Developer audience, and sample code

Article • 12/17/2024

The Antimalware Scan Interface is designed for use by two groups of developers.

- Application developers who want to make requests to antimalware products from within their apps.
- Third-party creators of antimalware products who want their products to offer the best features to applications.

Application developers

AMSI is designed in particular to combat "fileless malware". Application types that can optimally leverage AMSI technology include script engines, applications that need memory buffers to be scanned before using them, and applications that process files that can contain non-PE executable code (such as Microsoft Word and Excel macros, or PDF documents). However, the usefulness of AMSI technology is not limited to those examples.

There are two ways in which you can interface with AMSI in your application.

- By using the AMSI Win32 APIs. See Antimalware Scan Interface (AMSI) functions.
- By using the AMSI COM interfaces. See **IAmsiStream** interface.

For sample code showing how to consume AMSI within your COM application, see the IAmsiStream interface sample application .

Third-party creators of antimalware products

As a creator of antimalware products, you can choose to author and register your own in-process COM server (a DLL) to function as an AMSI provider. That AMSI provider must implement the **IAntimalwareProvider** interface, and it must run in-process.

Note that, after Windows 10, version 1709 (the Fall 2017 Creators' Update), your AMSI provider DLL may not work if it depends upon other DLLs in its path to be loaded at the same time. To prevent DLL hijacking, we recommend that your provider DLL load its dependencies explicitly (with a full path) using secure **LoadLibrary** calls, or equivalent. We recommend this instead of relying on the **LoadLibrary** search behavior.

The section below shows how to register your AMSI provider. For full sample code showing how to author your own AMSI provider DLL, see the IAntimalwareProvider interface sample application .

Register your provider DLL with AMSI

To begin with, you need to ensure that these Windows Registry keys exist.

- HKLM\SOFTWARE\Microsoft\AMSI\Providers
- HKLM\SOFTWARE\Classes\CLSID

An AMSI provider is an in-process COM server. Consequently, it needs to register itself with COM. COM classes are registered in `HKLM\SOFTWARE\Classes\CLSID`.

The code below shows how to register an AMSI provider, whose GUID (for this example) we will assume is `2E5D8A62-77F9-4F7B-A90C-2744820139B2`.

C++

```
#include <strsafe.h>
...
HRESULT SetKeyStringValue(_In_ HKEY key, _In_opt_ PCWSTR subkey, _In_opt_
PCWSTR valueName, _In_ PCWSTR stringValue)
{
    LONG status = RegSetKeyValue(key, subkey, valueName, REG_SZ, stringValue,
(wcslen(stringValue) + 1) * sizeof(wchar_t));
    return HRESULT_FROM_WIN32(status);
}

STDAPI DllRegisterServer()
{
    wchar_t modulePath[MAX_PATH];
    if (GetModuleFileName(g_currentModule, modulePath, ARRAYSIZE(modulePath))
>= ARRAYSIZE(modulePath))
    {
        return E_UNEXPECTED;
    }

    // Create a standard COM registration for our CLSID.
    // The class must be registered as "Both" threading model,
    // and support multithreaded access.
    wchar_t clsidString[40];
    if (StringFromGUID2(__uuidof(SampleAmsiProvider), clsidString,
ARRAYSIZE(clsidString)) == 0)
    {
        return E_UNEXPECTED;
```

```
    }

    wchar_t keyPath[200];
    HRESULT hr = StringCchPrintf(keyPath, ARRAYSIZE(keyPath),
L"Software\\Classes\\CLSID\\%ls", clsidString);
    if (FAILED(hr)) return hr;

    hr = SetKeyStringValue(HKEY_LOCAL_MACHINE, keyPath, nullptr,
L"SampleAmsiProvider");
    if (FAILED(hr)) return hr;

    hr = StringCchPrintf(keyPath, ARRAYSIZE(keyPath),
L"Software\\Classes\\CLSID\\%ls\\InProcServer32", clsidString);
    if (FAILED(hr)) return hr;

    hr = SetKeyStringValue(HKEY_LOCAL_MACHINE, keyPath, nullptr, modulePath);
    if (FAILED(hr)) return hr;

    hr = SetKeyStringValue(HKEY_LOCAL_MACHINE, keyPath, L"ThreadingModel",
L"Both");
    if (FAILED(hr)) return hr;

    // Register this CLSID as an anti-malware provider.
    hr = StringCchPrintf(keyPath, ARRAYSIZE(keyPath),
L"Software\\Microsoft\\AMSI\\Providers\\%ls", clsidString);
    if (FAILED(hr)) return hr;

    hr = SetKeyStringValue(HKEY_LOCAL_MACHINE, keyPath, nullptr,
L"SampleAmsiProvider");
    if (FAILED(hr)) return hr;

    return S_OK;
}
```

If your DLL implements the DllRegisterServer function, as the example above does, then you can register it by using the Windows-supplied executable `regsvr32.exe`. From an elevated command prompt, issue a command of this form.

Windows Command Prompt

```
C:>C:\Windows\System32\regsvr32.exe SampleAmsiProvider.dll
```

In this example, the command creates the following entries.

HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{2E5D8A62-77F9-4F7B-A90C-2744820139B2}

(Default) REG_SZ Sample AMSI Provider implementation

HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{2E5D8A62-77F9-4F7B-A90C-2744820139B2}\InprocServer32

(Default) REG_EXPAND_SZ %ProgramFiles%\TestProvider\SampleAmsiProvider.dll

ThreadingModel REG_SZ Both

In addition to regular COM registration, you also need to enroll the provider with AMSI. You do that by adding an entry to the following key.

HKLM\SOFTWARE\Microsoft\AMSI\Providers

For example,

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\AMSI\Providers\{2E5D8A62-77F9-4F7B-A90C-2744820139B2}

Known issues

Process didn't meet signing level requirements

If you have a non-Microsoft antimalware service that's Windows Protected Process Light (PPL) or Antimalware Protected Process Light (Anti-malware PPL) that tries to load in an AMSI provider, you might see the following information in the Code Integrity event log:

properties

```
Log Name: Microsoft-Windows-CodeIntegrity/Operational

Source: Microsoft-Windows-CodeIntegrity

Event ID: 3033

Description:

Code Integrity determined that a process (\Device\HarddiskVolume3\<Folder>\
<Folder w/ the ISV name>\<Folder w/ the product name>\<ProcessName>.exe) at-
tempted to load \Device\HarddiskVolume3\<Folder>\<Folder w/ the ISV name>\
<Folder w/ the product name>\<Your Amsi Provider>.dll that did not meet the
```

```
Custom 3 / Antimalware signing level requirements.
```

To view the Code Integrity event log, follow these steps:

1. Open Event Viewer.
2. In the navigation pane, expand **Applications and Services Logs** > **Microsoft** > **Windows** > **Code Integrity**, and then select **Operational**.

Or if you have System Audit Integrity auditing enabled, look for this:

- Log Name: Security
- Source: Microsoft Windows Security
- Event ID: `5038`

A file hash isn't valid

AMSI API's are designed to work with non-protected processes. ISV's are unable to sign their AMSI registered DLL's to be allowed to load into ELAM/PPL secured processes. In such cases, you might see the following information in the Windows Security event log:

properties

```
Description:

Code integrity determined that the image hash of a file is not valid.  The file
could be corrupt due to unauthorized modification, or the invalid hash could
indicate a potential disk device error.

File Name: \Device\HarddiskVolume3\<Folder> \<Folder w/ the ISV name> \<Folder
w/ the product name>\<Your Amsi Provider>.dll
```

To view the Windows Security event log, follow these steps:

1. Open Event Viewer.
2. In the Navigation pane, expand **Windows Logs**, and then select **Security**.

Workaround:

You can filter out events by following these steps:

1. To filter out an event, such as Event ID 3033 or 5038, open Event Viewer.
2. In the navigation pane, expand **Applications and Services Logs** > **Microsoft** > **Windows** > **Code Integrity**, and then select **Operational**.
3. Right-click **Operational**, and then select **Filter Current Log...**.
4. In the **<All Event IDs>** box, type `-3033` (or `-5038`), and then select **OK**.

Or, in Event Viewer, you can expand Windows Logs, right-click Security, select **Filter Current Log...**, and then specify `-3033` or `-5038`.

> **Tip**
>
> If you are using Windows Event Forwarding (WEF), you can filter details what event ids or descriptions are forwarded. For more information, see **Tech Community Blog: Advanced XML filtering in the Windows Event Viewer**

Filter out Events 3033 and 5038 for your SIEM

If your organization is using a SIEM server, make sure to filter out Event ID 3033 and/or Event ID 5038 specific to AMSI so that you don't ingest information that isn't useful for your Security Operations Center (SOC) analysts. For more information, see Use Windows Event Forwarding to help with intrusion detection.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A