Белорусский государственный университет

информатики и радиоэлектроники

Кафедра ПОИТ

Отчет по лабораторной работе №3

по дисциплине

«Операционные системы и системное программирование»

Выполнил:

Гладкий М.Г.

группа 851005

Проверил:

Деменковец Д.В.

Минск, 2020

**Исходный код программы**

Main.cpp

#include <iostream>  
#include <stdio.h>  
#include <new>  
#include <thread>  
#include <windows.h>  
  
#define PATH\_TO\_DLL "library.dll"  
#define FUNCTION\_NAME "FindString"  
#define PATH\_TO\_DLL\_FOR\_INJECTION "injectDll.dll"  
#define PATH\_TO\_KERNEL\_DLL "Kernel32.dll"  
#define NAME\_LOAD\_LIBRARY\_FUNCTION "LoadLibraryA"  
  
const char\* STRING = "Blank";  
const char\* REPLACE\_STRING = "ReplacedString123";  
  
typedef HMODULE(WINAPI\* LPLoadLibrary)(LPCSTR);  
  
typedef void ReplaceData(const char\* str, const char\* replaceStr);  
  
extern "C" \_declspec(dllimport) void FindString(const char\* str, const char\* replaceStr);  
  
char\* buf = new char[strlen(STRING) + 1];  
  
void useStaticConnection() {  
  
 FindString(STRING, REPLACE\_STRING);  
}  
  
void useDynamicConnection() {  
 HMODULE moduleDll = LoadLibraryA(PATH\_TO\_DLL);  
 if (moduleDll != NULL) {  
 ReplaceData\* FindStringFunc = (ReplaceData\*)GetProcAddress(moduleDll, FUNCTION\_NAME);  
 FindStringFunc(STRING, REPLACE\_STRING);  
 FreeLibrary(moduleDll);  
 }  
  
}  
  
void injectedDll(DWORD injectPid) {  
 HMODULE hdll = LoadLibraryA(PATH\_TO\_KERNEL\_DLL);  
 LPLoadLibrary LoadLibraryA = (LPLoadLibrary)GetProcAddress(hdll, NAME\_LOAD\_LIBRARY\_FUNCTION);  
 HANDLE hProc = OpenProcess(PROCESS\_VM\_READ | PROCESS\_VM\_OPERATION | PROCESS\_VM\_WRITE | PROCESS\_CREATE\_THREAD | PROCESS\_CREATE\_PROCESS, FALSE, injectPid);  
 LPVOID path = VirtualAllocEx(hProc, NULL, strlen(PATH\_TO\_DLL\_FOR\_INJECTION) + 1, MEM\_RESERVE | MEM\_COMMIT, PAGE\_READWRITE);  
 WriteProcessMemory(hProc, path, PATH\_TO\_DLL\_FOR\_INJECTION, strlen(PATH\_TO\_DLL\_FOR\_INJECTION) + 1, NULL);  
  
 DWORD threadID;  
  
 HANDLE HThread = CreateRemoteThread(hProc, NULL, NULL, (LPTHREAD\_START\_ROUTINE)LoadLibraryA, (LPVOID)path, NULL, &threadID);  
  
 Sleep(1000);  
  
  
  
 if (HThread == NULL)  
 {  
 printf("Error\n");  
 }  
 else {  
  
 WaitForSingleObject(HThread, INFINITE);  
 }  
  
 FreeLibrary(hdll);  
 CloseHandle(hProc);  
  
}  
  
void switchFunc(char mode)  
{  
 switch (mode)  
 {  
 case 0: {  
 DWORD injectPid;  
 printf("Enter process pid: ");  
 scanf("%d",&injectPid);  
 injectedDll(injectPid);  
 break;  
 }  
 case 1: {  
 printf("%s ",buf);  
 useStaticConnection();  
 printf(buf);  
 break;  
 }  
 case 2: {  
 printf("%s ",buf);  
 useDynamicConnection();  
 printf(buf);  
 break;  
 }  
 default: {  
 useStaticConnection();  
 break;  
 }  
 }  
}  
  
int main()  
{  
 strcpy\_s(buf, strlen(STRING) + 1, STRING);  
 int mode;  
 printf("Type of connection: 0 - injection; 1 - static; 2 - dynamic\n");  
 scanf("%d", &mode);  
 std::thread thread(switchFunc, mode);  
 thread.join();  
 return 0;  
}

Library.cpp

#include "library.h"  
#include <windows.h>  
#include <stdio.h>  
#include <new>  
#include <vector>  
  
void FindString(const char \*str, const char \*replaceStr) {  
 size\_t targetLen = strlen(str);  
 size\_t valueLen = strlen(replaceStr);  
  
 SYSTEM\_INFO si;  
 GetSystemInfo(&si);  
  
 MEMORY\_BASIC\_INFORMATION info;  
 auto baseAddress = (LPSTR) si.lpMinimumApplicationAddress;  
  
 while (baseAddress < si.lpMaximumApplicationAddress) {  
 if (VirtualQuery(baseAddress, &info, sizeof(info)) == sizeof(info)) {  
 if (info.State == MEM\_COMMIT && info.AllocationProtect == PAGE\_READWRITE) {  
 baseAddress = (LPSTR) info.BaseAddress;  
 char \*memory = (char \*) malloc(info.RegionSize);  
 SIZE\_T bytesRead;  
  
 if (ReadProcessMemory(GetCurrentProcess(), baseAddress, memory, info.RegionSize, &bytesRead)) {  
 for (SIZE\_T i = 0; i < bytesRead - targetLen; i++) {  
 if (strcmp(baseAddress + i, str) == 0) {  
 memcpy(baseAddress + i, replaceStr, valueLen + 1);  
 }  
 }  
 }  
 free(memory);  
 }  
 }  
 baseAddress += info.RegionSize;  
 }  
}  
  
#ifdef INJECT  
BOOL APIENTRY DllMain( HMODULE hModule,  
 DWORD ul\_reason\_for\_call,  
 LPVOID lpReserved  
)  
{  
  
 switch (ul\_reason\_for\_call)  
 {  
 case DLL\_PROCESS\_ATTACH:  
 case DLL\_THREAD\_ATTACH:  
 FindString("Blank", "ReplacedString123");  
 case DLL\_THREAD\_DETACH:  
 case DLL\_PROCESS\_DETACH:  
 break;  
 }  
 return TRUE;  
}  
#endif

Test.cpp

#include <iostream>  
#include <thread>  
#include <windows.h>  
  
int main()  
{  
 char text[] = "Blank";  
 long pid = GetCurrentProcessId();  
 printf("%d\n", pid);  
 int i = 0;  
 while (i < 1) {  
 printf("%s ",text);  
 std::this\_thread::sleep\_for(std::chrono::seconds(1));  
 }  
}

**Скриншоты выполнения программы**







