In today's laboratory we will try to do the on windows what we did on linux on last laboratory. So, in current laboratory archive you will find the following samples:

- Sha256
- WinHttp
- ReadDirectoryChanges

In folder sha256 you will find the sample from the last laboratory ported on windows. Also for each of the samples there is a visual studio solution which should help you build the projectes much easily.

If you don't know how to use a visual studio solution you can use the tutorial from here:

https://docs.microsoft.com/en-us/cpp/build/vscpp-step-2-build?view=vs-2019

So, using the visual studio solution and the above tutorial, do compile sha256 project and after compiling it you will obtain an executable which can be used to compute sha256 hash for a file.

Practic. The file eicar.txt which we have used in the last laboratory can be downloaded from here:

http://2016.eicar.org/85-0-Download.html

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

Winhttp

Lets take a look at winhttp project. Winhttp.dll is the same as curl but on windows, it is the library which implements http protocol on windows.

Lets take a look at the code:

```
#define FILE_NAME                   L"d:\\MyFile.html"
```

FILE_NAME is a define used for saving the answer from the server of an http request. Please change this name with the name of a file from your disk.

In main you will find:

```
  HANDLE hFile = CreateFile(FILE_NAME, GENERIC_READ|GENERIC_WRITE,
FILE_SHARE_READ|FILE_SHARE_WRITE|FILE_SHARE_DELETE, NULL, CREATE_ALWAYS,
        0, NULL);
```

Function CreateFile is similar with function open from linux, it is the function used on windows to open files. You can find its documentation here:

https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfilea

In the code above we opened a file, obtaining a HANDLE(similar to file descriptor on linux) to the file FILE_NAME,

having read/write rights (GENERIC_READ|GENERIC_WRITE),

which allows other processes to still modify it (sharing rights – FILE_SHARE_* represents the rights other processes are allowed to request when opening this file while the handle we obtained is still open

with security attributes NULL(this means it inherits the default security attributes)

with creation flags CREATE_ALWAYS, which means if the file does not exist, it have to create it.

0 – we didn't set any special flags for it,

NULL – we don't use a template in opening the file.

After this we open a communication session with the server:

```
  hSession = WinHttpOpen( L"WinHTTP Example/1.1",
                          WINHTTP_ACCESS_TYPE_DEFAULT_PROXY,
                          WINHTTP_NO_PROXY_NAME,
                          WINHTTP_NO_PROXY_BYPASS, 0 );
```

This function documentation can be found here:

https://docs.microsoft.com/en-us/windows/win32/api/winhttp/nf-winhttp-winhttpopen

In our example we open an http session which does not have a proxy set and which will use as user agent `WinHTTP Example/1.1`

After we open a session, we will use the session handle to open a connection with the server we want to communicate:

```
  if( hSession )
    hConnect = WinHttpConnect( hSession, L"www.virustotal.com",
                                                 INTERNET_DEFAULT_HTTPS_PORT, 0 );
```

Documentation for WinHttpConnect can be found here:

https://docs.microsoft.com/en-us/windows/win32/api/winhttp/nf-winhttp-winhttpconnect

In the code above we are trying to establish a connection with www.virustotal .com on 443 port, which is the port for https. (the last parameter will be 0, it will be used in the future by windows, at this moment it has to be 0).

Once we established a connection we are sending an http request:

```
// Create an HTTP request handle.
  if( hConnect )
    hRequest = WinHttpOpenRequest( hConnect, L"GET",
L"/vtapi/v2/file/report?apikey=1e5c8a20708ee442ef7393f7d7ed4c714137d2dd3ee7d9dcccf5c5eaee
ac5127&resource=275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f",
                                  NULL, WINHTTP_NO_REFERER,
                                  WINHTTP_DEFAULT_ACCEPT_TYPES,
                                                  WINHTTP_FLAG_SECURE );
```

Documentation for WinHttpOpenRequest o puteti gasi aici:

https://docs.microsoft.com/en-us/windows/win32/api/winhttp/nf-winhttp-winhttpopenrequest

Using the handle to the connection, hConnect, we are creating a request to the server and we initiat a get request for an url:

```
/vtapi/v2/file/report?apikey=1e5c8a20708ee442ef7393f7d7ed4c714137d2dd3ee7d9dcccf5c5eaeeac
5127&resource=275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f
```

VERY IMPORTANT:

Lets analyse the request:

```
apikey=1e5c8a20708ee442ef7393f7d7ed4c714137d2dd3ee7d9dcccf5c5eaeeac5127
&
resource=275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f
```

we are sending to the server 2 parameters an api key, which is a public key obtained after registering with virustotal . I recommend to register with virustotal using the link below, because each key is limited to 4 requests per minute and if you all are using this key you might receive errors which will not happen if anyone of you will have a separate key.

https://developers.virustotal.com/reference

(more precisely here https://www.virustotal.com/#/join-us , the first link is also useful cause it contains more explanations).

The second parameter is resource=sha256, which represents in this case the hash sha256 for the file we are searching for, in this case is the sha for eicar.txt

So we created the request, once we created we send it to the server using WinHttpSendRequest

```
    bResults = WinHttpSendRequest( hRequest,
                                   WINHTTP_NO_ADDITIONAL_HEADERS, 0,
                                   WINHTTP_NO_REQUEST_DATA, 0,
                                   0, 0 );
```

Documentation for this function WinHttpSendRequest:

https://docs.microsoft.com/en-us/windows/win32/api/winhttp/nf-winhttp-winhttpsendrequest

After we send the request to the server we are waiting the response with:

```
  if( bResults )
    bResults = WinHttpReceiveResponse( hRequest, NULL );
```

Function WinHttpReceiveResponse will block until we receive an answer from the server or a timeout occurs. The documentation for the function can be found here:

https://docs.microsoft.com/en-us/windows/win32/api/winhttp/nf-winhttp-winhttpreceiveresponse

Once we received the answer from the server we read the received data with functions WinHttpQueryDataAvailable and WinHttpReadData – the first tells us if there is still data to receive, the second read that data. Once we received the data we display them on the console with printf and we also write them in the filename FILE_NAME using function WriteFile.

Documentation for mentioned functions can be found here:

WinHttpQueryDataAvailable

https://docs.microsoft.com/en-us/windows/win32/api/winhttp/nf-winhttp-winhttpquerydataavailable

WinHttpReadData

https://docs.microsoft.com/en-us/windows/win32/api/winhttp/nf-winhttp-winhttpreaddata

WriteFile:

https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-writefile

Using this example we should be able to send any http(s) request to any server.

ReadDirectoryChanges:

Same as we did on linux, we need a way on windows to monitor changes on a folder or a file. In this laboratory we will show a easier method to do that, if anyone is interested in doing the right method to monitor file changes using a driver can read the sample from here:

http://joyasystems.com/sample-code%2FWindows%20Driver%20Samples%2FMinispy%20File%20System%20Minifilter%20Driver

Lets take a look at the code from ReadDirectoryChanges folder:

```
	HANDLE h = CreateFileW(L"d:\\Tests", FILE_LIST_DIRECTORY, FILE_SHARE_READ |
FILE_SHARE_WRITE | FILE_SHARE_DELETE,
		NULL, OPEN_EXISTING, FILE_FLAG_BACKUP_SEMANTICS, 0);
```

We obtain a handle to a folder. Please observe we use FILE_FLAG_BACKUP_SEMANTICS needed to have the necessary rights to obtain the notifications with what has changed in that folder.

We are using the above handle to read the changes using this function:

```
		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);
```

Using the function we are reading all the changes with any of these reasons

```
			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,
```

in buffer buffer, where we will have structures of this type `FILE_NOTIFY_INFORMATION`

Documentation for ReadDirectoryChanges can be found here:

https://docs.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-readdirectorychangesw

also structure FILE_NOTIFY_INFORMATION is described here:

https://docs.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-file_notify_information

Practic. 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.