EECS 349&444 (Fall 2019) – HW2



**Case Western Reserve University** Department of Computer and Data Sciences

EECS 349&444: Computer Security

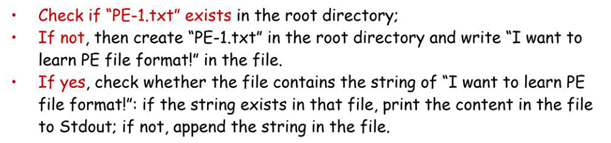
|  |  |  |
| --- | --- | --- |
|  | Assignment Date: | 10/08/2019 |
|  |  |  |
|  | Sumission Date: | 10/12/2019 @ 9:00 pm |
|  | First Name: | AMAN |
|  |  |  |
|  | Last Name: | ANAND |
|  |  |  |
|  |  |  |
|  | Git Hub Link: |  |
|  |  |  |

Abstract of the feedback: GOOD EXERCISE

EECS 349&444 (Fall 2019) – HW2

**Step 1:**

First a C program code is written as follows in order to implement the following functions.



The C program code is written as follows:

#include <string>

#include <iostream>

#include <stdio.h>

#include <fstream>

void update(char \*);

int main() {

char a[50]= "C:/Program Files/Common Files/system/PE-1.txt";

char \*f=a;

bool flag = 0;

std::ifstream ifile(f);

if((bool)ifile == true)

flag=1;

else

flag=0;

if (flag == 1){

update(a);

}

else{

std::ofstream fout;

fout.open(f);

fout<<"I want to learn PE file format!\n";

fout.close();

}

return 0;

}

void update(char \*filename){

bool file\_flag = false;

std::string line;

std::string comp\_line = "I want to learn PE file format!";

std::ifstream myfile (filename);

if (myfile.is\_open())

{

while ( getline (myfile,line) )

{

if (line == comp\_line){

myfile.close();

file\_flag=1;

}

}

myfile.close();

}

if (file\_flag == true){

std::string line;

std::ifstream myfile (filename);

if (myfile.is\_open())

{

while ( getline (myfile,line) )

{

std::cout << line << '\n';

}

myfile.close();

}

}else{

std::ofstream fout;

fout.open(filename,std::ofstream::app);

fout<<"I want to learn PE file format!\n";

fout.close();

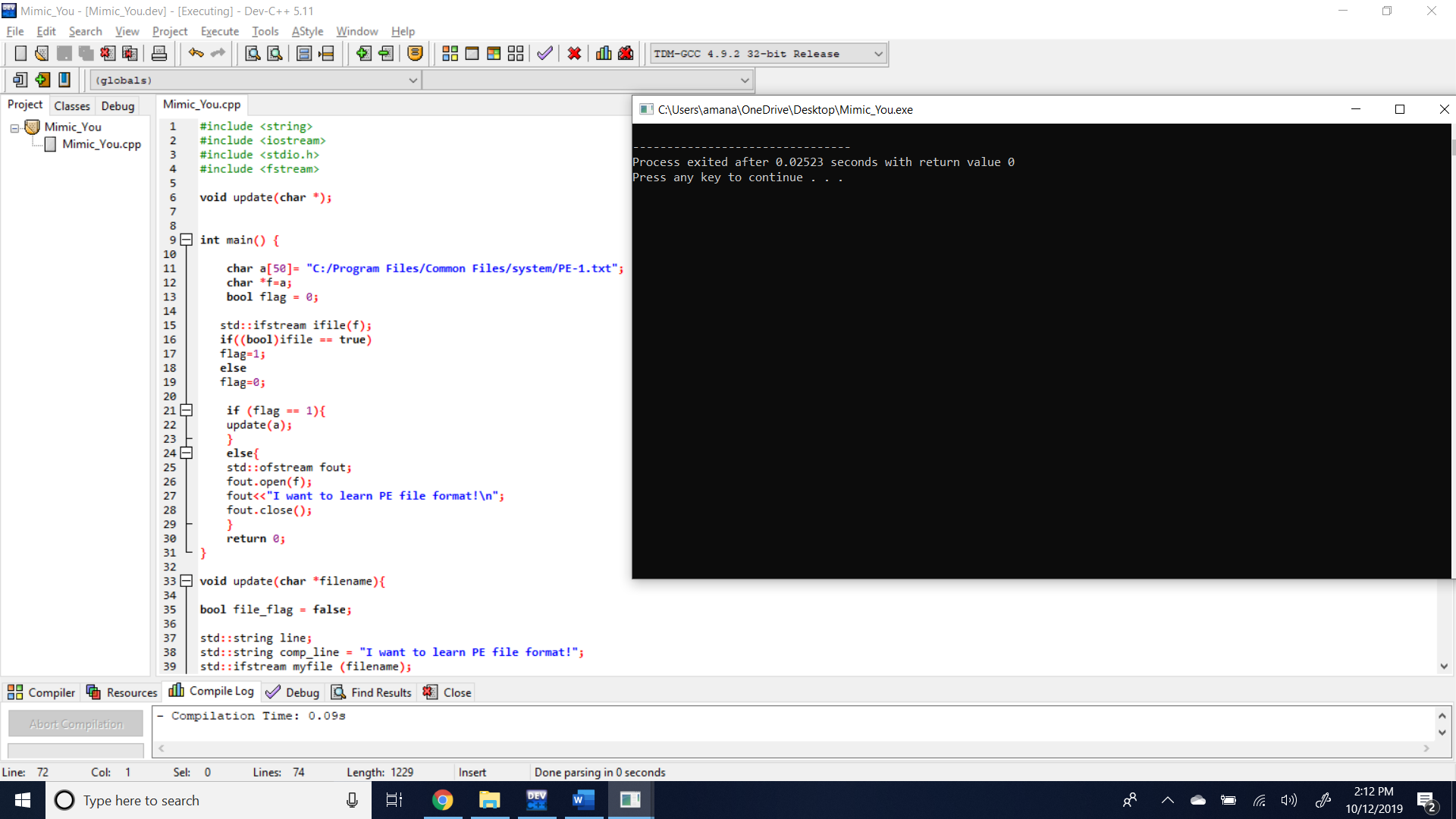
}

}

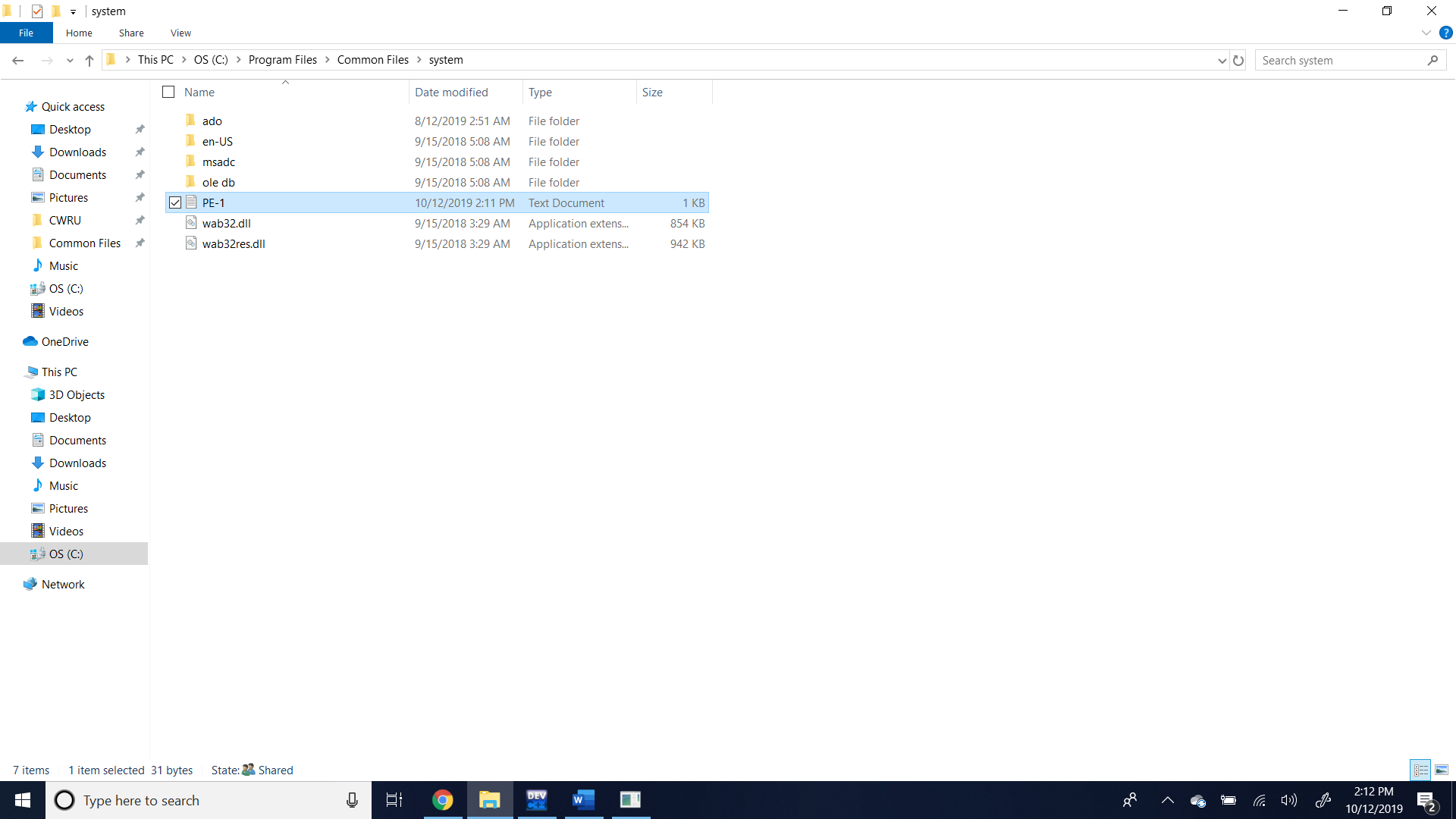
Explanation : The code will check the root directory i.e. C:/Program Files/Common Files/system/ to see if a PE-1 file exists or not. In case there is no file with this name , it will create a PE-1.txt file and type in “I want to learn PE file format!” in that file. If such a file does exist, it will append it and add the same line in the end. In the third case, if the file exists and does have this line in it, the code will print out all the data which has been written in the file.

Screenshots for these outputs are as follows,

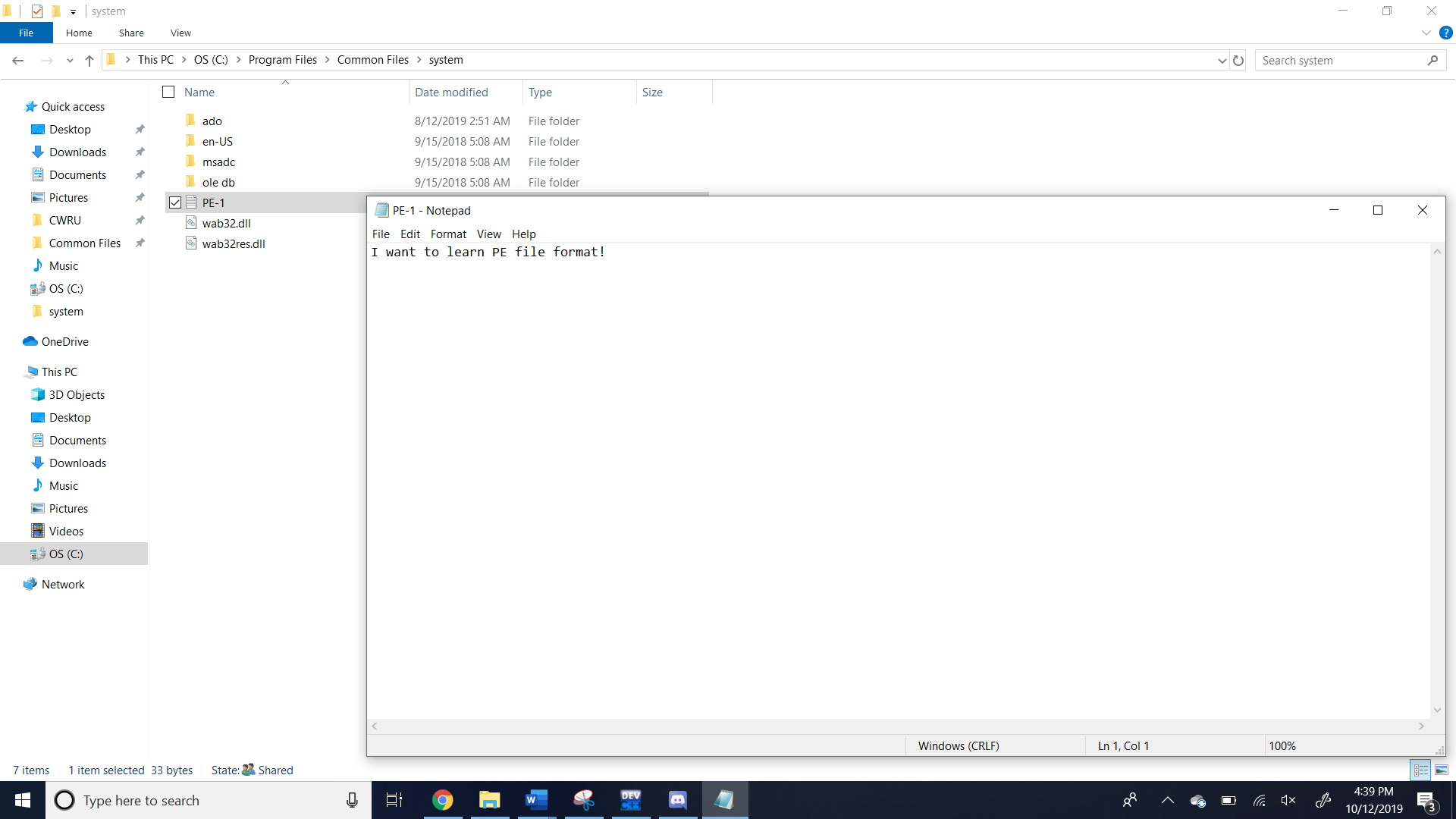
-Code is written and ran to check if file exists in C:/Program Files/Common Files/system/



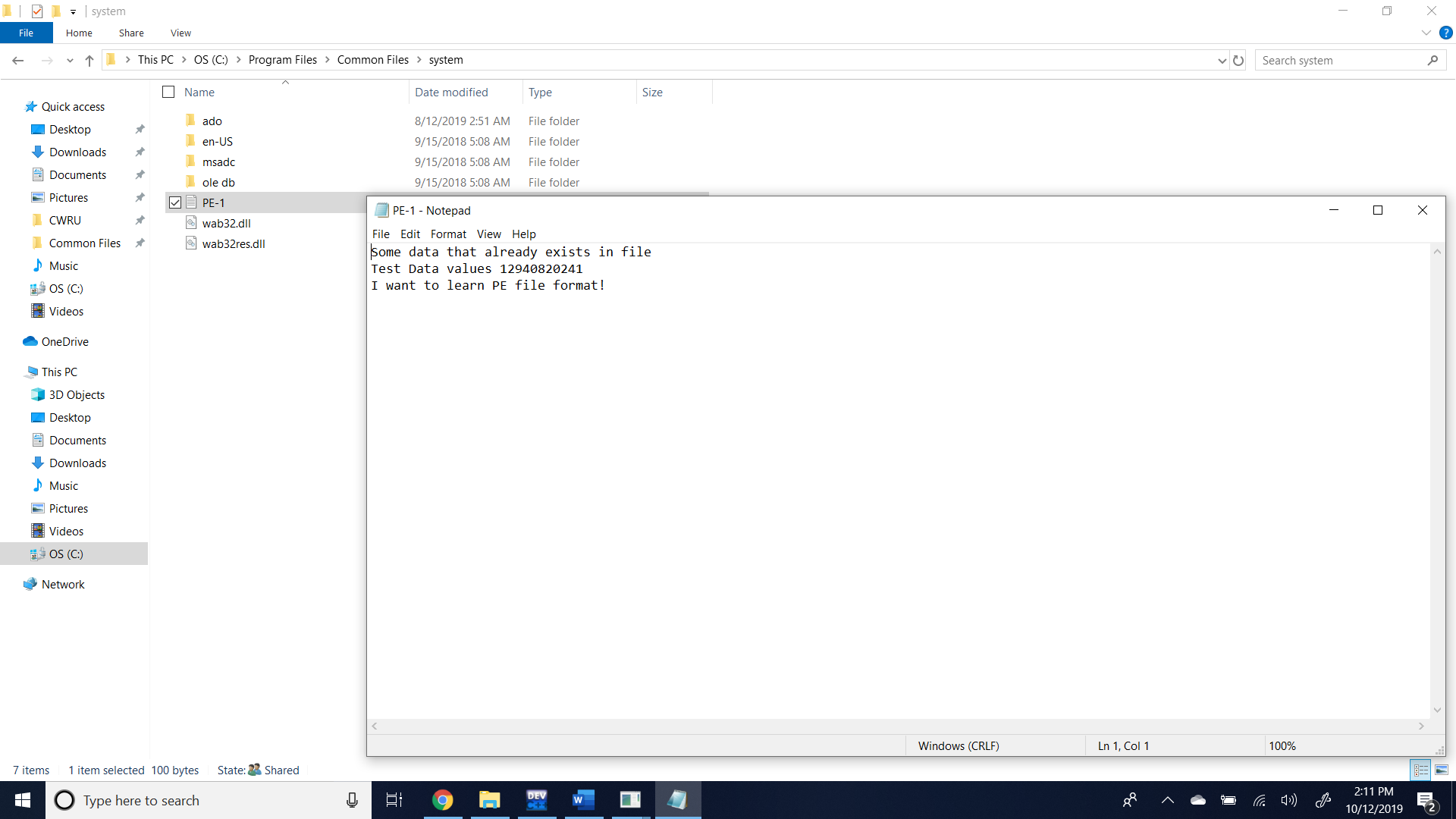
The file is then created and seen as follows:



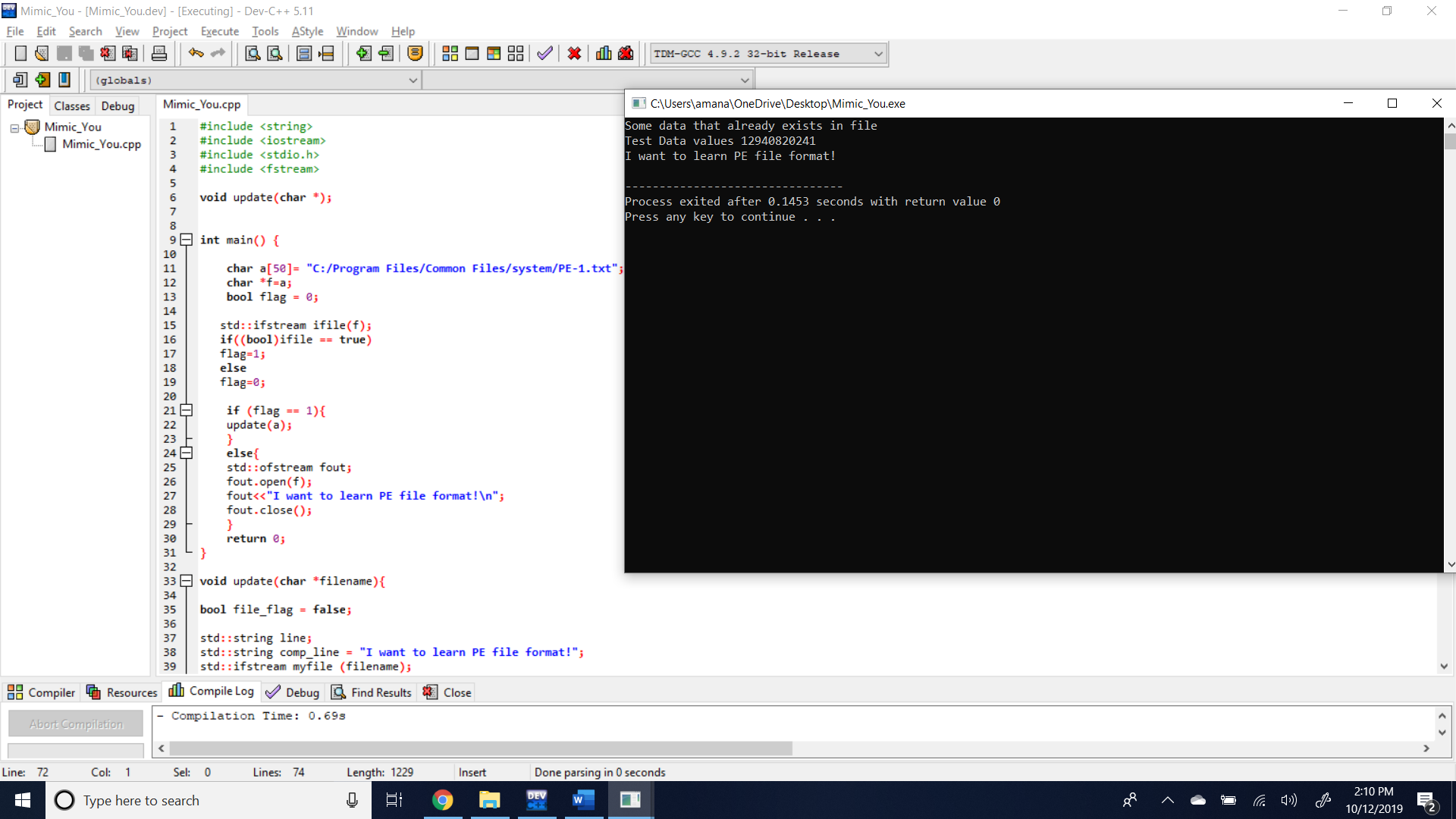
Now the code is ran again and this time as the file with that name exists, it types the line into the file and when we open the file we observe the following,



Now let us enter some additional data into the file along with the line I want to learn PE file format! To see if the program will give us whatever data has been fed to the file apart from that line.



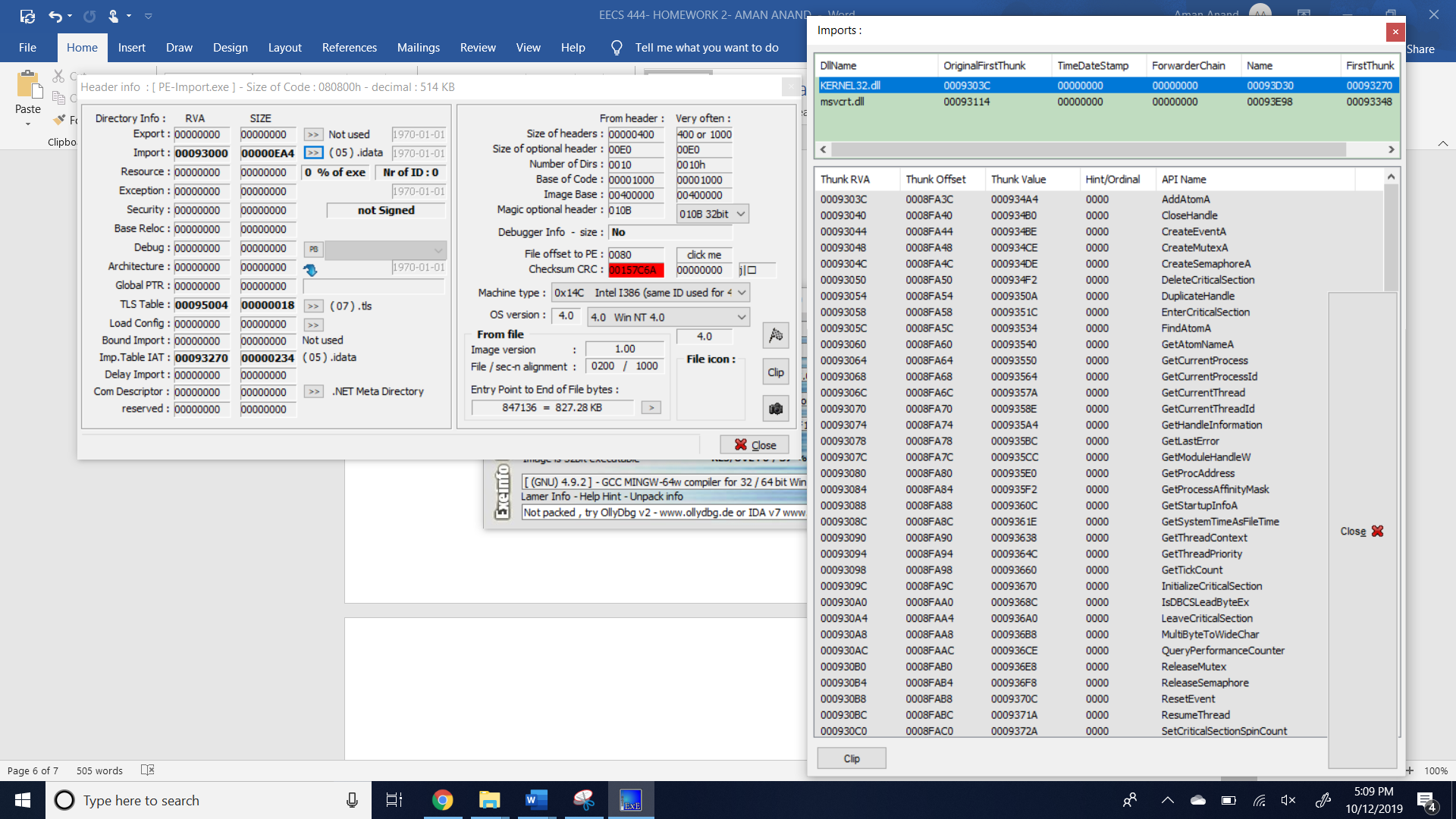
Now for the last time when we run the code we should be able to see the excess data that exists apart from the general line saying I want to learn PE file format!



Thus, it is observed that the program is able to spill and leak out the excess data that was stored along with the trigger line. In this manner all the conditions for the mimic you malware have been met.

**Step 2:**

Now to use the PE edit tool to open and see the Import table for the PE-Import.exe,

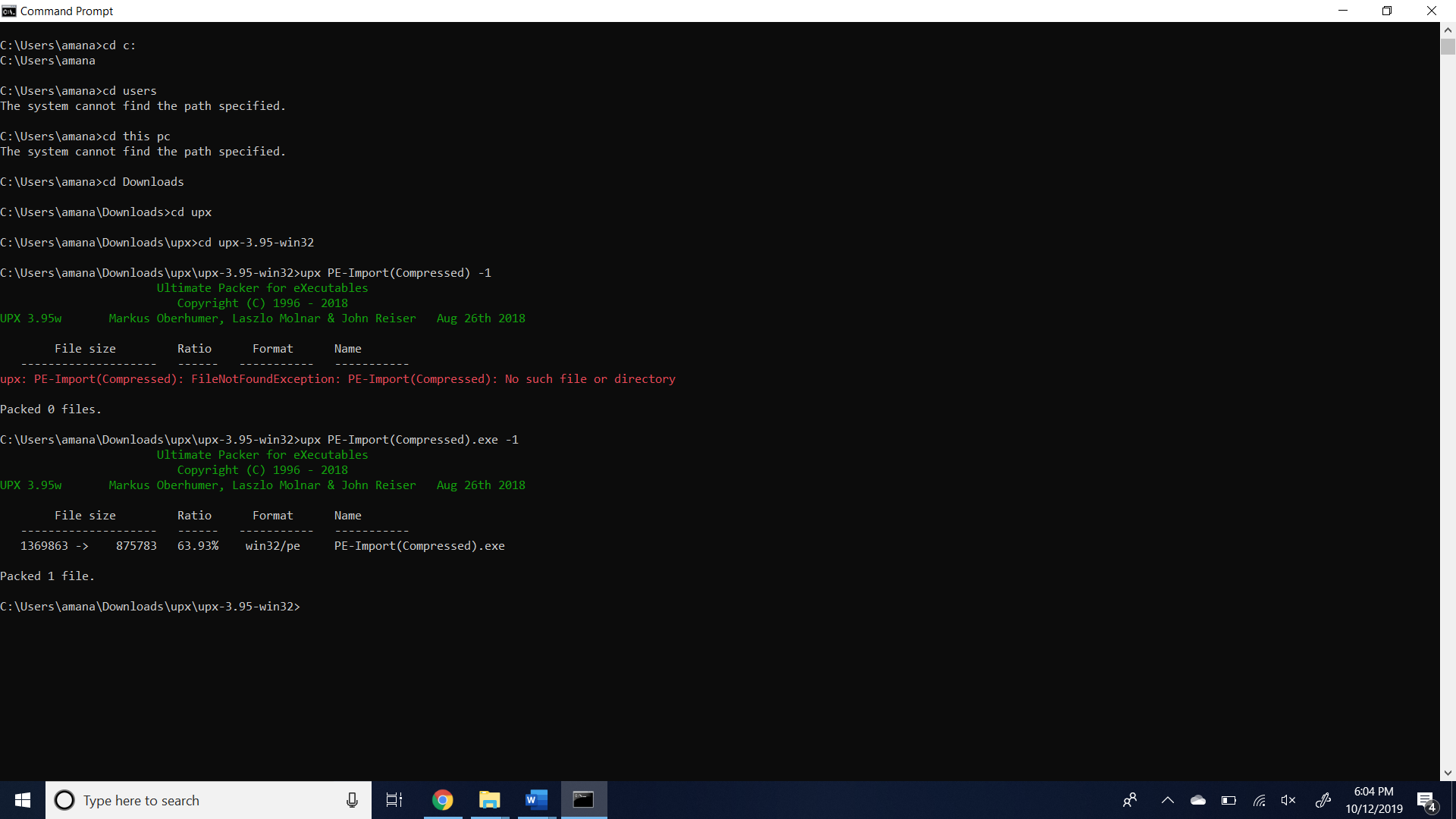


We observe the different rows for the KERNEL32.dll and msvcrt.dll which is in this version of the unpacked PE-Import before we use the packing tool of the UPX.

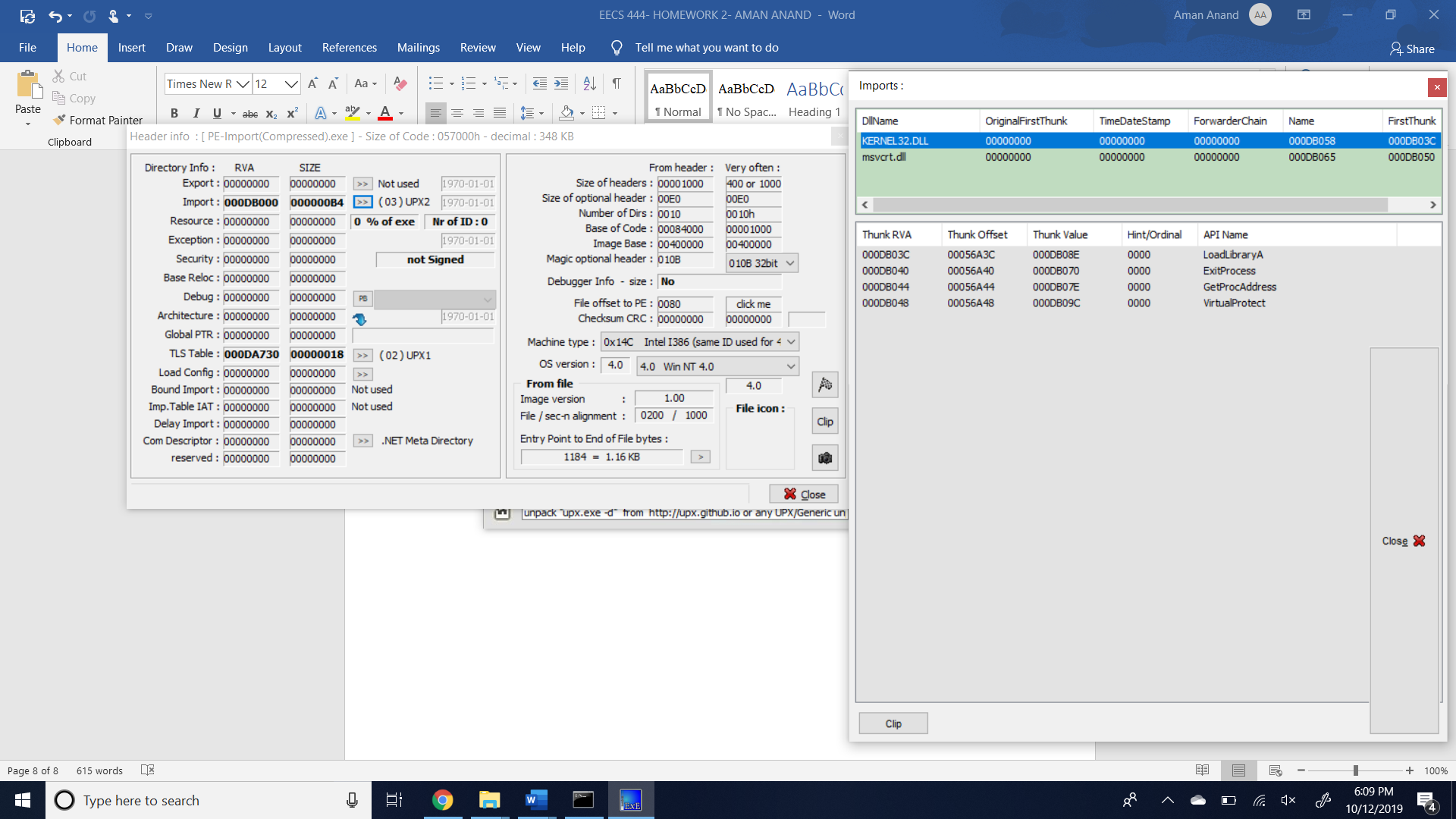
**Step 3:**

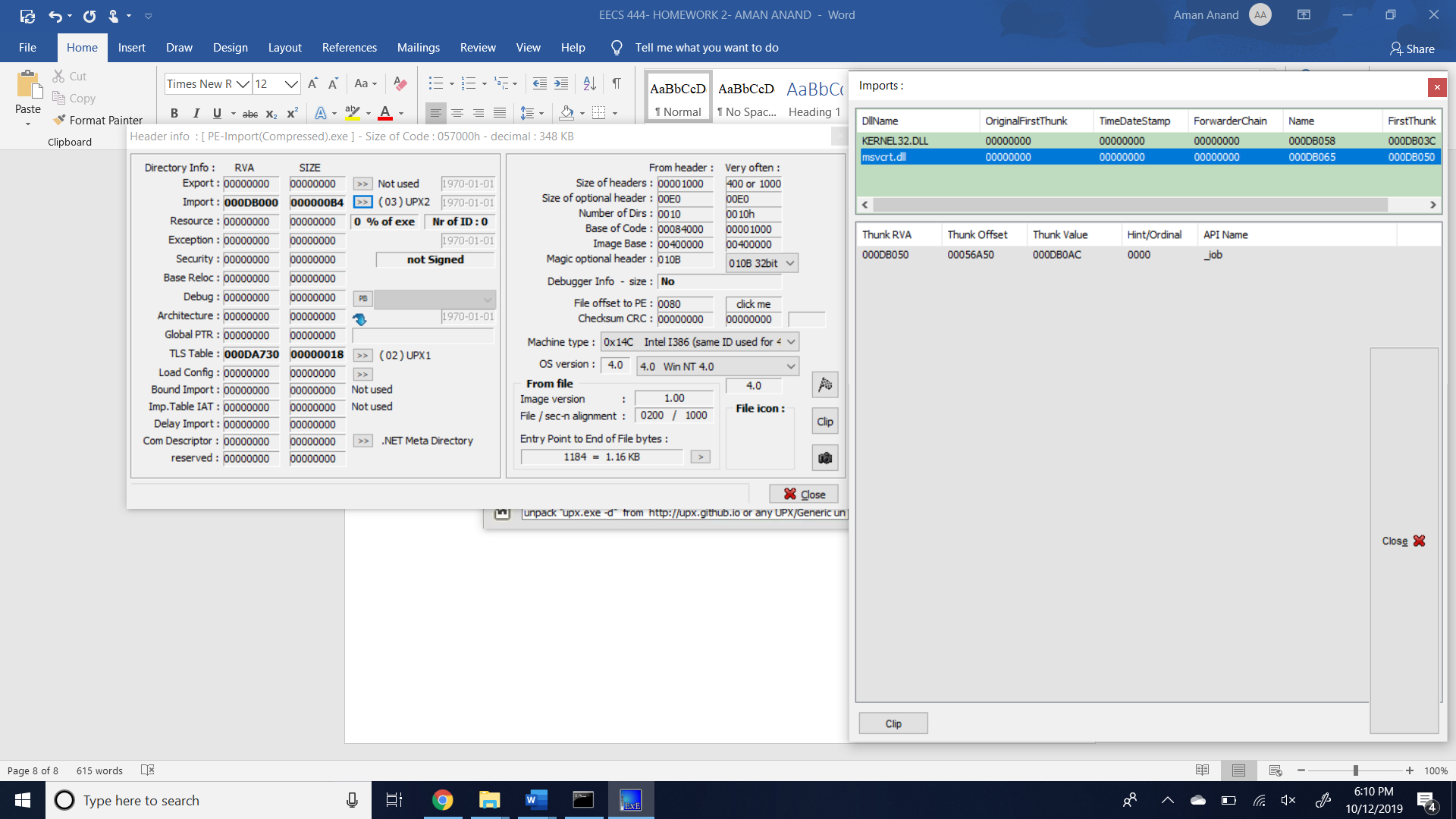
Now we need to pack the PE-Import using the UPX and then observe the difference using the PE edit tool for the file before, after packing and eventually after unpacking again.

Compressing the PE-Import file,



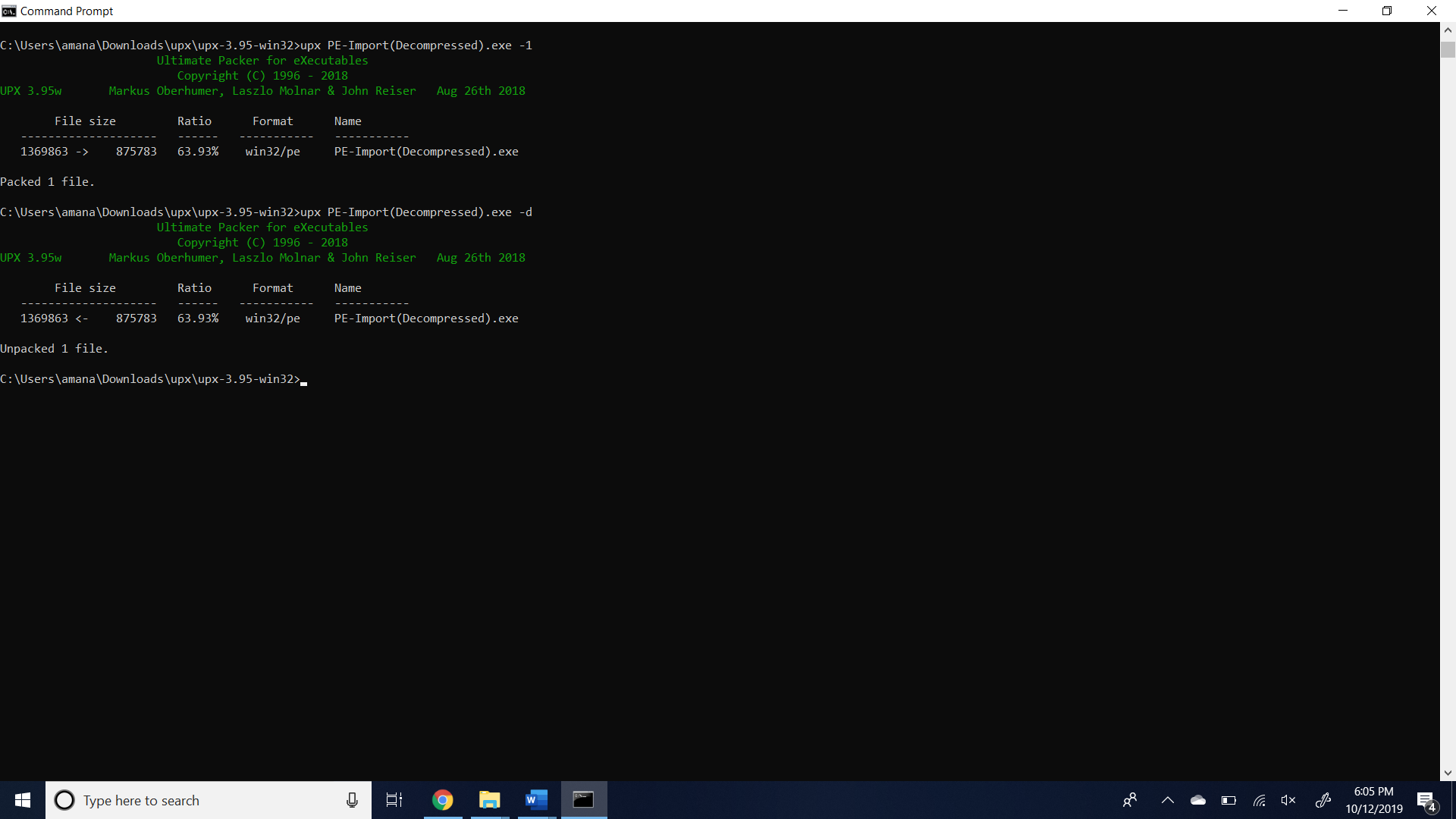
The results from the PE import tool after the packing are as follows,



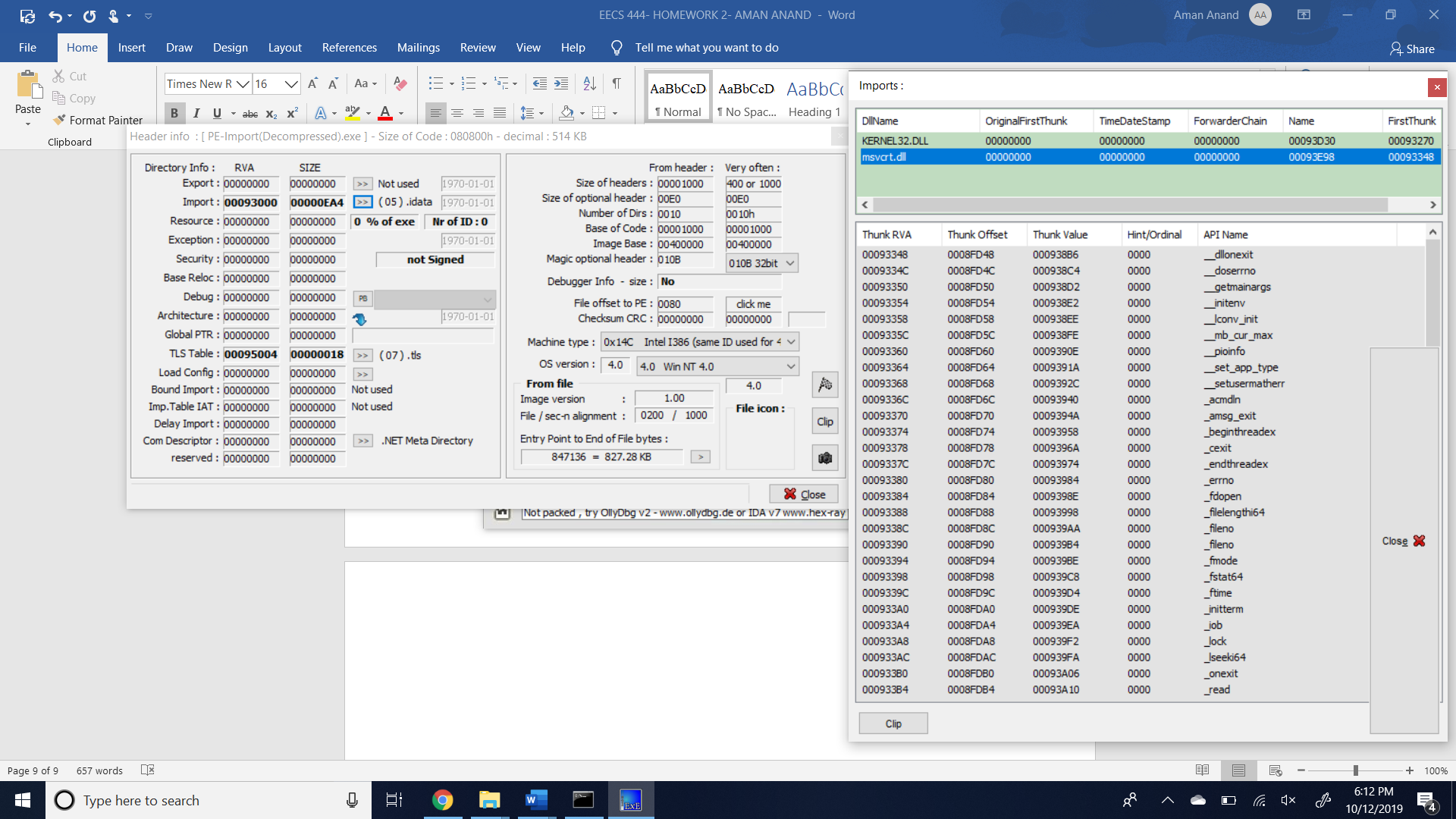


In the compressed version it is noted that the Import tables are compact and shrunk to one or couple of rows as the file has now been packed and hence the Import table is compact and not open for viewing as before.

Now to compress and then decompress the PE-Import file to see the difference,



After compressing this file, now we use the PE edit tool to see the Import table again as follows,

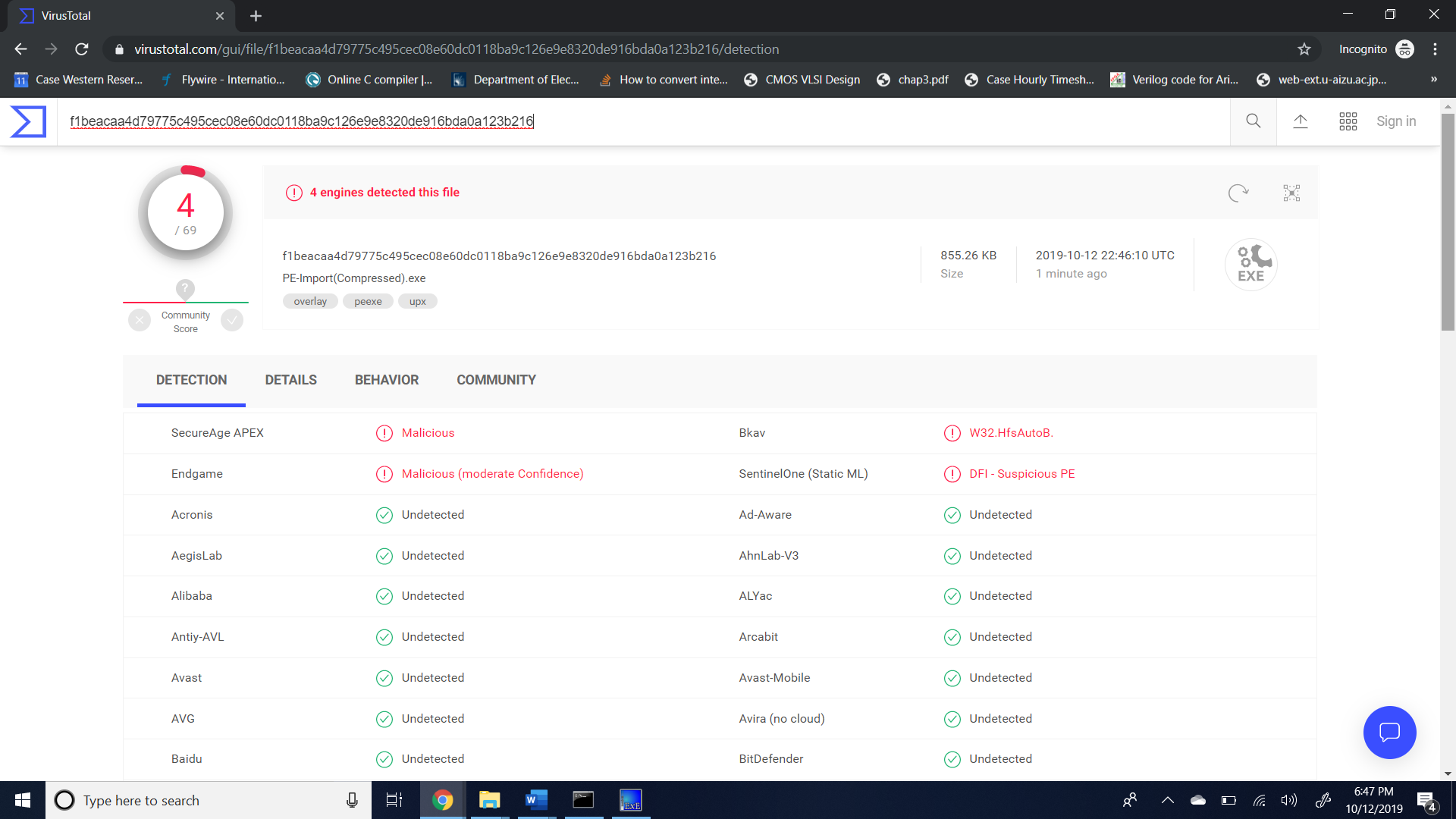


Thus, decompressing the file again brings out the compressed and compact rows of the Import table as multiple rows showing the contents of the executable C program.

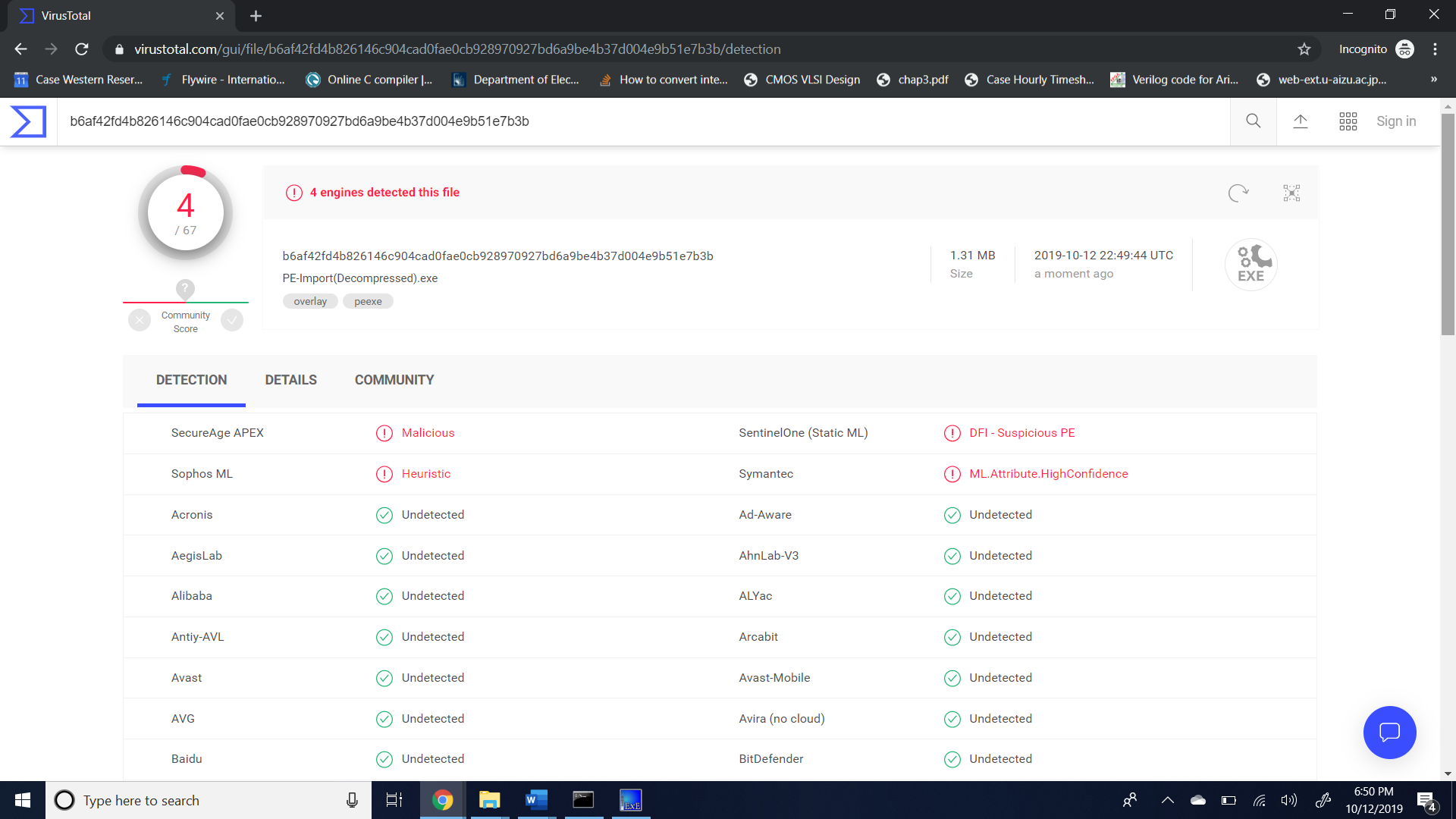
**Step 4:**

The final step is to prepare the code to bypass the Antivirus check by utilizing techniques and then checking if the virustotal website can detect the Mimicware file.

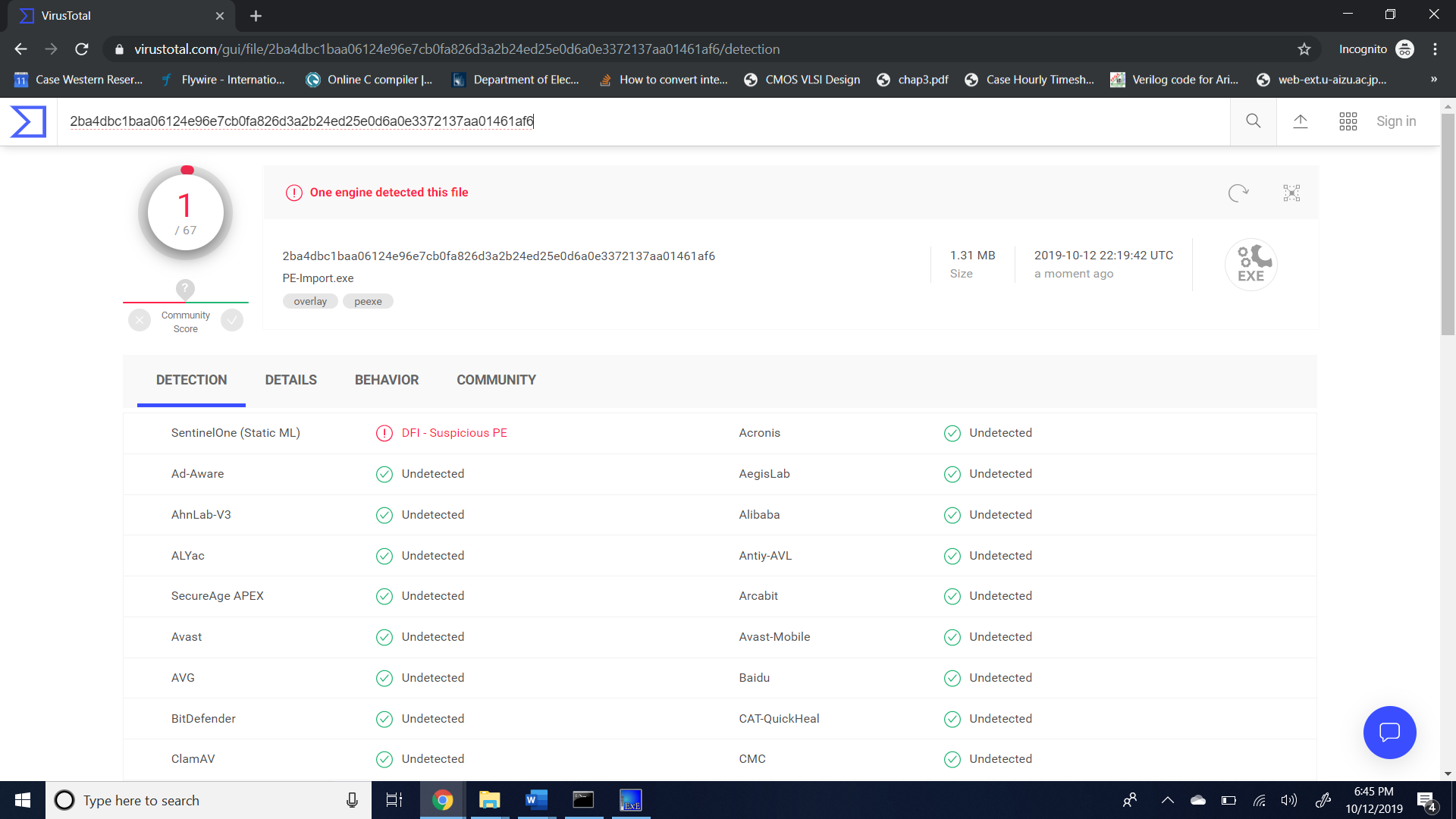
Now to check the packed file,



Now to check the file again after the unpacking step,



Now finally let’s see the file without compression and by using optimal techniques to write the code efficiently with obfuscation to hide the true intention of the malicious code.



Thus, the IP-Import file was successfully created and tested along with seeing the Import Tables at each step and finally by using packing and obfuscation techniques, fooling the virus total scan and then passing 1/67 scans of the antivirus software.