MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY

"KHARKIV POLYTECHNICAL INSTITUTE"

Department of Computer Engineering and Programming

«Software Means of Information Protection »

*Laboratory work report No 3*

*Topic: «* **Code injection in an executable file at the end of a section with API-function calls** *»*

Student:

Group. KH 919 I.E

Ayoub EL-HADDADI

Verified by:

Lecturer Viktor CHELAK

Kharkiv – 2022

***Purpose of work***:

To acquire practical skills for correcting errors in software that is in operation.

Creation of new functionality using implicit in use software for the x64 platform in the masm64 environment.

***Individual task:***

Variant 8: Instruction execution VPHMINPOSUW.

**Algorithm of the program**

First step of the program is to check the ability of the microprocessor to support the AVX and AVX2 commands. Therefore we use **cpuid** command to check bit 28 for **AVX** and bit 5 for **AVX2** of **ecx** register if it’s 1 then they are supported.

Then, we move to part of working with VPHMINPOSUW command. In a loop we load 8 numbers from array in each iteration with the following:

vmovups xmm2, [rsi]

And result of the VPHMINPOSUW command is minimum of those 8 numbers + index which define position of this element in xmm1 using the following:

vphminposuw xmm1, xmm2

We fill an empty array with results and output result in form of message box

Injection in an executable file

A message box must be injected with a title “infected” and a body contains name, surname and group number.

For that purpose we simply jump into an existing message box function and we binary edit the title and message body, or we can create a new string in memory by clicking in empty space with right click, and select binary edit.

After creating new string we can copy its address and replace the string used in message box. After this changes we can patch a new executable file.

After message box function is called we jump directly to the exit process.

The following step is to patch the file, and we can see how many bytes we have changed or added.

**Source Code**

Full source code of this lab you can find it in:

[**https://github.com/Elh-Ayoub/RP\_Labs/blob/main/lab3/lab3.asm**](https://github.com/Elh-Ayoub/RP_Labs/blob/main/lab3/lab3.asm)

**Results of the program:**



Figure 1 – Check support of AVX

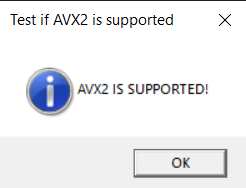


Figure 2 – Check support of AVX2

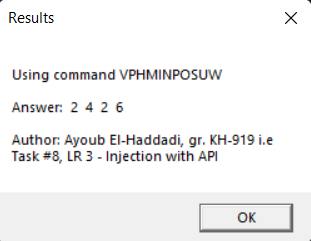


Figure 3 – Program results

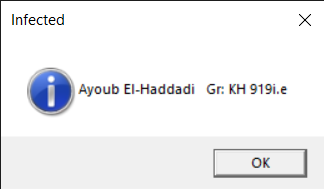


Figure 4 – patched exe file after injecting

**Conclusions:**

As a result of laboratory work we gained a practical skills in injecting and creation of new functionality using implicit in use software for the x64 platform in the masm64 environment.

**You can also find this report in:**

<https://github.com/Elh-Ayoub/RP_Labs/tree/main/Docs>