Microsoft

Microsoft Windows

Common Criteria Evaluation

Microsoft Windows Server 2016

Microsoft Windows Server 2012 R2

Microsoft Windows 10

Hyper-V Security Target

Document Information	
Version Number	0.07
Updated On	November 17, 2017

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 and conformance claims; and,
- Describes the organization of the security target.

1.1 Security Target, TOE, and Common Criteria (CC) Identification

ST Title: Microsoft Hyper-V Security Target

ST Version: version 0.07, November 17, 2017

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

- Microsoft Windows Server 2016
- Microsoft Windows Server 2016 Datacenter edition
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2012 R2 Datacenter edition
- Microsoft Windows 10 Enterprise Edition (64-bit version)

The following security updates and patches must be applied to the above Windows products:

- All critical updates as of June 30, 2017

Platform Hardware Identification: The following hardware platforms and components were used during testing:

- Surface Book
- Dell OptiPlex 3040
- HP ProDesk 600 G2

TOE Guidance Identification: The following administrator, user, and configuration guides were evaluated as part of the TOE:

- *Common Criteria Supplemental Admin Guidance* along with all the documents referenced therein.

Evaluation Assurance: As specified in section 5.2.1 and specific Evaluation 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 4, September 2012.

1.2 CC Conformance Claims

This TOE and ST 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 Server Virtualization, version 1.1, September 14, 2015 (SV PP)
- CC Part 3 assurance requirements specified in section 5.2.1 and evaluation activities specified in Section 5.2.2

1.3 Conventions, Terminology, Acronyms

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

1.3.1 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.

1.3.2 Terminology

1.3.2.1 Virtualization Terminology

Hypervisor and Partitions: A hypervisor is a layer of software that sits just above the hardware and beneath one or more operating systems. Its primary job is to provide isolated execution environments called partitions. Each partition is provided with its own set of (physical or virtual) hardware resources (memory, devices, CPU cycles). The hypervisor is responsible for controlling and arbitrating access to the underlying hardware where necessary.

Guests: Software running within a non-root partition is referred to as a guest. A guest might consist of a full-featured operating system like Windows 10 or a small, special-purpose kernel. The hypervisor is "guest-agnostic".

Specialized Partitions: In most respects, all partitions are equal. However, some partitions may be granted special privileges or assigned specialized functions.

In the evaluated version of Hyper-V there is only one partition that has special privileges, which is called the root partition. The root partition acts as the default owner of all hardware resources. It is also typically in charge of power management, plug and play, and hardware failure events. The root partition is also responsible for creating and managing other partitions and assigning hardware resources. In some respects, it acts like the service processor on a machine with hardware partitioning facilities. To start Hyper-V, first the Windows instance that is supposed to run in the root partition is started. This results in the Hypervisor starting, which then creates the root partition and “moves” the running instance of Windows into the root partition and assigns all devices to this partition.

Virtualization and Emulation: The hypervisor provides support for hardware virtualization. Virtualization provides multiple logical instances of CPUs and other hardware resources. These logical instances are mapped onto physical hardware resources using a variety of techniques. One such technique is emulation, the simulation of a processor or device using software. While the hypervisor facilitates processor and device emulation, the architecture attempts to provide or facilitate alternatives to emulation for performance reasons.

Legacy Guests, Worker Processes and Enlightenment: The hypervisor provides support for legacy guests. A legacy guest is an operating system that has no knowledge of the fact that it is running within a virtualized environment. Legacy guests require substantial infrastructure including a system BIOS and a wide variety of emulated devices. This infrastructure is not provided directly by the hypervisor. A separate Hyper-V service, called a “worker process” provides this infrastructure to legacy guests. Worker processes execute in the root partition and receive specific intercepts (i.e. notifications that specific events have occurred within a guest). This support for guest-mode intercept handlers provides added flexibility and reduced complexity within the hypervisor. Note that a worker process exists for each guest partition.

Device emulation provides broad compatibility, but it results in poor performance. Other operational assumptions made by legacy guests also add to virtualization overhead. This virtualization performance overhead can be mitigated by enlightening a legacy operating system. An enlightened guest has knowledge of the fact that it is running within a virtualized environment and changes its behavior accordingly. Various degrees of enlightenment are possible. For example, a guest might use a specialized block device driver to talk to an idealized virtual disk using a fast communication channel between itself and the root partition. More extensive enlightenment involves modifying or extending a guest hardware abstraction layer (HAL) so it talks to an idealized “synthetic” interrupt controller or access model specific register (MSR) that do not exist in real hardware.

Snapshots: A snapshot is a collection of data about a partition and its current state that allows restarting the partition in the same state. A Hyper-V snapshot therefore includes all of the information and data that is required to roll back the status of a partition to the state when the snapshot was taken. Information that is collected when taken a snapshot includes:

- Partition configuration settings (the contents of the vmc file)
- Virtual network settings
- The current state of all virtual hard disks (VHDs) that are attached to the partition
- State information for the partition

Live Migration: The live migration function allows moving a running partition from one Windows computer running Hyper-V (the "source" to another Windows computer also running Hyper-V (the "destination"). Hyper-V will verify that both Windows instances satisfy pre-defined compatibility criteria which ensure that software executed in the partition see the same operational environment from the underlying platform, which consists of the hardware visible to the partition and the Hyper-V interfaces and functions. Live migration requires all virtual hard disks used by the partition to be migrated to reside on cluster shared volumes. Live migration is not possible for partitions that use virtual hard disks on media local to the "source" instance of the TOE.

Live migration ensures that the security properties of a partition are maintained throughout the migration process and on the destination system regardless of the software executed in the partition being migrated.

1.3.2.2 Common Terminology

The following terminology is used in Windows security targets:

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[1] and the disclosure and modification of data[2].
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, 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[3], reliable access to IT resources.
Compromise	Violation of a security policy.
Confidentiality	A security policy pertaining to disclosure of data.

[1] Hardware and software
[2] Stored or communicated
[3] According to a defined metric

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.
Cryptographic module security policy	A precise specification of the security rules under which a cryptographic module must operate.
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 Server 2012 [Standard] and Windows Server 2012 Datacenter.
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-enforcing	A term used to indicate that the entity (e.g., module, interface, subsystem) is related to the enforcement of the TOE security policies.
Security-supporting	A term used to indicate that the entity (e.g., module, interface, subsystem) is not security-enforcing; however, the entity's implementation must still preserve the security of the TSF.
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 functions. 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.

1.3.3 Acronyms

The acronyms used in this security target are specified ***in Appendix A: List of Abbreviations*****Table 19 Requirement to Security Function Correspondence**

Appendix A: List of Abbreviations.

1.4 ST Overview and Organization

The Windows Server 2016, Windows Server 2012 R2, and Windows 10 TOE provides the following security services:

- Audit
- Cryptographic support
- User data protection
- Identification and Authentication (I&A)
- Protection of the TOE Security Functions (TSF)
- TOE access/session control
- Trusted path/channel
- Security management

This security target contains the following additional sections:

- **TOE Description** (Section 2): Provides an overview of the TSF and boundary.
- **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 Server 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 TOE Description

The TOE includes hypervisor and virtualization subsystem, known as “Hyper-V” in the Microsoft Windows Server 2016 operating system, the Microsoft Windows Server 2012 R2 operating system, the Microsoft 10 operating system, supporting operating system services, and those applications necessary to manage, support and configure the operating system and virtualization subsystem.

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.

2.1 Product Types

Windows Server 2016, Windows Server 2012 R2, and Windows 10 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 Server 2016, Windows Server 2012 R2, and Windows 10, collectively referred to as Windows, expand 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.

2.2 Product Description

The TOE includes five product variants of Windows:

- Microsoft Windows Server 2016
- Microsoft Windows Server 2016 Enterprise
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2012 RS Datacenter
- Microsoft Windows 10 Enterprise

Built for workloads ranging from the department to the enterprise to the cloud, Windows Server 2016 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 10 is suited for business desktops, notebook, tablet, 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.

In terms of security, Windows Server 2016, Windows Server 2012 R2, and Windows 10 share the same security characteristics.

2.3 Security Environment and TOE Boundary

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

2.3.1 Logical Boundaries

The logical boundary of the TOE includes:

- 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).[4]
- 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.
- **Windows Explorer** for Windows 10, Windows Server 2016, and Windows Server 2012 R2 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 **Key Isolation Service** which protects secret and private keys.
- 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.

[4] AuthIP key exchange was not examined in the Common Criteria portion of this evaluation.

- 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.

2.3.2 Physical Boundaries

Physically, each TOE computer executes on Intel-based x64 computer architectures. Refer to section 1.1 for the specific list of hardware included in the evaluation.

A set of devices may be attached as part of the TOE:

- Display Monitors
- Fixed Disk Drives (including disk drives and solid state drives)
- Removable Disk Drives (including USB storage)
- Network Adaptor
- Keyboard
- Mouse
- Printer
- Audio Adaptor
- Optical (CD-ROM or DVD) Drive
- Smart Card Reader
- Trusted Platform Module (TPM) version 2.0

While this list of devices is larger than is needed to evaluate the requirements in the Protection Profile for Server Virtualization, it is the same set of devices as the General Purpose Operating System Protection Profile evaluation for Windows Server 2016, Windows Server 2012 RS2, and Windows 10. By using the same set of devices for both evaluations, consumers can gain assurance by using both core OS capabilities and OS virtualization in combination.

2.4 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 validated cryptographic functions that support encryption/decryption, cryptographic signatures, cryptographic hashing, cryptographic key agreement (which is not studied in this evaluation), 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,[5] 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.
    - **IPsec:** Windows implements IPsec to provide protected, authenticated, confidential, and tamper-proof networking between two peer computers.
    - **TLS:** Windows implements TLS 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.
- **Identification and Authentication**: In the context of this evaluation, Windows provides the ability to use, store, and protect X.509 certificates that are used for IPsec and TLS authenticates the administrator to the computer.
- **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.
- **Session Locking:** in the context of this evaluation Windows allows an authorized administrator to configure the system to display a logon banner before the logon dialog.
- **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.

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

- **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.

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 Server Virtualization ("SV PP").

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 SV PP.

Table 1 Threats Addressed by Hyper-V

Threat	Description
T.DATA_LEAKAGE	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 medical data to be made accessible to unauthorized entities.
T.UNAUTHORIZED_UPDATE	A malicious party attempts to supply the Administrator with an update to the product that may compromise the security features of the TOE.
T.UNAUTHORIZED_MODIFICATION	Malware running on the physical host 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	An administrator may unintentionally install or configure the TOE incorrectly, resulting in ineffective security mechanisms.
T.3P_SOFTWARE	Vulnerabilities in 3rd party software can lead to VMM compromise. Where possible, the VS should mitigate the results of potential vulnerabilities or malicious content in third-party code.
T.VMM_COMPROMISE	Failure of security mechanisms could lead to unauthorized intrusion into or modification of the VMM or bypass of the VMM altogether.
T.PLATFORM_COMPROMISE	The hosting of untrusted or malicious domains by the VS cannot be permitted to compromise the security and integrity of the platform on which the VS executes.
T.UNAUTHORIZED_ACCESS	A user may gain unauthorized access to the TOE data and TOE executable code. A malicious user, process, or external IT entity may masquerade as an authorized entity in order to gain unauthorized access to data or TOE resources. A malicious user, process, or external IT entity may misrepresent itself as the TOE to obtain identification and authentication data.
T.WEAK_CRYPTO	A threat of weak cryptography may arise if the VMM does not provide sufficient entropy to support security-related features that depend on entropy to implement cryptographic algorithms.

T.UNMANAGEABLE_NW	The Virtualization System itself is generally part of a larger enterprise network and must be updated and patched as a normal part of enterprise network operations. Such basic network hygiene is more difficult if the enterprise network is unmanageable.

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 for Hyper-V

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

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 Secure Usage Assumptions for Hyper-V

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.

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 Security Objectives for the TOE

Security Objective	Source
O.VM_ISOLATION	As basic functionality, the VMM must support a security policy that mandates no information transfer between VMs.
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 VMs or of the physical platform. The overall objective is to ensure the integrity of critical components of a Virtualization System.
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	The VS responsible ensuring that software running in Guest VMs us not interfered with by VMs from other domains.
O.MANAGEMENT_ACCESS	Management functions must be exercised only by authorized Administrators.
O.MANAGEABLE_NETWORK	The VS must support standards and protocols that help enhance manageability of the VS as an IT product.
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.

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 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.

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 Server Virtualization, Version 1.1, September 14, 2015, or the Common Criteria.

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]]***).
- **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

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))
	Extended: Random Bit Generation (FCS_RBG_EXT.1)
	Extended: Entropy for Virtual Machines (FCS_ENT_EXT.1)
	IPsec Protocol (FCS_IPSEC_EXT.1)
	TLS Client Protocol (FCS_TLSC_EXT.1)
	TLS Server Protocol (FCS_TLSS_EXT.1)
	HTTPS Protocol (FCS_HTTPS_EXT.1)
User Data Protection (FDP)	VM Separation (FDP_VMS_EXT.1)
	Physical Platform Resource Controls (FDP_PPR_EXT.1)
	Virtual Networking Components (FDP_VNC_EXT.1)
	Residual Information in Memory (FDP_RIP_EXT.1)
	Residual Information on Disk (FDP_RIP_EXT.2)
	Hardware-Based Isolation Mechanisms (FDP_HBI_EXT.1)
Identification & Authentication (FIA)	Extended: Password Management (FIA_PMG_EXT.1)
	Password-based Authentication Mechanism (FIA_UAU_EXT.2)
	Administrator Identification and Authentication (FIA_UIA_EXT.1)
	X.509 Certificate Validation (FIA_X509_EXT.1)
	X.509 Certificate Authentication (FIA_X509_EXT.2)
Security Management (FMT)	Restrictions on Security Roles (FMT_SMR.2)
	Default Data Sharing Configuration (FMT_MSA_EXT.1)
	Management of Security Functions Behavior (FMT_MOF_EXT.1)
	Separation of Management and Operational Networks (FMT_SMO_EXT.1)
Protection of the TSF (FPT)	Trusted Updates to the Virtualization System (FPT_TUD_EXT.1)
	Trusted Update based on Certificates (FPT_TUD_EXT.2)
	VMM Isolation from VMs (FPT_VIV_EXT.1)
	Hypercall Controls (FPT_HCL_EXT.1)
	Virtual Device Parameters (FPT_VDP_EXT.1)
	Hardware Assists (FPT_HAS_EXT.1)
	Device Driver Isolation (FPT_DDI_EXT.1)
	Execution Environment Mitigations (FPT_EEM_EXT.1)
	Removable Devices and Media (FPT_RDM_EXT.1)
	Non-Existence of Disconnected Virtual Devices (FPT_DVD_EXT.1)
TOE Access (FTA)	TOE Access Banners (FTA_TAB.1)
Trusted Path/Channels (FTP)	Trusted Path for Remote Administration (FTP_TRP.1)
	User Interface: I/O Focus (FTP_UIF_EXT.1)
	User Interface: Identification of VM (FTP_UIF_EXT.2)

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 listed in Table **7** ~~1~~.
d. [***Specifically defined auditable events listed in Table B1***].

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; and
e. Additional information defined in Table **7** ~~1~~.

Requirement	Auditable Events	Additional 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.
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.
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.
FCS_TLSC_EXT.1	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.1	Failure to establish a TLS Session. Establishment/Termination of a TLS session.	Reason for failure. Non-TOE endpoint of connection (IP address).

Requirement	Auditable Events	Additional Record Contents
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.
FDP_VMS_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_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.
FDP_RIP_EXT.1	None.	None.
FDP_RIP_EXT.2	None.	None.
FDP_HBI_EXT.1	None.	None.
FIA_PMG_EXT.1	None.	None.
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).
FIA_X509_EXT.1	Failure to validate a certificate.	Reason for failure.
FIA_X509_EXT.2	None.	None.
FMT_SMR.2	None.	None.
FMT_MSA_EXT.1	None.	None.
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.

Requirement	Auditable Events	Additional Record Contents
	VM Start/Stop/Suspend events Start and end of remote management session. Account created, modified, enabled, disabled, removed.	
FMT_SMO_EXT.1	None.	None.
FPT_TUD_EXT.1	Initiation of update. Failure of signature verification.	No additional information
FPT_TUD_EXT.2	None.	None.
FPT_VIV_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_VDP_EXT.1	None.	None.
FPT_HAS_EXT.1	None.	None.
FPT_EEM_EXT.1	None.	None.
FPT_RDM_EXT.1	Transfer of removable media or device between VMs.	None.
FPT_DVD_EXT.1	None.	None.
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.
FTP_UIF_EXT.1	None.	None.
FTP_UIF_EXT.2	None.	None.

Table 7 Auditable Events

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 unauthorized 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 implementing the [***IPsec***] protocol.
FAU_STG_EXT.1.2	The TSF shall [***overwrite previous audit records according to the following rule: [overwrite the oldest stored audit event]*** [6]***, [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 [

- ***RSA schemes using cryptographic key sizes of 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 of 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)[7]

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.

[6] 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.
[7] This protection profile requirement was modified as part of NIAP Technical Decision 89.

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 SV PP.

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

[

- ***AES Key Wrap (KW) (as defined in NIST SP 800-38F),***[8]
- ***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,***[9]
- ***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 key sizes, 256-bit key sizes***].

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 SV 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.

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 SV 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 of 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 SV 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-SHA512***] and cryptographic key sizes [**128 and 256 bits**] and message digest sizes [***160, 256, 384, 512 bits***] that meet the following: FIPS

[8] For Windows 10 and Windows Server 2016 only.

[9] For Windows 10 and Windows Server 2016 only.

Pub 198-1, "The Keyed-Hash Message Authentication Code, and FIPS Pub 180-4, “Secure Hash Standard.

5.1.2.8 Extended: 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 Extended: 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 [***virtual device interface,***[10] ***passthrough access to hardware entropy source***].

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***].

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 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 Security Administrator based on [***
    - ***length of time, where the time values can configured within [any time up to 24] hours;***

    ];
- ***IKEv2 SA lifetimes can be configured by an Security Administrator based on [selection:***
    - ***number of packets/number of bytes;***

[10] For Windows Server 2016 and Windows 10 only.

    - ***length of time, where the time values can 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 a Security Administrator based on [***
    - ***number of packets/number of 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 a Security Administrator based on [***
    - ***number of packets/number of 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 (2048-bit 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 establish a trusted channel only to peers with valid certificates.

FCS_IPSEC_EXT.1.15 The TSF shall ensure that IKEv1 Phase 1 exchanges use only main mode.

5.1.2.11 TLS Client Protocol (FCS_TLSC_EXT.1)

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

- Mandatory Ciphersuites:
    - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
- [***Optional Ciphersuites:***
    - ***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_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***

].

FCS_TLSC_EXT.1.2 The TSF shall verify that the presented identifier matches the reference identifier according to RFC 6125.

FCS_TLSC_EXT.1.3 The TSF shall establish a trusted channel only if the peer certificate is valid.

FCS_TLSC_EXT.1.4 The TSF shall support mutual authentication using X.509v3 certificates.

FCS_TLSC_EXT.1.5 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 (FCS_TLSS_EXT.1)

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

- Mandatory Ciphersuites:
    - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
- [***Optional Ciphersuites:***
    - ***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_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***

].

FCS_TLSS _EXT.1.2	The TSF shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0, and [***none***].[11]
FCS_ TLSS_EXT.1.3	The TSF shall generate key agreement parameters [***over NIST curves [secp256r1, secp384r1] and no other curves; Diffie-Hellman parameters of size 2048 bits and [3072 bits***]].[12]
FCS_TLSS_EXT.1.4	The TSF shall support mutual authentication of TLS clients using X.509v3 certificates.
FCS_TLSS_EXT.1.5	The TSF shall not establish a trusted channel if the peer certificate is invalid.
FCS_TLSS_EXT.1.6	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.

5.1.3 Virtualization (User Data Protection) (FDP)

5.1.3.1 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, 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.

5.1.3.2 Physical Platform Resource Controls (FDP_PPR_EXT.1)

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

[11] This protection profile requirement was modified as part of NIAP Technical Decision 156.

[12] The Windows TLS server can also require 4096 DHE keys but that is not covered by this evaluation.

and only those resources explicitly assigned by the Hyper-V administrator from FDP_PPR_EXT.1.1].

5.1.3.3 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.

5.1.3.4 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.

5.1.3.5 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.

5.1.3.6 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***].

5.1.4 Identification and Authentication (FIA)

5.1.4.1 Extended: 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 settable and support passwords of 15 characters or greater.

5.1.4.2 Password-based Authentication Mechanism (FIA_UAU_EXT.2)[13]

FIA_UAU_EXT.2.1	The TSF shall provide a local password-based authentication mechanism, [***[smart card, virtual smart card]***]to perform administrative user authentication.

5.1.4.3 Administrator Identification and Authentication (FIA_UIA_EXT.1)

FIA_UIA_EXT.1.1 The TSF shall allow the following actions prior to requiring the Administrator to initiate the identification and authentication process:

- [***display the advisory warning message in FTA_TAB.1***]

FIA_UIA_EXT.1.2 The TSF shall require each administrative user to be successfully identified and authenticated before allowing any other TSF-mediated actions on behalf of that administrative user.

[13] This protection profile requirement was added as part of NIAP Technical Decision 100.

5.1.4.4 X.509 Certificate Validation (FIA_X509_EXT.1)

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

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted certificate.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates.
- The TSF shall validate the revocation status of the certificate using [***the Online Certificate Status Protocol (OCSP) as specified in RFC 2560, a Certificate Revocation List (CRL) as specified in RFC 5759***].
- The TSF shall validate the extendedKeyUsage field according to the following rules:
  - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the extendedKeyUsage field.
  - OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (idkp 9 with OID 1.3.6.1.5.5.7.3.9) in the extendedKeyUsage field.

5.1.4.5 X.509 Certificate Authentication (FIA_X509_EXT.2)

FIA_X509_EXT.2.1 The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [***IPsec, TLS, HTTPS***], and [***code signing for system software updates, code signing for integrity verification***].

FIA_X509_EXT.2.2 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***, ***not accept the certificate***].

5.1.5 Security Management (FMT)

5.1.5.1 Restrictions on Security Roles (FMT_SMR.2)

FMT_SMR.2.1 The TSF shall maintain the roles:

- Administrator.
- User.

FMT_SMR.2.2 The TSF shall be able to associate users with roles.

FMT_SMR.2.3 The TSF shall ensure that the conditions

- Administrator role shall be able to administer the TOE locally;
- Administrator role shall be able to administer the TOE remotely;
- Administrator role shall be able to manage the audit capabilities of the TOE.

are satisfied.

5.1.5.2 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.

5.1.5.3 Management of Security Functions Behavior (FMT_MOF_EXT.1)

FMT_MOF_EXT.1.1 The TSF shall be capable of performing the following management functions controlled by authorized Administrators only:

1) Ability to administer the Virtualization System locally and remotely;
2) Ability to update the Virtualization System, and to verify the updates using [***digital signature***] capability prior to installing those updates;
3) Ability to configure password policy
   - Minimum password length,
   - Minimum password complexity,
   - Maximum password lifetime.
4) Ability to create, delete, and configure VMs;
5) Ability to set default initial VM configurations;
6) Ability to configure virtual networks including VM;
7) Ability to manage the audit system and audit data;
8) Ability to configure VM access to physical devices;
9) Ability to configure inter-VM data sharing;
10) Ability to enable/disable VM access to Hypercall functions;
11) Ability to configure removable media policy;
12) [
    - ***Ability to configure the list of TOE-provided services available before an entity is identified and authenticated, as specified in FIA_UIA_EXT.1;***
    - ***Ability to configure the cryptographic functionality***
    - ***Ability to enable/disable screen lock,***
    - ***Ability to configure screen lock inactivity timeout,***
    - ***Ability to configure remote connection inactivity timeout,***
    - ***Ability to configure lockout policy for unsuccessful authentication attempts through [limiting number of attempts during a time period],***
    - ***Ability to configure name/address of directory server to bind with,***
    - ***Ability to configure name/address of audit/logging server to which to send audit/logging records,***
    - ***Ability to configure name/address of network time server***
    - ***Ability to enable/disable password authentication***[14]
    - ***Ability to configure advisory warning message in banner, as described in FTA_TAB.1.***[15]

].

[14] This protection profile requirement was modified as part of NIAP Technical Decision 100.

[15] This protection profile requirement was modified as part of NIAP Technical Decision 101.

FMT_MOF_EXT.1.2	The TSF shall be capable of performing the following management functions controlled by [***authorized Administrators, unprivileged Users***[16]]: [• ***Ability to connect/disconnect removable devices to/from a VM,*** • ***Ability to start a VM,*** • ***Ability to checkpoint a VM,*** • ***Ability to suspend a VM,*** • ***No other capabilities.***]

5.1.5.4 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, TLS, TLS/HTTPS, IPsec***].

5.1.6 Protection of the TSF (FPT)

5.1.6.1 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 [***support 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.

5.1.6.2 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.

5.1.6.3 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.

5.1.6.4 Hypercall Controls (FPT_HCL_EXT.1)

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 [***enable***] access to Hypercall functions.
FPT_HCL_EXT.1.3	The TSF shall permit exceptions to the configuration of the following Hypercall interface functions: [**no exceptions**].

[16] In the context of the Windows operating system, the protection profile term "unprivileged user" is equivalent to a Windows "standard user".

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.

5.1.6.5 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.

5.1.6.6 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.

5.1.6.7 Device Driver Isolation (FPT_DDI_EXT.1)

FPT_DDI_EXT.1.1 The TSF shall ensure that device drivers for physical devices are isolated from the VMM.

5.1.6.8 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,

d) Heap corruption detection

e) [control flow guard[17]]

].

5.1.6.9 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.

].

5.1.6.10 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 hardware configuration.

[17] For Windows Server 2016 and Windows 10 only.

5.1.7 TOE Access (FTA)

5.1.7.1 TOE Access Banners (FTA_TAB.1)

FTA_TAB.1.1	Before establishing an administrator session, the TSF shall display an advisory warning message regarding unauthorized use of the TOE.

5.1.8 Trusted Path / Channels (FTP)

5.1.8.1 Trusted Path for Remote Administration (FTP_TRP.1)

FTP_TRP.1.1	The TSF shall use [***IPsec, TLS, TLS/HTTPS***] 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 endpoints and protection of the communicated data from disclosure and detection of modification of the communicated data.
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.

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

FTP_UIF_EXT.1.1	The VMM shall indicate to the user which VM is currently connected to [***keyboard, mouse***].

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

FTP_UIF_EXT.2.1	The TSF shall uniquely identify a VM's output display to a user.

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 Server Virtualization.

Requirement Class	Requirement Component
ASE: Security Target	ASE_INT.1: ST Introduction
	ASE_CCL.1: Conformance Claims
	ASE_OBJ.1: Security Objectives
	ASE_ECD.1: Extended Components Definition
	ASE_REQ.1: Stated Security Requirements
	ASE_TSS.1: TOE Summary Specification
ADV: Design	ADV_FSP.1: Basic Functional Specification
AGD: Guidance Documents	AGD_OPE.1: Operational User Guidance
	AGD_PRE.1: Preparative Procedures
ALC: Life-cycle Support	ALC_CMC.1: Labeling of the TOE
	ALC_CMS.1: TOE CM Coverage

	ALC_TSU_EXT.1: Timely Security Updates
ATE: Tests	ATE_IND.1: Independent Testing - Sample
AVA: Vulnerability Assessment	AVA_VAN.1: Vulnerability Survey

Table 8 TOE Security Assurance Requirements

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.

Application Note: The software to be described includes the operating systems of the application processor and the baseband processor, as well as any firmware and applications. The process description includes the TOE developer processes as well as any third-party (carrier) processes. The process description includes each deployment mechanism (e.g., over- the-air updates, per-carrier updates, downloaded updates).

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.

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

ALC_TSU_EXT.1.3C The description shall include the mechanisms publicly available for reporting security issues pertaining to the TOE.

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

5.2.2 Server Virtualization 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

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 will be performed as specified in the associated assurance activities for the particular trusted channel mechanism. 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 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

5.2.2.2.3 Cryptographic Key Destruction (FCS_CKM_EXT.4)[18]

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.

[18] This protection profile assurance activity was modified as part of NIAP Technical Decision 89.

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).

5.2.2.2.4 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 SV 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.

AES-CBC Tests

AES-CBC Known Answer Tests

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

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

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

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

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

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

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

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

AES-CCM Tests

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

128 bit and 256 bit keys

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

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

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 5tuples 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 216 bits, whichever is smaller.

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

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

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

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

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

128 and 256 bit key encryption keys (KEKs)

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

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

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

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

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

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

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 SV 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 pseudorandomly 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 Signature Algorithms (FCS_COP.1(SIGN))

Application Note: FCS_COP.1(SIGN) corresponds to FCS_COP.1(3) in the SV 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 SV 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 Extended: Random Bit Generation (FCS_RBG_EXT.1)

Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Annex E, 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 80090A).

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 Extended: 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 AES-GCM-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

The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented.

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).

Tests

Tests are performed in conjunction with the other IPsec evaluation activities.

- (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, AES-CBC-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 evaluator shall verify that the TSS describes how the DN in the certificate is compared to the expected DN.

Operational Guidance

The evaluator shall ensure that the operational guidance includes configuration of the expected DN for the connection.

Tests

The evaluator shall, if necessary, configure the expected DN according to the operational guidance. The evaluator shall send a peer certificate signed by a trusted CA with a DN that does not match an expected DN and verify that the TOE denies the connection.

FCS_IPSEC_EXT.1.15

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

If the 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

The evaluator shall configure the TOE/platform as indicated in the operational guidance, 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.

5.2.2.2.11 TLS Client Protocol (FCS_TLSC_EXT.1)

FCS_TLSC_EXT.1.1

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 include those listed for this component.

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 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 ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA certificate while using one of the ECDSA ciphersuites.) 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 ciphersuite and verify that the client denies the connection.

Test 5: The evaluator 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 ciphersuite) or that the server denies the client's Finished handshake message.
- Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.
- 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.
- 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.1.2

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:

- 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.
- 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.1.3

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.1.4
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.1.5
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 (FCS_TLSS_EXT.1)
FCS_TLSS_EXT.1.1
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.
- 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.
- Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.
- 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.
- Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

FCS_TLSS_EXT.1.2

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 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.1.3

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

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.1.4

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.

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.1.5

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.

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.1.6

(conditional) If the TOE implements mutual authentication, the evaluator shall verify that the TSS describes how the DN or SAN in the certificate is compared to the expected identifier.

Operational Guidance

(conditional) If the TOE implements mutual authentication, and if 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 VM Separation (FDP_VMS_EXT.1)[19]

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.
c. As an Administrator, enable inter-VM communications between the VMs on the second VM.
d. Test that communications can be passed through the inter-VM channel.
e. As an Administrator again, disable inter-VM communications between the two VMs.
f. Test that communications can no longer be passed through the channel.

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

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

The evaluator must ensure that the ST includes the following statement attesting that there are no other ways for data to be transferred between VMs other than those listed in FDP_VMS_EXT.1.1:

> 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

[19] This protection profile assurance activity was modified as part of NIAP Technical Decision 109.

authorized Administrator. There are no ~~known~~[20] 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.

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.

[20] This protection profile assurance activity was modified as part of NIAP Technical Decision 109.

- 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 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 : 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 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.

5.2.2.3.4 Residual Information in Memory (FDP_RIP_EXT.1)[21]

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.

[21] This protection profile assurance activity was modified as part of NIAP Technical Decision 108.

5.2.2.3.5 Residual Information on Disk (FDP_RIP_EXT.2)
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 perform the following tests:

1. The evaluator (as an unprivileged VM user) must create a new, large file (10MB) in the VM's file system. The test is to read each location in the file to ensure that every location contains a value of 0. This can be done using a custom tool or a binary file editor or viewer.
2. The evaluator (as VS Administrator) must create a virtual disk and connect it to a VM. As an unprivileged VM user, the evaluator must then create a large (10 MB) memory-mapped file on the virtual disk. The test is to read each location in the file to ensure that every location contains a value of 0. This can be done using a custom tool or a binary file editor or viewer.

5.2.2.3.6 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.

5.2.2.4 Identification and Authentication (FIA)

5.2.2.4.1 Extended: 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.

5.2.2.4.2 Password-based Authentication Mechanism (FIA_UAU_EXT.2)[22]
The protection profile does not specify any assurance activities for this requirement.

5.2.2.4.3 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

[22] This protection profile requirement was added as part of NIAP Technical Decision 100.

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.

The evaluator shall perform the following tests for each method by which administrators access the TOE (local and remote), as well as for each type of credential supported by the login method:

- Test 1: The evaluator shall use the operational guidance to configure the appropriate credential supported for the login method. For that credential/login method, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.
- Test 2: The evaluator shall configure the services allowed (if any) according to the operational guidance, and then determine the services available to a remote or local administrator. The evaluator shall determine that the list of services available is limited to those specified in the requirement.

5.2.2.4.4 X.509 Certificate Validation (FIA_X509_EXT.1)

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.

- Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function (application validation, trusted channel setup, or trusted software update) failing. 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 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 or OCSP is selected; if both 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 then attempts the test with a

certificate that will be revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.

- Test 4: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.
- Test 5: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension not set. The validation of the certificate path fails.
- Test 6: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.

5.2.2.4.5 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 Restrictions on Security Roles (FMT_SMR.2)

The evaluator shall examine the TSS to verify that they describe the administrator role and the powers granted to and limitations of the role. The evaluator shall review the operational guidance to ensure that it contains instructions for administering the TOE both locally and remotely, including any configuration that needs to be performed on the client for remote administration. In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this PP be tested; for instance, if the TOE can be administered through a local hardware interface; SSH; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

5.2.2.5.2 Default Data Sharing Configuration (FMT_MSA_EXT.1)

This requirement is met if FDP_VMS_EXT.1 is met.

5.2.2.5.3 Management of Security Functions Behavior (FMT_MOF_EXT.1)

FMT_MOF_EXT.1.1

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 the security management functions and actions can be executed only by authorized Administrators.

The evaluator must attempt to access each of the functions and policies in FMT_MOF_EXT.1.1 as a nonAdministrative user and verify that the attempts fail. In addition, the evaluator must attempt to perform the below-listed management functions. The requirement is met if all attempts fail.

Test 1. The evaluator must attempt to change a VS configuration setting.

Test 2: The evaluator must attempt to create, delete, and reconfigure a VM.

FMT_MOF_EXT.1.2

The evaluator shall examine the TSS and Operational Guidance to ensure that it describes which privileges are required for each security management function selected in FMT_MOF_EXT.1.2. The evaluator shall verify that the security management functions and actions can be executed by users or administrators with the identified privileges.

5.2.2.5.4 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 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)

5.2.2.6.2 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.

5.2.2.6.3 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 known design or implementation flaws that bypass or defeat VM isolation.

5.2.2.6.4 Hypercall Controls (FPT_HCL_EXT.1)
The evaluator shall examine the TSS or operational guidance to ensure it includes the documentation of the interface, including all possible functions available via the interface. 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. 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, 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.

5.2.2.6.5 Virtual Device Parameters (FPT_VDP_EXT.1)

The evaluator shall examine the TSS to ensure it documents all virtual device interfaces, including I/O ports, protocols, and data formats.

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.

5.2.2.6.6 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.

5.2.2.6.7 Device Driver Isolation (FPT_DDI_EXT.1)

The evaluator shall examine the TSS documentation to verify that it describes the mechanism used for device driver isolation.

5.2.2.6.8 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.

5.2.2.6.9 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.

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

The evaluator shall perform the following tests:

- Test 1: 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.

5.2.2.7 TOE Access (FTA)

5.2.2.7.1 TOE Access Banners (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 in can occur.

5.2.2.8 Trusted Path / Channels (FTP)

5.2.2.8.1 Trusted Path for Remote Administration (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.

5.2.2.8.2 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.

5.2.2.8.3 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.

6 TOE Summary Specification (TSS)

This chapter describes the Windows security functions which 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.

Unless otherwise noted in this section, all statements apply to Windows Server 2016, Windows Server 2012 R2 and Windows 10.

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 Path / Channels

Hyper-V provides separation of partitions, controls access of partitions to resources like virtual hard disks or virtual network adapter, controls the management of the virtualization subsystem and enforces a privilege-based management policy, allows auditing of security critical events, controls access of administrative users in the root partition to the TOE management functions, and enforces quota for processor time for partitions. Hyper-V also offers live migration of partitions from one instance of the TOE to another instance, provided the underlying platforms of both instances of the TOE satisfy the compatibility criteria required for live migration. Live migration will neither allow a partition to increase its privileges nor undermine the security of any of the two instances of the TOE involved in the migration.

Hyper-V can be used to consolidate several physical computers based on the Intel architecture onto one machine. Hyper-V enables the IT administrator to define partitions; each instantiation of the TOE has one dedicated partition, called the root partition, and a variable number of so called “guest partitions”. Resources accesses by guest partitions are virtualized by Hyper-V, i. e. Hyper-V performs a “translation” of the virtual resource accesses by a guest partition onto the real resources available to the TOE. Such resources include virtual CPUs, main memory, virtual hard disks, virtual network adapters, virtual CD/DVD drives or floppy disk drives as well as virtual video adapter and virtual mouse and keyboard. The root partition is used for support of the resource virtualization and for TOE management activities.

Each guest partition can take over the tasks of one physical server. Each guest will have its own operating system installed and is restricted by Hyper-V to the use of the resources that are assigned to the partition. The assignment of resources to partitions is performed by administrative roles defined in the root partition. Hyper-V allows separating each guest partition from others with a comparable degree as if they were executing on separate physical servers.

6.1 Audit

The TOE Audit security function performs:

- Audit Collection
- Audit Review
- 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:

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.

Table 9 Standard Fields in a Windows Audit Entry

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.[23] 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.

Appendix B lists the set of audit events and the format for each event record which is used in this evaluation.

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 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 Windows 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

[23] 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.

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 Mapping

The **Audit** function satisfies the following SFRs:

- **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
- **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 6.8.1 and 6.2.4.

6.2 Cryptographic Support

6.2.1 Cryptographic Algorithms and Operations

Cryptography API: Next Generation (CNG) API is designed to be extensible at many levels and agnostic to cryptographic algorithm suites. An important feature of CNG is its native implementation of the Suite B algorithms, including algorithms for AES (128, 192, 256 key sizes)[24], 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. The entropy pool is populated using the following values:

[24] Note that the 192-bit key size is not used in this evaluation.

- An initial entropy value from a seed file provided to the Windows OS Loader at boot time (512 bits of entropy).[25]
- 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).
- Random values gathered periodically by calling the RDRAND CPU instruction, (256 bits of entropy).

The main source of entropy in the system is the CPU cycle counter which tracks hardware interrupts. This is a sufficient health test; if the computer were not accumulating hardware and software interrupts every processor clock cycle it would not be running and therefore there would be no need for random bit generation. In the same manner, a failure of the TPM chip or processor would be a critical error that halts the computer. In addition, when the user follows 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.[26]

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

[25] 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.
[26] Running Windows in FIPS validated mode is required according to the administrative guidance.

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:

HMAC Algorithm	Hash function Used	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-SHA-256	SHA-256	512 bits	32 bytes	Same as HMAC-SHA-1
HMAC-SHA-384	SHA-384	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. The key size is variable.
HMAC-SHA-512	SHA-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.

Table 10 HMAC Characteristics

The HMAC function forms the basis for a FIPS Approved implementation of a password based key derivation function (PBKDF). Windows inputs the password as a text string without any optional padding or blocking into a HMAC 512 function. The hash functions supported by the Windows implementation of SP 800-132 are SHA-1, SHA-256, SHA-384 or SHA-512.

The TSF includes a key isolation service designed specifically to host secret and private keys in a protected process to mitigate tampering or access to sensitive key materials. The TSF performs a key error detection check on each transfer of key (internal and intermediate transfers). The TSF prevents archiving of expired (private) signature keys. The TSF destroys non-persistent cryptographic keys by a single direct overwrite consisting of zeros.

The following table describes the keys and secrets used for networking, signature verification, and TPM-based attestation; all keys originate within Windows unless stated otherwise. When non-persistent keys or secrets are no longer needed for a network session, 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 11 Types of Keys Used by Windows

Key (Storage Location)	Description
Symmetric encryption/decryption keys (non-persistent)	Keys used for AES (FIPS 197) encryption/decryption for IPsec ESP, TLS.
HMAC keys (non-persistent)	Keys used for HMAC-SHA1, HMAC-SHA256, HMAC-SHA384, and HMAC-SHA512 (FIPS 198-1) as part of IPsec
Asymmetric ECDSA Public Keys (non-persistent)	Keys used for the verification of ECDSA digital signatures (FIPS 186-4) for IPsec traffic and peer authentication.
Asymmetric ECDSA Private Keys (non-persistent)	Keys used for the calculation of ECDSA digital signatures (FIPS 186-4) for IPsec traffic and peer authentication.
Asymmetric RSA Public Keys (non-persistent for IPsec, TLS, persistent for update verification)	Keys used for the verification of RSA digital signatures (FIPS 186-4) for IPsec, TLS and signed product updates.
Asymmetric RSA Private Keys (non-persistent for IPsec, TLS, persistent for attestation)	Keys used for the calculation of RSA digital signatures (FIPS 186-4) for IPsec, TLS as well as TPM-based health attestations.[27] The key size is 2048 for IPsec and can be 2048 or 3072 bits for TLS and attestation.
Asymmetric DSA Private Keys (non-persistent)	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.
DH Private and Public values (non-persistent)	Private and public values used for Diffie-Hellman key establishment for TLS.
ECDH Private and Public values (non-persistent)	Private and public values used for EC Diffie-Hellman key establishment for TLS.

6.2.2 Cryptographic Algorithm Evaluation

Table 12 Windows Server 2016 Cryptographic Algorithm Standards and Evaluation Methods[28]

Cryptographic Operation	Standard	Requirement	Evaluation Method
Encryption/Decryption	FIPS 197 AES	FCS_COP.1(SYM)	NIST CAVP #4061, #4062, #4063, #4064, #4074
	NIST SP 800-38A CBC mode		NIST CAVP #4063, #4064, #4074

[27] The keys used for health attestation are stored in the TPM, which is a physical device that is part of the physical computer.

[28] See http://www.commoncriteriaportal.org/files/epfiles/Windows%2010%20AU%20and%20Server%202016%20GP%20OS%20Security%20Target%20-%20Public.pdf

	NIST SP 800-38C CCM mode		NIST CAVP #4061, #4064
	NIST SP 800-38A CTR mode		NIST CAVP #4064, #4074
	NIST SP 800-38E XTS mode		NIST CAVP #4064
	NIST SP 800-38F KW mode		NIST CAVP #4062
	NIST SP 800-38D GCM mode		NIST CAVP #4064
Digital signature (key generation)	FIPS 186-4 RSA	FCS_CKM.1	NIST CAVP #2195
Digital signature (generation and verification)	FIPS 186-4 RSA	FCS_COP.1(SIGN)	NIST CAVP #2192, #2193, #2194
Digital signature	FIPS 186-4 DSA	FCS_CKM.1	NIST CAVP #1098
Digital signature	FIPS 186-4 ECDSA	FCS_CKM.1, FCS_COP.1(SIGN)	NIST CAVP #911, #920
Hashing	FIPS 180-4 SHA-2	FCS_COP.1(HASH)	NIST CAVP #3346, #3347
Keyed-Hash Message Authentication Code	FIPS 198-1 HMAC	FCS_COP.1(HMAC)	NIST CAVP #2651, #2661
Random number generation	NIST SP 800-90A DRBG	FCS_RBG_EXT.1	NIST CAVP #1217, #1222
Key agreement	NIST SP 800-56A ECDH	FCS_CKM.2	NIST CAVP #92, #93
Key establishment	NIST SP 800-56B RSA	FCS_CKM.2	Vendor affirmed
IKEv1	NIST SP 800-56A	FCS_CKM.2, FCS_IPSEC_EXT.1	NIST CVL #886
IKEv2	NIST SP 800-56A	FCS_CKM.2, FCS_IPSEC_EXT.1	NIST CVL #886
TLS	NIST SP 800-56A	FCS_CKM.2, FCS_TLSC_EXT.1, FCS_TLSS_EXT.1	NIST CVL #886

Table 13 Windows Server 2012 R2 Cryptographic Algorithm Standards and Evaluation Methods[29]

Cryptographic Operation	Standard	Requirement	Evaluation Method
Encryption/Decryption	FIPS 197 AES	FCS_COP.1(SYM)	NIST CAVP #2832, #2848, #2853
	NIST SP 800-38A CBC mode		NIST CAVP #2832, #2853
	NIST SP 800-38C CCM mode		NIST CAVP #2832, #2848
	NIST SP 800-38A CTR mode		NIST CAVP #2832
	NIST SP 800-38D GCM mode		NIST CAVP #2832
Digital signature	FIPS 186-4 RSA	FCS_CKM.1	NIST CAVP #1487

[29] See http://www.commoncriteriaportal.org/files/epfiles/st_windows10.pdf.

(key generation)			
Digital signature (generation and verification)	FIPS 186-4 RSA	FCS_COP.1(SIGN)	NIST CAVP #1493, #1494, #1519
Digital signature	FIPS 186-4 DSA	FCS_CKM.1	NIST CAVP #855
Digital signature	FIPS 186-4 ECDSA	FCS_CKM.1, FCS_COP.1(SIGN)	NIST CAVP #505, #1263
Hashing	FIPS 180-4 SHA-2	FCS_COP.1(HASH)	NIST CAVP #2373, #2396
Keyed-Hash Message Authentication Code	FIPS 198-1 HMAC	FCS_COP.1(HMAC)	NIST CAVP #1773
Random number generation	NIST SP 800-90A_DRBG	FCS_RBG_EXT.1	NIST CAVP #489
Key agreement	NIST SP 800-56A ECDH	FCS_CKM.2	NIST CAVP #47
Key establishment	NIST SP 800-56B RSA	FCS_CKM.2	Vendor affirmed
IKEv1	NIST SP 800-56A	FCS_CKM.2, FCS_IPSEC_EXT.1	NIST CVL #323
IKEv2	NIST SP 800-56A	FCS_CKM.2, FCS_IPSEC_EXT.1	NIST CVL #323
TLS	NIST SP 800-56A	FCS_CKM.2, FCS_TLSC_EXT.1, FCS_TLSS_EXT.1	NIST CVL #323

Table 14 Windows 10 Creators Update Cryptographic Algorithm Standards and Evaluation Methods

Cryptographic Operation	Standard	Requirement	Evaluation Method
Encryption/Decryption	FIPS 197 AES	FCS_COP.1(SYM)	NIST CAVP #4624, #4625, #4626, #4627
	NIST SP 800-38A CBC mode		NIST CAVP #4624, #4627
	NIST SP 800-38C CCM mode		NIST CAVP #4624, #4625
	NIST SP 800-38A CTR mode		NIST CAVP #4624, 4627
	NIST SP 800-38E XTS mode		NIST CAVP #4624
	NIST SP 800-38F KW mode		NIST CAVP #4626
	NIST SP 800-38D GCM mode		NIST CAVP #4624
Digital signature (key generation)	FIPS 186-4 RSA	FCS_CKM.1	NIST CAVP #2521, #2522
Digital signature (generation and verification)	FIPS 186-4 RSA	FCS_COP.1(SIGN)	NIST CAVP #2521, #2522, #2523, #2524
Digital signature	FIPS 186-4 DSA	FCS_CKM.1	NIST CAVP #1223

Digital signature (key generation)	FIPS 186-4 ECDSA	FCS_CKM.1	NIST CAVP #1133, #1135, #1136
Digital signature (generation and verification)	FIPS 186-4 ECDSA	FCS_COP.1(SIGN)	NIST CAVP #1133, #1135
Hashing	FIPS 180-4 SHA-2	FCS_COP.1(HASH)	NIST CAVP #3790
Keyed-Hash Message Authentication Code	FIPS 198-1 HMAC	FCS_COP.1(HMAC)	NIST CAVP #3061, #3062
Random number generation	NIST SP 800-90A_DRBG	FCS_RBG_EXT.1	NIST CAVP #1555. 1556
Key agreement	NIST SP 800-56A ECDH	FCS_CKM.2	NIST CAVP #127, #128
Key establishment	NIST SP 800-56B RSA	FCS_CKM.2	Vendor affirmed
IKEv1	NIST SP 800-56A	FCS_CKM.2, FCS_IPSEC_EXT.1	NIST CVL #1278
IKEv2	NIST SP 800-56A	FCS_CKM.2, FCS_IPSEC_EXT.1	NIST CVL #1278
TLS	NIST SP 800-56A	FCS_CKM.2, FCS_TLSC_EXT.1, FCS_TLSS_EXT.1	NIST CVL #1278

6.2.3 TLS, HTTPS

Windows implements TLS to enable a trusted network path that is used for client and server authentication, as well as HTTPS/ HTTP/TLS.

The following table summarizes the TLS RFCs implemented in Windows:

RFC #	Name	How Used
2246	The TLS Protocol Version 1.0	Specifies requirements for TLS 1.0.
3268	Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)	Specifies additional ciphersuites implemented by Windows.
3546	Transport Layer Security (TLS) Extensions	Updates RFC 2246 with TLS 1.0 extensions 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.
SSL3	The SSL Protocol Version 3	Specifies requirements for SSL3.

Table 15 TLS RFCs Implemented by Windows

These protocols are described at:

- MS-TLSP Transport Layer Security (TLS) Profile
- RFC 2246 The TLS Protocol Version 1.0
- RFC 3268 -AES Ciphersuites for TLS
- RFC 3546 Transport Layer Security (TLS) Extensions
- RFC 4366 Transport Layer Security (TLS) Extensions
- RFC 4492 ECC Cipher Suites for TLS
- RFC 4681 TLS User Mapping Extension
- RFC 5246 - The Transport Layer Security (TLS) Protocol, Version 1.2
- RFC 5289 - TLS ECC Suites with SHA-256384 and AES GCM

The Cipher Suites in Schannel article describes the complete set of TLS cipher suites implemented in Windows (reference: http://msdn.microsoft.com/en-us/library/windows/desktop/aa374757(v=vs.85).aspx), of which the following are used in the evaluated configuration:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- 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_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_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_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289.

SSL versions 1.0, v2.0, v3.0 and TLS version 1.0 are disabled by default.

When negotiating a TLS 1.2 elliptic curve cipher suite, Windows will include automatically as part of the Client Hello message both its supported elliptic curves extension, i.e., secp256r1, secp384r1, and secp521r1 as well as signature algorithm, i.e., SHA256, SHA384, and SHA512. 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 the applicable extended key usages. 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. A certificate the 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.

6.2.4 IPsec

The Windows IPsec implementation 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 and covers Windows 8, Windows RT, and Server 2012.[30]

Windows implements both RFC 2409, Internet Key Exchange (IKEv1), and RFC 4306, Internet Key Exchange version 2, (IKEv2).[31] Windows IPsec supports both tunnel mode and transport mode and provides an option for NAT transversal (reference: section 7.5.5, IPsec Encapsulations).[32] 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.[33] These rules are described in Understanding Firewall Rules and section 4 of *Microsoft Windows Common Criteria Supplemental Admin Guidance for IPsec VPN Clients*. 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 Behavior).[34] 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[35] as authentication algorithms for key exchange as well as Diffie-Hellman Groups 14, 19, 24, and 20 (reference: section 6, Appendix A, Product Behavior).[36] The IPsec VPN client will propose a cryptosuite to the IPsec VPN server; if the server responds with a cryptosuite that the client

---

[30] Also available as [MS-WSO], *Windows System Overview*, page 43 for offline reading.
[31] [MS-IKEE], *Internet Key Exchange Protocol Extensions*, page 8.
[32] [MS-WSO], page 45.
[33] [MS-WPO], page 44.
[34] [MS-IKEE], pages 74 – 75.
[35] 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.
[36] *Ibid*.

supports, the client will use the server's proposed cryptosuite instead. 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.[37] 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, 224, or 384 bits for DH groups 14, 19, 24, and 20 respectively.[38] See the TSS section for **Cryptographic Algorithms and Operations** for the NIST CAVP validation numbers.

Windows implements peer authentication using 2048 bit RSA certificates,[39] or ECDSA certificates using the P-256 and P-384 curves for both IKEv1 and IKEv2.[40]

While Windows supports pre-shared IPsec keys, it is not recommended due to the potential use of weak pre-shared keys.[41] Windows simply uses the pre-shared key that was entered by the authorized administrator, there is no additional processing on the input data.

Windows operating systems do not implement the IKEv1 aggressive mode option during a Phase 1 key exchange.

Windows will validate certificates as described in section **6.4.1** by comparing the Common Name of the certificate presented by the VPN gateway to the expected values for the IP address or Fully Qualified Domain Name of the VPN gateway or the user FQDN.

6.2.4.1 RFC Summary

The following table summarizes the use of RFCs and Windows 10:

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.

[37] [MS-IKEE], page 51.
[38] http://technet.microsoft.com/en-us/library/cc962035.aspx.
[39] [MS-IKEE], page 73.
[40] http://technet.microsoft.com/en-us/library/905aa96a-4af7-44b0-8e8f-d2b6854a91e6.
[41] http://technet.microsoft.com/en-us/library/cc782582(v=WS.10).aspx.

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 of the Internet Key Exchange version 2 (IKEv2) Protocol	Certain IPsec cryptosuites implemented by Windows.
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.

Table 16 IPsec RFCs Implemented by Windows

6.2.5 SFR Mapping

The **Cryptographic Support** function satisfies the following SFRs:

FCS_CKM.1, FCS_CKM.2: See

Table 13 Windows Server 2012 R2 Cryptographic Algorithm Standards and Evaluation Methods

- ,
- **Table** 14 Windows 10 Creators Update Cryptographic Algorithm Standards and Evaluation Methods , **Error! Reference source not found.**.
- **FCS_CKM_EXT.4**: Windows overwrites critical cryptographic parameters immediately after that data is no longer needed

FCS_COP.1(SYM), **FCS_COP.1(HASH)**, **FCS_COP.1(SIGN), FCS_COP.1(HMAC), FCS_RBG_EXT.1**: See

Table 13 Windows Server 2012 R2 Cryptographic Algorithm Standards and Evaluation Methods

- ,
- **Table** 14 Windows 10 Creators Update Cryptographic Algorithm Standards and Evaluation Methods , **Error! Reference source not found.**.
- **FCS_ENT_EXT.1**: The root partition provides entropy to virtual machines executing in child partitions.
- **FCS_IPSEC_EXT.1**: Windows provides an IPsec implementation as described above in **6.2.4**
- **FCS_TLSC_EXT.1, FCS_TLSS_EXT.1, FCS_HTTPS_EXT.1**: Windows implements a TLS 1.1 and 1.2 client and server to provide mutual authentication, confidentiality and integrity to upper-layer protocols such as HTTP.

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 Second-Level Address Translation (SLAT), hardware-enforced support for Data Execution Protection (DEP) and virtualization support from the underlying UEFI firmware.[42]

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 Windows Server 2016 and Windows 10 Hypervisor Top-Level Functional Specification and Windows Server 2012 R2 Hypervisor Top-Level Functional Specification describes the externally visible behavior of the hypervisor.

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.

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 vTPM. Furthermore, the Hyper-V administrator can grant "direct" access to a physical hard disk and removable storage.

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

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.

"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~~[43] 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." [44]

"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."[45]

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.

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 accuracy of these claims, and there are no known design or implementation flaws that bypass or defeat the security of the virtual device interfaces."[46]

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.

[43] This protection profile assurance activity was modified as part of NIAP Technical Decision 109.
[44] Protection Profile for Server Virtualization, page 38.
[45] Protection Profile for Server Virtualization, page 54.
[46] Protection Profile for Server Virtualization, page 56.

“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.”[47]

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.

6.3.6 SFR Mapping

The **Virtualization (User Data Protection)** function satisfies the following SFRs:

- **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_PPR_EXT.1**: The physical hard disk and removable media is the only non-virtualized resource that a guest virtual machine can access.
- **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.
- **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.

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 that must be provided to the TSF.

6.4.1 X.509 Certificate Validation

Every Windows component that uses X.509 certificates is responsible for performing certificate validation, however all components use a common subcomponent, which validates certificates as described in RFC 5280 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.

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, however it will inform the user and seek their consent before establishing a HTTPS web browsing session. Certification validation for updates to

[47] Protection Profile for Server Virtualization, page 40.

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 Updates.**

6.4.2 SFR Mapping

The **Identification and Authentication** function satisfies the following SFRs:

- **FIA_PMG_EXT.1**: Windows devices support logon passwords at least 14 characters in length to as long as 127 characters. Windows logon passwords can be composed from uppercase characters, lowercase characters, digits, and special characters to be used in passwords.
- **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_UAU_EXT.2**: 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.
- **FIA_X509_EXT.1**: Windows validates X.509 certificates according to RFC 5280 and provides OCSP and CRL services for applications to check certificate revocation status.
- **FIA_X509_EXT.2**: Windows uses X.509 certificates for EAP-TLS exchanges, TLS, DTLS, HTTPS, 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 **Security Management (FMT)**, the following table maps which activities can be done by a standard Windows user or an administrator, which may be logged into the computer locally or from a network connection. A checkmark indicates which entity can invoke the management function. 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.

Management Function	Local Administrator	Hyper-V Administrator	Standard User
Ability to administer the Virtualization System locally and remotely	√		
Ability to update the Virtualization System, and to verify the updates using a digital signature capability prior to installing those updates	√		
Ability to configure password policy	√		
Minimum password length	√		
Minimum password complexity	√		
Maximum password lifetime	√		
Ability to create, delete, and configure VMs	√		
Ability to set default initial VM configurations	√		
Ability to configure virtual networks including VM	√		
Ability to manage the audit system and audit data	√		
Ability to configure VM access to physical devices	√		

Ability to configure inter-VM data sharing	√		
Ability to enable/disable VM access to Hypercall functions	√		
Ability to configure removable media policy	√		
Ability to configure the list of services available before an entity is identified and authenticated	√		
Ability to configure the cryptographic functionality	√		
Ability to enable/disable screen lock	√		
Ability to configure screen lock inactivity timeout	√		√
Ability to configure remote connection inactivity timeout	√		
Ability to configure lockout policy for unsuccessful authentication attempts by limiting number of attempts during a time period	√		
Ability to configure name/address of directory server to bind with	√		
Ability to configure name/address of audit/logging server to which to send audit/logging records	√		
Ability to configure name/address of network time server	√		
Ability to enable/disable password authentication	√		
Ability to configure the logon banner	√		
Ability to connect/disconnect removable devices to/from a VM	√		
Ability to start a VM	√	√	
Ability to checkpoint a VM	√	√	
Ability to suspend a VM	√	√	

Table 17 SV PP Management Functions

6.5.1 SFR Mapping

The **Security Management** function satisfies the following SFRs:

- **FMT_SMR.2**: In the context of this evaluation Windows has a local administrator role which manages the virtualization system in the root partition and a user role which operates the guest virtual machines in each child partition.
- **FMT_MSA_EXT.1**: Data can be shared between guest virtual machines only through virtual networking.
- **FMT_MOF_EXT.1**: Windows provides the management functions that are described by FMT_MOF_EXT.1.1.
- **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 (Windows 10 only)

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.

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.

6.6.2 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 and user-mode applications implement Address Space Layout Randomization (ASLR) in order to load executable code at unpredictable base addresses. 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. [48]

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 implement stack buffer overrun protection by being complied with the /GS option, which is used for all Windows binaries.

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.

6.6.3 Windows Updates

Updates to Windows are delivered as Microsoft Update Standalone Package files (.msu files) and are signed by Microsoft with two digital signatures, a SHA1 signature for legacy applications and a SHA256 signature for modern applications. The RSA SHA256 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.

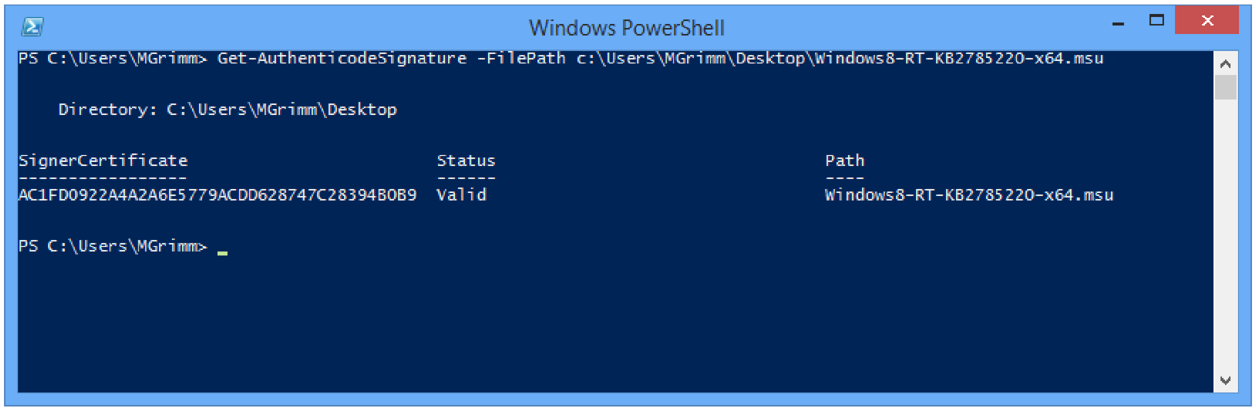
The Windows operating system will check that the certificate is valid and has not been revoked using a standard PKI CRL. Once the Trusted Installer determines that the package is valid, it will update Windows; otherwise the installation will abort and there will be an error message in the event log. 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.

[48] The PRNG is seeded by the TPM RBG, the RDRAND instruction and other sources.

Updates to Windows are delivered through the Windows Update capability, which is enabled by default, or the user can go to http://www.microsoft.com/security/default.aspx 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:



```
PS C:\Users\MGrimm> Get-AuthenticodeSignature -FilePath c:\Users\MGrimm\Desktop\Windows8-RT-KB2785220-x64.msu


    Directory: C:\Users\MGrimm\Desktop


SignerCertificate                         Status                                      Path
-----------------                         ------                                      ----
AC1FD0922A4A2A6E5779ACDD628747C28394B0B9  Valid                                       Windows8-RT-KB2785220-x64.msu


PS C:\Users\MGrimm> _
```

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.

6.6.4 SFR Mapping

The **Protection of the TSF** function satisfies the following SFRs:

- **FPT_TUD_EXT.1, FPT_TUD_EXT.2:** Windows provides a means to identify the current version of the Windows software and the hardware model. Windows has an update mechanism to deliver updated binaries and a means for a user to confirm that the digital signatures, which ensure the integrity of the update.
- **FPT_EEM_EXT.1**: Windows implements a collection of runtime mechanisms to prevent and mitigate harm from implementation weaknesses in Windows and other programs executing on the computer.
- **FPT_VIV_EXT.1**: The Hypervisor ensures that virtual machines executing in child partitions can not interfere with each other nor the operating system in the root partition.
- **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_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_HAS_EXT.1**: The Hypervisor will leverage VMX or VMT machine instructions, Second Level Address Translation (SLAT), and IOMMU hardware support to provide isolation between partitions.
- **FPT_DDI_EXT.1**: Windows isolates kernel-mode device drivers in guest VM from the user-mode Virtual Machine Manager service by running the guest VM in a different partition than the Windows instance running in the root partition.
- **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 and USB 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_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.

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 Mapping

The **TOE Access** function satisfies the following SFRs:

- **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 Path / Channels

6.8.1 Network Trusted Path

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

- Using IPsec for remote management of Hyper-V and the Windows OS.
- Using TLS (HTTPS) for certificate enrollment; CRL checking; authentication to network resources such as web (HTTPS) and directory (LDAP-S) servers.

Administration tools such as the Hyper-V Manager, PowerShell, and Remote Desktop (RDP) rely on Ipsec for secure networking.

In order to establish a trusted channel, these communications are protected as described above in sections 6.2.3 and 6.2.4.

6.8.2 User Interface Trusted Path

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.

6.8.3 SFR Mapping

The **Trusted Path / Channels** function satisfies the following SFRs:

- **FTP_TRP.1:** Windows provides several trusted network channels that protect data in transit from disclosure, provide data integrity, and endpoint identification that is used by IPsec, TLS, and HTTPS.
- **FTP_UIF_EXT.1**: The Hyper-V client user interfaces indicates which virtual machine is connected to the physical keyboard and mouse.
- **FTP_UIF_EXT.2**: 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://technet.microsoft.com/en-us/security/ff852094), 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://technet.microsoft.com/en-us/security/dn436305. If the sender wishes to send secure email, there is a public PGP key for S/MIME at https://technet.microsoft.com/en-us/security/dn606155.aspx. Security updates for Microsoft products – operating system, firmware, and applications – are delivered as described in section 6.6.3.

7 Protection Profile Conformance Claim

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

7.1 Rationale for Conformance to Protection Profile

This Security Target is in strict compliance with the Protection Profile for Server Virtualization, version 1.1, September 14, 2015 ("SV PP").

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.

The requirements in the protection profile are assumed to represent a complete set of requirements that serve to address any interdependencies. Given that all of the functional requirements in the protection profile have been copied into this security target, the dependency analysis for those requirements is not reproduced here.

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 18 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.

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

PP Requirement	ST Requirement	Operation and Rationale
FDP_PPR_EXT.1	FDP_PPR_EXT.1	A selection and assignment which are allowed by the PP.
FDP_VNC_EXT.1	FDP_VNC_EXT.1	Copied from the PP without changes.
FDP_RIP_EXT.1	FDP_RIP_EXT.1	Copied from the PP without changes.
FDP_RIP_EXT.2	FDP_RIP_EXT.2	Copied from the PP without changes.
FDP_HBI_EXT.1	FDP_HBI_EXT.1	Two selections which are allowed by the PP.
FIA_PMG_EXT.1	FIA_PMG_EXT.1	Copied from the PP without changes.
FIA_UAU_EXT.2	FIA_UAU_EXT.2	A selection and assignment which are allowed by the PP.
FIA_UIA_EXT.1	FIA_UIA_EXT.1	A selection which is allowed by the PP.
FIA_X509_EXT.1	FIA_X509_EXT.1	A selection which is allowed by the PP.
FIA_X509_EXT.2	FIA_X509_EXT.2	Three selections which are allowed by the PP.
FMT_SMR.2	FMT_SMR.2	Copied from the PP without changes.
FMT_MSA_EXT.1	FMT_MSA_EXT.1	A selection which is allowed by the PP.
FMT_MOF_EXT.1	FMT_MOF_EXT.1	Four selections which are allowed by the PP.
FMT_SMO_EXT.1	FMT_SMO_EXT.1	A selection which is allowed by the PP.
FPT_TUD_EXT.1	FPT_TUD_EXT.1	Two selections which are allowed by the PP.
FPT_TUD_EXT.2	FPT_TUD_EXT.2	Copied from the PP without changes.
FPT_VIV_EXT.1	FPT_VIV_EXT.1	Copied from the PP without changes.
FPT_HCL_EXT.1	FPT_HCL_EXT.1	An assignment and a selection which is allowed by the PP.
FPT_VDP_EXT.1	FPT_VDP_EXT.1	Copied from the PP without changes.
FPT_HAS_EXT.1	FPT_HAS_EXT.1	Two assignments which are allowed by the PP.
FPT_DDI_EXT.1	FPT_DDI_EXT.1	Copied from the PP without changes.
FPT_EEM_EXT.1	FPT_EEM_EXT.1	A selection which is allowed by the PP.
FPT_RDM_EXT.1	FPT_RDM_EXT.1	An assignment and a selection which is allowed by the PP.
FPT_DVD_EXT.1	FPT_DVD_EXT.1	Copied from the PP without changes.
FTA_TAB.1	FTA_TAB.1	Copied from the PP without changes.
FTP_TRP.1	FTP_TRP.1	A selection which is allowed by the PP.
FTP_UIF_EXT.1	FTP_UIF_EXT.1	A selection which is allowed by the PP.
FTP_UIF_EXT.2	FTP_UIF_EXT.2	Copied from the PP without changes.

Table 18 Rationale for Operations

8.2 Security Assurance Requirements

The statement of security assurance requirements (SARs) found in section **Error! Reference source not found. Error! Reference source not found.**, is in strict conformance with the Protection Profile for Server 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 19 Requirement to Security Function Correspondence**

. 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.

Requirement	Audit	Cryptographic Protection	User Data Protection	I & A	Security Management	TSF Protection	Resource Utilization	TOE Access	Trusted Path / Channel
FAU_GEN.1	X								
FAU_SAR.1	X								
FAU_STG.1	X								
FAU_STG_EXT.1	X								
FCS_CKM.1		X							
FCS_CKM.2		X							
FCS_CKM_EXT.4		X							
FCS_COP.1SYM		X							
FCS_COP.1HASH		X							
FCS_COP.1SIGN		X							
FCS_COP.1HMAC		X							
FCS_RBG_EXT.1		X							
FCS_ENT_EXT.1		X							
FCS_IPSEC_EXT.1		X							
FCS_TLSC_EXT.1		X							

Requirement	Audit	Cryptographic Protection	User Data Protection	I & A	Security Management	TSF Protection	Resource Utilization	TOE Access	Trusted Path / Channel
FCS_TLSS_EXT.1		X							
FCS_HTTPS_EXT.1		X							
FDP_VMS_EXT.1			X						
FDP_PPR_EXT.1			X						
FDP_VNC_EXT.1			X						
FDP_RIP_EXT.1			X						
FDP_RIP_EXT.2			X						
FDP_HBI_EXT.1			X						
FIA_PMG_EXT.1				X					
FIA_UAU_EXT.2				X					
FIA_UIA_EXT.1				X					
FIA_X509_EXT.1				X					
FIA_X509_EXT.2				X					
FMT_SMR.2					X				
FMT_MSA_EXT.1					X				
FMT_MOF_EXT.1					X				
FMT_SMO_EXT.1					X				
FPT_TUD_EXT.1						X			
FPT_TUD_EXT.2						X			
FPT_VIV_EXT.1						X			
FPT_HCL_EXT.1						X			
FPT_VDP_EXT.1						X			
FPT_HAS_EXT.1						X			
FPT_EEM_EXT.1						X			
FPT_DDI_EXT.1						X			
FPT_RDM_EXT.1						X			
FPT_DVD_EXT.1						X			
FTA_TAB.1								X	
FTP_TRP.1									X
FTP_UIF_EXT.1									X
FTP_UIF_EXT.2									X

Table 19 Requirement to Security Function Correspondence

Appendix A: List of Abbreviations

Abbreviation	Meaning
3DES	Triple DES
ACE	Access Control Entry
ACL	Access Control List
ACP	Access Control Policy
AD	Active Directory
ADAM	Active Directory Application Mode
AES	Advanced Encryption Standard
AGD	Administrator Guidance Document
AH	Authentication Header
ALPC	Advanced Local Process Communication
ANSI	American National Standards Institute
API	Application Programming Interface
APIC	Advanced Programmable Interrupt Controller
BTG	BitLocker To Go
CA	Certificate Authority
CBAC	Claims Basic Access Control, see DYN
CBC	Cipher Block Chaining
CC	Common Criteria
CD-ROM	Compact Disk Read Only Memory
CIFS	Common Internet File System
CIMCPP	Certificate Issuing and Management Components For Basic Robustness Environments Protection Profile, Version 1.0, April 27, 2009
CM	Configuration Management; Control Management
COM	Component Object Model
CP	Content Provider
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
DES	Data Encryption Standard
DH	Diffie-Hellman
DHCP	Dynamic Host Configuration Protocol
DFS	Distributed File System
DMA	Direct Memory Access
DNS	Domain Name System
DS	Directory Service
DSA	Digital Signature Algorithm

DYN	Dynamic Access Control
EAL	Evaluation Assurance Level
ECB	Electronic Code Book
EFS	Encrypting File System
ESP	Encapsulating Security Protocol
FEK	File Encryption Key
FIPS	Federal Information Processing Standard
FRS	File Replication Service
FSMO	Flexible Single Master Operation
FTP	File Transfer Protocol
FVE	Full Volume Encryption
GB	Gigabyte
GC	Global Catalog
GHz	Gigahertz
GPC	Group Policy Container
GPO	Group Policy Object
GPOSPP	US Government Protection Profile for General-Purpose Operating System in a Networked Environment
GPT	Group Policy Template
GPT	GUID Partition Table
GUI	Graphical User Interface
GUID	Globally Unique Identifiers
HTTP	Hypertext Transfer Protocol
HTTPS	Secure HTTP
I/O	Input / Output
I&A	Identification and Authentication
IA	Information Assurance
ICF	Internet Connection Firewall
ICMP	Internet Control Message Protocol
ICS	Internet Connection Sharing
ID	Identification
IDE	Integrated Drive Electronics
IETF	Internet Engineering Task Force
IFS	Installable File System
IIS	Internet Information Services
IKE	Internet Key Exchange
IOMMU	Input Output Memory Management Unit
IP	Internet Protocol
IPv4	IP Version 4
IPv6	IP Version 6
IPC	Inter-process Communication
IPI	Inter-process Interrupt
IPSec	IP Security
ISAPI	Internet Server API
IT	Information Technology
KDC	Key Distribution Center

LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LPC	Local Procedure Call
LSA	Local Security Authority
LSASS	LSA Subsystem Service
LUA	Least-privilege User Account
MAC	Message Authentication Code
MB	Megabyte
MMC	Microsoft Management Console
MSR	Model Specific Register
NAC	(Cisco) Network Admission Control
NAP	Network Access Protection
NAT	Network Address Translation
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
NLB	Network Load Balancing
NMI	Non-maskable Interrupt
NTFS	New Technology File System
NTLM	New Technology LAN Manager
OS	Operating System
PAE	Physical Address Extension
PC/SC	Personal Computer/Smart Card
PIN	Personal Identification Number
PKCS	Public Key Certificate Standard
PKI	Public Key Infrastructure
PP	Protection Profile
RADIUS	Remote Authentication Dial In Service
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RAS	Remote Access Service
RC4	Rivest’s Cipher 4
RID	Relative Identifier
RNG	Random Number Generator
RPC	Remote Procedure Call
RSA	Rivest, Shamir and Adleman
RSASSA	RSA Signature Scheme with Appendix
SA	Security Association
SACL	System Access Control List
SAM	Security Assurance Measure
SAML	Security Assertion Markup Language
SAR	Security Assurance Requirement
SAS	Secure Attention Sequence
SD	Security Descriptor
SHA	Secure Hash Algorithm
SID	Security Identifier
SIP	Session Initiation Protocol

SIPI	Startup IPI
SF	Security Functions
SFP	Security Functional Policy
SFR	Security Functional Requirement
SMB	Server Message Block
SMI	System Management Interrupt
SMTP	Simple Mail Transport Protocol
SP	Service Pack
SPI	Security Parameters Index
SPI	Stateful Packet Inspection
SRM	Security Reference Monitor
SSL	Secure Sockets Layer
SSP	Security Support Providers
SSPI	Security Support Provider Interface
ST	Security Target
SYSVOL	System Volume
TCP	Transmission Control Protocol
TDI	Transport Driver Interface
TLS	Transport Layer Security
TOE	Target of Evaluation
TPM	Trusted Platform Module
TSC	TOE Scope of Control
TSF	TOE Security Functions
TSS	TOE Summary Specification
UART	Universal Asynchronous Receiver / Transmitter
UI	User Interface
UID	User Identifier
UNC	Universal Naming Convention
US	United States
UPN	User Principal Name
URL	Uniform Resource Locator
USB	Universal Serial Bus
USN	Update Sequence Number
v5	Version 5
VDS	Virtual Disk Service
VPN	Virtual Private Network
VSS	Volume Shadow Copy Service
WAN	Wide Area Network
WCF	Windows Communications Framework
WebDAV	Web Document Authoring and Versioning
WebSSO	Web Single Sign On
WDM	Windows Driver Model
WIF	Windows Identity Framework
WMI	Windows Management Instrumentation
WSC	Windows Security Center
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

Appendix B: Audit Events for the Server Virtualization Protection Profile

Table 20 Audit Events for SV PP Functional Requirements

SFR	Auditable Events	Additional Audit Record Contents	Log: Event Id
FAU_GEN.1	Startup and shutdown of audit functions		Windows Logs\Security: **1100**, **4608**
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	**N/A:** not applicable based on the ST selecting overwriting oldest events with newest events when log fills
FCS_RBG_EXT.1	Failure of the randomization process.	No additional information.	Windows Logs -> 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.	Windows Server 2012 R2: Applications and Services Logs\Microsoft\Windows\Hyper-V-SynthStor\Admin: **12140** (Failure, Policy Violation) Windows 10 Enterprise / Windows Server 2016: Applications and Services Logs\Microsoft\Windows\Hyper-V-Worker\Admin: **12140** (Failure, Policy Violation)
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.	Windows Logs\Security: **4656** (Failure, Policy Violation) Windows Server 2012 R2: Applications and Services Logs\Microsoft\Windows\Hyper-V-SynthNic\Admin: **12597** (Success, Configuration) Windows 10 Enterprise / Windows Server 2016:

SFR	Auditable Events	Additional Audit Record Contents	Log: Event Id
			Applications and Services Logs\Microsoft\Windows\Hyper-V-Worker\Admin: **12597** (Success, Configuration)
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.	Initiation: Windows Logs\Security:**4651, 5451** Termination: Windows Logs\Security:**4655, 5452** Failure: Windows Logs\Security:**4652, 4653, 4654**
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).	Windows Logs\Security: **4624, 4625**
	Administrator session start time and end time.	None.	Start time: Windows Logs\Security: **4624** End time: Windows Logs\Security: **4634**
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 VM Start/Stop/Suspend events. Start and end of remote management session.	Configuration changes.	Updates to the TOE: Windows Logs->Setup: **1, 2, 3** Configuration changes: see **Table 2** Start-up and shutdown of the TOE: Windows Logs\Security: **1100, 4608** VM Start/Stop/Suspend events: Applications and Services Logs\Microsoft\Windows\Hyper-V-Worker\Admin: **18500, 18502, 18504, 18510**

SFR	Auditable Events	Additional Audit Record Contents	Log: Event Id
			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.	Initiation: Windows Logs/Setup: **1** Failure: Windows Logs/Setup: **3**
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_RDM_EXT.1	Transfer of removable media or device between VMs.	None.	
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.	Initiation: Windows Logs\Security:**4651, 5451** Termination: Windows Logs\Security:**4655, 5452**

SFR	Auditable Events	Additional Audit Record Contents	Log: Event Id
			Failure: Windows Logs\Security:**4652, 4653, 4654**
FCS_TLSC_EXT.1	Failure to establish a TLS Session. Establishment/Termination of a TLS session.	Reason for failure. Non-TOE endpoint of connection (IP address).	Failure: Windows Logs -> System: **36888** Establishment: Windows Logs -> System: **36880** Terminate: Microsoft-Windows-SChannel-Events/Perf: **1793**
FCS_TLSS_EXT.1	Failure to establish a TLS Session. Establishment/Termination of a TLS session.	Reason for failure. Non-TOE endpoint of connection (IP address).	Failure: Windows Logs -> System: **36888** Establishment: Windows Logs -> System: **36880** Terminate: Microsoft-Windows-SChannel-Events/Perf: **1793**
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.	Failure: Windows Logs -> System: **36888** Establishment: Windows Logs -> System: **36880** Terminate: Microsoft-Windows-SChannel-Events/Perf: **1793**

Table 21 Audit Events for SV PP Management Requirements

Administrative Action/Management Functions	ID
Ability to administer the Virtualization System locally and remotely;	Windows Logs/Security: **4624, 4625**
Ability to update the Virtualization System, and to verify the updates using [digital signature] capability prior to installing those updates;	Windows Logs/Setup: **1, 2, 3**
Ability to configure password policy	Windows Logs/Security: **4739**
Ability to create, delete, and configure VMs	Applications and Services Microsoft-Windows-Hyper-V-VMMS/Admin: **13002, 13003** Applications and Services Microsoft-Windows-Hyper-V-VMMS/Analytic: **12170, 12180**
~~**Ability to set default initial VM configurations;**~~	N/A

Administrative Action/Management Functions	ID
Ability to configure virtual networks including VMs;	Applications and Services Microsoft-Windows-Hyper-V-VMMS/Networking: **26000, 26004, 26012, 26016, 26074**
Ability to manage the audit system and audit data;	Windows Logs/Security: **4719**
Ability to configure VM access to physical devices;	Applications and Services 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
Ability to configure removable media policy;	Applications and Services Microsoft-Windows-Hyper-V-VMMS/Analytic: **12170, 12180**
Ability to configure the list of TOE-provided services available before an entity is identified and authenticated, as specified in FIA_UIA_EXT.1;	Windows Logs/Security: **4657**
Ability to configure the cryptographic functionality	TLS: Windows Logs/Security: **4657** IPsec: Windows Logs/Security: **5449**
Ability to enable/disable screen lock	Windows Logs/Security: **4663**
Ability to configure screen lock inactivity timeout	Windows Logs/Security: **4663**
Ability to configure remote connection inactivity timeout	Windows Logs/Security: **4663**
Ability to configure lockout policy for unsuccessful authentication attempts through limiting number of attempts during a time period	Windows Logs/Security: **4739**
Ability to configure name/address of directory server to bind with	Windows Logs/System: **3260**
Ability to configure name/address of audit/logging server to which to send audit/logging records	Windows Logs/Security: **4947**
Ability to configure name/address of network time server	Windows Logs/System: **37**
Ability to configure advisory warning message in banner, as described in FTA_TAB.1	Windows Logs/Security: **4657**
Ability to enable/disable password authentication.	Windows Logs/Security: **4662**
Ability to checkpoint a VM	Applications and Services Logs\Microsoft\Windows\Hyper-V-Worker\Admin: **18596**

Table 22 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>
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>
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>
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>
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>

Id	Log location	Message	Fields
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>
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>
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)> **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>

Id	Log location	Message	Fields
			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 >
4653	**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**: <User ID, Remote source IP address>
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>

Id	Log location	Message	Fields
4656	**Windows Logs->Security** Subcategory: File System and Handle Manipulation	A handle to an object was requested.	**N/A**:<User ID> **EventData->RemoteAddress**: <Remote source IP address> **System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData ->SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure>
4657	**Windows Logs->Security** Subcategory: Registry	A registry value was modified.	**EventData->ObjectName:** <Configuration change> **System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData ->SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure>
4662	**Windows Logs->Security** Subcategory: Other Object Access Events	An operation was performed on an object.	**EventData->ObjectName:** <Requested file> **System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData ->SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure>
4663	**Windows Logs->Security** Subcategory: Registry	An attempt was made to access an object.	**EventData->ObjectName:** <Configuration change> **System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData ->SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure>
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>
4720	**Windows Logs->Security** Subcategory: User Account Management	A user account was created.	**EventData -> *:** <Configuration changes> **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**	A user account was enabled.	**System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <Type of event>

Id	Log location	Message	Fields
	Subcategory: User Account Management		**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>
4947	**Windows Logs -> Security** Subcategory: MPSSVC Rule-Level Policy Change	A change was made to the Windows Firewall exception list. A rule was modified.	**EventData -> *:** <Configuration changes> **System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData -> SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure>
5447	**Windows Logs -> Security** Subcategory: Other Policy Change Events	A Windows Filtering Platform filter has been changed.	**EventData -> *:** <Configuration changes> **System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Task**: <type of event> **EventData -> SubjectUserSid**: <subject identifier > **System->Keywords**: <Outcome as Success or Failure>
5449	**Windows Logs -> Security** Subcategory: Filtering Platform Policy Change	A Windows Filtering Platform provider context has been changed.	**EventData -> *:** <Configuration changes> **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>
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>
12140	Windows Server 2012 R2: **Applications and Services -> Microsoft -> Windows -> Hyper-V-SynthStor -> Admin** Windows Server 2016: **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>

Id	Log location	Message	Fields
			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>
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**	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>

Id	Log location	Message	Fields
	-> Hyper-V-VMMS -> Admin		**System->Security[UserID]**: <Subject identifier > **System->Level**: <Outcome as Success or Failure>
18500	**Applications and Services -> Microsoft -> Windows -> Hyper-V-Worker/Admin**	<VM name> started successfully. (Virtual machine ID <VM ID>)	**System->EventID:** <Configuration change> **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>
18502	**Applications and Services -> Microsoft -> Windows -> Hyper-V-Worker/Admin**	<VM name> was turned off. (Virtual machine ID <VM ID>)	**System->EventID:** <Configuration change> **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>
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->EventID:** <Configuration change> **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>
18510	**Applications and Services -> Microsoft -> Windows -> Hyper-V-Worker/Admin**	<VM name> saved successfully. (Virtual machine ID <VM ID>)	**System->EventID:** <Configuration change> **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>
18596	**Applications and Services -> Microsoft -> Windows -> Hyper-V-Worker/Admin**	<VM name> was restored successfully. (Virtual machine ID <VM ID>)	**System->EventID:** <Configuration change> **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>
26000	**Applications and Services -> Microsoft -> Windows**	Switch created, name= <switch ID>, friendly name=<switch name>.	**System->EventID:** <Configuration change> **System->TimeCreated[SystemTime]**: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <Subject identifier >

Id	Log location	Message	Fields
	-> Hyper-V-VMMS -> Networking		**System->Level**: <Outcome as Success or Failure>
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->EventID:** <Configuration change> **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>
26012	**Applications and Services -> Microsoft -> Windows -> Hyper-V-VMMS -> Networking**	Internal miniport created, name = <miniport ID>, friendly name = <miniport name>, MAC = <MAC address>.	**System->EventID:** <Configuration change> **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>
26016	**Applications and Services -> Microsoft -> Windows -> Hyper-V-VMMS -> Networking**	External ethernet port <port ID> bound.	**System->EventID:** <Configuration change> **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>
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->EventID:** <Configuration change> **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>
36880	**Windows Logs -> System**	An TLS server handshake completed successfully. The negotiated cryptographic parameters are as follows:	**System->EventID:** <Configuration change> **System->TimeCreated**[**SystemTime**]: <Date and time of event> **System->Provider[Name]**: <type of event> **System->Security[UserID]**: <subject identifier >
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.	**UserData->EventXML->TargetName**: <Non-TOE endpoint> **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 >

Id	Log location	Message	Fields
			UserData->EventXML->AlertDesc: < Reason for failure> **UserData->EventXML->ErrorState:** < Reason for failure > 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