


```
1  #include <windows.h>
2  #include <stdio.h>
3  #include <tlhelp32.h>
4  #include <iostream>
5
6  using namespace std;
```

Code

Blame

54 lines (46 loc) · 1.77 KB

Raw



```
11  HANDLE hSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
12  PROCESSENTRY32W pe32;
13  pe32.dwSize = sizeof(PROCESSENTRY32W);
14  if (Process32FirstW(hSnap, &pe32) != FALSE) {
15      while (pid == 0 && Process32NextW(hSnap, &pe32) != FALSE) {
16          if (wcscmp(pe32.szExeFile, procName) == 0) {
17              pid = pe32.th32ProcessID;
18          }
19      }
20  }
21  CloseHandle(hSnap);
22  return pid;
23  }
24
25  ... bool DLLinjector(DWORD pid, const wchar_t* dllPath) {
26      typedef LPVOID memory_buffer;
27
28      HANDLE hProc = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pid);
29      if (hProc == NULL) {
30          cout << "OpenProcess() failed: " << GetLastError() << endl;
31          return false;
32      }
33
34      HMODULE hKernel32 = GetModuleHandleW(L"Kernel32");
35      FARPROC lb = GetProcAddress(hKernel32, "LoadLibraryW");
36      memory_buffer allocMem = VirtualAllocEx(hProc, NULL, wcslen(dllPath) * sizeof(wchar
37      if (allocMem == NULL) {
38          cout << "VirtualAllocEx() failed: " << GetLastError() << endl;
39          return false;
40      }
41      WriteProcessMemory(hProc, allocMem, dllPath, wcslen(dllPath) * sizeof(wchar_t), NUL
42      HANDLE rThread = CreateRemoteThread(hProc, NULL, 0, (LPTHREAD_START_ROUTINE)lb, all
43      if (rThread == NULL) {
44          cout << "CreateRemoteThread() failed: " << GetLastError() << endl;
45          return false;
46      }
47
48      cout << "Code Injected";
49
50      CloseHandle(hProc);
51      FreeLibrary(hKernel32);
52      VirtualFreeEx(hProc, allocMem, wcslen(dllPath) * sizeof(wchar_t), MEM_RELEASE);
53      return true;
54  }
```