

DNAlerter

1.0

Создано системой Doxygen 1.13.2

Глава 1

DNAlert

Оповещение о небезопасном подключении к сети

Глава 2

Список файлов

2.1 Файлы

Полный список документированных файлов.

code.cpp	
Код для определения профиля сети	??

Глава 3

Файлы

3.1 Файл code.cpp

Код для определения профиля сети

```
#include <windows.h>
#include <iostream>
#include <thread>
#include <fstream>
#include <atomic>
#include <cwchar>
```

Макросы

```
#define IDI_MYICON 101
#define ID_TRAY_APP_ICON 1
#define WM_TRAYNOTIFY (WM_USER + 1)
#define IDM_EXIT 1000
```

Функции

```
std::atomic< bool > running (true)
int startup ()
LRESULT CALLBACK WndProc (HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)
void InitTray (HWND hwnd)
void RemoveTray (HWND hwnd)
std::chrono::system_clock::time_point makeTimePoint (int year, int month, int day, int hour, int minute, int second)
DWORD WINAPI ShowUnsafeNetworkWarning (LPVOID lpParam)
    Отображение MessageBox.
int check_win_registry ()
    Получение профиля сети
int main ()
    Основная функция
```

Переменные

HWND hwnd

3.1.1 Подробное описание

Код для определения профиля сети

Открывает реестр Windows и проверяет тип сети, к которой выполнялось последнее подключение.

3.1.2 Функции

3.1.2.1 check_win_registry()

int check_win_registry ()

Получение профиля сети

Каждые десять секунд сканирует реестр виндовс

Получает последний профиль сети

Записывает в файл is_threat тип сети

```
00166 {
00167     HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
00168     WCHAR lastProfile[255] = L"None";
00169     auto lastDate = makeTimePoint(0,0,0,0,0);
00170     int lastCategory = 0;
00171     while (running){
00172         DWORD index = 0;
00173         WCHAR subKeyName[255];
00174         DWORD cbName = 255;
00175
00176
00177         int year;
00178         int month;
00179         int day;
00180         int hour;
00181         int minute;
00182         int second;
00183
00184
00185         HKEY hKey;
00186
00187         LONG result = RegOpenKeyExW(HKEY_LOCAL_MACHINE,
00188                                     L"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\NetworkList\\Profiles",
00189                                     0,
00190                                     KEY_READ,
00191                                     &hKey);
00192         if (result != ERROR_SUCCESS) {
00193             std::wcerr « L"Failed to open registry key. Error code: " « result « std::endl;
00194             return 1;
00195         }
00196
00197         while (RegEnumKeyExW(hKey, index, subKeyName, &cbName, NULL, NULL, NULL, NULL) ==
00198             ERROR_SUCCESS) {
00199             HKEY hSubKey;
00200             result = RegOpenKeyExW(hKey, subKeyName, 0, KEY_READ, &hSubKey);
00201             if (result == ERROR_SUCCESS) {
00202                 WCHAR profileName[255];
00203                 DWORD cbData = sizeof(profileName);
00204                 result = RegQueryValueExW(hSubKey, L"ProfileName", NULL, NULL, (LPBYTE)profileName, &cbData);
00205                 if (result == ERROR_SUCCESS) {
00206                     DWORD type;
00207                     cbData = 0;
00208                     result = RegQueryValueExW(hSubKey, L"DateLastConnected", NULL, &type, NULL, &cbData);
00209                     if (result == ERROR_SUCCESS && type == REG_BINARY) {
00210                         BYTE* dateData = new BYTE[cbData];
00211                         result = RegQueryValueExW(hSubKey, L"DateLastConnected", NULL, NULL, dateData, &cbData);
00212                         if (result == ERROR_SUCCESS) {
```



```

00213         year = int(dateData[0]+dateData[1]*256);
00214         month = int(dateData[2]+dateData[3]*256);
00215         day = int(dateData[6]+dateData[7]*256);
00216         hour = int(dateData[8]+dateData[9]*256);
00217         minute = int(dateData[10]+dateData[11]*256);
00218         second = int(dateData[12]+dateData[13]*256);
00219
00220
00221         auto date = makeTimePoint(year,month,day,hour,minute,second);
00222
00223         DWORD category;
00224         RegQueryValueExW(hSubKey, L"Category", NULL, &type, (LPBYTE)&category, &cbData);
00225
00226         if (date>lastDate or (wcsncmp(profileName,lastProfile)==0 and category!=lastCategory)){
00227             lastDate=date;
00228             lastCategory=category;
00229             WriteConsoleW(hConsole, L"Changed Profile Name: ", 22, nullptr, nullptr);
00230             WriteConsoleW(hConsole, profileName, wcslen(profileName), nullptr, nullptr);
00231             WriteConsoleW(hConsole, "\n", 1, nullptr, nullptr);
00232             if (category==0){
00233                 CreateThread(NULL, 0, ShowUnsafeNetworkWarning, NULL, 0, NULL);
00234             }
00235             wcsncpy(lastProfile, profileName, 255);
00236         }
00237     } else {
00238         std::wcerr << L"Failed to read DateLastConnected. Error code: " << result << std::endl;
00239     }
00240 } else if (result == ERROR_FILE_NOT_FOUND) {
00241     std::wcerr << L"DateLastConnected not found for profile: " << profileName << std::endl;
00242 } else {
00243     std::wcerr << L"Failed to query DateLastConnected. Error code: " << result << std::endl;
00244 }
00245 }
00246 else{
00247     std::wcerr << L"Error" << std::endl;
00248 }
00249 RegCloseKey(hSubKey);
00250 }
00251 index++;
00252 cbName = 255;
00253 }
00254 WriteConsoleW(hConsole, L"LastProfile Name: ", wcslen(L"LastProfile Name: "), nullptr, nullptr);
00255 WriteConsoleW(hConsole, lastProfile, wcslen(lastProfile), nullptr, nullptr);
00256 WriteConsoleW(hConsole, "\n", 1, nullptr, nullptr);
00257
00258 std::wcout << L"Last Category: " << lastCategory << std::endl;
00259 RegCloseKey(hKey);
00260
00261 std::ofstream myFile("./is_thread", std::ios::out | std::ios::trunc);
00262 if (myFile.is_open()) {
00263     myFile << lastCategory;
00264     myFile.close();
00265 }
00266 std::this_thread::sleep_for(std::chrono::seconds(10));
00267 }
00268 return 0;
00269 }

```

3.1.2.2 InitTray()

```

void InitTray (
    HWND hwnd)

```

Создание иконки на панели задач при запуске программы

```

00093 {
00094     NOTIFYICONDATAW nid = {0};
00095     nid.cbSize = sizeof(NOTIFYICONDATA);
00096     nid.hWnd = hwnd;
00097     nid.uID = ID_TRAY_APP_ICON;
00098     nid.uFlags = NIF_ICON | NIF_MESSAGE | NIF_TIP;
00099     nid.uCallbackMessage = WM_TRAYNOTIFY;
00100     nid.hIcon = LoadIcon(GetModuleHandle(NULL), MAKEINTRESOURCE(IDI_MYICON));
00101     wcsncpy(nid.szTip, L"Оповещение о небезопасном подключении к сети");
00102     Shell_NotifyIconW(NIM_ADD, &nid);
00103 }

```

3.1.2.3 main()

```
int main ()
```

Основная функция

Запускает функцию для создания иконки на панели задач.

Запускает поток проверки подключения.

```
00280 {
00281     startup();
00282
00283     WNDCLASSEXW wc = {0};
00284     wc.cbSize = sizeof(WNDCLASSEXW);
00285     wc.lpfnWndProc = WndProc;
00286     wc.hInstance = GetModuleHandleW(NULL);
00287     wc.lpszClassName = L"TrayAppClass";
00288     RegisterClassExW(&wc);
00289
00290     hwnd = CreateWindowW(L"TrayAppClass", L"TrayApp", WS_OVERLAPPEDWINDOW,
00291         0, 0, 0, 0, NULL, NULL, wc.hInstance, NULL);
00292
00293     if (!hwnd) {
00294         std::cerr << "Error creating icon!" << std::endl;
00295         return 1;
00296     }
00297
00298
00299
00300     InitTray(hwnd);
00301
00302     std::thread cwrThread(check_win_registry);
00303
00304     MessageBoxW(NULL,
00305         L"Приложение запущено!",
00306         L"Успешно",
00307         MB_OK | MB_ICONINFORMATION | MB_SYSTEMMODAL);
00308
00309
00310     MSG msg;
00311     while (GetMessageW(&msg, NULL, 0, 0)) {
00312         TranslateMessage(&msg);
00313         DispatchMessage(&msg);
00314     }
00315
00316
00317     RemoveTray(hwnd);
00318     cwrThread.join();
00319
00320     std::ofstream myFile("./is_threat", std::ios::out | std::ios::trunc);
00321     if (myFile.is_open()) {
00322         myFile << "1";
00323         myFile.close();
00324     }
00325
00326     return 0;
00327 }
00328 }
```

3.1.2.4 makeTimePoint()

```
std::chrono::system_clock::time_point makeTimePoint (
    int year,
    int month,
    int day,
    int hour,
    int minute,
    int second)
```

Получает на вход год, месяц, день, час, минуту и секунду в формате числа

И конвертирует их в формат времени

```
00133 {
00134     std::tm tm = {0};
00135     tm.tm_year = year - 1900;
```

```

00136     tm.tm_mon = month - 1;
00137     tm.tm_mday = day;
00138     tm.tm_hour = hour;
00139     tm.tm_min = minute;
00140     tm.tm_sec = second;
00141
00142     std::time_t tt = std::mktime(&tm);
00143     return std::chrono::system_clock::from_time_t(tt);
00144 }

```

3.1.2.5 RemoveTray()

```

void RemoveTray (
    HWND hwnd)

```

Удаление иконки на панели задач при выключении программы

```

00111 {
00112     NOTIFYICONDATA nid = {0};
00113     nid.cbSize = sizeof(NOTIFYICONDATA);
00114     nid.hWnd = hwnd;
00115     nid.uID = ID_TRAY_APP_ICON;
00116     Shell_NotifyIcon(NIM_DELETE, &nid);
00117 }

```

3.1.2.6 ShowUnsafeNetworkWarning()

```

DWORD WINAPI ShowUnsafeNetworkWarning (
    LPVOID lpParam)

```

Отображение MessageBox.

Так-как создание MessageBox останавливает процесс выполнения программы, создаётся отдельный поток для предотвращения этого.

```

00152 {
00153     MessageBoxW(NULL, L"Вы подключены к небезопасной сети!", L"Внимание", MB_OK | MB_ICONWARNING |
        MB_SYSTEMMODAL);
00154     return 0;
00155 }

```

3.1.2.7 startup()

```

int startup ()

```

Добавляет путь к программе в список автозагрузки windows

```

00030 {
00031     wchar_t path[MAX_PATH];
00032     GetModuleFileNameW(NULL, path, MAX_PATH);
00033     std::wstring progPath = path;
00034     HKEY hkey = NULL;
00035     LONG createStatus = RegCreateKeyW(HKEY_CURRENT_USER,
        L"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",
00036         &hkey);
00037
00038
00039     if (createStatus == ERROR_SUCCESS) {
00040         LONG status = RegSetValueExW(hkey,
00041             L"DAlert",
00042             0,
00043             REG_SZ,
00044             (BYTE*)progPath.c_str(),
00045             (progPath.size() + 1) * sizeof(wchar_t));
00046
00047         RegCloseKey(hkey);
00048     }
00049
00050     return 0;
00051 }

```

3.1.2.8 WndProc()

```
LRESULT CALLBACK WndProc (
    HWND hwnd,
    UINT msg,
    WPARAM wParam,
    LPARAM lParam)
```

Функция для выключения программы, если нажата кнопка "Выключить"

```
00057 {
00058     switch (msg) {
00059     case WM_TRAYNOTIFY:
00060         if (lParam == WM_RBUTTONUP) {
00061             POINT pt;
00062             GetCursorPos(&pt);
00063             HMENU hMenu = CreatePopupMenu();
00064             AppendMenuW(hMenu, MF_STRING, IDM_EXIT, L"Выключить");
00065             SetForegroundWindow(hwnd);
00066             TrackPopupMenu(hMenu, TPM_RIGHTALIGN | TPM_BOTTOMALIGN | TPM_RIGHTBUTTON,
00067                 pt.x, pt.y, 0, hwnd, NULL);
00068             DestroyMenu(hMenu);
00069         }
00070         break;
00071     case WM_COMMAND:
00072         if (LOWORD(wParam) == IDM_EXIT) {
00073             running = false;
00074             PostQuitMessage(0);
00075         }
00076         break;
00077     case WM_DESTROY:
00078         running = false;
00079         PostQuitMessage(0);
00080         break;
00081     default:
00082         return DefWindowProcW(hwnd, msg, wParam, lParam);
00083     }
00084     return 0;
00085 }
```