Microsoft Windows

FIPS 140 Validation

Microsoft Windows Server 2019

Microsoft Azure Stack Edge

Microsoft Azure Stack Hub

Microsoft Azure Stack Edge Rugged

Non-Proprietary

Security Policy Document

Document Information	
Version Number	1.2
Updated On	April 19, 2023

Version History

Version	Date	Summary of changes
1.0	November 2, 2020	Draft sent to NIST CMVP
1.1	November 3, 2022	Updates in response to NIST feedback
1.2	April 19, 2023	Added CMVP certificate numbers for bounded modules

TABLE OF CONTENTS

1 Introduction

The Windows Operating System (OS) Loader, which consists of the binary WINLOAD.EFI, is the operating system loader that loads the operating system kernel (ntoskrnl.exe) and other boot stage binary image files. The Windows OS Loader is a part of BitLocker Drive Encryption, which is a data protection feature of the Windows Server operating system which encrypts data on a storage volume. For the purpose of this validation, Windows OS Loader is classified as a Software-Hybrid cryptographic module because the validated platforms all implement the AES-NI instruction set.

1.1 List of Cryptographic Module Binary Executables

The Windows OS Loader module contains the following binary:

- WINLOAD.EFI

The Windows builds covered by this validation are:

- Windows Server 2019 build 10.0.17763.10021 and 10.0.17763.10127

1.2 Validated Platforms

The editions covered by this validation are:

- Windows Server 2019 Datacenter Core

The Windows OS Loader components listed in Section 1.1 were validated using the combination of computers and Windows operating system editions specified in the table below.

All the computers for Windows Server listed in the table below are all 64-bit Intel architecture and implement the AES-NI instruction set but not the SHA Extensions.

Table 1 Validated Platforms

Computer	Windows Server 2019 Datacenter Core	Processor Image
Microsoft Azure Stack Edge – Dell XR2 – Intel Xeon Silver 4114	√	wikichip.org
Microsoft Azure Stack Hub – Dell PowerEdge R640 – Intel Xeon Gold 6230	√	

		wikichip.org
Microsoft Azure Stack Hub – Dell PowerEdge R840 – Intel Xeon Platinum 8260	√	wikichip.org
Microsoft Azure Stack Edge Rugged – Rugged Mobile Appliance – Intel Xeon D-1559	√	wikichip.org

2 Cryptographic Module Specification

Windows OS Loader is a multi-chip standalone module that operates in FIPS-approved mode during normal operation of the computer and Windows operating system boot sequence.

The following configurations and modes of operation will cause Windows OS Loader to operate in a non-approved mode of operation:

- Boot Windows in Debug mode
- Boot Windows with Driver Signing disabled

2.1 Cryptographic Boundary

The software-hybrid cryptographic boundary for Windows OS Loader consists of disjoint software and hardware components within the same physical boundary of the host platform. The software components are defined as the binary WINLOAD.EFI, and the hardware components are the CPUs running on each host platform.

2.2 FIPS 140-2 Approved Algorithms

Windows OS Loader implements the following FIPS 140-2 Approved algorithms:[1]

Table 2 Approved Algorithms

Algorithm	Windows Server 2019 build 10.0.17763.10021	Windows Server 2019 build 10.0.17763.10127
FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 1024, 2048, and 3072 moduli; supporting SHA-1, SHA-256, SHA-384, and SHA-512	#C1586	#C2052
FIPS 180-4 SHS SHA-1, SHA-256, SHA-384, and SHA-512	#C1577	#C2044
FIPS 197 AES CBC 128 and 256 (Decrypt)	#C1577	#C2044
NIST SP 800-38E AES XTS 128 and 256 (Decrypt)	#C1577	#C2044
NIST SP 800-38C AES CCM 256 (Decrypt)	#C1583	#C2049

2.3 Non-Approved Algorithms

Windows OS Loader implements the following non-approved algorithms:

- A non-determinic random number generator for entropy that is a not a FIPS Approved algorithm but is allowed by FIPS 140. See **Collecting Initial Entropy for the OS** in Services.

2.4 FIPS 140-2 Approved Algorithms from Bounded Modules

A bounded module is a FIPS 140 module which provides cryptographic functionality that is relied on by a downstream module. As described in the Integrity Chain of Trust section, the Windows OS Loader depends on the following algorithms:

The Boot Manager for Windows Server 2019 (version 1809) build 10.0.17763.10021 (module certificate #4484) provides:

- CAVP certificates #C1586 for FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 2048 moduli; supporting SHA-256
- CAVP certificates #C1577 for FIPS 180-4 SHS SHA-256

[1] This module may not use some of the capabilities described in each CAVP certificate.

The Boot Manager for Windows Server 2019 (version 1809) build 10.0.17763.10127 (module certificate #4484) provides:

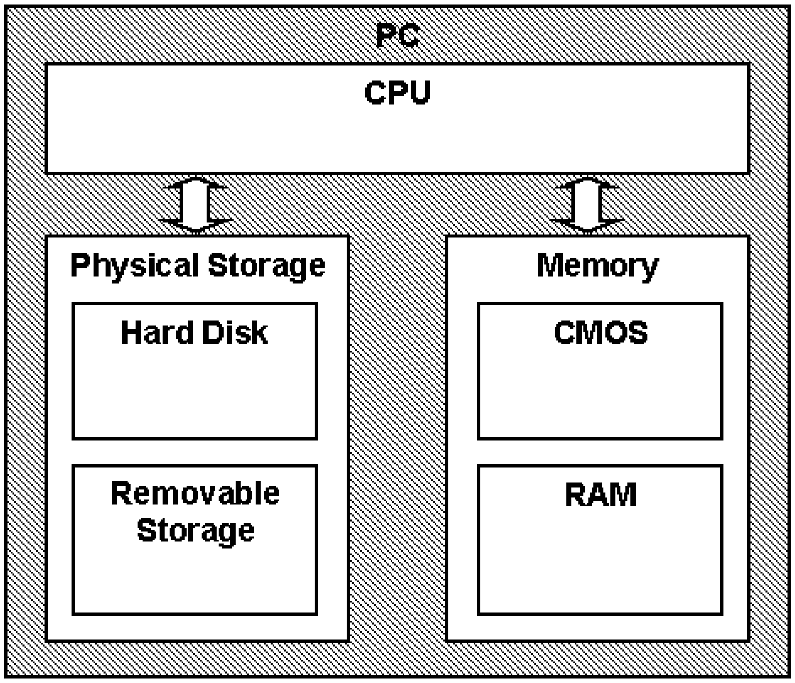
- CAVP certificates #C2052 for FIPS 186-4 RSA PKCS#1 (v1.5) digital signature verification with 2048 moduli; supporting SHA-256
- CAVP certificates #C2044 for FIPS 180-4 SHS SHA-256

2.5 Cryptographic Bypass

Cryptographic bypass is not supported by Windows OS Loader.

2.6 Hardware Components of the Cryptographic Module

The physical boundary of the module is the physical boundary of the computer that contains the module. The following diagram illustrates the hardware components used by the Windows OS Loader module:



Note: The CPU provides Processor Algorithm Accelerator (PAA)

3 Cryptographic Module Ports and Interfaces

3.1 Control Input Interface

The Windows OS Loader Control Input Interface is the set of internal functions responsible for intercepting control input. These functions are:

1. OslMain (WINLOAD) – This function receives and parses the Boot Application parameters, which are passed to the module when execution is passed from Boot Manager.

2. BlBdInitialize – Reads the system status to determine if a boot debugger is attached.
3. BlInitializeLibrary – Performs the parsing Boot Application parameters.
4. BlXmiRead – Reads the operator selection from the Windows OS Loader user interface.

3.2 Status Output Interface

The Status Output Interface is the BlXmiWrite function that is responsible for displaying any integrity verification errors to the display. The Status Output Interface is also defined as the BlLogData responsible for writing the name of the corrupt driver to the bootlog.

3.3 Data Output Interface

The Data Output Interface is represented by the AhCreateLoadOptionsString function. OslArchTransferToKernel is responsible for transferring the execution from Windows OS Loader to the initial execution point of the Windows Server kernel. Data exits the module in the form of the initial instruction address of the Windows Server kernel.

Data exits the module from the AhCreateLoadOptionsString function in the form of boot application parameters passed to the Windows Server kernel.

3.4 Data Input Interface

The Data Input Interface is represented by the BlFileReadEx function and the BlDeviceRead function. BlFileReadEx is responsible for reading the binary data of unverified components from the computer hard drive. In addition the BitLocker Full Volume Encryption Key (FVEK) can also be entered into the module over the module's data input interface. BlDeviceRead is responsible for reading data directly from devices.

4 Roles, Services and Authentication

4.1 Roles

In Windows Server, authentication and assignment of roles happens after the OS boots. Since Windows OS Loader functions only during the period between power-on and OS initialization, the module's functions are fully automatic and not configurable. FIPS 140 validations define formal "User" and "Cryptographic Officer" roles. Both roles can use any Windows OS Loader service.

4.2 Services

Windows OS Loader services are described below. It does not export any cryptographic functions.

1. **Loading the OS (Trusted Boot)** - The main service is to load the Windows Server operating system kernel (ntoskrnl.exe), the Windows hypervisor (hvix64.exe for Intel processors and hvax64.exe for AMD processors), and other boot stage binary image files, including Code Integrity (ci.dll), Secure Kernel Code Integrity (skci.dll), and Kernel Mode Cryptographic Primitives Library (cng.sys) after it validates their integrity using the FIPS 140-2 Approved cryptographic algorithm implementations described below. After the verified kernel and boot stage binary image files are loaded, Windows OS Loader passes the execution control to the NT

operating system kernel, and the Virtual Secure Mode kernel, and it terminates its own execution. If the integrity of any module is not verified, Windows OS Loader does not transfer the execution to the kernel and displays the boot failure page.

2. **Show Status** – If Windows OS Loader fails any of the checks specified in the Finite State Model then it reports a boot failure status on the computer monitor.
3. **Perform Self-Tests** - The module provides a power-up self-tests service that is automatically executed when the module is loaded into memory. See Self-Tests.
4. **Zeroizing Cryptographic Material -** see Cryptographic Key Management.
5. **Collecting Initial Entropy** for the OS - Windows OS Loader also implements an entropy source for the operating system. Entropy is gathered from the following sources, when they are available on the computer:
   a. The contents of the registry value HKLM\System\RNG\Seed, which is written by the Kernel Mode Cryptographic Primitives Library (cng.sys) during its normal operation.
   b. The contents of the registry value HKLM\System\RNG\ExternalEntropy, which can be populated by system administrators. This value is overwritten after reading, to ensure that it is not reused.
   c. If a Trusted Platform Module (TPM) is available, the output of a TPM_GetRandom call to the TPM.
   d. The current system time.
   e. The contents of the OEM0 ACPI table in the machine firmware.
   f. If the CPU supports instructions for the RDSEED (used when available) or RDRAND (used if RDSEED is not available), the output from RDSEED or RDRAND.
   g. The output of the UEFI random number generator.
   h. The CPU timings.

   These inputs are then combined using SHA-512, and the entropy source is conditioned using a non-Approved RNG. From this NDRNG, a block of output bytes is passed to the Windows kernel at boot time. This block of output bytes is used by CNG.SYS as one of its entropy sources.

   This service is considered a “Non-Approved, but Allowed” service. It is only “Allowed” in the context that it is being used by another FIPS 140-2 Approved module, i.e., the Kernel Mode Cryptographic Primitives Library (cng.sys), in order to provide entropy to one of the FIPS-approved DRBGs.

6. **Measured Boot** – The combination of Windows Server and the computer’s firmware logs measurements from the boot process that can be sent to a remote trusted attestation server which objectively assesses the health of early boot stage components.

The following table maps the services to their corresponding algorithms and critical security parameters (CSPs) as described in Cryptographic Key Management.

Table 3 Services

Service	Algorithms	CSPs	Invocation
Loading the OS (Trusted Boot)	FIPS 186-4 RSA PKCS#1 (v1.5) FIPS 180-4 SHS: SHA-256 hash SHA-384 hash SHA-512 hash AES CBC 128 and 256 bits AES XTS 128 and 256 bits[2] AES CCM 256 bits	RSA public key to verify the integrity of components mentioned in Integrity Chain of Trust Full Volume Encryption Key (FVEK) (to load the BitLocker encrypted data containing the OS)	This service is fully automatic.
Show Status	None	None	This service is fully automatic.
Perform Self-Tests	FIPS 186-4 RSA PKCS#1 (v1.5) verify with public key KAT and signature verification KAT FIPS 180-4 SHS: SHA-1 KAT SHA-256 KAT SHA-512 KAT FIPS 197 AES: AES CBC KAT AES CCM KAT AES XTS KAT	None	This service is fully automatic.
Zeroizing Cryptographic Material	None	Full Volume Encryption Key (FVEK)	See Zeroization.
Collecting Initial Entropy	SHA-512	None	This service is fully automatic.
Measured Boot	SHA-1 SHA-256	None	This service is fully automatic after Measured Boot has been enabled.

4.3 Authentication

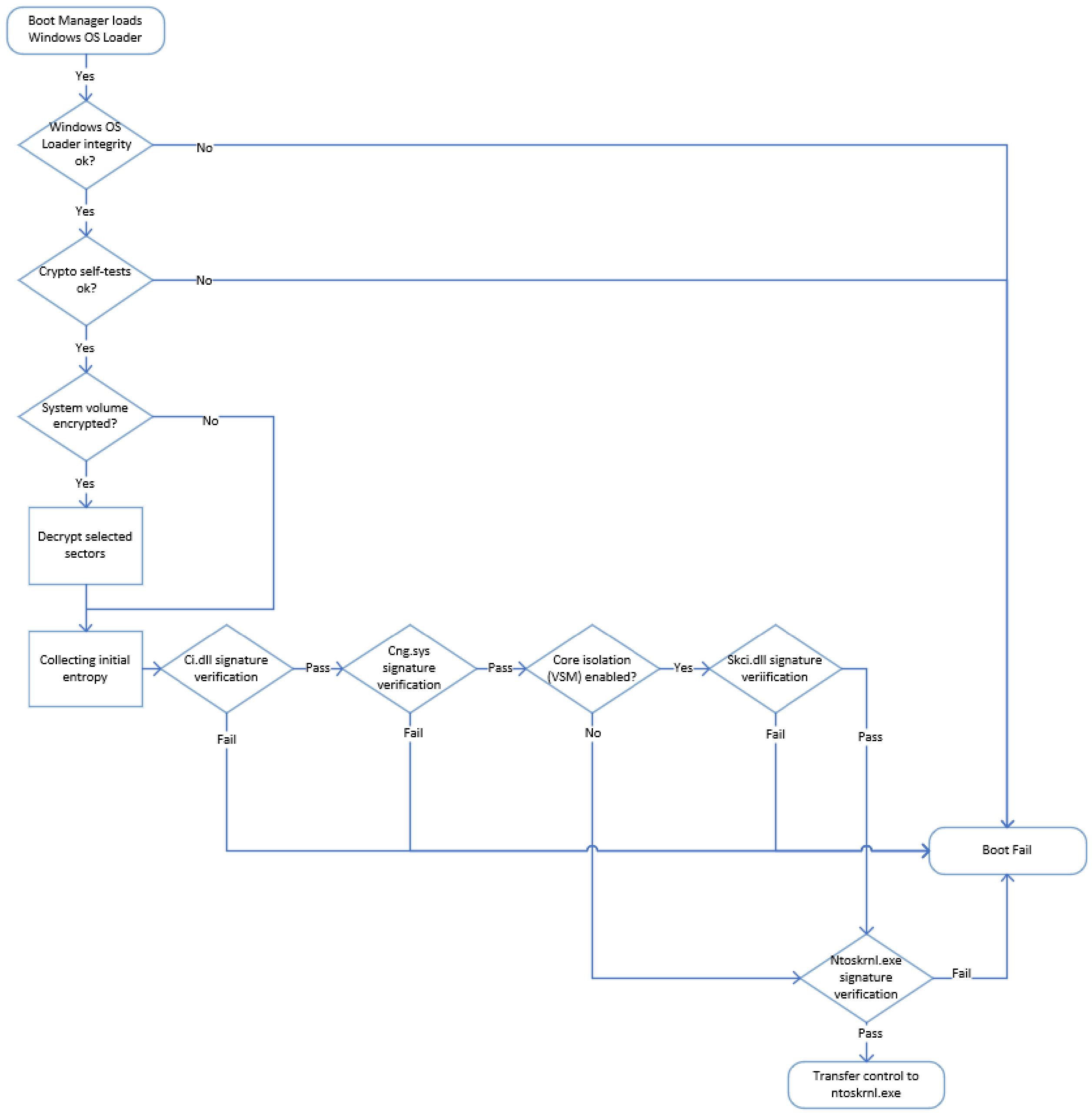
The Windows OS Loader does not implement any authentication services.

5 Finite State Model

[2] The length of the data unit does not exceed 2^{20} AES blocks for storage applications such as BitLocker..

5.1 Specification

The following diagram shows the finite state model for Windows OS Loader:


Boot Manager loads Windows OS Loader
Yes
Windows OS Loader integrity ok?
No
Yes
Crypto self-tests ok?
No
Yes
System volume encrypted?
No
Yes
Decrypt selected sectors
Collecting initial entropy
Ci.dll signature verification
Pass
Cng.sys signature verification
Pass
Core isolation (VSM) enabled?
Yes
Skci.dll signature veriification
Fail
Fail
No
Fail
Pass
Boot Fail
Ntoskrnl.exe signature verification
Fail
Pass
Transfer control to ntoskrnl.exe

6 Operational Environment

The operational environment for Windows OS Loader is the Windows operating system running on a supported hardware platform.

6.1 Single Operator

During the operating system boot process there is no logged on user, so the single operator requirement is met.

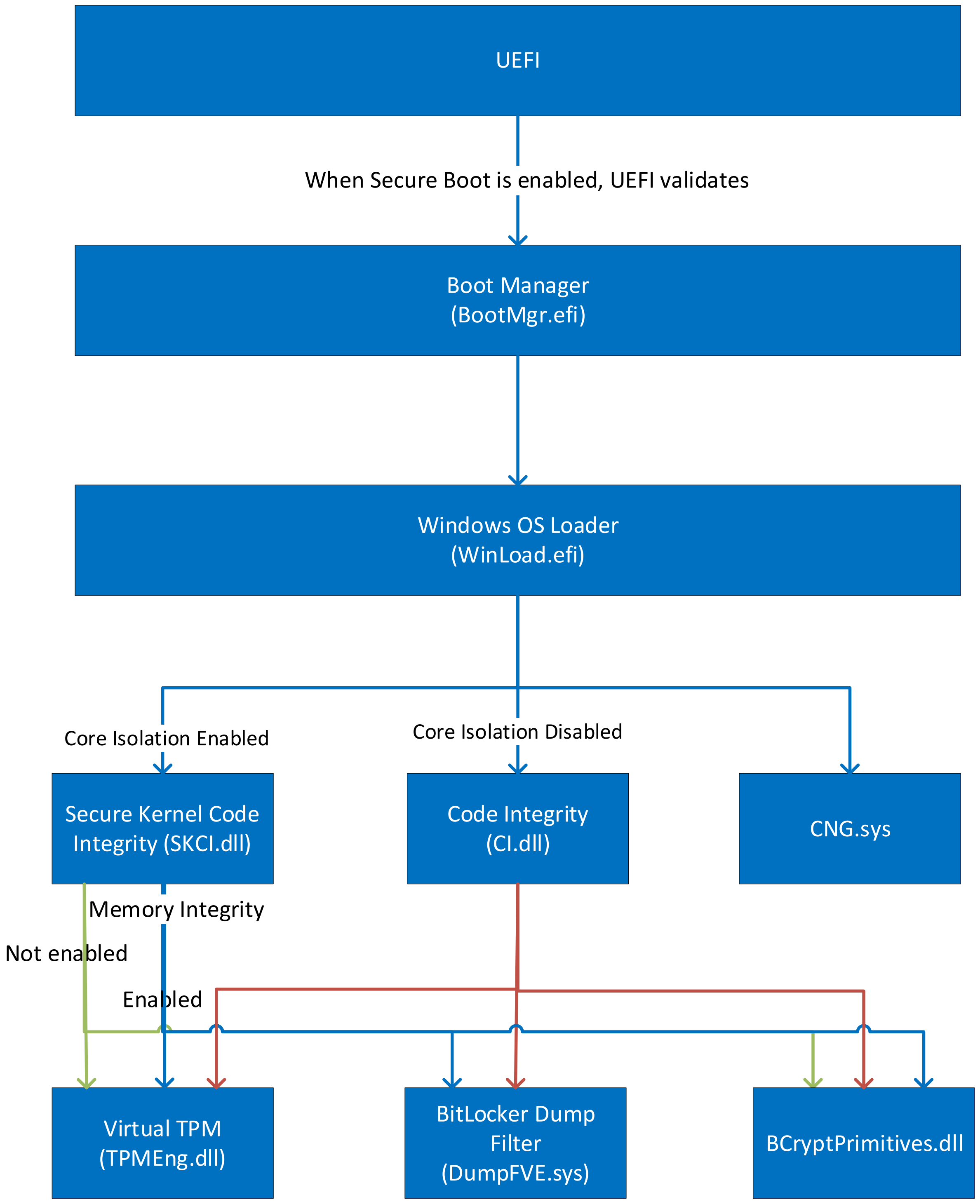
6.2 Cryptographic Isolation

While it is running, Windows OS Loader is the only process running on the computer.

6.3 Integrity Chain of Trust

Windows uses several mechanisms to provide integrity verification depending on the stage in the boot sequence and also on the hardware and configuration. The following diagram describes the Integrity Chain of trust for each supported configuration for the following versions:

- Windows Server 2019 build 10.0.17763.10021 and 10.0.17763.10127


UEFI
When Secure Boot is enabled, UEFI validates
Boot Manager (BootMgr.efi)
Windows OS Loader (WinLoad.efi)
Core Isolation Enabled
Core Isolation Disabled
Secure Kernel Code Integrity (SKCI.dll)
Code Integrity (CI.dll)
CNG.sys
Memory Integrity
Not enabled
Enabled
Virtual TPM (TPMEng.dll)
BitLocker Dump Filter (DumpFVE.sys)
BCryptPrimitives.dll

Boot Manager checks the integrity of the WINLOAD.EFI component of the Windows OS Loader before it is loaded.

Windows binaries include a SHA-256 hash of the binary signed with the 2048 bit Microsoft RSA code-signing key (i.e., the key associated with the Microsoft code-signing certificate). The integrity check uses the public key component of the Microsoft code signing certificate to verify the signed hash of the binary.

The Windows OS Loader component verifies the integrity of multiple kernel mode cryptographic modules using the same process described above. The following binaries are verified:

- CNG.SYS
- CI.DLL
- SKCI.DLL (When VSM is enabled)

Windows OS Loader also verifies the integrity of other early boot kernel mode drivers that are not cryptographic modules.

7 Cryptographic Key Management

7.1 Critical Security Parameters

When the System Volume is encrypted with Bitlocker, the WINLOAD.EFI component of the Windows OS Loader uses these critical security parameters (CSP):

- Full Volume Encryption Key (FVEK) – 128 or 256-bit AES key that is used to decrypt data on disk sectors of the hard drive.

The FVEK is provided to the WINLOAD.EFI component of the Windows OS Loader by Boot Manager.

Windows OS Loader also uses as a CSP the public key component of the Microsoft code signing certificate as described in Integrity Chain of Trust:

- Microsoft Root Certificate Authority (CA) Public Key – 2048-bit RSA key with SHA-256.

7.2 Zeroization

7.2.1 Volatile Keys

The FVEK is zeroized when the module component WINLOAD.EFI is unloaded from memory after control is transferred to ntoskrnl.exe.

7.3 Access Control Policy

The Windows OS Loader does not allow access to the cryptographic keys contained within it, so an access control table is not included in this document.

The FVEK is received from outside the Windows OS and then managed appropriately once received.

Windows OS Loader prevents access to volatile keys and CSPs by zeroizing them after use.

8 Self-Tests

8.1 Power-On Self-Tests

The WINLOAD.EFI component of the Windows OS Loader performs the following power-on (startup) self-tests:

- RSA PKCS#1 (v1.5) verify with public key Known Answer Test
    - RSA signature verification Known Answer Test with 1024-bit key and SHA-1 message digest
    - RSA signature verification Known Answer Test with 2048-bit key and SHA-256 message digest
- SHS (SHA-1) Known Answer Test
- SHS (SHA-256) Known Answer Test
- SHS (SHA-512) Known Answer Test
- AES-CCM 256 Encrypt/Decrypt Known Answer Tests
- AES-CBC 128 and 256 Encrypt/Decrypt Known Answer Tests
- XTS-AES 128 and 256 Encrypt/Decrypt Known Answer Tests

If the self-test fails, the module will not load and status will be returned. If the status is not STATUS_SUCCESS, then that is the indicator a self-test failed. While there are various potential failure status indicators, STATUS_INTERNAL_ERROR is one example for a failure status.

8.2 Conditional Self-Tests

Windows OS Loader performs the following conditional self-tests:

- A CRNGT is performed for the non-approved NDRNG per FIPS 140-2 IG 9.8.

9 Design Assurance

The secure installation, generation, and startup procedures of this cryptographic module are part of the overall operating system secure installation, configuration, and startup procedures for the Windows Server operating system.

The Windows Server operating system must be pre-installed on a computer by an OEM, installed by the end-user, by an organization's IT administrator, or updated from a previous Windows Server version downloaded from Windows Update.

An inspection of authenticity of the physical medium can be made by following the guidance at this Microsoft web site: https://www.microsoft.com/en-us/howtotell/default.aspx

The installed version of Windows Server must be checked to match the version that was validated. See Appendix A for details on how to do this.

For Windows Updates, the client only accepts binaries signed with Microsoft certificates. The Windows Update client only accepts content whose signed SHA-2 hash matches the SHA-2 hash specified in the metadata. All metadata communication is done over a Secure Sockets Layer (SSL) port. Using SSL ensures that the client is communicating with the real server and so prevents a spoof server from sending the client harmful requests. The version and digital signature of new cryptographic module releases must be verified to match the version that was validated. See Appendix A for details on how to do this.

10 Mitigation of Other Attacks

The following table lists the mitigations of other attacks for this cryptographic module:

Table 4 Mitigation of Other Attacks

Algorithm	Protected Against	Mitigation
SHA1	Timing Analysis Attack	Constant time implementation
	Cache Attack	Memory access pattern is independent of any confidential data
SHA2	Timing Analysis Attack	Constant time implementation
	Cache Attack	Memory access pattern is independent of any confidential data
AES	Timing Analysis Attack	Constant time implementation
	Cache Attack	Memory access pattern is independent of any confidential data
		Protected against cache attacks only when running on a processor that implements AES-NI

11 Security Levels

The security level for each FIPS 140-2 security requirement is given in the following table.

Table 5 Security Levels

Security Requirement	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	1
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	2
Mitigation of Other Attacks	1

Boot Manager is a multi-chip standalone software-hybrid module whose host platforms meet the level 1 physical security requirements. The module consists of production-grade components that include standard passivation techniques and is entirely contained within a metal or hard plastic production-grade enclosure that may include doors or removable covers.

12 Additional Details

For the latest information on Microsoft Windows, check out the Microsoft web site at:

https://www.microsoft.com/en-us/windows

For more information about FIPS 140 validations of Microsoft products, please see:

https://docs.microsoft.com/en-us/windows/security/threat-protection/fips-140-validation

13 Appendix A – How to Verify Windows Versions and Digital Signatures

13.1 How to Verify Windows Versions

The installed version of Windows Server must be verified to match the version that was validated using the following method:

1. In the Search box type "cmd" and open the Command Prompt desktop app.
2. The command window will open.
3. At the prompt, enter "ver”.
4. The version information will be displayed in a format like this:
   ```
   Microsoft Windows [Version 10.0.xxxxx]
   ```

If the version number reported by the utility matches the expected output, then the installed version has been validated to be correct.

13.2 How to Verify Windows Digital Signatures

After performing a Windows Update that includes changes to a cryptographic module, the digital signature and file version of the binary executable file must be verified. This is done like so:

1. Open a new window in Windows Explorer.
2. Type “C:\Windows\” in the file path field at the top of the window.
3. Type the cryptographic module binary executable file name (for example, “CNG.SYS”) in the search field at the top right of the window, then press the Enter key.
4. The file will appear in the window.
5. Right click on the file’s icon.
6. Select Properties from the menu and the Properties window opens.
7. Select the Details tab.
8. Note the File version Property and its value, which has a number in this format: xx.x.xxxxx.xxxx.
9. If the file version number matches one of the version numbers that appear at the start of this security policy document, then the version number has been verified.
10. Select the Digital Signatures tab.
11. In the Signature list, select the Microsoft Windows signer.
12. Click the Details button.
13. Under the Digital Signature Information, you should see: “This digital signature is OK.” If that condition is true, then the digital signature has been verified.