ERNW
providing security.

Device Guard Image Integrity: Function Invocation Paths between ci.dll and skci.dll

Aleksandar Milenkoski✉
amilenkoski@ernw.de

This work is part of the *Windows Insight* series. This series aims to assist efforts on analysing inner working principles, functionalities, and properties of the Microsoft Windows operating system. For general inquiries contact Aleksandar Milenkoski (amilenkoski@ernw.de) or Dominik Phillips (dphillips@ernw.de). For inquiries on this work contact the corresponding author (✉).

The content of this work has been created in the course of the project named 'Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10 (SiSyPHuS Win10)' (ger.) - 'Study of system design, logging, hardening, and security functions in Windows 10' (eng.). This project has been contracted by the German Federal Office for Information Security (ger., Bundesamt für Sicherheit in der Informationstechnik - BSI).

Required Reading

In addition to referenced work, related work focussing on Device Guard Image Integrity and Virtual Secure Mode, part of the *Windows Insight* series, are relevant for understanding concepts and terms mentioned in this document.

Technology Domain

The operating system in focus is Windows 10, build 1607, 64-bit, long-term servicing branch (LTSB).

1 Introduction

ci.dll statically exports 8 functions: *CiCheckSignedFile*, *CiFindPageHashesInCatalog*, *CiFindPageHashesInSignedFile*, *CiFreePolicyInfo*, *CiGetPEInformation*, *CiInitialize*, *CiValidateFileObject*, and *CiVerifyHashInCatalog*. However, when initialized, it exports additional 18 functions in the form of callback functions. Once they are exported, the normal kernel can invoke these functions. The exported callback functions are: *CiValidateImageHeader*, *CiValidateImageData*, *CiQueryInformation*, *CiSetFileCache*, *CiGetFileCache*, *CiHashMemory*, *KappxIsPackageFile*, *CiCompareSigningLevels*, *CiValidateFileAsImageType*, *CiRegisterSigningInformation*, *CiUnregisterSigningInformation*, *CiInitializePolicy*, *SIPolicyQueryPolicyInformation*, *CiValidateDynamicCodePages*, *SIPolicyQuerySecurityPolicy*, *CiGetStrongImageReference*, *CiReleaseContext*, and *CiHvciSetImageBaseAddress*.

The kernel invokes the *SepInitializeCodeIntegrity* and *CiInitialize* functions in order to initialize code integrity. This function exports the previously mentioned callback functions. Figure 1 depicts several of these functions. The functions depicted in Figure 1 are extracted from the execution context of the normal kernel. Figure 2 depicts the kernel invoking the *CiInitializePolicy* callback function after the function has been exported to it.

skci.dll statically exports 9 functions. These functions are: *SkciCreateCodeCatalog*, *SkciCreateSecureImage*, *SkciFinalizeSecureImageHash*, *SkciFinishImageValidation*, *SkciFreeImageContext*, *SkciInitialize*, *SkciTransferVersionResource*, *SkciValidateDynamicCodePages*, and *SkciValidateImageData*.

The functions exported by *skci.dll* are invoked when Windows 10 routes code integrity functionalities from the normal to the secure kernel. This work provides an overview of the invocation paths of the functions exported by *skci.dll*, from the triggering of their invocation in the context of the normal environment (Section 3), to their execution in the context of the secure environment (Section 2).

```
kd> .for (r $t0=1;@$t0<=18;r $t0=@$t0+1){ ln poi(nt!SeCiCallbacks+8*@$t0) }
[...]
(fffff800`d2ef4a50)  CI!CiSetFileCache  |  (fffff800`d2ef4eb4)   CI!CipGetFileCache
Exact matches:
    CI!CiSetFileCache (<no parameter info>)
[...]
(fffff800`d2ef51d0)  CI!CiGetFileCache  |  (fffff800`d2ef5220)   CI!CiGetCatalogHint
Exact matches:
    CI!CiGetFileCache (<no parameter info>)
[...]
(fffff800`d2ef2c70)  CI!CiQueryInformation  |  (fffff800`d2ef2e1c)   CI!CipCheckConfigOptions
Exact matches:
    CI!CiQueryInformation (<no parameter info>)
[...]
(fffff800`d2efb400)  CI!CiValidateImageHeader  |  (fffff800`d2efbe10)   CI!CiValidateImageData
Exact matches:
    CI!CiValidateImageHeader (<no parameter info>)
[...]
(fffff800`d2efbe10)  CI!CiValidateImageData  |  (fffff800`d2efbf00)   CI!CiValidateDynamicCodePages
Exact matches:
    CI!CiValidateImageData (<no parameter info>)
[...]
```

Figure 1: Code integrity callback functions

```
kd> bp CI!CiInitializePolicy
[...]
Breakpoint 2 hit
CI!CiInitializePolicy:
fffff800`d2ef2130 48895c2420      mov     qword ptr [rsp+20h],rbx
kd> kc
 # Call Site
00 CI!CiInitializePolicy
01 nt!SeCodeIntegrityInitializePolicy
02 nt!Phase1InitializationDiscard
03 nt!Phase1Initialization
04 nt!PspSystemThreadStartup
05 nt!KiStartSystemThread
```

Figure 2: The kernel invoking the exported callback function *CIInitializePolicy*

2 Secure Environment

In the context of the secure kernel, the functions exported by *skci.dll* are primarily invoked by functions with the prefix *Skm* (an exception is the *SkInitSystem* function). The column 'skci.dll' of Table 2 lists the functions exported by *skci.dll*. The column 'Secure kernel' of this table lists the functions that invoke the functions exported by *skci.dll* – functions with prefix *Skm* and *SkInitSystem*.

The functions with prefix *Skm* and *SkInitSystem* are primarily invoked when the secure kernel processes specific secure services. *SkinitSystem* is also invoked during the initialization of the secure kernel, by the *SkiSystemStartupFunction*. Secure services are requested by the normal environment. They can be uniquely identified by their secure service call numbers (SSCNs). The column 'SSCN' in Table 2 lists the SSCNs of the secure services that execute the functions with prefix *Skm*, or *SkInitSystem*. These functions are invoked in the *IumInvokeSecureService* function, which is where secure service requests are processed. *SkmiDeleteImage* is invoked through

a function pointer. Therefore, this function cannot be associated with a specific SSCN based on static analysis only (/ in Table 2).

3 Normal Environment

In order to invoke functions implemented in *skci.dll*, the normal kernel and functions implemented in *ci.dll* request secure services. In order to request a secure service, they use the *g_CiVslHvciInterface* variable implemented in *ci.dll*. This variable stores pointers to functions implemented in the normal kernel. These functions have names with the prefix *Vsl*. They invoke the *VslpEnterIumSecureMode* function. This function requests secure services from the secure kernel by issuing Virtual Trust Level (VTL) calls. The SSCNs of requested secure services are stored as the second parameter of *VslpEnterIumSecureMode*. Table 1 lists the positions, or offsets, in the *g_CiVslHvciInterface* variable (column 'g_CiVslvciInterface position') at which pointers to functions with the prefix *Vsl* are stored (column 'Function'). The column 'SSCN' of Table 1 lists the SSCNs identifying the secure services requested by the functions with the prefix *Vsl*.

g_CiVslHvciInterface position	Function	SSCN
g_CiVslHvciInterface	*VslCreateSecureAllocation*	*0x13*
g_CiVslHvciInterface + 0x08	*VslFillSecureAllocation*	*0x14*
g_CiVslHvciInterface + 0x10	*VslMakeCodeCatalog*	*0x15*
g_CiVslHvciInterface + 0x18	*VslCreateSecureImageSection*	*0x16*
g_CiVslHvciInterface + 0x20	*VslValidateSecureImagePages*	*0xC1*
g_CiVslHvciInterface + 0x28	*VslFinalizeSecureImageHash*	*0x17*
g_CiVslHvciInterface + 0x30	*VslFinishSecureImageValidation*	*0x18*
g_CiVslHvciInterface + 0x38	*VslPrepareSecureImageRelocations*	*0x19*
g_CiVslHvciInterface + 0x40	*VslRelocateImage*	*0x1A*
g_CiVslHvciInterface + 0x48	*VslCloseSecureHandle*	*0x1B*
g_CiVslHvciInterface + 0x50	*VslGetNestedPageProtectionFlags*	*0xE7*
g_CiVslHvciInterface + 0x58	*VslValidateDynamicCodePages*	*0x1C*
g_CiVslHvciInterface + 0x60	*VslTransferSecureImageVersionResource*	*0x1D*

Table 1: Functions referenced by *g_CiVslHvciInterface*

The invocation of the functions referenced by *g_CiVslHvciInterface* is protected by ControlFlowGuard. Figure 3 depicts an invocation of the function referenced at offset *0x20* of *g_CiVslHvciInterface – VslValidateSecureImagePages* (see Table 1). In accordance with the design of ControlFlowGuard, the *rax* register, at the time the *_guard_dispatch_icall_fptr* function is invoked, points to the function that is ultimately invoked. Therefore, the places where the functions referenced by *g_CiVslHvciInterface* are invoked can be identifed by searching for invocations of *_guard_dispatch_icall_fptr* in the implementations of *ci.dll* and the normal kernel, such that the *rax* register points at a given offset of *g_CiVslHvciInterface*.

```
CI!CiHvciValidateImageData+0x93:
fffff80d`c6c16643 b900000000      mov     ecx,0
fffff80d`c6c16648 488bf3          mov     rsi,rbx
[...]
fffff80d`c6c1668b 488b05ce35feff  mov     rax,qword ptr [CI!g_CiVslHvciInterface+0x20 (fffff80d`c6bf9c60)]
fffff80d`c6c16692 03d6            add     edx,esi
[...]
fffff80d`c6c166a1 ff15c98ffeff    call    qword ptr [CI!_guard_dispatch_icall_fptr (fffff80d`c6bff670)]
fffff80d`c6c166a7 428b94f59c000000 mov     edx,dword ptr [rbp+r14*8+9Ch]
[...]
```

Figure 3: ControlFlowGuard protecting functions referenced by *g_CiVslHvciInterface*

A brief analysis revealed that most of the functions referenced by *g_CiVslHvciInterface* are invoked by functions with the prefix *CiHvci*. These functions are implemented in *ci.dll*.

The column 'ci.dll' of Table 2 lists the functions with prefix *CiHvci* that ultimately trigger the execution of functions exported by *skci.dll* (the → symbol marks function invocation). The column 'Normal kernel' of this table lists functions implemented in the normal kernel that request secure services in order to trigger the execution of functions exported by *skci.dll*. Some of these functions are referenced by the *g_CiVslHvciInterface* variable and are invoked by the functions with prefix *CiHvci* (see Table 1). Others are invoked directly by the normal kernel, such as *VslCreateSecureImageSection*. The column 'SSCN' of Table 2 lists the SSCNs identifying the secure services requested by the functions listed in the column 'Normal kernel' of this table. The execution of these services in the context of the secure kernel results in the execution of functions exported by *skci.dll*.

ci.dll	Normal kernel	SSCN	Secure kernel	skci.dll
	VslpInumPhase4Initialize	*0x1*	*SkInitSystem*	*SkciInitialize*
	VslpInumPhase0Initialize	*0xD0*		
	VslMakeCodeCatalog	*0x15*	*SkmmConvertSecureAllocationToCatalog*	*SkciCreateCodeCatalog*
CiHvciCalculateHeaderHash *CiHvciCalculateImageHash*	*VslCreateSecureImageSection*	*0x16*	*SkmmCreateSecureImageSection*	*SkciCreateSecureImage*
CiHvciAddNonSectionDataToFileHash *CiHvciCalculateImageHash* *CiHvciValidateImageData*	*VslValidateSecureImagePages*	*0xC1*	*SkmmValidateSecureImagePages*	*SkciValidateImageData*
CiHvciTransferRelocationInformation	*VslPrepareSecureImageRelocations*	*0x19*	*SkmmPrepareImageRelocations*	
CiHvciValidateDynamicCodePages	*VslValidateDynamicCodePages*	*0x1C*	*SkmmValidateDynamicCodePages*	*SkciValidateDynamicCodePages*
CiHvciCalculateHeaderHash *CiHvciCalculateImageHash*	*VslFinalizeSecureImageHash*	*0x17*	*SkmmFinalizeSecureImageHash*	*SkciFinalizeSecureImageHash*
CiHvciVerifyFileHashInCatalogs → *CipHvciVerifyHashInCatalogs* *CiHvciVerifyPageHashInCatalogs* → *CipHvciVerifyHashInCatalogs* *CiHvciVerifyFileHashSignedFile*	*VslFinishSecureImageValidation*	*0x18*	*SkmmFinishSecureImageValidation*	*SkciFinishImageValidation*
	VslCreateSecureImageSection	*0x16*	*SkmmCreateSecureImageSection*	*SkciFreeImageContext*
		/	*SkmiDeleteImage*	
CiHvciSetFileVersionInformation	*VslTransferSecureImageVersionResource*	*0x1D*	*SkmmTransferImageVersionResource*	*SkciTransferVersionResource*

Table 2: *ci.dll* and *skci.dll*: Invocation paths to functions statically exported by *skci.dll*