Windows 7 Code Integrity (ci.dll) Security Policy

For FIPS 140-2 Validation

v 4.3
01/16/2013

1 Introduction

Code Integrity (CI.DLL) is a Windows 7 feature that verifies the integrity of some key Windows 7 binary image files as they are loaded into memory from the disk.

CI.DLL is not a general purpose cryptographic module. It is validated under FIPS 140-2 because it implements cryptographic algorithms and provides the integrity checks of the Windows 7 general purpose cryptographic modules.

1.1 Cryptographic Boundary for CI.DLL

The Windows 7 CI.DLL consists of a dynamically-linked library (CI.DLL, versions 6.1.7600.16385, 6.1.7600.17122, 6.1.7600.21320, 6.1.7601.17514, 6.1.7601.17950, and 6.1.7601.22108). The cryptographic boundary for CI.DLL is defined as the enclosure of the computer system, on which CI.DLL is to be executed. The physical configuration of CI.DLL, as defined in FIPS 140-2, is multi-chip standalone.

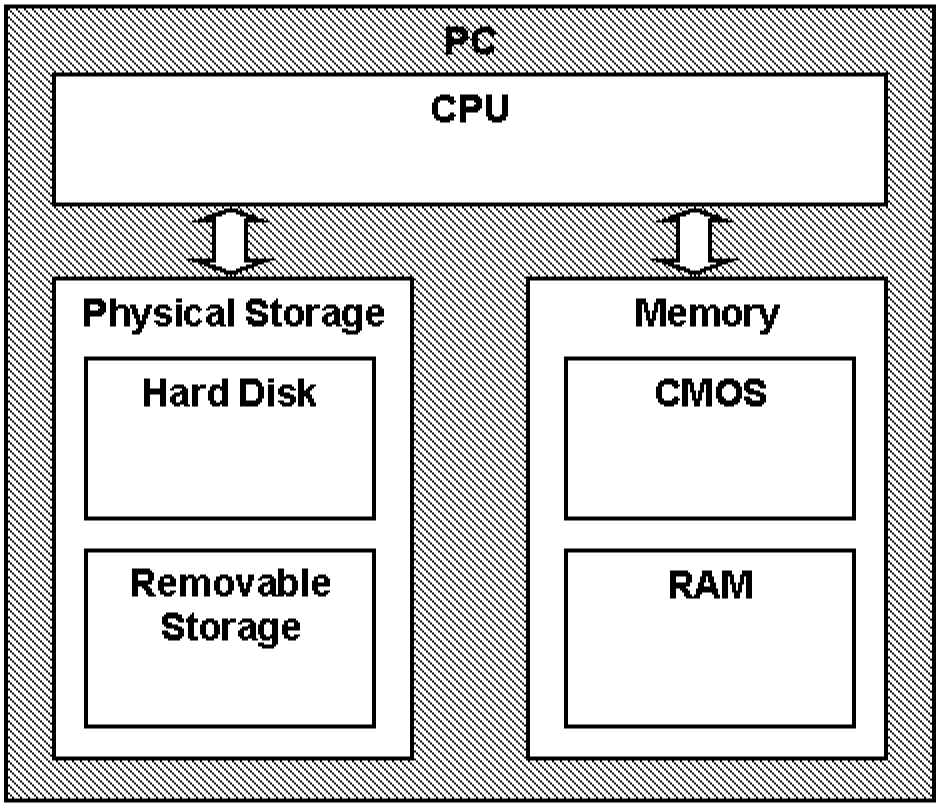
2 Security Policy

2.1 CI.DLL Security Policy Elements

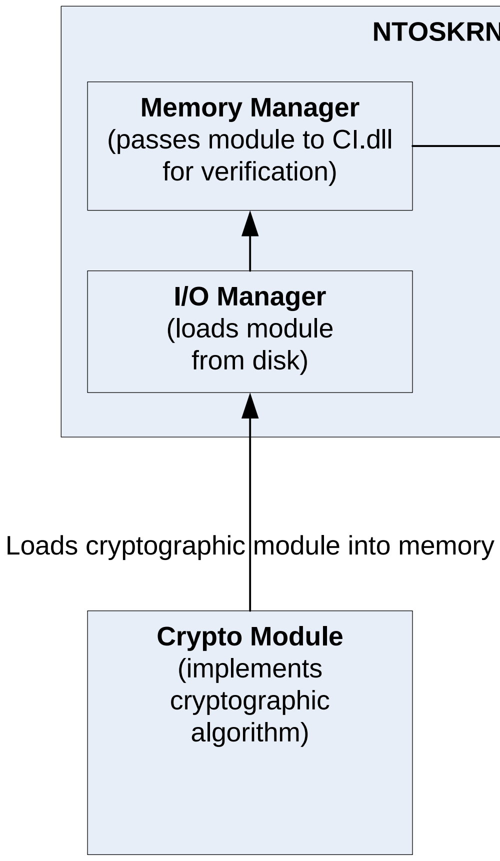
CI.DLL is considered to be in a FIPS mode of operation when the following rules are followed.

- CI.DLL is supported on Windows 7 and Windows 7 SP1.
- Windows 7 is an operating system supporting a “single user” mode where there is only one interactive user during a logon session.
- CI.DLL is only in the “Approved mode of operation” when Windows is booted normally, meaning Debug mode has not been enabled and Driver Signing enforcement has not been disabled.
- CI.DLL operates in FIPS mode of operation only when used with the FIPS approved version of Windows 7 Winload OS Loader (winload.exe) validated to FIPS 140-2 Cert. #1326 operating in FIPS mode

The following diagram illustrates the master components of the CI.DLL module



The following diagram illustrates the interaction of CI.DLL with cryptographic modules


NTOSKRNL.EXE
Memory Manager (passes module to CI.dll for verification)
CI.DLL (verifies cryptographic module integrity using signed hashes prior to execution)
I/O Manager (loads module from disk)
Loads cryptographic module into memory
Crypto Module (implements cryptographic algorithm)

2.2 Code Integrity security policy

- CI.DLL's main service is to verify the integrity of digitally signed drivers and components within the computer (such as bcryptprimitives.dll, rsaenh.dll, dssenh.dll). In addition to this service, CI.DLL also provides status services. These status services indicate whether the aforementioned integrity checks passed.
- All services implemented within CI.DLL are available to the User and Crypto officer roles. The User and Crypto officer roles are assumed by the operating system or application processes that will invoke binary image verification in CI.dll. The Window Memory Manager is an example.
- CI.DLL implements the following FIPS-140-2 Approved algorithms.
  - RSA PKCS#1 (v1.5) digital signature verification (Cert. #557)
  - SHA-1 hash (Cert. #1081)
  - SHA-256 hash (Cert. #1081)
  - SHA-384 hash (Cert. #1081)
  - SHA-512 hash (Cert. #1081)
- CI.DLL performs the following power-on (start up) self-tests.
  - SHS (SHA-1) Known Answer Test
  - SHS (SHA-256) Known Answer Test
  - SHS (SHA-512) Known Answer Test
  - RSA verify using a verify test with a Known Signature of the PKCS#1 v1.5 format
- CI.DLL verifies the integrity of the Windows 7 general purpose cryptographic modules using the following FIPS-140-2 Approved algorithms.
  - RSA PKCS#1 (v1.5) verify with public key
  - SHA-1 hash

    - SHA-256 hash
    - SHA-384 hash
    - SHA-512 hash

Cryptographic bypass is not supported by CI.DLL.

CI.DLL (versions: 6.1.7600.16385, 6.1.7600.17122, and 6.1.7600.21320) were tested using the following machine configurations:

x86	Windows 7 Ultimate – HP Compaq dc7600
x64	Windows 7 Ultimate – HP Compaq dc7600

CI.DLL (versions: 6.1.7601.17514, 6.1.7601.17950, and 6.1.7601.22108) were tested using the following machine configurations:

x86	Windows 7 Ultimate SP1 – HP Compaq dc7600
x64	Windows 7 Ultimate SP1 – HP Compaq dc7600

3 CI.DLL Ports and Interfaces

3.1 CI.DLL export functions

The following list contains all the functions exported by CI.DLL to its callers. Note that CI.DLL is not callable outside the kernel. They are explained further in the subsequent subsections.

- CiInitialize()
- CiValidateImageHeader()
- CiValidateImageData()

CiInitialize() is directly callable by a kernel mode caller. However, the other two are not.

3.1.1 CiInitialize()

CiInitialize() is the function exported by CI.DLL for initializing the image file integrity validation capability of CI.DLL.
As the power-on (start up) function of CI.DLL, CiInitialize() conducts the following power-on (start up) self-tests.

- SHS (SHA-1) Known Answer Test
- SHS (SHA-256) Known Answer Test
- SHS (SHA-512) Known Answer Test
- RSA verify using a verify test with a Known Signatures of the PKCS#1 v1.5 format:
    - RSA signature with 1024-bit key and SHA-1 message digest
    - RSA signature with 2048-bit key and SHA-256 message digest

If a self test fails, CiInitialize() returns STATUS_UNSUCCESSFUL. On the other hand, after the successful initialization, CiInitialize() returns a callback structure consisting of the following two functions. A caller subsequently can use these two functions to obtain the image file integrity validation service from CI.DLL.

- CiValidateImageHeader()
- CiValidateImageData()

3.1.2 CiValidateImageHeader()

When a caller (such as the Memory Manager) wants to obtain the set of trusted per-page hashes of an image file, it calls CiValidateImageHeader(). Trusted per-page hashes can use the following algorithms:

- SHS (SHA-1)
- SHS (SHA-256)

- SHS (SHA-384)
- SHS (SHA-512)

In the case of a Windows 7 general purpose cryptographic module (namely, bcryptprimitives.dll, rsaenh.dll, or dssenh.dll), if CiValidateImageHeader() does not find the set of trusted per-page hashes for the cryptographic module, then CiValidateImageHeader() verifies the full cryptographic module image by verifying a trusted file hash. The trusted file hash may be:

- SHS (SHA-1)
- SHS (SHA-256)
- SHS (SHA-384)
- SHS (SHA-512)

If this validation process fails, the cryptographic module is not valid. Subsequently, the Windows 7 Memory Manager does not load any page of the cryptographic module.

Both the trusted file image hash and trusted page hashes are signed using the RSA signature algorithm with PKCS#1 v1.5 padding.

CI.DLL has a different verification procedure for kernel mode crypto modules that are loaded into memory all at once (not in a per-page fashion as the other user mode general purpose crypto modules). As a result, when CiValidateImageHeader() is called by the memory manager, the CI_VALIDATE_DRIVER_IMAGE flag is set, and the entire image is validated by verifying a trusted image hash. This is similar to the user mode module verification when page hashes are not present.

3.1.3 CiValidateImageData()

After calling CiValidateImageHeader() to obtain the set of trusted per-page hashes of an image file, a caller (such as the Memory Manager) would want to know the integrity of a page of the image file. The caller uses CiValidateImageData() to check the page integrity.

If the computed hash matches the identified trusted hash, then CiValidateImageData confirms the integrity of the page. Otherwise, CiValidateImageData() returns STATUS_INVALID_IMAGE_HASH. The Windows 7 Memory Manager does not load invalid pages.

3.2 CI.DLL Data Input Interface

The Data Input Interface for CI.DLL consists of the three CI export functions. Data and options are passed to the interface as input parameters to the CI export functions. Data Input is kept separate from Control Input by passing Data Input in separate parameters from Control Input.

3.3 CI.DLL Data Output Interface

The Data Output Interface for CI.DLL also consists of the three CI export functions. For CiInitialize(), data is returned to its caller as the Callbacks output parameter. For CiValidateImageHeader(), data is returned to its caller as the SePool output parameter.

3.4 CI.DLL Control Input Interface

The Control Input Interface for CI.DLL also consists of the three CI export functions. Options for control operations are passed as input parameters to the CI export functions. The SecureRequired parameter in CiValidateImageHeader() is the only control option provided by CI.DLL in the Control Input Interface.

3.5 CI.DLL Status Output Interface

The Status Output Interface for CI.DLL also consists of the three CI export functions. For each function, the status information is returned to the caller as the return value (e.g. STATUS_SUCCESS, STATUS_UNSUCCESSFUL, STATUS_INVALID_IMAGE_HASH) from the function.

4 Specification of Roles

CI.DLL supports both User and Cryptographic Officer roles (as defined in FIPS-140-2). Both roles have access to all services implemented in CI.DLL through a caller component running in the kernel mode. The module does not provide authentication, as such both roles are implicitly assumed when the services exported by the module are invoked.

4.1 Maintenance Roles

Maintenance roles are not supported by CI.DLL.

4.2 Multiple Concurrent Interactive Operators

There is only one interactive operator during a logon session. Multiple concurrent interactive operators sharing a logon session are not supported.

5 Cryptographic Key Management

CI.DLL does not handle security-relevant information such as secret and private cryptographic key, authentication data or any other protected information. Hence, there is no operation related to any of the below.

- Key generation
- Key output
- Key storage

The only CSPs the module supports are the RSA PKCS#1 public keys used to verify integrity. These CSPs are accessible by both approved roles. The CSPs are stored on the hard-drive. Zeroization is performed by deleting the CI.DLL module, ntph.cat, and ntpe.cat files.

6 CI.DLL Self Tests

CI.DLL performs the following power-on (start up) self-tests when a caller calls its CiInitialize().

- SHS (SHA-1) Known Answer Test
- SHS (SHA-256) Known Answer Test
- SHS (SHA-512) Known Answer Test

RSA verify using a verify test with Known Signatures of the PKCS#1 v1.5 format with both 1024-bit keys with SHA1 digest and 2048-bit keys with SHA-256 digest.

The integrity of CI.DLL itself is protected by an RSA signature with a 2048-bit key and SHA-256 message digest, which is verified by Winload.exe before CI.DLL is loaded into memory.

7 Additional details

For the latest information on Windows 7, check out the Microsoft web site at http://www.microsoft.com.

CHANGE HISTORY			
AUTHOR	DATE	VERSION	COMMENT
Ben Nick	5/22/2009	4.0	Updated for Windows 7 RTM
	6/1/2010	4.1	Updates based on CMVP comments