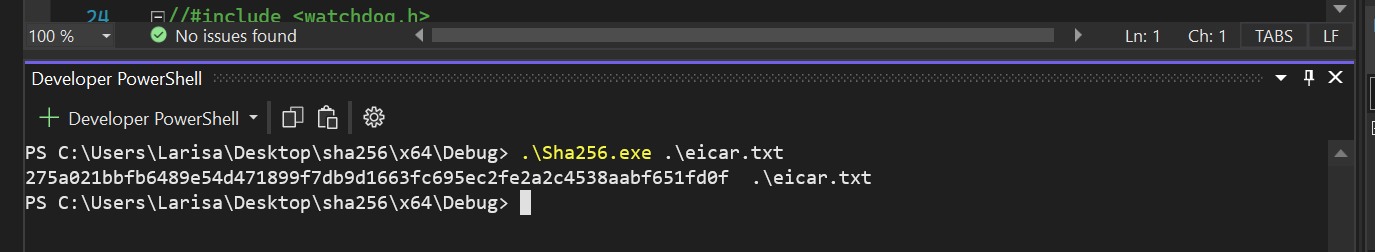
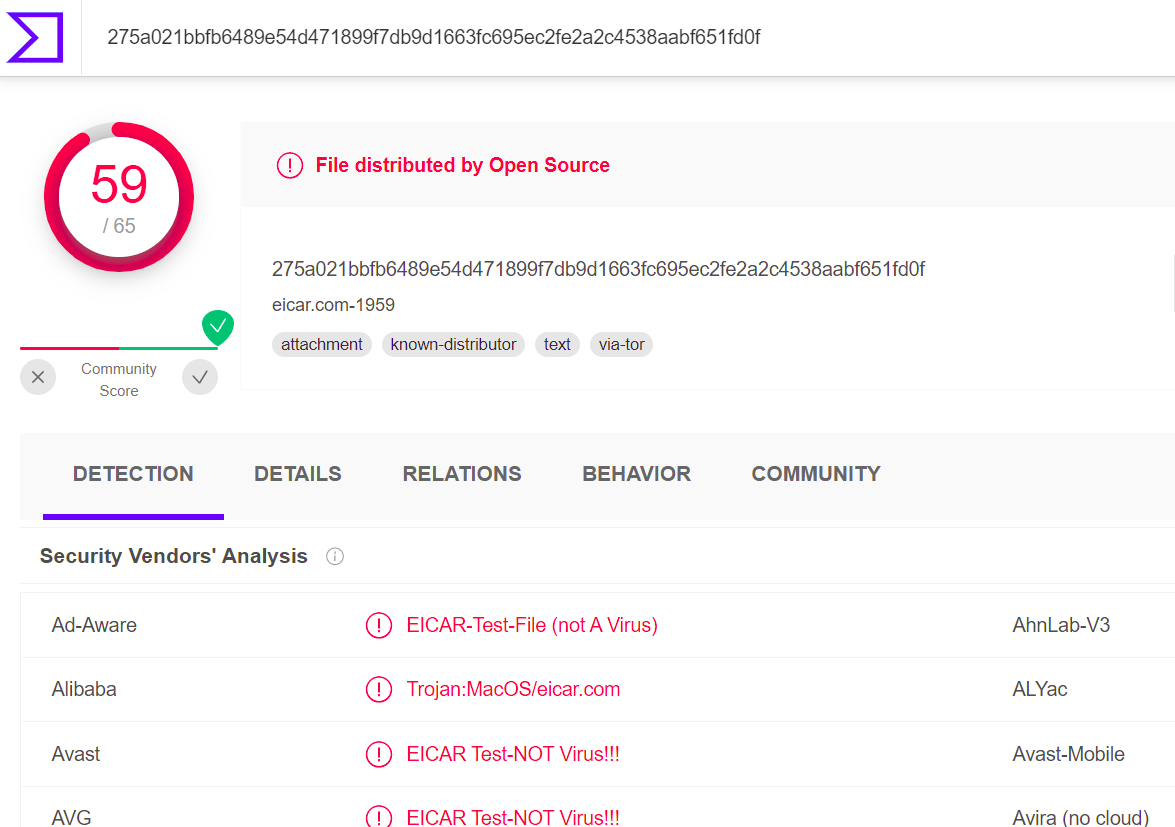
Laborator 4 – Rezolvare

Laboratorul 4 este similar cu 3, doar că este pe Windows.

**Ex 1**. *Please download one of the eicar files and compute with your executable the sha256 hash for it.*

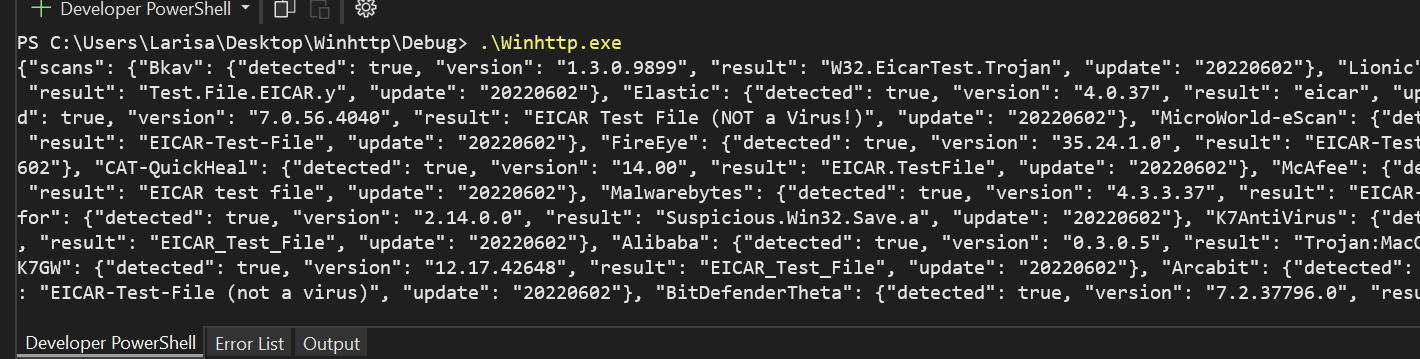
Teoretic, puteam folosi și terminalul de Ubuntu (Linux Subsystem for Windows).

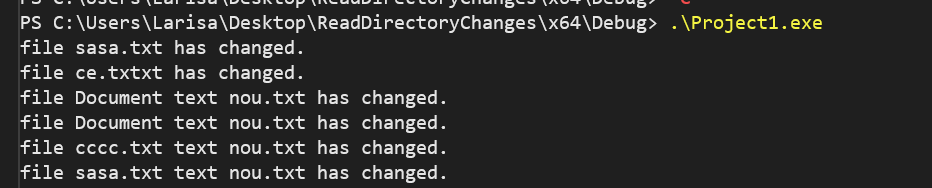




**Ex 2**. *Please run all the 3 samples and analyse the code until everything is clear. After that please create a program which monitories changes in a folder received as a parameter and for every file modified ask its hash on virustotal and if it is infected, delete it.*

Rulare proiecte oferite:





Rezolvarea problemei va fi diferită față de cea din Laboratorul 3, deoarece vom folosi fișierele executabile pentru a procesa anumite comenzi (spre exemplu, să obținem hash-ul).

Codul:

#include <windows.h>

#include <winhttp.h>

#include <stdio.h>

#include <string>

#include <fstream>

#include <iostream>

#include <codecvt>

#define FILE\_NAME L"tmp.txt"

#define MAXCHAR 10000

using namespace std;

char buffer[0x100000];

char response\_buffer[MAXCHAR];

// Compute hash of file using the exe

string compute\_hash(string full\_path)

{

string command = "C:\\Users\\Larisa\\Desktop\\Ex\\Sha256.exe " + full\_path + " > D:\\temp\_output.txt";

const char\* aux = command.c\_str();

system(aux);

ifstream output\_file("D:\\temp\_output.txt");

string file\_hash;

getline(output\_file, file\_hash, ' ');

output\_file.close();

remove("D:\\temp\_output.txt");

return file\_hash;

}

// API to VirusTotal.com

void API(string hash)

{

DWORD dwSize = 0;

DWORD dwDownloaded = 0;

LPSTR pszOutBuffer;

BOOL bResults = FALSE;

HANDLE hFile = CreateFile(FILE\_NAME, GENERIC\_READ | GENERIC\_WRITE, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE | FILE\_SHARE\_DELETE, NULL, CREATE\_ALWAYS,

0, NULL);

HINTERNET hSession = NULL,

hConnect = NULL,

hRequest = NULL;

// Use WinHttpOpen to obtain a session handle.

hSession = WinHttpOpen(L"WinHTTP Example/1.1",

WINHTTP\_ACCESS\_TYPE\_DEFAULT\_PROXY,

WINHTTP\_NO\_PROXY\_NAME,

WINHTTP\_NO\_PROXY\_BYPASS, 0);

// Specify an HTTP server.

if (hSession)

hConnect = WinHttpConnect(hSession, L"www.virustotal.com",

INTERNET\_DEFAULT\_HTTPS\_PORT, 0);

// Create an HTTP request handle.

if (hConnect)

{

string request\_parameter = "/vtapi/v2/file/report?apikey=abffc16654d1b01774d2c36f400d762f5038391f3898caa4ba3c4c61bf8ed136&resource=" + hash;

wstring\_convert<codecvt<wchar\_t, char, mbstate\_t>> converter;

wstring w\_request\_parameter = converter.from\_bytes(request\_parameter);

hRequest = WinHttpOpenRequest(hConnect, L"GET", w\_request\_parameter.c\_str(),

NULL, WINHTTP\_NO\_REFERER,

WINHTTP\_DEFAULT\_ACCEPT\_TYPES,

WINHTTP\_FLAG\_SECURE);

}

// Send a request.

if (hRequest)

bResults = WinHttpSendRequest(hRequest,

WINHTTP\_NO\_ADDITIONAL\_HEADERS, 0,

WINHTTP\_NO\_REQUEST\_DATA, 0,

0, 0);

// End the request.

if (bResults)

bResults = WinHttpReceiveResponse(hRequest, NULL);

// Keep checking for data until there is nothing left.

if (bResults)

{

do

{

// Check for available data.

dwSize = 0;

if (!WinHttpQueryDataAvailable(hRequest, &dwSize))

printf("Error %u in WinHttpQueryDataAvailable.\n",

GetLastError());

// Allocate space for the buffer.

pszOutBuffer = new char[dwSize + 1];

if (!pszOutBuffer)

{

printf("Out of memory\n");

dwSize = 0;

}

else

{

// Read the data.

ZeroMemory(pszOutBuffer, dwSize + 1);

if (!WinHttpReadData(hRequest, (LPVOID)pszOutBuffer,

dwSize, &dwDownloaded))

printf("Error %u in WinHttpReadData.\n", GetLastError());

else

{

DWORD dwWritten = 0;

WriteFile(hFile, pszOutBuffer, dwSize, &dwWritten, NULL);

//printf("%s", pszOutBuffer);

}

// Free the memory allocated to the buffer.

delete[] pszOutBuffer;

}

} while (dwSize > 0);

}

// Report any errors.

if (!bResults)

printf("Error %d has occurred.\n", GetLastError());

// Close any open handles.

if (hRequest) WinHttpCloseHandle(hRequest);

if (hConnect) WinHttpCloseHandle(hConnect);

if (hSession) WinHttpCloseHandle(hSession);

CloseHandle(hFile);

}

int main()

{

DWORD bRet = 0;

string working\_directory = "C:\\Users\\Larisa\\Desktop\\lab3";

// Handle to monitor the folder

HANDLE h = CreateFileW(L"C:\\Users\\Larisa\\Desktop\\lab3", FILE\_LIST\_DIRECTORY, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE | FILE\_SHARE\_DELETE,

NULL, OPEN\_EXISTING, FILE\_FLAG\_BACKUP\_SEMANTICS, 0);

// Watch for changes in the folder

while (1)

{

ReadDirectoryChangesW(h, buffer, 0x100000, TRUE,

FILE\_NOTIFY\_CHANGE\_FILE\_NAME |

FILE\_NOTIFY\_CHANGE\_DIR\_NAME |

FILE\_NOTIFY\_CHANGE\_ATTRIBUTES |

FILE\_NOTIFY\_CHANGE\_SIZE |

FILE\_NOTIFY\_CHANGE\_LAST\_WRITE |

FILE\_NOTIFY\_CHANGE\_LAST\_ACCESS |

FILE\_NOTIFY\_CHANGE\_CREATION |

FILE\_NOTIFY\_CHANGE\_SECURITY,

&bRet,

NULL,

NULL);

// Change Produced

FILE\_NOTIFY\_INFORMATION \*p = (FILE\_NOTIFY\_INFORMATION \*)buffer;

//wprintf(L"file %s has changed.\n", p->FileName);

// Get the name of the file

// Put the name of the changed file in temp\_file.txt

FILE\* temp\_file;

temp\_file = fopen("D:\\temp\_file.txt", "w");

fwprintf(temp\_file, L"%s\n", p->FileName);

fclose(temp\_file);

// Use temp\_file to get back the name of the changed file

ifstream aux\_file;

aux\_file.open("D:\\temp\_file.txt");

string file\_name;

getline(aux\_file, file\_name);

aux\_file.close();

remove("D:\\temp\_file.txt");

file\_name = file\_name.substr(0, p->FileNameLength / sizeof(WCHAR));

// Full Path Name

string full\_path = working\_directory + "\\" + file\_name;

ifstream f(full\_path.c\_str());

// Check the file was not deleted -> delete is monitored

if(f)

{

f.close();

// Compute hash of file

string hash = compute\_hash(full\_path);

// Get the response from the API

API(hash);

// Check if it is virus

FILE\* file = NULL;

if (!(file = fopen("C:\\Users\\Larisa\\Desktop\\Ex\\Debug\\tmp.txt", "r")))

{

perror("fopen");

return(1);

}

while (fgets(response\_buffer, MAXCHAR - 1, file) != NULL) {

int no\_positives;

char\* positive\_ptr = strstr(response\_buffer, "positives");

if (positive\_ptr != NULL) {

positive\_ptr = positive\_ptr + strlen("positives\": ");

positive\_ptr = strtok(positive\_ptr, ",");

no\_positives = atoi(positive\_ptr);

if (no\_positives > 0) {

cout << "Deleting file " << full\_path.c\_str() << endl;

remove(full\_path.c\_str()); // remove the file

}

}

}

}

}

CloseHandle(h);

return 0;

}

