_TLSC_EXT.2	FCS_TLSC_EXT.2	Three selections which are allowed by the PP.

PP or EP	Requirement	ST Requirement	Operation & Rationale
PP	FCS_TLSS_EXT.2	FCS_TLSS_EXT.2	Multiple selections which are allowed by the PP.
PP	FCS_HTTPS_EXT.1	FCS_HTTPS_EXT.1	Copied from the PP without changes.
PP	FDP_HBI_EXT.1	FDP_HBI_EXT.1	Two selections and assignments which are allowed by the PP.
PP	FDP_PPR_EXT.1	FDP_PPR_EXT.1	Multiple selections and assignments which are allowed by the PP.
PP	FDP_RIP_EXT.1	FDP_RIP_EXT.1	Copied from the PP without changes.
PP	FDP_RIP_EXT.2	FDP_RIP_EXT.2	Copied from the PP without changes.
PP	FDP_VMS_EXT.1	FDP_VMS_EXT.1	Multiple selections and assignments which are allowed by the PP.
PP	FDP_VNC_EXT.1	FDP_VNC_EXT.1	Copied from the PP without changes.
PP	FIA_AFL_EXT.1	FIA_AFL_EXT.1	Three selections and two assignments which are allowed by the PP.
PP	FIA_PMG_EXT.1	FIA_PMG_EXT.1	A selection which is allowed by the PP.
PP	FIA_UAU.5	FIA_UAU.5	Multiple selections and an assignment which is allowed by the PP.
PP	FIA_UIA_EXT.1	FIA_UIA_EXT.1	Copied from the PP without changes.
PP	FIA_X509_EXT.1	FIA_X509_EXT.1	A refinement and a selection which is allowed by the PP.
PP	FIA_X509_EXT.2	FIA_X509_EXT.2(TLS)	Three selections which are allowed by the PP.
PP	FIA_X509_EXT.2	FIA_X509_EXT.2(IPSEC)	Three selections which are allowed by the PP.
SV EP	FMT_MOF_EXT.1	FMT_MOF_EXT.1	Two selections and a refinement which is allowed by the Server Virtualization EP.
PP	FMT_MSA_EXT.1	FMT_MSA_EXT.1	A selection which is allowed by the PP.
PP	FMT_SMO_EXT.1	FMT_SMO_EXT.1	A selection which is allowed by the PP.
PP	FPT_DVD_EXT.1	FPT_DVD_EXT.1	Copied from the PP without changes.
PP	FPT_EEM_EXT.1	FPT_EEM_EXT.1	A selection and assignment which is allowed by the PP.

PP or EP	Requirement	ST Requirement	Operation & Rationale
PP	FPT_GVI_EXT.1	FPT_GVI_EXT.1	An assignment which is allowed by the PP.
PP	FPT_HAS_EXT.1	FPT_HAS_EXT.1	Two assignments which are allowed by the PP.
PP	FPT_HCL_EXT.1	FPT_HCL_EXT.1	An assignment which are allowed by the PP.
PP	FPT_ML_EXT.1	FPT_ML_EXT.1	A selection and two assignments which are allowed by the PP.
PP	FPT_RDM_EXT.1	FPT_RDM_EXT.1	A selection and an assignment which are allowed by the PP.
PP	FPT_TUD_EXT.1	FPT_TUD_EXT.1	Two selections which are allowed by the PP.
PP	FPT_TUD_EXT.2	FPT_TUD_EXT.2	Copied from the PP without changes.
PP	FPT_VDP_EXT.1	FPT_VDP_EXT.1	Copied from the PP without changes.
PP	FPT_VIV_EXT.1	FPT_VIV_EXT.1	Copied from the PP without changes.
PP	FTA_TAB.1	FTA_TAB.1	Copied from the PP without changes.
PP	FTP_ITC_EXT.1	FTP_ITC_EXT.1	Two selections which are allowed by the PP.
PP	FTP_TRP.1	FTP_TRP.1	Copied from the PP without changes.
PP	FTP_UIF_EXT.1	FTP_UIF_EXT.1	Copied from the PP without changes.
PP	FTP_UIF_EXT.2	FTP_UIF_EXT.2	Copied from the PP without changes.

8.2 Security Assurance Requirements

The statement of security assurance requirements (SARs) found in section 5.2.1 is in exact conformance with the Protection Profile for Virtualization.

8.3 Rationale for the TOE Summary Specification

This section, in conjunction with section 6, the TOE Summary Specification (TSS), provides evidence that the security functions are suitable to meet the TOE security requirements.

Each subsection in section 6, TOE Security Functions (TSFs), describes a Security Function (SF) of the TOE. Each description is followed with rationale that indicates which requirements are satisfied by aspects of the corresponding SF. The set of security functions work together to satisfy all of the functional requirements. Furthermore, all the security functions are necessary in order for the TSF to provide the required security functionality.

The set of security functions work together to provide all of the security requirements as indicated in **Table 24**. The security functions described in the TOE Summary Specification and listed in the tables below are all necessary for the required security functionality in the TSF.

Table 24 Requirement to Security Function Correspondence

PP or EP	Requirement	Audit	Cryptographic Protection	User Data Protection	I & A	Security Management	TSF Protection	Resource Utilization	TOE Access	Trusted Path / Channel
PP	FAU_GEN.1	X								
PP	FAU_SAR.1	X								
PP	FAU_STG.1	X								
PP	FAU_STG_EXT.1	X								
PP	FCS_CKM.1		X							
PP	FCS_CKM.2		X							
PP	FCS_CKM_EXT.4		X							
PP	FCS_COP.1(SYM)		X							
PP	FCS_COP.1(HASH)		X							
PP	FCS_COP.1(SIGN)		X							
PP	FCS_COP.1(HMAC)		X							
PP	FCS_RBG_EXT.1		X							
PP	FCS_ENT_EXT.1		X							
PP	FCS_IPSEC_EXT.1		X							
PP	FCS_TLSC_EXT.2		X							
PP	FCS_TLSS_EXT.2		X							
PP	FCS_HTTPS_EXT.1		X							
PP	FDP_HBI_EXT.1			X						
PP	FDP_PPR_EXT.1			X						
PP	FDP_RIP_EXT.1			X						
PP	FDP_RIP_EXT.2			X						
PP	FDP_VMS_EXT.1			X						
PP	FDP_VNC_EXT.1			X						
PP	FIA_AFL_EXT.1				X					
PP	FIA_PMG_EXT.1				X					
PP	FIA_UAU.5				X					
PP	FIA_UIA_EXT.1				X					
PP	FIA_X509_EXT.1				X					
PP	FIA_X509_EXT.2(TLS)				X					
PP	FIA_X509_EXT.2(IPSEC)				X					
SV EP	FMT_MOF_EXT.1					X				

PP or EP	Requirement	Audit	Cryptographic Protection	User Data Protection	I & A	Security Management	TSF Protection	Resource Utilization	TOE Access	Trusted Path / Channel
PP	FMT_MSA_EXT.1					X				
PP	FMT_SMO_EXT.1					X				
PP	FPT_DVD_EXT.1						X			
PP	FPT_EEM_EXT.1						X			
PP	FPT_GVI_EXT.1						X			
PP	FPT_HAS_EXT.1						X			
PP	FPT_HCL_EXT.1						X			
PP	FPT_ML_EXT.1						X			
PP	FPT_RDM_EXT.1						X			
PP	FPT_TUD_EXT.1						X			
PP	FPT_TUD_EXT.2						X			
PP	FPT_VDP_EXT.1						X			
PP	FPT_VIV_EXT.1						X			
PP	FTA_TAB.1								X	
PP	FTP_ITC_EXT.1									X
PP	FTP_TRP.1									X
PP	FTP_UIF_EXT.1									X
PP	FTP_UIF_EXT.2									X

9 Appendix A: List of Abbreviations

Abbreviation	Meaning
ACE	Access Control Entry
ACL	Access Control List
ACPI	Advanced Configuration and Power Interface
AD	Active Directory
AES	Advanced Encryption Standard
AGD	Administrator Guidance Document
AH	[IPsec] Authentication Header
ALPC	Advanced Local Process Communication
API	Application Programming Interface
ARP	Address Resolution Protocol
BCD	Boot Configuration Database
CA	Certificate Authority
CBC	Cipher Block Chaining
CC	Common Criteria
CM	Configuration Management; Control Management
CPU	Central Processing Unit
CRL	Certificate Revocation List
CryptoAPI	Cryptographic API
CSP	Cryptographic Service Provider
DAC	Discretionary Access Control
DACL	Discretionary Access Control List
DC	Domain Controller
DEP	Data Execution Prevention
DH	Diffie-Hellman
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
DNS	Domain Name System
DS	Directory Service
DSA	Digital Signature Algorithm
DVD	Digital Versatile Disk
EAL	Evaluation Assurance Level
ESP	Encapsulating Security Protocol
FIPS	Federal Information Processing Standard
GP OS PP	Protection Profile for General Purpose Operating Systems
GUI	Graphical User Interface
GUID	Globally Unique Identifiers
HTTP	Hypertext Transfer Protocol
HTTPS	Secure HTTP
HV	Hypervisor
I/O	Input / Output
I&A	Identification and Authentication
IA	Information Assurance

IC	Integrated Circuit (aka “chip”)
ID	Identification
IETF	Internet Engineering Task Force
IIS	Internet Information Services
IKE	Internet Key Exchange
IP	Internet Protocol
IPv4	IP Version 4
IPv6	IP Version 6
IPsec	IP Security
ISAKMP	ISA Key Management Protocol
IT	Information Technology
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LSA	Local Security Authority
LSASS	LSA Subsystem Service
MAC	Message Authentication Code
MSR	Model Specific Register
NAT	Network Address Translation
NDP	Network Discovery Protocol
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
NMI	Non-maskable Interrupt
NTFS	New Technology File System
NTLM	New Technology LAN Manager
OCSP	Online Certificate Status Protocol
OS	Operating System
PAE	Physical Address Extension
PIN	Personal Identification Number
PKCS	Public Key Certificate Standard
PKI	Public Key Infrastructure
PKIX	PKI Exchange
PP	Protection Profile
PRNG	Pseudo Random Number Generator
RAM	Random Access Memory
RAS	Remote Access Service
RNG	Random Number Generator
RSA	Rivest, Shamir and Adleman
SA	Security Association
SACL	System Access Control List
SAR	Security Assurance Requirement
SAS	Secure Attention Sequence
SD	Security Descriptor
SHA	Secure Hash Algorithm
SID	Security Identifier
SF	Security Functions
SFP	Security Functional Policy

SFR	Security Functional Requirement
SLAT	Second Level Address Translation
SMI	System Management Interrupt
SP	[NIST] Special Publication
SPD	[IPsec] Security Policy Database
SRM	Security Reference Monitor
SSL	Secure Sockets Layer
SSP	Security Support Providers
SSPI	Security Support Provider Interface
ST	Security Target
TCP	Transmission Control Protocol
TLFS	Top-Level Functional Specification
TLS	Transport Layer Security
TOE	Target of Evaluation
TPM	Trusted Platform Module
TSC	TOE Scope of Control
TSF	TOE Security Functions
TSP	TOE Security Policy
TSS	TOE Summary Specification
UEFI	Unified Extensible Firmware Interface
UI	User Interface
UPN	User Principal Name
URL	Uniform Resource Locator
USB	Universal Serial Bus
USN	Update Sequence Number
UUID	Universally Unique Identifier
VMM	Virtual Machine Monitor
VPN	Virtual Private Network
WMI	Windows Management Instrumentation
WSUS	Windows Software Update Service
WU	Windows Update
WSDL	Web Service Description Language
WWW	World-Wide Web
X64	A 64-bit instruction set architecture
X86	A 32-bit instruction set architecture

10 Appendix B: Audit Events for the Protection Profile for Virtualization

10.1 Audit Events for Virtualization PP Requirements

Table 25 Audit Events for Virtualization PP Requirements

SFR	Auditable Events	Additional Audit Record Contents	Log: Event Id
FAU_GEN.1	Start-up and shutdown of audit functions		Security: **4608** (Startup) Security: **1100** (Shut down)
FAU_STG_EXT.1	Failure of audit data capture due to lack of disk space or pre-defined limit. On failure of logging function, capture record of failure and record upon restart of logging function.	None	Security: **1104**
FCS_RBG_EXT.1	Failure of the randomization process.	No additional information.	System: 20
FDP_PPR_EXT.1	Successful and failed VM connections to physical devices where connection is governed by configurable policy. Security policy violations.	VM and physical device identifiers. Identifier for the security policy that was violated.	Applications and Services Logs\Microsoft\Windows\Hyper-V-VMMS\Networking: **26074** (Success) Applications and Services Logs\Microsoft\Windows\Hyper-V-Worker\Admin: **12140** (Failure, Policy Violation) Applications and Services Logs\Microsoft\Windows\WMI-Activity\Operational: **5858** (Policy Violation due to Insufficient Privilege)
FDP_VNC_EXT.1	Successful and failed attempts to connect VMs to virtual and physical networking components. Security policy violations.	VM and virtual or physical networking component identifiers. Identifier for the security policy that was violated.	Applications and Services Logs\Microsoft\Windows\Hyper-V-Worker\Admin: **12597, 12598** (Success, Configuration) Security: **4656** (Failure, Policy Violation)

SFR	Auditable Events	Additional Audit Record Contents	Log: Event Id
	Administrator configuration of inter-VM communications channels between VMs.		Applications and Services Logs\Microsoft\Windows\WMI-Activity\Operational: **5858** (Policy Violation due to Insufficient Privilege)
FTP_ITC_EXT.1	Initiation of the trusted channel. Termination of the trusted channel. Failures of the trusted path functions	User ID and remote source (IP Address) if feasible.	Security: **4651, 5451** (IPsec Initiation) Security: **4655, 5452** (IPsec Termination) Security: **4652** (Failure, Main Mode)**, 4654** (Failure, Quick Mode) System: **36880** (TLS, HTTPS Initiation) Microsoft-Windows-SChannel-Events/Perf: **1793** (TLS, HTTPS Termination) System: **36888** (TLS, HTTPS Failure)
FTP_TRP.1	Initiation of the trusted channel. Termination of the trusted channel. Failures of the trusted path functions.	User ID and remote source (IP Address) if feasible.	Security: **4651, 5451** (Initiation) Security: **4655, 5452** (Termination)
FIA_UIA_EXT.1	Administrator authentication attempts All use of the identification and authentication mechanism. Administrator session start time and end time.	Provided user identity, origin of the attempt (e.g. console, remote IP address). None.	Security: **4624** (Authentication attempt, successful), **4625** (Authentication attempt, failed) Security: **4624** (Start time) Security: **4634** (End time)
FIA_X509_EXT.1	Failure to validate a certificate.	Reason for failure.	Applications and Services Logs > Microsoft > Windows >CAPI2 > Operational: **11**
FMT_MOF_EXT.1	Updates to the TOE. Configuration changes (system, network, audit function, Guest VM time, etc.). Start-up and shutdown of the TOE	Configuration changes.	Setup: **1** (Initiation) Setup: **2** (Success) Setup: **3** (Failure) Start-up and shutdown of the TOE: Windows Logs\Security: **1100**, **4608**

SFR	Auditable Events	Additional Audit Record Contents	Log: Event Id
	VM Start/Stop/Suspend events. Start and end of remote management session.		VM Start/Stop/Suspend events: Applications and Services Logs\Microsoft\Windows\Hyper-V-Worker\Admin: **18500, 18502, 18504, 18510, 18516** Start and end of remote management session: Windows Logs\Security: **4624, 4634**
	Account created, modified, enabled, disabled, removed	None.	Create: Windows Logs\Security: **4720** Modify: Windows Logs\Security: **4738** Enable: Windows Logs\Security: **4722** Disable: Windows Logs\Security: **4725** Remove: Windows Logs\Security: **4726**
FPT_TUD_EXT.1	Initiation of update. Failure of signature verification.	No additional information.	Setup: **1** (Initiation) Setup: **3** (Failure)
FPT_GVI_EXT.1	Actions taken due to failed integrity check.	None.	Applications and Services Logs\Microsoft\Windows\Hyper-V-VMMS\Admin: **2014**
FPT_HCL_EXT.1	Attempts to access disabled hypercall interfaces. Security policy violations.	Interface for which access was attempted. Identifier for the security policy that was violated.	**N/A**: Hypercall functions may not be disabled.
FPT_ML_EXT.1	Integrity measurements collected	Integrity measurement values	System: **20**

SFR	Auditable Events	Additional Audit Record Contents	Log: Event Id
FPT_RDM_EXT.1	Connection/disconnection of removable media or device to/from a VM. 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.)	Applications and Services Logs\Microsoft\Windows\Hyper-V-VMMS\Analytic: **12170** (Connect) Applications and Services Logs\Microsoft\Windows\Hyper-V-VMMS\Analytic: **12180** (Disconnect)
FCS_IPSEC_EXT.1	Failure to establish an IPsec SA. Establishment/Termination of an IPsec SA.	Reason for failure. Non-TOE endpoint of connection (IP address) for both successes and failures.	Security: **4651, 5451** (Initiation) Security: **4655, 5452** (Termination) Security: **4652, 4654** (Failure)
FCS_TLSC_EXT.2	Failure to establish a TLS Session. Establishment/Termination of a TLS session.	Reason for failure. Non-TOE endpoint of connection (IP address).	System: **36888** (Failure) System: **36880** (Establishment) Microsoft-Windows-SChannel-Events/Perf: **1793** (Terminate)
FCS_TLSS_EXT.2	Failure to establish a TLS Session. Establishment/Termination of a TLS session.	Reason for failure. Non-TOE endpoint of connection (IP address).	System: **36888** (Failure) System: **36880** (Establishment) Microsoft-Windows-SChannel-Events/Perf: **1793** (Terminate)
FCS_HTTPS_EXT.1	Failure to establish a HTTPS Session. Establishment/Termination of a HTTPS session.	Reason for failure. Non-TOE endpoint of connection (IP address) for both successes and failures.	System: **36888** (Failure) System: **36880** (Establishment) Microsoft-Windows-SChannel-Events/Perf: **1793** (Terminate)

10.2 Audit Events for Extended Package Server Virtualization Management Requirements

Table 26 Audit Events for Extended Package Server Virtualization Management Requirements

Administrative Action/Management Functions	ID
Ability to update the Virtualization System	Setup: **1** (Initiation) Setup: **2** (Success) Setup: **3** (Failure)
Ability to configure Administrator password policy as defined in FIA_PMG_EXT.1	Security: **4739**
Ability to create, configure and delete VMs	Applications and Services Logs\Microsoft\Windows\Hyper-V-VMMS\Admin: **13002** (Create)**, 13003** (Delete) Applications and Services Logs\Microsoft\Windows\Hyper-V-VMMS\Analytic: **12170** (Configure, adding components), **12180** (Configure, removing components) Applications and Services Logs\Microsoft\Windows\VHDMP: **12** (Configure, creating virtual disk), **16** (Configure, deleting virtual disk)
Ability to set default initial VM configurations	Applications and Services Logs\Microsoft\Windows\Hyper-V-VMMS\Analytic: **12170** (Configure, adding components), **12180** (Configure, removing components) Applications and Services Logs\Microsoft\Windows\VHDMP: **12** (Configure, creating virtual disk), **16** (Configure, deleting virtual disk)
Ability to configure virtual networks including VM	Applications and Services Logs\Microsoft\Windows\Hyper-V-VMMS\Networking: **26000, 26004, 26012, 26016, 26074**
Ability to configure and manage the audit system and audit data	Security: **4719**
Ability to configure VM access to physical devices	Applications and Services Logs\Microsoft\Windows\Hyper-V-VMMS\Analytic: **12170, 12180**
Ability to configure inter-VM data sharing	Applications and Services Microsoft-Windows-Hyper-V-VMMS/Admin: **12514**
Ability to enable/disable VM access to Hypercall functions	**N/A,** as the hypercall interfaces may not be disabled for an enlightened guest operating system.
Ability to configure removable media policy	Applications and Services Microsoft-Windows-Hyper-V-VMMS/Analytic: **12170, 12180**
Ability to configure the cryptographic functionality	Security: **4657** (TLS) Security: **5449** (IPsec)
Ability to change default authorization factors	Security: **4657** (ObjectValueName: **ScForceOption**)
Ability to enable/disable screen lock	Security: **4663**
Ability to configure screen lock inactivity timeout	Security: **4663**
Ability to configure remote connection inactivity timeout	Security: **4657** (ObjectValueName: **MaxIdleTime**)

Administrative Action/Management Functions	ID
Ability to configure lockout policy for unsuccessful authentication attempts, specifically: timeouts between attempts, limiting number of attempts during a time period	Security: **4739**
Ability to configure name/address of directory server to bind with	System: **3260**
Ability to configure name/address of audit/logging server to which to send audit/logging records	Security: **4947**
Ability to configure name/address of network time server	System: **37**
Ability to configure banner	Security: **4657** (ObjectValueName: **legalnoticecaption** and/or **legalnoticetext**)
Ability to connect/disconnect removable devices to/from a VM	Applications and Services Microsoft-Windows-Hyper-V-VMMS/Analytic: **12170** (Connect) Applications and Services Microsoft-Windows-Hyper-V-VMMS/Analytic: **12180** (Disconnect)
Ability to start a VM	Applications and Services Logs\Microsoft\Windows\Hyper-V-Worker\Admin: **18500**
Ability to stop/halt a VM	Applications and Services Logs\Microsoft\Windows\Hyper-V-Worker\Admin: **18502, 18516**
Ability to checkpoint a VM	Applications and Services Logs\Microsoft \Windows\Hyper-V-Worker\Admin: **18596**
Ability to suspend a VM	Applications and Services Logs\Microsoft \Windows\Hyper-V-Worker\Admin: **18510**
Ability to resume a VM	Applications and Services Logs\Microsoft\Windows\Hyper-V-Worker\Admin: **18518**

10.3 Format for Audit Event Records

Table 27 Format for Audit Event Records

Id	Log location	Message	Fields
1	Windows Logs->Setup	Initiating changes for package	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <Type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure>

Id	Log location	Message	Fields
2	Windows Logs->Setup	Package was successfully changed to the Installed state	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <subject identifier > **System->Level**: <Outcome as Success or Failure>
3	Windows Logs->Setup	Windows update could not be installed because … "The data is invalid"	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <Type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure>
11	Applications and Services Logs-> Microsoft->Windows->CAPI2-> Operational	For more details for this event, please refer to the "Details" section	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **UserData->CertGetCertificateChain->Result**: <Reason for failure of validation>
12	Applications and Services Logs\Microsoft\Windows\VHDMP	Handle for virtual disk <disk name> created successfully. VM ID = <VM ID, Type = <disk type>, Version = <version>, Flags = <flags>, AccessMask = <access mask>, WriteDepth = <write depth>, GetInfoOnly = <true/false>, ReadOnly = <true/false>, HandleContext = <GUID>, VirtualDisk = <GUID>.	**System->TimeCreated[SystemTime]:** <Date and time of event> **System->Provider[Name]:** <type of event> **System->Security[UserID]:** <Subject identifier > **System->Level:** <Outcome as Success or Failure> **UserData->VmId,VmName:** <VM identifier> **UserData->Device: <Virtual device identifier>**
16	Applications and Services Logs\Microsoft\Windows\VHDMP	Virtual disk object destroyed: <Virtual Disk GUID>	**System->TimeCreated[SystemTime]:** <Date and time of event> **System->Provider[Name]:** <type of event> **System->Security[UserID]:** <Subject identifier> **System->Level:** <Outcome as Success or Failure> **UserData->VmId,VmName:** <VM identifier> **UserData->Device:** <Virtual device identifier>
20	Windows Logs -> System	The last boot's success was <LastBootGood event data>.	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System-> Security[UserID]**: <subject identifier > **EventData->LastBootGood**: <Outcome as true or false indicating if the kernel-mode cryptographic self-tests and RNG initialization succeeded or failed>

Id	Log location	Message	Fields
37	Windows Logs -> System	The time provider NtpClient is currently receiving valid time data from <NTP server address>.	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->EventID,Level:** <Outcome as Success or Failure> **EventData->Data:** <Configuration change>
1100	Windows Logs->Security Subcategory: Security State Change	The event logging service has shut down	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Task**: <Type of event> **System->Keywords**: <Outcome as Success or Failure> **N/A**: <Subject identifier>
1104	Windows Logs->Security	The security log is now full.	**System->TimeCreated[SystemTime]: <Date and time of event>** **System->Provider[Name]: <type of event>** **System->Security[UserID]: <Subject identifier >** **System->EventID,Level: <Outcome as Success or Failure>**
1793	Microsoft-Windows-SChannel-Events/Perf	A TLS Security Context handle is being deleted	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System-> Security[UserID]**: <subject identifier > **System->Level**: <Outcome as Success or Failure> **EventData->ContextHandle:** <non-TOE endpoint>
3260	Windows Logs -> System	This computer has been successfully joined to domain <Domain Name>.	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Computer**: <subject identifier > **System->EventID,Level:** <Outcome as Success or Failure> **EventData->Data:** <Configuration change>
4608	Windows Logs->Security Subcategory: Security State Change	Startup of audit functions	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Task**: <type of event> **EventData -> SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure>
4624	Windows Logs -> Security Subcategory: Logon	An account was successfully logged on.	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Task**: <type of event> **EventData -> SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure> **EventData ->LogonType**: <type of logon (e.g. interactive)>

Id	Log location	Message	Fields
			EventData ->LogonID: <unique logon identification> **EventData ->TargetUserName**: <name of enabled account> **EventData ->TargetDomainName**: <domain of enabled account if applicable, otherwise computer> **EventData ->WorkstationName**: <name of computer user logged on> **EventData ->IpAddress**: <IP address of computer logged on>
4625	Windows Logs -> Security Subcategory: Logon	An account failed to log on.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData-> SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure> **EventData ->LogonType**: <type of logon (e.g. interactive)> **EventData ->LogonID**: <unique logon identification> **EventData ->TargetUserName**: <name of enabled account> **EventData ->TargetDomainName**: <domain of enabled account if applicable, otherwise computer> **EventData ->WorkstationName**: <name of computer user logged on> **EventData ->IpAddress**: <IP address of computer logged on>
4634	Windows Logs -> Security Subcategory: Logoff	An account was logged off.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData -> SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure>
4651	Windows Logs -> Security Subcategory: IPsec Main Mode	Ipsec main mode security association was established. A certificate was used for authentication.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **System->Keywords**: <Outcome as Success or Failure > **EventData->LocalMMPrincipalName**: <Subject identity > **EventData->RemoteMMPrincipalName:** <Remote User ID> **EventData->RemoteAddress**: <User ID, Remote source IP address>
4652	Windows Logs -> Security Subcategory: IPsec Main Mode	IPsec main mode negotiation failed	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData->FailureReason**: <Outcome as Success or Failure; reason for failure> **EventData->LocalAddress**: <Subject identity as IP address> **EventData->RemoteAddress**: < Remote source IP address >

Id	Log location	Message	Fields
4654	Windows Logs -> Security Subcategory: IPsec Main Mode	IPsec quick mode negotiation failed	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData->FailureReason**: <Outcome as Success or Failure; reason for failure> **EventData->LocalAddress**: <Subject identity as IP address> **EventData->RemoteAddress**: <User ID, Remote source IP address>
4655	Windows Logs -> Security Subcategory: IPsec Main Mode	IPsec main mode security association ended	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **System ->Keywords**: <Outcome as Success or Failure > **EventData->LocalAddress**: <Subject identity as IP address> **N/A**:<User ID> **EventData->RemoteAddress**: <Remote source IP address>
4656	Windows Logs->Security Subcategory: File System and Handle Manipulation	A handle to an object was requested.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData ->SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure> **EventData->ObjectName:** <Configuration change>
4657	Windows Logs->Security Subcategory: Registry	A registry value was modified.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData ->SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure> **EventData->ObjectName:** <Requested file>
4662	Windows Logs->Security Subcategory: Other Object Access Events	An operation was performed on an object.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData ->SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure> **EventData->ObjectName:** <Configuration change>
4663	Windows Logs->Security Subcategory: Registry	An attempt was made to access an object.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData ->SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure>

Id	Log location	Message	Fields
4719	Windows Logs->Security Subcategory: Audit Policy Change	System audit policy was changed.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData ->SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure> **EventData -> *:** <Configuration changes>
4720	Windows Logs->Security Subcategory: User Account Management	A user account was created.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <Type of event> **System->Keywords**: <Outcome as Success or Failure> **EventData->SubjectUserSid**: <Subject identifier>
4722	Windows Logs->Security Subcategory: User Account Management	A user account was enabled.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <Type of event> **System->Keywords**: <Outcome as Success or Failure> **EventData->SubjectUserSid**: <Subject identifier>
4725	Windows Logs->Security Subcategory: User Account Management	A user account was disabled.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <Type of event> **System->Keywords**: <Outcome as Success or Failure> **EventData->SubjectUserSid**: <Subject identifier>
4726	Windows Logs->Security Subcategory: User Account Management	A user account was deleted.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <Type of event> **System->Keywords**: <Outcome as Success or Failure> **EventData->SubjectUserSid**: <Subject identifier>
4738	Windows Logs->Security Subcategory: User Account Management	A user account was changed	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <Type of event> **System->Keywords**: <Outcome as Success or Failure> **EventData->SubjectUserSid**: <Subject identifier>
4739	Windows Logs -> Security Subcategory: Authentication Policy Change	Domain Policy was changed.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData -> SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure> **EventData -> *:** <Configuration changes>
4947	Windows Logs -> Security Subcategory: MPSSVC Rule-Level Policy Change	A change was made to the Windows Firewall exception list. A rule was modified.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData -> SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure>

Id	Log location	Message	Fields
			EventData -> *: <Configuration changes>
5447	Windows Logs -> Security Subcategory: Other Policy Change Events	A Windows Filtering Platform filter has been changed.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData -> SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure> **EventData -> *:** <Configuration changes>
5449	Windows Logs -> Security Subcategory: Filtering Platform Policy Change	A Windows Filtering Platform provider context has been changed.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData -> SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure> **EventData -> *:** <Configuration changes>
5451	Windows Logs -> Security Subcategory: IPsec Quick Mode	IPsec quick mode security association was established	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **System ->Keywords**: <Outcome as Success or Failure > **EventData->LocalAddress**: <Subject identity as IP address> **N/A**:<User ID> **EventData->RemoteAddress**: <Remote source IP address>
5452	Windows Logs -> Security Subcategory: IPsec Quick Mode	IPsec quick mode security association ended	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **System ->Keywords**: <Outcome as Success or Failure > **EventData->LocalAddress**: <Subject identity as IP address> **N/A**:<User ID> **EventData->RemoteAddress**: <Remote source IP address>
5858	Applications and Services Logs\Microsoft\Windows\WMI-Activity\Operational	Id = <Guid>; ClientMachine = <Machine Name>; User = <User Name>; ClientProcessId = <Process ID>; Component = <Component Name>; Operation = <Attempted Operation Details>; ResultCode = <Result Code>; PossibleCause = <Possible Cause>	**Source** = WMI-Activity **Task Category** = None **Computer** = <Machine Name> **Result Code** = 0x80041003, whenever the WMI filter is accessed without sufficient permission, e.g., a non-admin attempts an admin-only task.

Id	Log location	Message	Fields
12140	Applications and Services -> Microsoft -> Windows -> Hyper-V-Worker -> Admin (Source: Hyper-V-SynthStor)	Attachment <disk identifier> failed to open because of error: <Error Message> <ErrorCode>. <Virtual machine ID>	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **UserData->VmId,VmName**: <VM identifier> **UserData->String:** <Physical device identifier>
12170	Applications and Services Microsoft-Windows-Hyper-V-VMMS/Analytic	Virtual device <Device Id> added to Virtual machine <Virtual machine ID>	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **UserData->VmId,VmName**: <VM identifier> **UserData->Device**: <Virtual device identifier>
12180	Applications and Services Microsoft-Windows-Hyper-V-VMMS/Analytic	Virtual device <Device Id> removed from the virtual machine <Virtual machine ID>	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **UserData->VmId,VmName**: <VM identifier> **UserData->Device**: <Virtual device identifier>
12514	Applications and Services Microsoft-Windows-Hyper-V-VMMS/Admin	Found a certificate for server authentication. Remote access to virtual machines is now possible.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure>
12597	Windows Server 2012 R2: Applications and Services -> Microsoft -> Windows -> Hyper-V-SynthNic -> Admin Windows Server 2016: Applications and Services -> Microsoft -> Windows -> Hyper-V-Worker -> Admin (Source: Hyper-V-SynthNic)	<VM Name> Network Adapter <Virtual Switch ID> Connected to virtual network. <Virtual Machine ID>	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **UserData->VmId,VmName**: <VM identifier> **UserData->NicGuid,NicName**: <Networking component identifier>

Id	Log location	Message	Fields
12598	Applications and Services > Microsoft > Windows > Hyper-V-Worker > Admin (Source: Hyper-V-SynthNic)	<VM Name> Network Adapter <Virtual Switch ID> Disconnected from virtual network. <Virtual Machine ID>	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **UserData->VmId,VmName**: <VM identifier> **UserData->NicGuid,NicName**: <Networking component identifier>
12670	Windows Server 2012 R2: Applications and Services -> Microsoft -> Windows -> Hyper-V-SynthNic -> Admin Windows Server 2016: Applications and Services -> Microsoft -> Windows -> Hyper-V-Worker -> Admin (Source: Hyper-V-SynthNic)	<VM Name> failed to allocate resources while connecting to a virtual network: Insufficient system resources exist to complete the requested service. (<Error Code>) (Virtual Machine ID <Virtual Machine ID>). The Ethernet switch may not exist.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **UserData->VmId,VmName**: <VM identifier> **UserData->String:** <Networking component identifier>
13002	Applications and Services -> Microsoft -> Windows -> Hyper-V-VMMS/ -> Admin	A new virtual machine <VM name> was created. (Virtual machine ID <VM ID>)	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **System->EventID:** <Configuration change>
13003	Applications and Services -> Microsoft -> Windows -> Hyper-V-VMMS -> Admin	The virtual machine <VM Name> was deleted. (Virtual machine ID <VM ID>)	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **System->EventID:** <Configuration change>
18500	Applications and Services -> Microsoft -> Windows -> Hyper-V-Worker/Admin	<VM name> started successfully. (Virtual machine ID <VM ID>)	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **System->EventID:** <Configuration change>

Id	Log location	Message	Fields
18502	Applications and Services -> Microsoft -> Windows -> Hyper-V-Worker/Admin	<VM name> was turned off. (Virtual machine ID <VM ID>)	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **System->EventID:** <Configuration change>
18504	Applications and Services -> Microsoft -> Windows -> Hyper-V-Worker/Admin	<VM name> was shut down using the Shutdown Integration Component with parameters: Force = false, Reason = 'Shutdown initiated by <Username> using Hyper-V management tools.' (Virtual machine ID <VM ID>)	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **System->EventID:** <Configuration change>
18510	Applications and Services -> Microsoft -> Windows -> Hyper-V-Worker/Admin	<VM name> saved successfully. (Virtual machine ID <VM ID>)	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **System->EventID:** <Configuration change>
18516	Applications and Services > Microsoft > Windows > Hyper-V-Worker/Admin	<VM name> was paused. (Virtual machine ID <VM ID>)	**System->TimeCreated[SystemTime]:** <Date and time of event> **System->Provider[Name]:** <type of event> **System->Security[UserID]:** <Subject identifier> **System->Level:** <Outcome as Success or Failure> **System->EventID:** <Configuration change>
18596	Applications and Services -> Microsoft -> Windows -> Hyper-V-Worker/Admin	<VM name> was restored successfully. (Virtual machine ID <VM ID>)	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **System->EventID:** <Configuration change>
26000	Applications and Services -> Microsoft -> Windows -> Hyper-V-VMMS -> Networking	Switch created, name= <switch ID>, friendly name=<switch name>.	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **System->EventID:** <Configuration change>

Id	Log location	Message	Fields
26004	Applications and Services -> Microsoft -> Windows -> Hyper-V-VMMS -> Networking	Switch port created, switch name = <switch ID> switch friendly name = <switch name>, port name = <port ID>, port friendly name=<port name>.	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **System->EventID:** <Configuration change>
26012	Applications and Services -> Microsoft -> Windows -> Hyper-V-VMMS -> Networking	Internal miniport created, name = <miniport ID>, friendly name = <miniport name>, MAC = <MAC address>.	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **System->EventID:** <Configuration change>
26016	Applications and Services -> Microsoft -> Windows -> Hyper-V-VMMS -> Networking	External ethernet port <port ID> bound.	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **System->EventID:** <Configuration change>
26074	Applications and Services -> Microsoft -> Windows -> Hyper-V-VMMS -> Networking	Ethernet switch port connected (switch name = <switch name>, port name = <port name>, adapter GUID = <adapter ID>).	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure> **System->EventID:** <Configuration change>
36880	Windows Logs -> System	An TLS server handshake completed successfully. The negotiated cryptographic parameters are as follows:	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <subject identifier > **UserData->EventXML->TargetName**: <Non-TOE endpoint>
36888	Windows Logs -> System	A fatal alert was generated and sent to the remote endpoint. This may result in termination of the connection. The TLS protocol defined fatal error code is %1.	**System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <subject identifier > Windows Server 2016: **UserData->EventXML->TargetName**: <Non-TOE endpoint > **UserData->EventXML->AlertDesc:** < Reason for failure> **UserData->EventXML->ErrorState:** < Reason for failure >

Id	Log location	Message	Fields
			Windows Server 2012 R2: **EventData->AlertDesc:** < Reason for failure > **EventData->ErrorState:** < Reason for failure > The following are the possible error codes: **Description** — **Error Code Value** Unexpected message — 10 Bad record MAC — 20 Record overflow — 22 Decompression fail — 30 Handshake failure — 40 Illegal parameter — 47 Unknown CA — 48 Access denied — 49 Decode error — 50 Decrypt error — 51 Protocol version — 70 Insufficient security — 71 Internal error — 80 Unsupported extension — 110