**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут 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.»

Виконав: Перевірив:

Студент 2 курсу ФІОТ доц. Павлов В.Г.

Групи ІМ-13

Ситник Денис Олександрович

Номер у списку групи: 22

Київ – 2023

**Мета:** Ознайомлення із специфікацією COFF (Common Object File Format). Вивчення прийомів дослідження структури файлів PEформату.

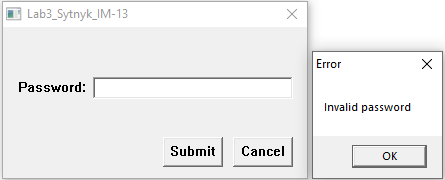
**1)** є

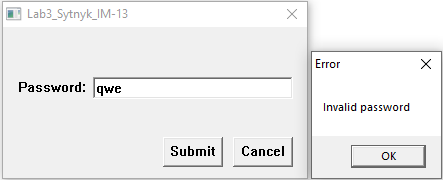
**2)** є

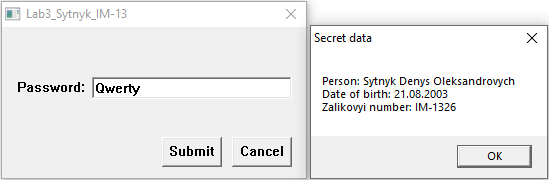
**3)** є

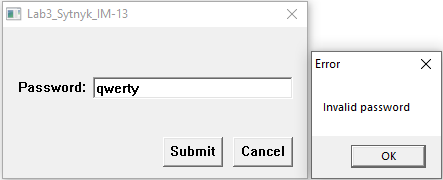
**4)**

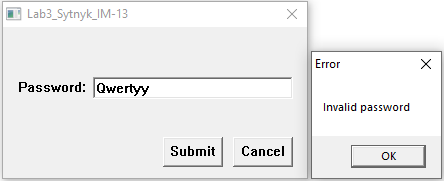
**4.1) Без шифрування**

****

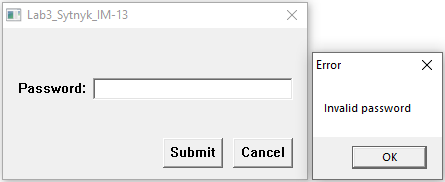
****

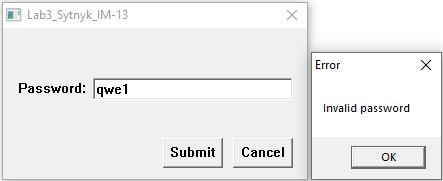
****

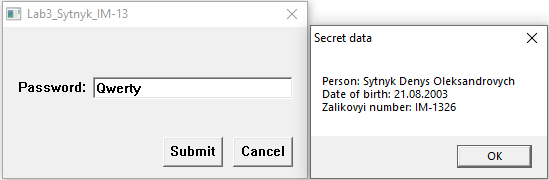
****

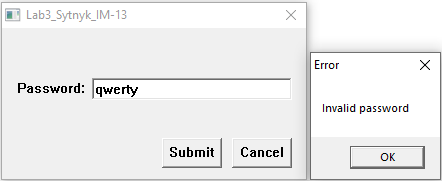
****

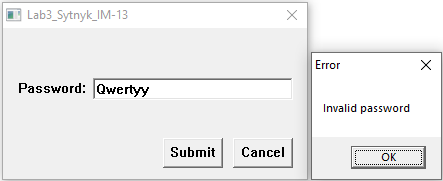
**4.2) З шифруванням**

****

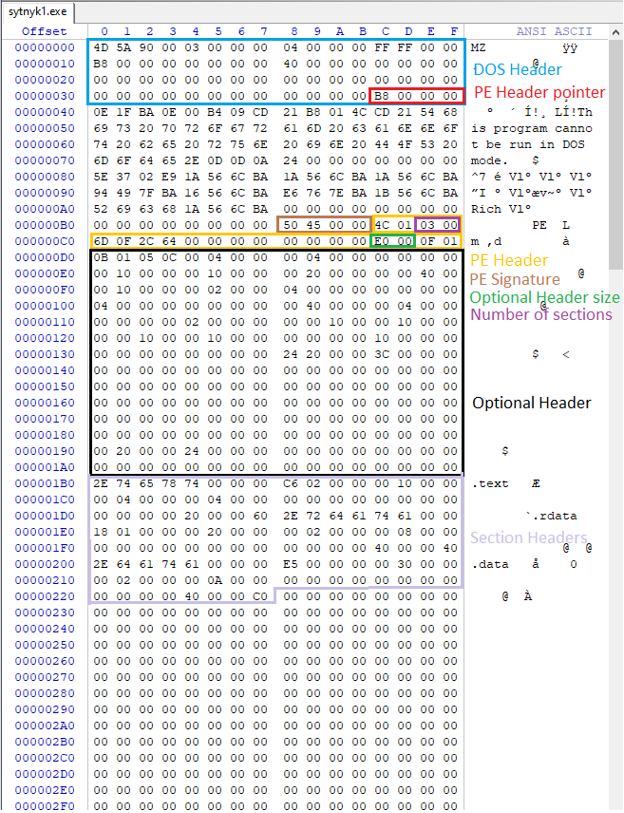
****

****

****



**5-6)**

****

Отже, яка ж структура моєї програми. Перші 64 байти відповідають за DOS Header. В цьому ж хедері останні 4 байти – це вказівник на PE Header, який прописаний у зворотньому порядку, що означає, що він здвинутий на 000000B8 – 184 байти.

Йдемо до 185 байту і бачимо PE Signature 4-байтової сигнатури «PE\0\0». Одразу після нього знаходиться 20-байтовий PE Header. За ним йде Optional Header (чорний), розмір якого можна дізнатись у двобайтовому полі, що виділено зеленим. Звідси дізнаємось, що Optional Header має довжину 224 байти.   
Відразу після Optional Header починається Section Headers, розмір якого також дізнаємось з двобайтового поля у PE Header: 0300 (виділене фіолетовим). Це означає, що у нас є 3 секції, кожна по 40 байтів.

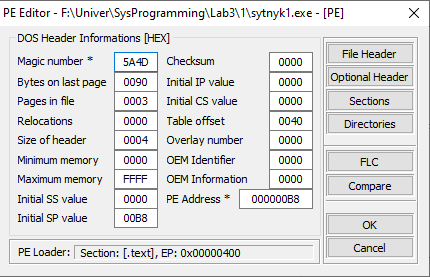
**7-8)**

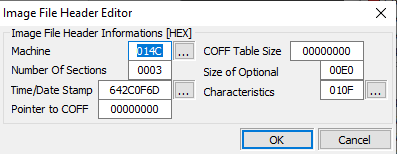
|  |  |  |
| --- | --- | --- |
| 1)Розділ 3.3.1 **Machine Types** | | |
| Поле | Значення | Опис |
| Machine type | 014С | Тип машини: *Intel 386* або пізніше або інші схожі процесори |

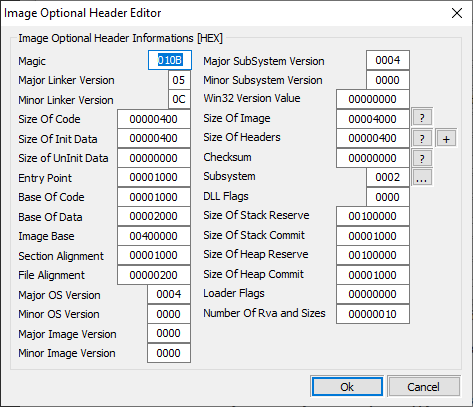
|  |  |  |
| --- | --- | --- |
| 2) Розділ 3.4.1 **Optional Header Standard Fields (Image Only)** | | |
| Поле | Значення | Опис |
| Magic | 010В | Беззнакове ціле число (unsigned int), що ідентифікує стан файлу.  *0x10B* – ідентифікатор звичайного виконуваного файлу (.exe). |
| MajorLinkerVersion | 05 | Номер старшої версії компонувальника |
| MinorLinkerVersion | 0С | Номер другорядної версії компонувальника |
| SizeOfCode | 00000400 | Розмір усіх секцій з програмним кодом |
| SizeOfInitializedData | 00000400 | Розмір усіх секцій з ініціалізованими даними |
| SizeOfUninitializedData | 00000000 | Розмір усіх секцій з неініціалізованими даними |
| AddressOfEntryPoint | 00001000 | Адреса вхідної точки програми |
| BaseOfCode | 00001000 | Адреса початку розділу коду |
| BaseOfData | 00002000 | Адреса початку розділу змінних |

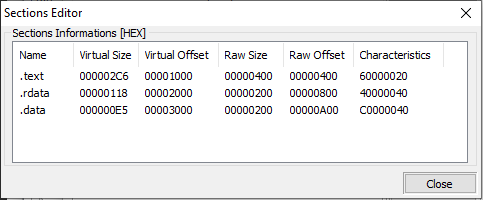
|  |  |  |
| --- | --- | --- |
| 3) Розділ 4 **Section Table (Section Headers), section .data** | | |
| Поле | Значення | Опис |
| Name | .text\0\0\0 | 8-байтовий рядок ASCII із доповненням нульовими значеннями |
| VirtualSize | 000000E5 | Загальний розмір секції в пам’яті |
| VirtualAdress | 00003000 | Адреса першого байту розділу |
| SizeOfRawData | 00000200 | Розмір розділу у файлі |
| PointerToRawData | 00000A00 | Вказівник на першу сторінку розділу в COFF файл |
| PointerToRelocations | 00000000 | Вказівник файлу на початок записів переміщення для розділу (0 для .exe) |
| PointerToLinenumbers | 00000000 | Вказівник файлу на початок записів номерів рядків для розділу |
| NumberOfRelocations | 0000 | Кількість релокаційних входжень в розділ (0 для .ехе) |
| NumberOfLinenumbers | 0000 | Кількість номерних записів розділу |
| Characteristics | C0000040 | Прапори, що описують характеристики розділу |

**9) Скріншоти з PE Tools:**

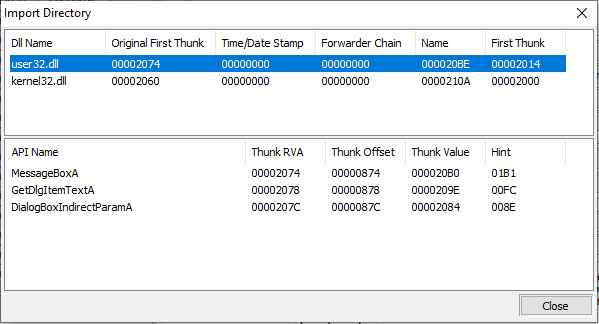
****

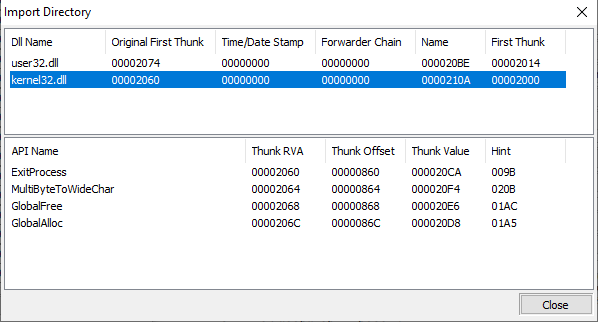
****

****

****

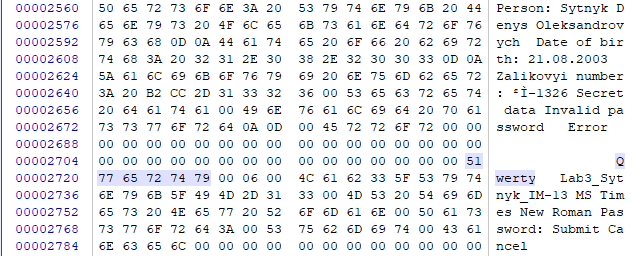
**10) Import Directory**

****

****

У розділі API Name видно які функції імпортуються.

**11)** За даними з PE Editor, .rdata має зсув 00002000, отже шукаємо



І знаходимо тут пароль у відкритому вигляді.

**12)** Шифрування паролю за допомогою фукнції XOR

Зашифровуємо пароль:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Пароль | | | | | | |
| **ASCII** | Q | w | e | r | t | y |
| **Binary** | 01010001 | 01110111 | 01100101 | 01110010 | 01110100 | 01111001 |

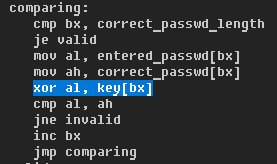
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ключ | | | | | | |
| **ASCII** | p | V | C | L | W | N |
| **Binary** | 01110000 | 01010110 | 01000011 | 01001100 | 01010111 | 01001110 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Шифрування | | | | | | |
| **Binary** | 00100001 | 00100001 | 00100110 | 00111110 | 00100011 | 00110111 |
| **ASCII** | ! | ! | & | > | # | 7 |

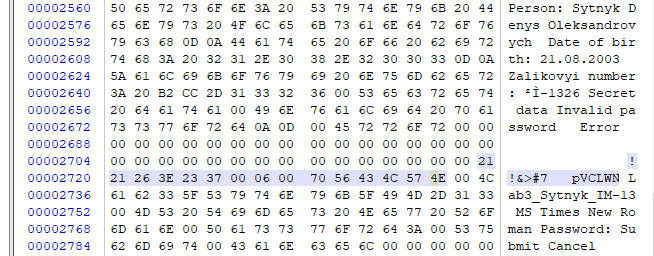
Вставляємо в програму зашифрований пароль «!!&>#7» замість відкритого і ключ:



а також додаємо у функцію перевірки пароля шифрування введеного користувачем в поле вводу рядка:



Створюємо та відкриваємо .exe з шифруванням в WinHex



І бачимо тут зашифрований пароль і ключ замість відкритого.  
**13)** Висновок

Отже, виконавши лабораторну роботу я ознайомися із специфікацією **COFF** (Common Object File Format) і вивчив прийоми дослідження структури файлів PE-формату, навчився знаходити заголовки, які містять службову інформацію, і секції, які описують різні властивості виконуваного файлу і його структуру, за допомогою допоміжних програм (**WinHex та PE Editor**), а також шифрувати дані функцією XOR, з метою приховати секретні дані від зловмисників. Таким чином пароль у явному вигляді зберігатись не буде, але все ж, якщо людина розуміється хоч трохи на структурі файлів PE-формату і шифруванні, то без проблем зможе отримати зашифрований пароль, і ключ, за допомогою якого і дешифрує його.

Насправді цей метод шифрування на сьогодні вже є неактуальним і вразливим. Натомість використовується асиметричне шифрування, яке є в рази безпечнішим, оскільки зловмисникам для реалізації своїх планів потрібно якимось чином дізнатись закритий ключ, який зберігається лише на локальних машинах і серверах, і нікому не видається.