Software Licensing Guidance for Products that Use Process Patching and Hooking

February 4, 2008 **‑** Version 1.1b

Abstract

The Windows Vista® and Windows Server® 2008 operating systems introduced a new software licensing service—Slsvc.exe—that manages all activity that is related to Windows® licensing. This software licensing service implements several antitampering technologies that are designed to detect modifications to the service.

This paper provides guidance to help providers of antivirus, antispyware, and other software products to understand the behavior of the software licensing service in Windows Vista and Windows Server 2008 and to adhere to the guidance for interactions with this service.

This information applies for the following operating systems:
 Windows Server 2008
 Windows Vista

For the latest information, see:
 [http://www.microsoft.com/whdc/system/vista/SLSVCguide.mspx](http://www.microsoft.com/whdc/system/vista/SLSVCguide_Vista.mspx)

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

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

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

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

© 2006–2008 Microsoft Corporation. All rights reserved.

Microsoft, Windows, Windows Server, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

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

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Change |  |  |  |
| 02/04/2008 | Updated to include Windows Server 2008 |
| 12/13/2006 | First publication |

Contents

[Introduction to the Software Licensing Service 3](#_Toc189888978)

[Guidance 3](#_Toc189888979)

[How to Identify the Software Licensing Service 4](#_Toc189888980)

[Contact Information 4](#_Toc189888981)

[Appendix – Sample C Code 5](#_Toc189888982)

# Introduction to the Software Licensing Service

The Windows Vista® and Windows Server® 2008 operating systems introduced a new software licensing service—Slsvc.exe—that manages all activity that is related to Windows® licensing. This software licensing service implements several antitampering technologies that are designed to detect modifications to the service. This paper provides guidance to software vendors for interactions with this software licensing service.

Important: When modifications are detected, the licensing state of the system is affected and the user might be notified of this change in the licensing state. For example, Windows may display a notification that states: “An unauthorized change was made to your license,” which indicates that the Windows operating system on that computer is now in a nongenuine state.

# Guidance

The guidance in this paper is intended to help antivirus, antispyware, and other software product providers to understand the behavior of the software licensing service in Windows Vista and Windows Server 2008 and to adhere to the guidance for interactions with that service.

On-Disk Image and Run-Time Image Memory Guidance. The specific guidance is as follows:

* For all Microsoft binaries, you must not change any on-disk image.
* Important: We do not recommend and do not support changing any operating system binary in memory. This guidance applies to all running processes on a system.
* For the software licensing service process, you must not change any run-time memory. This includes changing memory images of loaded binaries, changing API entry points (such as Import Address Table), and so on.

You can still check and verify that no malware is running in the software licensing service process.

Examples of Disallowed Operations. The following operations are examples of those not allowed:

* Changing the on-disk image of Ntdll.dll.
* Changing the first 5 bytes of the CreateProcess function in memory in the software licensing service.
* Modifying in memory the Import Address Table entry of User32.dll in the software licensing service.

Guidance for Hooking Processes. If your software hooks or injects code into system processes, follow this guidance:

* If you are using the AppInit\_DLLs mechanism to load your DLL, then no action is required. Your DLL will not be loaded into the software licensing service process.
* If you are not using the AppInit\_DLLs mechanism, then you should add code to exclude the hooking or injection from the software licensing service process. The following section describes how to identify the software licensing service process.

# How to Identify the Software Licensing Service

Identify the software licensing service (Slsvc.exe) by using the service security identifier (SID). The SID is as follows:

L"S-1-5-80-2119565420-4155874467-2934723793-509086461-374458824"

Before injecting code into a process, check the identity of that process. If it matches the identity of the software licensing service, then abort your injection operation.

The appendix provides sample C source code that shows how to check whether the process is the software licensing service.

# Contact Information

If you represent a provider of antivirus, antispyware, or other software products and require additional information, contact:
 slsup@microsoft.com

# Appendix – Sample C Code

This sample C code shows how to check whether a process is the software licensing service.

Note:

* To perform the actions shown in the sample code, your process must run as “local system.”
* The following is sample code and is not intended to compile or run as is.

#include <windows.h>

#include <stdio.h>

#include <stdlib.h>

#ifndef SECURITY\_SERVICE\_ID\_BASE\_RID

// defines taken from the Vista SDK/WDK.

#define SECURITY\_SERVICE\_ID\_BASE\_RID (0x00000050L)

#define SECURITY\_SERVICE\_ID\_RID\_COUNT (6L)

#endif

BOOL

CheckSLSvc(

 HANDLE hProcess // handle needs to be opened for PROCESS\_QUERY\_INFORMATION\_ACCESS

 )

{

 HANDLE hProcessToken = NULL;

 HANDLE hImpersonationToken = NULL;

 SID\_IDENTIFIER\_AUTHORITY IDAuthorityNT = SECURITY\_NT\_AUTHORITY;

 PSID pSidToCheck = NULL;

#define MEMBER\_ACCESS 1

 SECURITY\_DESCRIPTOR SecDesc;

 PACL pDacl = NULL;

 ULONG DaclSize;

 GENERIC\_MAPPING GenericMapping = {

 STANDARD\_RIGHTS\_READ | MEMBER\_ACCESS,

 STANDARD\_RIGHTS\_EXECUTE,

 STANDARD\_RIGHTS\_WRITE,

 STANDARD\_RIGHTS\_ALL | MEMBER\_ACCESS};

 BYTE PrivilegeSetBuffer[sizeof(PRIVILEGE\_SET) + 3\*sizeof(LUID\_AND\_ATTRIBUTES)];

 PPRIVILEGE\_SET PrivilegeSet = (PPRIVILEGE\_SET) PrivilegeSetBuffer;

 ULONG PrivilegeSetLength = sizeof(PrivilegeSetBuffer);

 ACCESS\_MASK AccessGranted = 0;

 BOOL AccessStatus = FALSE;

 BOOL fServiceIdentity = FALSE;

 //

 // always check against the process token.

 //

 // the caller must be running as LocalSystem for this to succeed against another

 //process. (process != GetCurrentProcess)

 //

 if(!OpenProcessToken(hProcess, TOKEN\_QUERY | TOKEN\_DUPLICATE, &hProcessToken))

 {

 goto Cleanup;

 }

 //

 // duplicate the process token to an impersonation token for the AccessCheck call.

 //

 if(!DuplicateToken( hProcessToken, SecurityImpersonation, &hImpersonationToken ))

 {

 goto Cleanup;

 }

 //

 // allocate the SID(s).

 //

 //

 // NT SERVICE\slsvc

 // S-1-5-80-2119565420-4155874467-2934723793-509086461-374458824

 //

 if(!AllocateAndInitializeSid(

 &IDAuthorityNT,

 SECURITY\_SERVICE\_ID\_RID\_COUNT,

 SECURITY\_SERVICE\_ID\_BASE\_RID,

 2119565420,

 4155874467,

 2934723793,

 509086461,

 374458824,

 0, 0,

 &pSidToCheck

 ))

 {

 goto Cleanup;

 }

 //

 // Construct a security descriptor to pass to access check

 //

 //

 // The size is equal to the size of twice the length of the SID

 // (for owner and group) + size of the DACL + sizeof ACL + size of the

 // ACE, which is an ACE + length of

 // ths SID.

 //

 DaclSize = sizeof(ACL) +

 (1 \* sizeof(ACCESS\_ALLOWED\_ACE)) + // 1 per SID

 (3 \* GetLengthSid(pSidToCheck)); // 1 per SID + 2 total for group+owner

 pDacl = (PACL) LocalAlloc( LMEM\_ZEROINIT, DaclSize );

 if( pDacl == NULL )

 {

 goto Cleanup;

 }

 if(!InitializeSecurityDescriptor(&SecDesc, SECURITY\_DESCRIPTOR\_REVISION))

 {

 goto Cleanup;

 }

 //

 // Fill in fields of security descriptor

 //

 if(!SetSecurityDescriptorOwner(&SecDesc, pSidToCheck, FALSE))

 {

 goto Cleanup;

 }

 if(!SetSecurityDescriptorGroup(&SecDesc, pSidToCheck, FALSE))

 {

 goto Cleanup;

 }

 if(!InitializeAcl( pDacl, DaclSize, ACL\_REVISION ))

 {

 goto Cleanup;

 }

 //

 // add the SID(s) to the ACL.

 //

 if(!AddAccessAllowedAce(

 pDacl,

 ACL\_REVISION,

 MEMBER\_ACCESS,

 pSidToCheck

 ))

 {

 goto Cleanup;

 }

 //

 // Set the DACL on the security descriptor

 //

 if(!SetSecurityDescriptorDacl( &SecDesc, TRUE, pDacl, FALSE ))

 {

 goto Cleanup;

 }

 if(!AccessCheck(

 &SecDesc,

 hImpersonationToken,

 MEMBER\_ACCESS,

 &GenericMapping,

 PrivilegeSet,

 &PrivilegeSetLength,

 &AccessGranted,

 &AccessStatus

 ))

 {

 goto Cleanup;

 }

 //

 // if the access check failed, then the sid(s) are not a member of the

 // token

 //

 if ((AccessStatus) && (AccessGranted == MEMBER\_ACCESS))

 {

 fServiceIdentity = TRUE;

 }

Cleanup:

 if( hProcessToken )

 {

 CloseHandle( hProcessToken );

 }

 if( hImpersonationToken )

 {

 CloseHandle( hImpersonationToken );

 }

 //

 // free the allocated SID(s).

 //

 if( pSidToCheck )

 {

 FreeSid( pSidToCheck );

 }

 if( pDacl )

 {

 LocalFree( pDacl );

 }

 return fServiceIdentity;

}

BOOL SetPrivilege(

 HANDLE hToken, // token handle

 LPCTSTR Privilege, // Privilege to enable/disable

 BOOL bEnablePrivilege // TRUE to enable. FALSE to disable

)

{

 TOKEN\_PRIVILEGES tp = { 0 };

 // Initialize everything to zero

 LUID luid;

 DWORD cb=sizeof(TOKEN\_PRIVILEGES);

 memset(&tp, 0, sizeof(tp));

 if(!LookupPrivilegeValue( NULL, Privilege, &luid ))

 return FALSE;

 tp.PrivilegeCount = 1;

 tp.Privileges[0].Luid = luid;

 if(bEnablePrivilege) {

 tp.Privileges[0].Attributes = SE\_PRIVILEGE\_ENABLED;

 } else {

 tp.Privileges[0].Attributes = 0;

 }

 AdjustTokenPrivileges( hToken, FALSE, &tp, cb, NULL, NULL );

 if (GetLastError() != ERROR\_SUCCESS)

 return FALSE;

 return TRUE;

}

int

\_\_cdecl

main(

 int argc,

 char \*argv[]

 )

{

 HANDLE hToken = NULL;

 HANDLE hProcess = NULL;

 DWORD dwProcessID;

 BOOL fAssertDebug = FALSE;

 BOOL fSLSvc = FALSE;

 if(argc == 2)

 {

 dwProcessID = atoi(argv[1]);

 fAssertDebug = TRUE;

 } else {

 // default to current process.

 dwProcessID = GetCurrentProcessId();

 }

 // assert the SeDebugPrivilege, if you are opening another process...

 if(fAssertDebug)

 {

 if(!OpenThreadToken(GetCurrentThread(), TOKEN\_ADJUST\_PRIVILEGES | TOKEN\_QUERY,

 FALSE, &hToken))

 {

 if (GetLastError() == ERROR\_NO\_TOKEN)

 {

 if (!ImpersonateSelf(SecurityImpersonation))

 {

 return 13;

 }

 if(!OpenThreadToken(GetCurrentThread(), TOKEN\_ADJUST\_PRIVILEGES |

 TOKEN\_QUERY, FALSE, &hToken))

 {

 return 13;

 }

 } else {

 return 13;

 }

 }

 if(!SetPrivilege(hToken, SE\_DEBUG\_NAME, TRUE))

 {

 printf("Failed to assert SeDebugPrivilege! errorcode=%lu\n", GetLastError());

 // fall through and try anyway.

 }

 }

 //

 hProcess = OpenProcess(PROCESS\_QUERY\_INFORMATION, FALSE, dwProcessID);

 if( hProcess == NULL )

 {

 printf("OpenProcess error! errorcode=%lu\n", GetLastError());

 return 1;

 }

 if(fAssertDebug)

 {

 RevertToSelf();

 }

 fSLSvc = CheckSLSvc(hProcess);

 if(fSLSvc == TRUE)

 {

 printf("PID %lu is SLSvc!\n", dwProcessID);

 } else {

 printf("PID %lu is NOT SLSvc!\n", dwProcessID);

 }

 CloseHandle(hProcess);

 return 0;

}