Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Обчислювальної техніки

Лабораторна робота № 3

з дисципліни «Системне програмування»

на тему

« Дослідження структури програм формату EXE»

Виконала:

студентка гр. ІП-93

Узунлу Гамзенур

Викладач:

доц. Павлов В. Г.

Київ – 2021

**Мета роботи**: Ознайомлення із специфікацією COFF (Common Object File Format). Вивчення прийомів дослідження структури файлів PE-формату.

**Програма 1. Код програми без XOR-шифрування**

.386

.model flat, stdcall

option casemap :none

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Imports \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

include c:/masm32/include/windows.inc

include c:/masm32/include/user32.inc

include c:/masm32/include/kernel32.inc

includelib c:/masm32/include/user32.lib

includelib c:/masm32/include/kernel32.lib

;\*\*\*\*\*\*\*\*\*\*\*\*\*\* Procedure prototypes \*\*\*\*\*\*\*\*\*\*\*\*\*\*

WinMain proto :dword, :dword, :dword, :dword

WndProc proto :dword, :dword, :dword, :dword

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.data

; Window properties

WndTitle DB "Log In Window", 0

WndXPos equ 20

WndYPos equ 20

WndHeight equ 120

WndWidth equ 300

; Label properties

LabelField db "static", 0

LabelText db "Enter the password", 0

LabelID equ 2001

LabelXPos equ 20

LabelYPos equ 20

LabelHeight equ 20

LabelWidth equ 160

; Editing field properties

EditField db "edit", 0

EditID equ 2000

EditXPos equ 20

EditYPos equ 50

EditHeight equ 20

EditWidth equ 160

; Button properties

Button db "button", 0

ButtonText db "Try", 0

ButtonID equ 2002

ButtonXPos equ 200

ButtonYPos equ 20

ButtonHeight equ 20

ButtonWidth equ 70

; Exit button

ExitButtonText db "Exit", 0

ExitButtonID equ 2003

ExitButtonXPos equ 200

ExitButtonYPos equ 50

ExitButtonHeight equ 20

ExitButtonWidth equ 70

; Specific parameters

KeyboardHookWParam equ 40

ErrorMessage db "Try again :(", 0

MsgBoxTitle db "Message", 0

WndClassName DB "BasicWindow", 0

; Student related data

userInfo db "Uzunlu Gamzenur", 13,

"9882", 13,

"11.10.1999", 0

password db "abcdefgh"

len dw 8

.data?

hInstance dd ?

lpCmdLine dd ?

hEditText HWND ?

input db 64 DUP (?)

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.code

start:

; Invoke GUI hooks

invoke GetModuleHandle, NULL

mov hInstance, eax

invoke GetCommandLine

mov lpCmdLine, eax

; Launch main window

invoke WinMain, hInstance, NULL, lpCmdLine, SW\_SHOWDEFAULT

; After it finishes, exit

invoke ExitProcess, eax

WinMain proc hInst :dword, hPrevInst :dword, CmdLine :dword, nShowCmd :dword

; WC - window context

local wc :WNDCLASSEX

; Msg - message

local msg :MSG

local hWnd :HWND

; Apply styles to the window

mov wc.cbSize, sizeof WNDCLASSEX

mov wc.style, CS\_HREDRAW or CS\_VREDRAW or CS\_BYTEALIGNWINDOW

mov wc.lpfnWndProc, WndProc

mov wc.cbClsExtra, NULL

mov wc.cbWndExtra, NULL

; Apply instance

push hInst

pop wc.hInstance

; Apply more window properties

mov wc.hbrBackground, COLOR\_BTNFACE + 1

mov wc.lpszMenuName, NULL

mov wc.lpszClassName, offset WndClassName

; Initialize app icon

invoke LoadIcon, hInst, IDI\_APPLICATION

mov wc.hIcon, eax

mov wc.hIconSm, eax

; Initialize cursor hooks

invoke LoadCursor, hInst, IDC\_ARROW

mov wc.hCursor, eax

; Register window context as a new window

invoke RegisterClassEx, addr wc

; Create GUI window representation

invoke CreateWindowEx, WS\_EX\_APPWINDOW, addr WndClassName, addr WndTitle,

WS\_OVERLAPPEDWINDOW,

WndXPos, WndYPos, WndWidth, WndHeight,

NULL, NULL, hInst, NULL

; Register creation into eax register

mov hWnd, eax

; Show window and update every time it changes

invoke ShowWindow, hWnd, nShowCmd

invoke UpdateWindow, hWnd

; Event emitting

EventEmitter:

invoke GetMessage, addr msg, NULL, 0, 0

cmp eax, 0

je EndOfEvents

invoke TranslateMessage, addr msg

invoke DispatchMessage, addr msg

jmp EventEmitter

EndOfEvents:

mov eax, msg.wParam

ret

WinMain endp

; Window events processor

WndProc proc hWnd :dword, uMsg :dword, wParam :dword, lParam :dword

; If the current event is creating the window

.if uMsg==WM\_CREATE

; Create edit field

invoke CreateWindowEx, NULL,

addr EditField, NULL,

WS\_VISIBLE or WS\_CHILD or ES\_LEFT or ES\_AUTOHSCROLL or ES\_AUTOVSCROLL or WS\_BORDER,

EditXPos, EditYPos, EditWidth, EditHeight,

hWnd, EditID, hInstance, NULL

mov hEditText, eax

; Create label field

invoke CreateWindowEx, NULL,

addr LabelField, addr LabelText,

WS\_VISIBLE or WS\_CHILD,

LabelXPos, LabelYPos, LabelWidth, LabelHeight,

hWnd, LabelID, hInstance, NULL

; Create "Try" button

invoke CreateWindowEx, NULL,

addr Button, addr ButtonText,

WS\_VISIBLE or WS\_CHILD,

ButtonXPos, ButtonYPos, ButtonWidth, ButtonHeight,

hWnd, ButtonID, hInstance, NULL

; Create "Exit" button

invoke CreateWindowEx, NULL,

addr Button, addr ExitButtonText,

WS\_VISIBLE or WS\_CHILD,

ExitButtonXPos, ExitButtonYPos, ExitButtonWidth, ExitButtonHeight,

hWnd, ExitButtonID, hInstance, NULL

; If the event is the window's command

; (button click, whatever)

.elseif uMsg==WM\_COMMAND

; If clicked "Exit", exit

cmp wParam, ExitButtonID

je stopProgram

; If clicked "Try", start checking the password

cmp wParam, ButtonID

jne stop

; Take out the string inside edit field

invoke SendMessage, hEditText, WM\_GETTEXT, KeyboardHookWParam, addr input

; If the length of input and password aren't equal,

; there is an error

cmp ax, len

jne inputError

; Otherwise, start a cycle

mov di, 0

cycle:

; If we checked every letter and it's correct, success

cmp di, len

je inputSuccess

; If we didn't reach the end, check

mov dh, input[di]

cmp dh, password[di]

; If the letters aren't equal, error

jne inputError

inc di

jmp cycle

; Error dialog

inputError:

invoke MessageBox, hWnd, addr ErrorMessage, addr MsgBoxTitle, MB\_OK

jmp stop

; Success dialog

inputSuccess:

invoke MessageBox, hWnd, addr userInfo, addr MsgBoxTitle, MB\_OK

.elseif uMsg == WM\_DESTROY

stopProgram:

invoke PostQuitMessage, 0

jmp stop

.else

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

ret

.endif

; Leave the procedure

stop:

xor eax,eax

ret

WndProc endp

end start

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Поведінка програми: при запуску з’являється вікно, у якому є текст “Enter the password”, поле для вводу та дві кнопки “Try”, “Exit”. При вводі правильного паролю та натискання на кнопку “Try”, показуються дані про студента. Якщо пароль не правильний, програма пропонує спробувати ще раз. При натисканні на кнопку “Exit” програма завершує роботу.

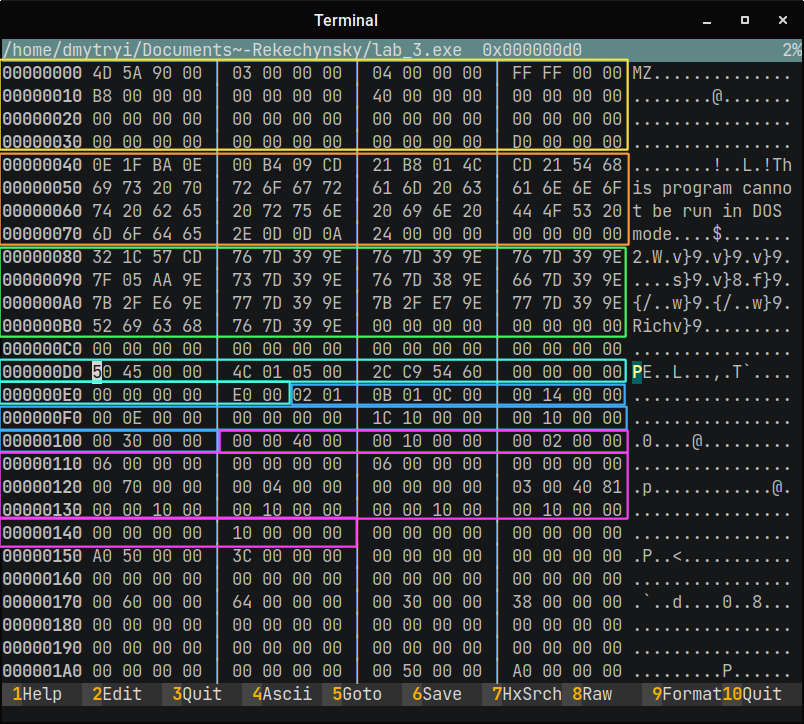
Перевіримо програму на коректність роботи:





Як бачимо, програма показує адекватну поведінку.

Тепер дослідимо її нутрощі:



Перша зона: заголовок MS-DOS (жовтий колір). Починається на всім відомі літери MZ, містить нулі для вирівнювання та посилання на наступну зону: ДОС-заглушку.

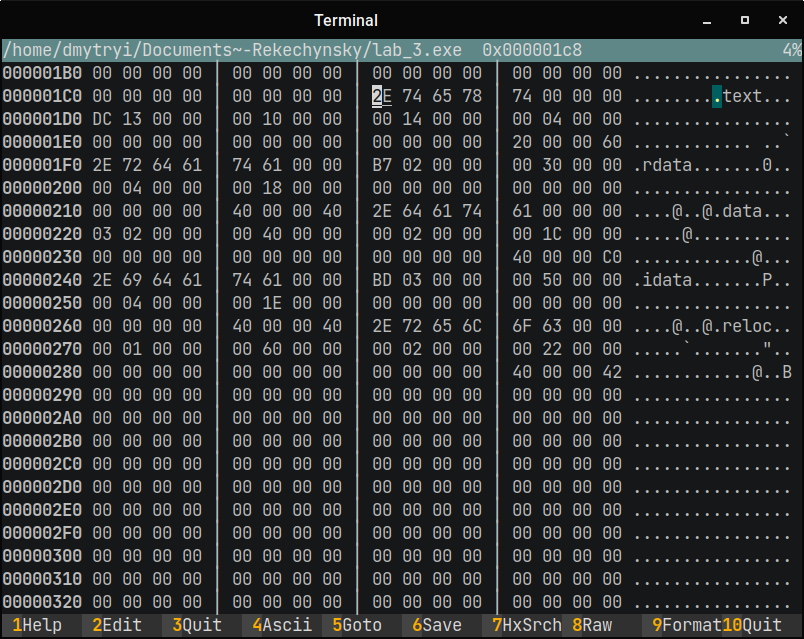
Друга зона (оранжева): виводить на екран “Ця програма не можу бути запущена у середовищі ДОС” та тікає. Ця заглушка була розроблена у часи створення 32-бітної системи Windows, коли вона все ще запускала 16-бітні програми DOS. На даний час її присутність радше традиція, ніж необхідність.

Третя зона (зелена) не розпізнана мною. Радше за все, компілятор вирішив, що туди непогано було б накидати сміття.

Четверта зона (бірюзова) починається із сигнатури PE (Portable Executable) та містить заголовок файлу формату PE.

П’ята зона (синя) містить стандартні поля PE.

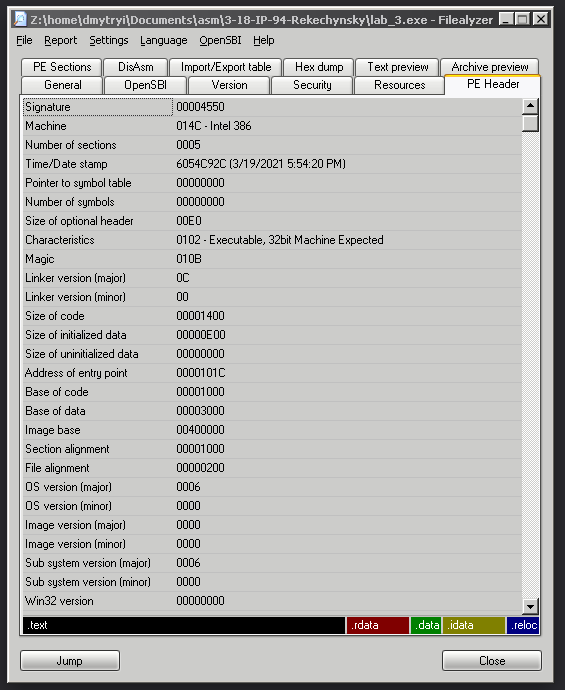
Шоста зона (фіолетова) містить NT (New Technology, Windows версії 3.1 та вище) поля PE.

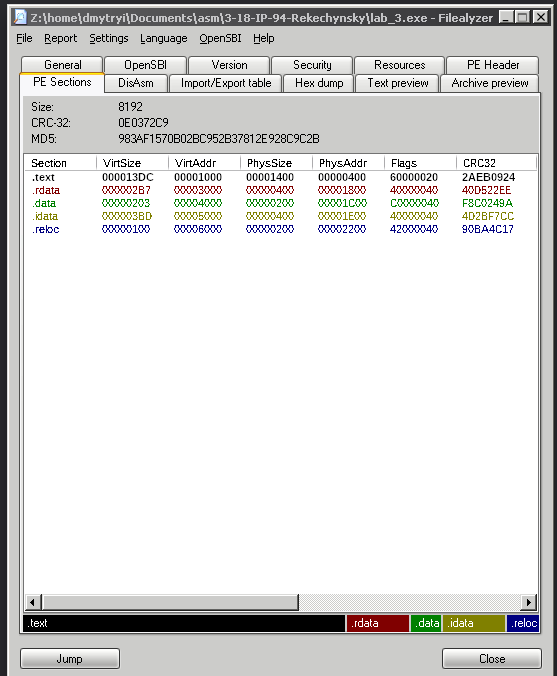
На цьому скриншоті показано секційні заголовки (починаючи з рядка 0x1С, з того місця, де показує курсор).

Дані:

|  |  |
| --- | --- |
| Machine | 0x14C, Intel 386 |
| Magic | 0x10B, PE32 |
| MajorLinkerVersion | 0xС |
| MinorLinkerVersion | 0x0 (Версія лінкера 12.0) |
| SizeOfCode | 0x1400 (5120 секцій коду) |
| SizeOfInitializedData | 0xE00 (3584 секцій ініціалізованих даних) |
| SizeOfUninitializedData | 0x0 (неініціалізованих даних нема) |
| AddressOfEntryPoint | 0x101C (адреса точки входу) |
| BaseOfCode | 0x1000 (початок розідлу коду, який завантажується у пам’ять) |
| BaseOfData | 0x3000 (початок розділу даних) |
| Name | .text: 74 78 65 74 2E  .rdata: 61 74 61 64 72 2E  .data: 61 74 61 64 2E |
| VirtualSize | .text: 0x13DC  .rdata: 0x2B7  .data: 0x203 |
| VirtualAddress | .text: 0x1000  .rdata: 0x3000  .data: 0x4000 |
| SizeOfRawData | .text: 0x1400  .rdata: 0x4000  .data: 0x4000 |
| PointerToRawData | .text: 0x4000  .rdata: 0x1800  .data: 0x200 |
| PointerToRelocations | .text: 0x0  .rdata: 0x0  .data: 0x0 |
| PointerToLineNumbers | .text: 0x0  .rdata: 0x0  .data: 0x0 |
| NumberOfRelocations | .text: 0x0  .rdata: 0x0  .data: 0x0 |
| NumberOfLineNumbers | .text: 0x0  .rdata: 0x0  .data: 0x0 |
| Charecteristics | .text: 0x60000020  Є доступ на зчитування та виконування. Містить виконуваний код.  .rdata: 0x40000040  Є доступ на зчитування. Містить початкові дані.  .data: 0xС0000040  Є доступ на запис та зчитування. Містить ініціалізовані дані. |

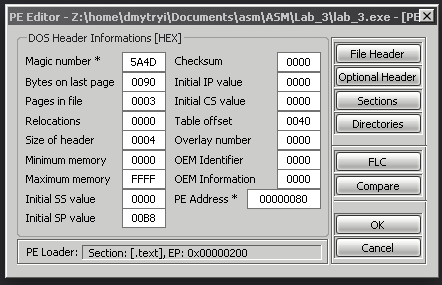
Скріншоти, що підтверджують правильність таблиці (FileAlyzer 1.6.0.4):

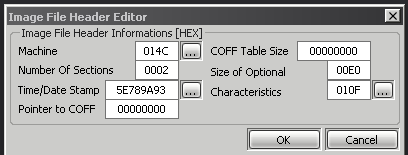


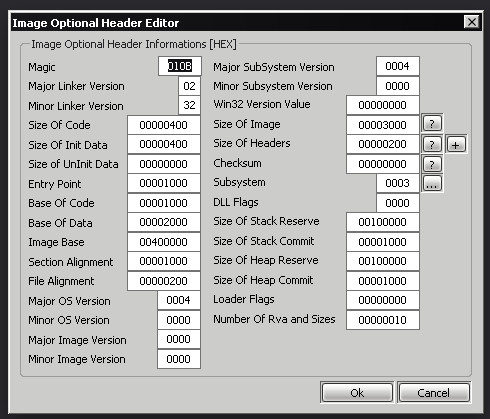


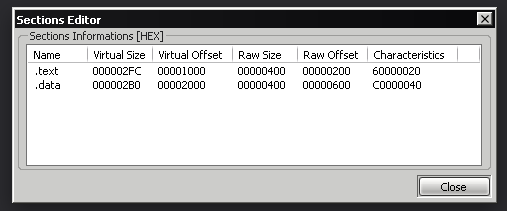
Отже, все правильно.

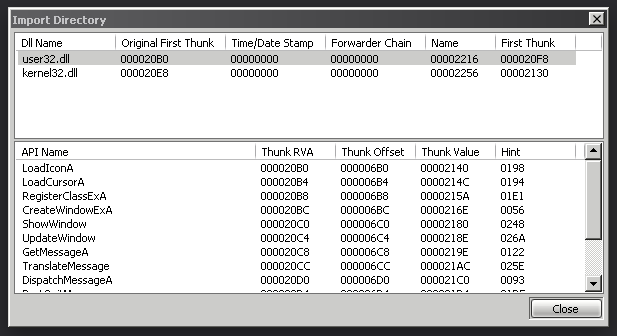
Тепер розглянемо дані, надані PETools:

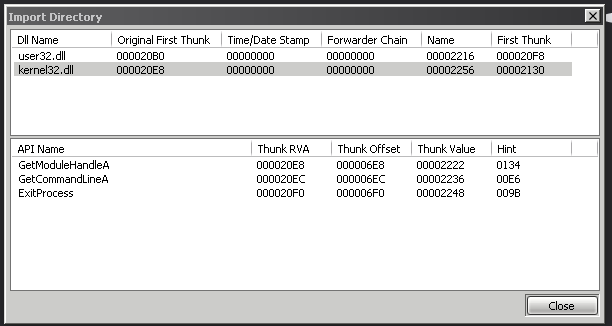




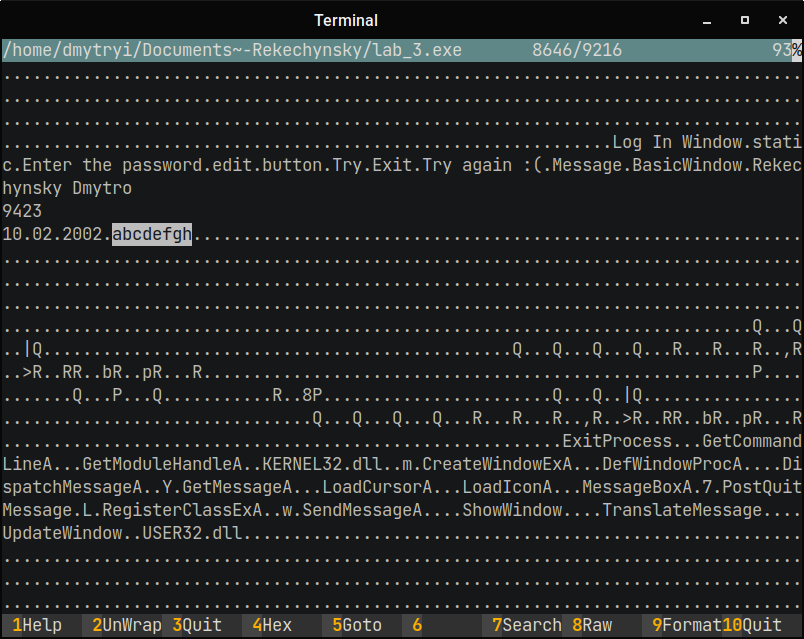


Тепер, що ж імпортувалось?

З user32.dll імпортувалось чимало процедур, які стосуються взаємодії з графічним інтерфейсом.

З kernel32.dll імортувались процедури, які стосуються міжпроцесової взаємодії.

До речі, зверніть увагу, що пароль у коді видимий:



А тепер, як сховати пароль? Найпростіше, але елегантно робоче, це XOR-шифрування.

Нехай ключем буде число 5 (000001012), а паролем так і лишиться abcdefgh. Тоді:

a: 97 = 01100001

01100001

XOR

00000101

=

01100100 (d)

b: 98 = 01100010

01100010

XOR

00000101

=

01100111 (g)

c: 99 = 01100011

01100011

XOR

00000101

=

01100110 (f)

d: 100 = 01100100

01100100

XOR

00000101

=

01100001 (a)

e: 101= 01100101

01100101

XOR

00000101

=

01100000 (`)

f: 102 = 01100110

01100110

XOR

00000101

=

01100011 (c)

g: 103 = 01100111

01100111

XOR

00000101

=

01100010 (b)

h: 104 = 01101000

01101000

XOR

00000101

=

01101101 (m)

Нескладним чином отримаємо dgfa`cbm.

Якщо провернути ту саму операцію над зашифрования паролем, отримаємо розшифрований

Лишається до програми дописати рядок, який відповідає за операцію xor над символом, а ще замість відкритого паролю вставити зашифрований. А також вставити ключ.

Маємо:

.386

.model flat, stdcall

option casemap :none

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Imports \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

include c:/masm32/include/windows.inc

include c:/masm32/include/user32.inc

include c:/masm32/include/kernel32.inc

includelib c:/masm32/include/user32.lib

includelib c:/masm32/include/kernel32.lib

;\*\*\*\*\*\*\*\*\*\*\*\*\*\* Procedure prototypes \*\*\*\*\*\*\*\*\*\*\*\*\*\*

WinMain proto :dword, :dword, :dword, :dword

WndProc proto :dword, :dword, :dword, :dword

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.data

; Window properties

WndTitle DB "Log In Window", 0

WndXPos equ 20

WndYPos equ 20

WndHeight equ 120

WndWidth equ 300

; Label properties

LabelField db "static", 0

LabelText db "Enter the password", 0

LabelID equ 2001

LabelXPos equ 20

LabelYPos equ 20

LabelHeight equ 20

LabelWidth equ 160

; Editing field properties

EditField db "edit", 0

EditID equ 2000

EditXPos equ 20

EditYPos equ 50

EditHeight equ 20

EditWidth equ 160

; Button properties

Button db "button", 0

ButtonText db "Try", 0

ButtonID equ 2002

ButtonXPos equ 200

ButtonYPos equ 20

ButtonHeight equ 20

ButtonWidth equ 70

; Exit button

ExitButtonText db "Exit", 0

ExitButtonID equ 2003

ExitButtonXPos equ 200

ExitButtonYPos equ 50

ExitButtonHeight equ 20

ExitButtonWidth equ 70

; Specific parameters

KeyboardHookWParam equ 40

ErrorMessage db "Try again :(", 0

MsgBoxTitle db "Message", 0

WndClassName DB "BasicWindow", 0

; Student related data

userInfo db "Rekechynsky Dmytro", 13,

"9423", 13,

"10.02.2002", 0

; The password is "abcdefgh"

password db "dgfa`cbm"

key db 00000101b

len dw 8

.data?

hInstance dd ?

lpCmdLine dd ?

hEditText HWND ?

input db 64 DUP (?)

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.code

start:

; Invoke GUI hooks

invoke GetModuleHandle, NULL

mov hInstance, eax

invoke GetCommandLine

mov lpCmdLine, eax

; Launch main window

invoke WinMain, hInstance, NULL, lpCmdLine, SW\_SHOWDEFAULT

; After it finishes, exit

invoke ExitProcess, eax

WinMain proc hInst :dword, hPrevInst :dword, CmdLine :dword, nShowCmd :dword

; WC - window context

local wc :WNDCLASSEX

; Msg - message

local msg :MSG

local hWnd :HWND

; Apply styles to the window

mov wc.cbSize, sizeof WNDCLASSEX

mov wc.style, CS\_HREDRAW or CS\_VREDRAW or CS\_BYTEALIGNWINDOW

mov wc.lpfnWndProc, WndProc

mov wc.cbClsExtra, NULL

mov wc.cbWndExtra, NULL

; Apply instance

push hInst

pop wc.hInstance

; Apply more window properties

mov wc.hbrBackground, COLOR\_BTNFACE + 1

mov wc.lpszMenuName, NULL

mov wc.lpszClassName, offset WndClassName

; Initialize app icon

invoke LoadIcon, hInst, IDI\_APPLICATION

mov wc.hIcon, eax

mov wc.hIconSm, eax

; Initialize cursor hooks

invoke LoadCursor, hInst, IDC\_ARROW

mov wc.hCursor, eax

; Register window context as a new window

invoke RegisterClassEx, addr wc

; Create GUI window representation

invoke CreateWindowEx, WS\_EX\_APPWINDOW, addr WndClassName, addr WndTitle,

WS\_OVERLAPPEDWINDOW,

WndXPos, WndYPos, WndWidth, WndHeight,

NULL, NULL, hInst, NULL

; Register creation into eax register

mov hWnd, eax

; Show window and update every time it changes

invoke ShowWindow, hWnd, nShowCmd

invoke UpdateWindow, hWnd

; Event emitting

EventEmitter:

invoke GetMessage, addr msg, NULL, 0, 0

cmp eax, 0

je EndOfEvents

invoke TranslateMessage, addr msg

invoke DispatchMessage, addr msg

jmp EventEmitter

EndOfEvents:

mov eax, msg.wParam

ret

WinMain endp

; Window events processor

WndProc proc hWnd :dword, uMsg :dword, wParam :dword, lParam :dword

; If the current event is creating the window

.if uMsg==WM\_CREATE

; Create edit field

invoke CreateWindowEx, NULL,

addr EditField, NULL,

WS\_VISIBLE or WS\_CHILD or ES\_LEFT or ES\_AUTOHSCROLL or ES\_AUTOVSCROLL or WS\_BORDER,

EditXPos, EditYPos, EditWidth, EditHeight,

hWnd, EditID, hInstance, NULL

mov hEditText, eax

; Create label field

invoke CreateWindowEx, NULL,

addr LabelField, addr LabelText,

WS\_VISIBLE or WS\_CHILD,

LabelXPos, LabelYPos, LabelWidth, LabelHeight,

hWnd, LabelID, hInstance, NULL

; Create "Try" button

invoke CreateWindowEx, NULL,

addr Button, addr ButtonText,

WS\_VISIBLE or WS\_CHILD,

ButtonXPos, ButtonYPos, ButtonWidth, ButtonHeight,

hWnd, ButtonID, hInstance, NULL

; Create "Exit" button

invoke CreateWindowEx, NULL,

addr Button, addr ExitButtonText,

WS\_VISIBLE or WS\_CHILD,

ExitButtonXPos, ExitButtonYPos, ExitButtonWidth, ExitButtonHeight,

hWnd, ExitButtonID, hInstance, NULL

; If the event is the window's command

; (button click, whatever)

.elseif uMsg==WM\_COMMAND

; If clicked "Exit", exit

cmp wParam, ExitButtonID

je stopProgram

; If clicked "Try", start checking the password

cmp wParam, ButtonID

jne stop

; Take out the string inside edit field

invoke SendMessage, hEditText, WM\_GETTEXT, KeyboardHookWParam, addr input

; If the length of input and password aren't equal,

; there is an error

cmp ax, len

jne inputError

; Otherwise, start a cycle

mov di, 0

cycle:

; If we checked every letter and it's correct, success

cmp di, len

je inputSuccess

; If we didn't reach the end, check

mov dh, input[di]

xor dh, key

cmp dh, password[di]

; If the letters aren't equal, error

jne inputError

inc di

jmp cycle

; Error dialog

inputError:

invoke MessageBox, hWnd, addr ErrorMessage, addr MsgBoxTitle, MB\_OK

jmp stop

; Success dialog

inputSuccess:

invoke MessageBox, hWnd, addr userInfo, addr MsgBoxTitle, MB\_OK

.elseif uMsg == WM\_DESTROY

stopProgram:

invoke PostQuitMessage, 0

jmp stop

.else

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

ret

.endif

; Leave the procedure

stop:

xor eax,eax

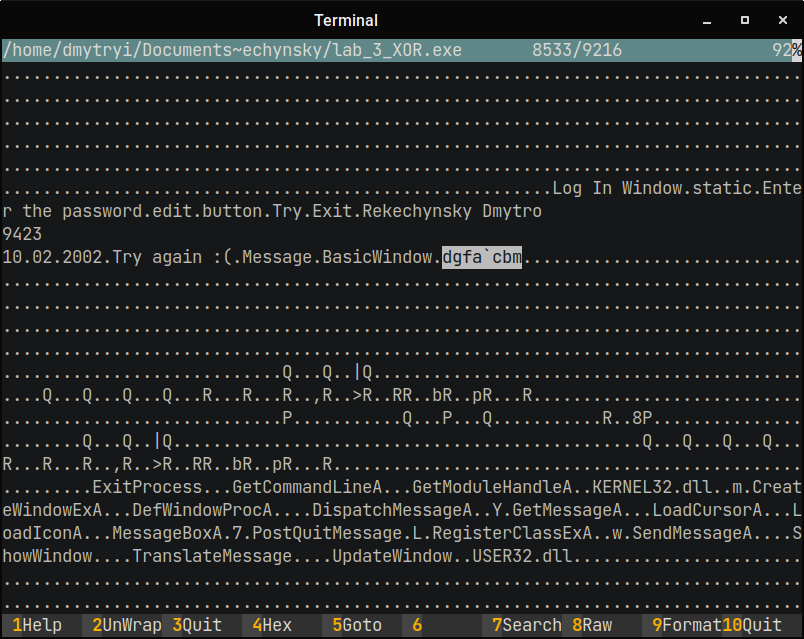
ret

WndProc endp

end start

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Спробуємо виявити пароль у скомпільованому коді:



Паролю більше не виявлено, до побачення.

ВИСНОВОК. Розроблено програму із графічним інтерфейсом, яка запитує пароль. Проаналізовано вміст файлу та визначено за ним характеристики програми та її секцій. Написано нову програму, в якій задіяне XOR-шифрування. Складено звіт. Роботу виконано успішно.