

To create a DLL injection code, you need to perform the following steps:

1. Open Visual Studio and create a new C++ Win32 Console Application project.

2. Add the following code to your project's main.cpp file:

```c++

#include <windows.h>

#include <iostream>

#include <TlHelp32.h>

using namespace std;

int main()

{

// Get the process ID of the target executable

DWORD processId = 0;

cout << "Enter the process ID of the target executable: ";

cin >> processId;

// Open the process for injection

HANDLE hProcess = OpenProcess(PROCESS\_ALL\_ACCESS, FALSE, processId);

if (hProcess == NULL)

{

cout << "Failed to open the process." << endl;

return 1;

}

// Get the base address of the target executable module

HMODULE hModule = GetModuleHandle("target.exe");

if (hModule == NULL)

{

cout << "Failed to get the module handle." << endl;

CloseHandle(hProcess);

return 1;

}

// Allocate memory within the target process

LPVOID lpRemoteBuffer = VirtualAllocEx(hProcess, NULL, sizeof(DWORD), MEM\_COMMIT, PAGE\_READWRITE);

if (lpRemoteBuffer == NULL)

{

cout << "Failed to allocate memory." << endl;

CloseHandle(hProcess);

return 1;

}

// Inject the DLL into the target process

FARPROC hLoadLibrary = GetProcAddress(GetModuleHandle("kernel32"), "LoadLibraryA");

if (hLoadLibrary == NULL)

{

cout << "Failed to get the LoadLibrary function address." << endl;

CloseHandle(hProcess);

return 1;

}

// Write the name of the DLL to the allocated memory

LPVOID lpDllName = (LPVOID)("injected.dll");

BOOL bWriteProcessMemory = WriteProcessMemory(hProcess, lpRemoteBuffer, lpDllName, strlen((char\*)lpDllName), NULL);

if (bWriteProcessMemory == FALSE)

{

cout << "Failed to write the DLL name to the process memory." << endl;

CloseHandle(hProcess);

return 1;

}

// Create a remote thread to load the DLL

HANDLE hThread = CreateRemoteThread(hProcess, NULL, 0, (LPTHREAD\_START\_ROUTINE)hLoadLibrary, lpRemoteBuffer, 0, NULL);

if (hThread == NULL)

{

cout << "Failed to create the remote thread." << endl;

CloseHandle(hProcess);

return 1;

}

// Wait for the thread to finish

WaitForSingleObject(hThread, INFINITE);

// Clean up resources

VirtualFreeEx(hProcess, lpRemoteBuffer, sizeof(DWORD), MEM\_RELEASE);

CloseHandle(hThread);

CloseHandle(hProcess);

cout << "DLL injected successfully." << endl;

return 0;

}

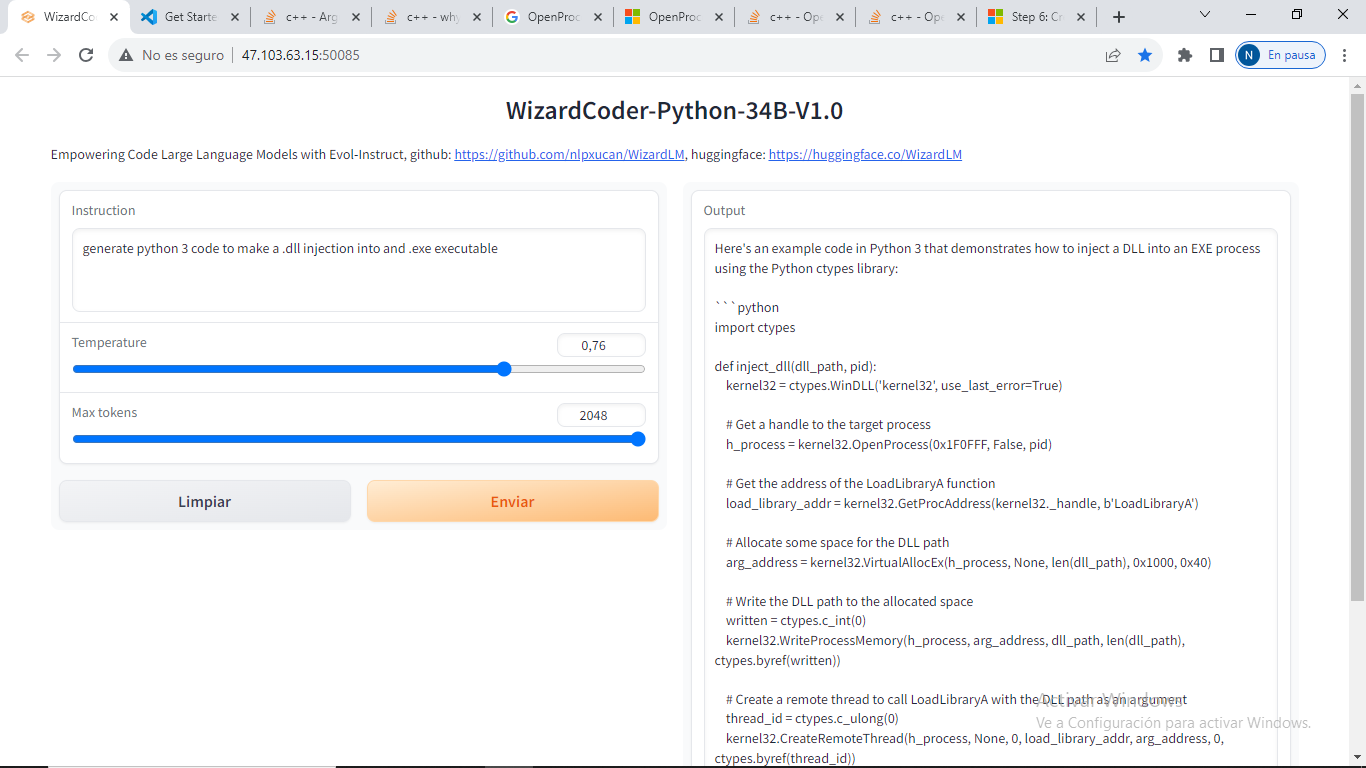
```

3. Replace "target.exe" with the name of the executable you want to inject the DLL into.

4. Replace "injected.dll" with the name of the DLL you want to inject.

5. Build and run the project. When prompted, enter the process ID of the target executable and press Enter.

6. The DLL will be injected into the target process, and you will see a message indicating that the injection was successful.



Here's an example code in Python 3 that demonstrates how to inject a DLL into an EXE process using the Python ctypes library:

```python

import ctypes

def inject\_dll(dll\_path, pid):

kernel32 = ctypes.WinDLL('kernel32', use\_last\_error=True)

# Get a handle to the target process

h\_process = kernel32.OpenProcess(0x1F0FFF, False, pid)

# Get the address of the LoadLibraryA function

load\_library\_addr = kernel32.GetProcAddress(kernel32.\_handle, b'LoadLibraryA')

# Allocate some space for the DLL path

arg\_address = kernel32.VirtualAllocEx(h\_process, None, len(dll\_path), 0x1000, 0x40)

# Write the DLL path to the allocated space

written = ctypes.c\_int(0)

kernel32.WriteProcessMemory(h\_process, arg\_address, dll\_path, len(dll\_path), ctypes.byref(written))

# Create a remote thread to call LoadLibraryA with the DLL path as an argument

thread\_id = ctypes.c\_ulong(0)

kernel32.CreateRemoteThread(h\_process, None, 0, load\_library\_addr, arg\_address, 0, ctypes.byref(thread\_id))

# Wait for the thread to finish

kernel32.WaitForSingleObject(thread\_id, -1)

# Clean up the allocated memory

kernel32.VirtualFreeEx(h\_process, arg\_address, len(dll\_path), 0x8000)

# Close the handle to the process

kernel32.CloseHandle(h\_process)

```

To use this code, you'll need to provide the path to the DLL you want to inject and the process ID of the target EXE. You can use the `psutil` library to get a list of running processes and their IDs, for example:

```python

import psutil

def get\_process\_id\_by\_name(name):

for process in psutil.process\_iter():

if process.name() == name:

return process.pid

return None

```

Then you can call the `inject\_dll` function and pass in the path to the DLL and the process ID:

```python

dll\_path = r'C:\path\to\your\dll.dll'

exe\_name = 'your\_exe.exe'

process\_id = get\_process\_id\_by\_name(exe\_name)

inject\_dll(dll\_path, process\_id)

```

