Product Solutions Resources Open Source Enterprise Pricing Sign in Sign up

gtworek / **PSBits** Public

Notifications Fork 525 Star 3.2k

Code Issues Pull requests Actions Projects Security Insights

PSBits / IFilter / **Dll.cpp**

gtworek Source code added e6905d7 · 3 years ago History

```
// THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
// ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
// THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
// PARTICULAR PURPOSE.
//
// Copyright (c) Microsoft Corporation. All rights reserved

// code is based on the https://docs.microsoft.com/en-us/samples/microsoft/windows-clas
// the original license may be seen at https://github.com/microsoft/Windows-classic-sam

#ifndef UNICODE
#error Unicode environment required. Some day, I will fix, if anyone needs it.
#endif

#include <Windows.h>
#include <new>
#include <Shlwapi.h>
#include <tchar.h>
#include <Psapi.h>

#define LOGFUNCTION TCHAR _strMsg[1024] = { 0 };\
                                        _stprintf_s(_strMsg, _countof(_strMsg), TEXT("[
                                        if (_tcslen(_strMsg) > 0)\
                                        {\
                                                OutputDebugString(_strMsg);\
                                        }

#define SZ_FILTERSAMPLE_CLSID L"{AF9925E4-9A8A-4927-994E-EFC65F2EC6DF}"
#define SZ_FILTERSAMPLE_HANDLER L"{80423D65-47FF-4C4E-B7BD-C91627824A93}"

#define USERNAME_LENGTH 512
#define DOMAINNAME_LENGTH 512

HRESULT CFilterSample_CreateInstance(REFIID riid, void** ppv);

// Handle to the DLL's module
HINSTANCE g_hInst = NULL;

// Module Ref count
long c_cRefModule = 0;


BOOL GetProcessUsername(HANDLE hProcess, LPTSTR lpUserName)
{
        HANDLE hToken = nullptr;
        PTOKEN_USER ptuTokenInformation = nullptr;
        DWORD dwTokenLength;
        DWORD dwUserNameLen = USERNAME_LENGTH;
        DWORD dwDomainNameLen = DOMAINNAME_LENGTH;
        TCHAR szUserName[USERNAME_LENGTH];
        TCHAR szDomainName[DOMAINNAME_LENGTH];
        SID_NAME_USE snuSidUse;
        TCHAR strNameBuf[USERNAME_LENGTH + 1 + DOMAINNAME_LENGTH] = {0};

        if (!OpenProcessToken(hProcess, TOKEN_QUERY, &hToken))
        {
                _tcscpy_s(strNameBuf, _countof(strNameBuf), TEXT("UNKNOWN"));
                _tcscpy_s(strNameBuf, _countof(strNameBuf), TEXT("UNKNOWN"));
                lpUserName = strNameBuf;
                return FALSE;
        }

        GetTokenInformation(hToken, TokenUser, nullptr, 0, &dwTokenLength);
        ptuTokenInformation = static_cast<PTOKEN_USER>(LocalAlloc(LPTR, dwTokenLength))
        if (nullptr == ptuTokenInformation)
        {
                CloseHandle(hToken);
                _tcscpy_s(strNameBuf, _countof(strNameBuf), TEXT("UNKNOWN"));
                lpUserName = strNameBuf;
                return FALSE;
        }

        if (!GetTokenInformation(hToken, TokenUser, ptuTokenInformation, dwTokenLength,
        {
                CloseHandle(hToken);
                LocalFree(ptuTokenInformation);
```

```
                                hr = CLASS_E_CLASSNOTAVAILABLE;
                        }
                }
                return hr;
        }

        IFACEMETHODIMP LockServer(BOOL bLock)
        {
                if (bLock)
                {
                        DllAddRef();
                }
                else
                {
                        DllRelease();
                }
                return S_OK;
        }

private:
        ~CClassFactory()
        {
                DllRelease();
        }

        long m_cRef;
        CLSID m_clsid;
};


// Standard DLL functions
STDAPI_(BOOL) DllMain(HINSTANCE hInstance, DWORD dwReason, void*)
{
        if (dwReason == DLL_PROCESS_ATTACH)
        {
                TCHAR strMsg[1024] = {0};
                TCHAR szFilePath[MAX_PATH] = {0};
                TCHAR szUserName[USERNAME_LENGTH + 1 + DOMAINNAME_LENGTH];

                GetProcessImageFileName(GetCurrentProcess(), szFilePath, MAX_PATH);
                GetProcessUsername(GetCurrentProcess(), szUserName);
                _stprintf_s(strMsg, _countof(strMsg), TEXT("[GTIFilter] USERNAME = %s,
                OutputDebugString(strMsg);
                g_hInst = hInstance;
                DisableThreadLibraryCalls(hInstance);
        }
        return TRUE;
}

__control_entrypoint(DllExport)
STDAPI DllCanUnloadNow(void)
{
        LOGFUNCTION;
        return (c_cRefModule == 0) ? S_OK : S_FALSE;
}


_Check_return_
STDAPI DllGetClassObject(_In_ REFCLSID clsid, _In_ REFIID riid, _Outptr_ LPVOID FAR* pp
{
        *ppv = NULL;
        CClassFactory* pClassFactory = new(std::nothrow) CClassFactory(clsid);
        HRESULT hr = pClassFactory ? S_OK : E_OUTOFMEMORY;
        if (SUCCEEDED(hr))
        {
                hr = pClassFactory->QueryInterface(riid, ppv);
                pClassFactory->Release();
        }
        return hr;
}

// A struct to hold the information required for a registry entry
struct REGISTRY_ENTRY
{
        HKEY hkeyRoot;
        PCWSTR pszKeyName;
        PCWSTR pszValueName;
        PCWSTR pszData;
};

// Creates a registry key (if needed) and sets the default value of the key
HRESULT CreateRegKeyAndSetValue(const REGISTRY_ENTRY* pRegistryEntry)
{
        LOGFUNCTION;
        HRESULT hr;
        HKEY hKey;

        LONG lRet = RegCreateKeyExW(pRegistryEntry->hkeyRoot, pRegistryEntry->pszKeyNam
                                    0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, N
        if (lRet != ERROR_SUCCESS)
        {
                hr = HRESULT_FROM_WIN32(lRet);
        }
        else
        {
                lRet = RegSetValueExW(hKey, pRegistryEntry->pszValueName, 0, REG_SZ,
                                      (LPBYTE)pRegistryEntry->pszData,
                                      ((DWORD)wcslen(pRegistryEntry->pszData) + 1) * si

                hr = HRESULT_FROM_WIN32(lRet);

```

Files

8d76789

Go to file

- AppLockerBypass
- CERTPL2Hosts
- CopyEAs
- DFIR
- DNS
- EnableAllParentPrivileges
- FMAPI

PSBits / IFilter / Dll.cpp

Top

Code Blame 346 lines (303 loc) · 8.86 KB

Raw

```
// Registers this COM server
STDAPI DllRegisterServer()
{
        LOGFUNCTION;
        HRESULT hr;
        WCHAR szModuleName[MAX_PATH];

        if (!GetModuleFileNameW(g_hInst, szModuleName, ARRAYSIZE(szModuleName)))
        {
                hr = HRESULT_FROM_WIN32(GetLastError());
        }
        else
        {
```

- FakeAMSI
- FakeCmdLine
- GPO
- GetWindowFlag
- HashSrv
- HideSnapshot
- IFilter
  - Dll.cpp
  - FilterBase.h
  - FilterSample.cpp
  - FilterSample.def
  - FilterSample.dll
  - README.md
- IOCTL_VOLSNAP_SET_MAX_DIFF...
- LSASecretDumper
- LoLBIN
- MSI_Payload
- Misc
- NTDSdiff
- NTFSObjectID
- NetstatWithTimestamps
- NoDLP
- NoRebootSvc
- NoRunDll
- NtPowerInformation
- OfflineSAM
- PasswordStealing
- ProcessMitigations
- RDPHoneyPot
- RegExport
- SIP
- ServerLevelPluginDll
- Services
- ShutdownPriority
- SuspendProcess

```
            // List of registry entries we want to create
            const REGISTRY_ENTRY rgRegistryEntries[] =
            {
                // RootKey             KeyName
                {HKEY_LOCAL_MACHINE, L"Software\\Classes\\CLSID\\" SZ_FILTERSAM
                {
                    HKEY_LOCAL_MACHINE, L"Software\\Classes\\CLSID\\" SZ_FI
                    szModuleName
                },
                {
                    HKEY_LOCAL_MACHINE, L"Software\\Classes\\CLSID\\" SZ_FI
                    L"ThreadingModel", L"Both"
                },
                {
                    HKEY_LOCAL_MACHINE, L"Software\\Classes\\CLSID\\" SZ_FI
                    L"Filter Sample Persistent Handler"
                },
                {
                    HKEY_LOCAL_MACHINE,
                    L"Software\\Classes\\CLSID\\" SZ_FILTERSAMPLE_HANDLER L
                },
                {
                    HKEY_LOCAL_MACHINE,
                    L"Software\\Classes\\CLSID\\" SZ_FILTERSAMPLE_HANDLER
                    L"\\PersistentAddinsRegistered\\{89BCB740-6119-101A-BCB
                },
                {HKEY_LOCAL_MACHINE, L"Software\\Classes\\.filtersample", NULL,
                {HKEY_LOCAL_MACHINE, L"Software\\Classes\\.filtersample\\Persis
            };
            hr = S_OK;
            for (int i = 0; i < ARRAYSIZE(rgRegistryEntries) && SUCCEEDED(hr); i++)
            {
                hr = CreateRegKeyAndSetValue(&rgRegistryEntries[i]);
            }
        }
        return hr;
    }

    // Unregisters this COM server
    STDAPI DllUnregisterServer()
    {
        LOGFUNCTION;
        HRESULT hr = S_OK;
        const PCWSTR rgpszKeys[] =
        {
            L"Software\\Classes\\CLSID\\" SZ_FILTERSAMPLE_CLSID,
            L"Software\\Classes\\CLSID\\" SZ_FILTERSAMPLE_HANDLER,
            L"Software\\Classes\\.filtersample"
        };

        // Delete the registry entries
        for (int i = 0; i < ARRAYSIZE(rgpszKeys) && SUCCEEDED(hr); i++)
        {
            DWORD dwError = RegDeleteTree(HKEY_LOCAL_MACHINE, rgpszKeys[i]);
            if (ERROR_FILE_NOT_FOUND == dwError)
            {
                // If the registry entry has already been deleted, say S_OK.
                hr = S_OK;
            }
            else
            {
                hr = HRESULT_FROM_WIN32(dwError);
            }
        }
        return hr;
    }
```