**Міністерство освіти і науки України Національний технічний університет України «Київський політехнічний інститут імені 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 ІІ курсу ФІОТ доц. Павлов В. Г.

групи ІМ-12

Сутулов Нікіта Олегович

номер у списку групи (варіант): 20

номер залікової книжки: 1229

Київ 2023

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

**Порядок виконання роботи**

1. Вивчити структуру програм формату EXE [1].

2. Розробити програму на мові Асемблер, за допомогою якої у віконному інтерфейсі по введеному паролю виводяться персональні дані студента – ПІБ, дата народження, номер залікової книжки тощо (див. лаб. роботу 1).

3. Виконати компіляцію розробленого файлу у формат EXE.

4. Перевірити роботу програми шляхом введення як правильного, так і невірного паролів.

5. Отриманий виконавчий файл дослідити за допомогою програми HEX-редактора HIEW32 (див. лаб. роботу 2) або WinHex (http://rainbowsky.ru/system/winhex/ - trial версия\*)[2].

6. На скріншоті перших 25 рядків вмісту файлу обвести кольоровим олівцем або фломастером області MS-DOS заголовка (DOS\_HEADER), PE заголовка (PE\_HEADER) і таблиці секцій (SECTION\_HEADERS). Скріншот привести в звіті по лабораторній роботі.

7. Відповідно до опису секцій [1] скласти таблицю, в яку занести параметри свого файлу, вказані в розділах 3.3.1, 3.4.1 і 4 (перша таблиця).

8. У останньому стовпчику таблиці розшифрувати виписані значення полів заголовка файлу. Таблицю привести в звіті по лабораторній роботі.

9. Провести дослідження того ж файлу за допомогою меню "PE Editor" безкоштовної програми PE Tools (http://soft.mydiv.net/win/download-PE-Tools.html\*). Все скріншоти вікон програми з даними, відповідними раніше побудованій таблиці, привести в звіті по лабораторній роботі.

10

10. Дослідити таблицю імпорту (Import Directory) даного файлу і визначити, які саме функції використовуються з бібліотек, що підключаються. Скріншоти вікон Import Directory з функціями, що імпортуються, з кожного бібліотечного файлу привести в звіті по лабораторній роботі.

11. Знайти в тексті файлу по зсуву, узятому з побудованої таблиці, секцію з даними і переконатися, що текст оригінала пароля, що міститься в тексті програми, може бути легко виявлений за допомогою HEX-редактора. Привести скріншот цього фрагмента програми у вигляді HEX - коду в звіті по лабораторній роботі.

12. Виконати шифрування пароля за допомогою функції XOR, знову скомпілювати EXE -файл і переконатися, що тепер вони не виявляються явним чином в тексті виконуваного EXE - файлу. Привести скріншоти цієї програми в режимах «Hex» і «Text» в звіті по лабораторній роботі.

13. Зробити висновки по лабораторній роботі.

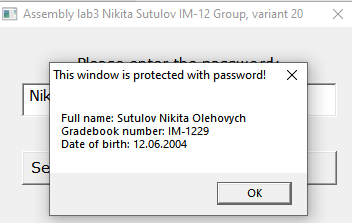
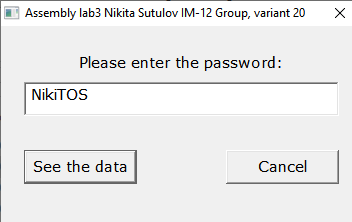
**Виконання роботи**

4. **Перевірити роботу програми шляхом введення як правильного, так і невірного паролів.**

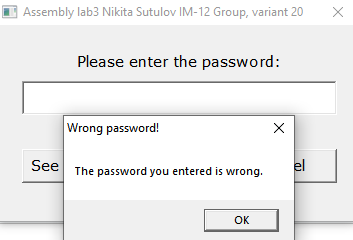
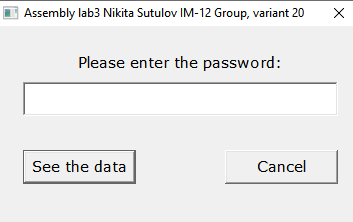
Правильний пароль – NikiTOS

* програма без шифрування:

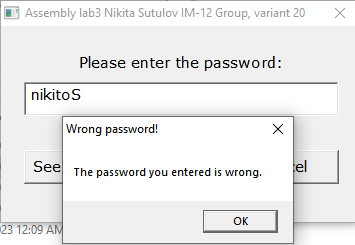
1. введення правильного пароля:



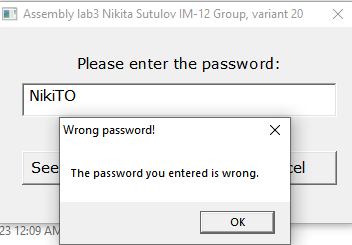
1. введення порожнього пароля:



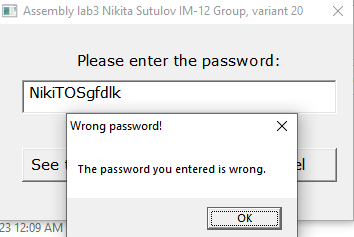
1. введення неправильного пароля (символи в іншому регістрі):



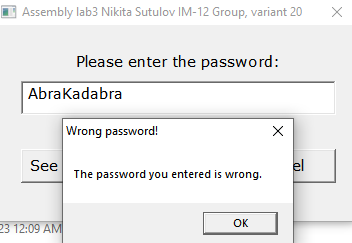
1. введення неправильного пароля (недостатньо літер):



1. введення неправильного пароля (з зайвими літерами):

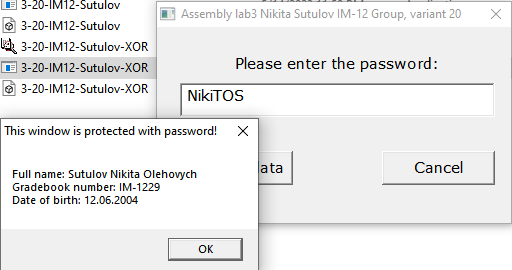


1. введення неправильного пароля:

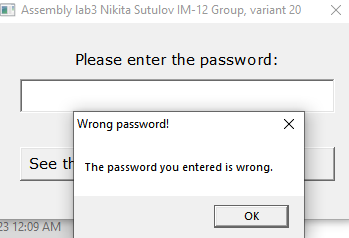


* програма з шифруванням:

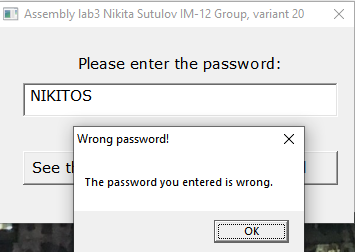
1. введення правильного пароля:



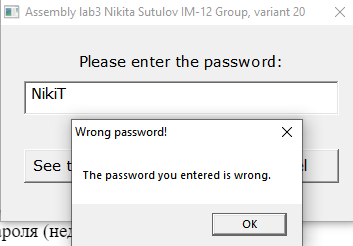
1. введення порожнього пароля:



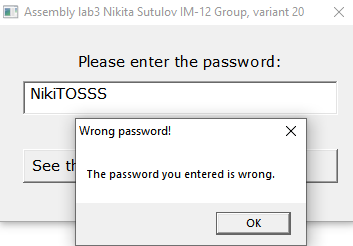
1. введення неправильного пароля (символи в іншому регістрі):



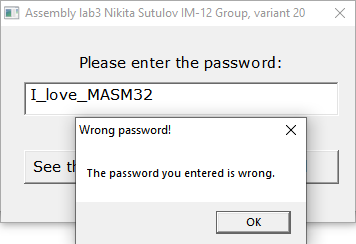
1. введення неправильного пароля (недостатньо літер):



1. введення неправильного пароля (з зайвими літерами):



1. введення неправильного пароля:



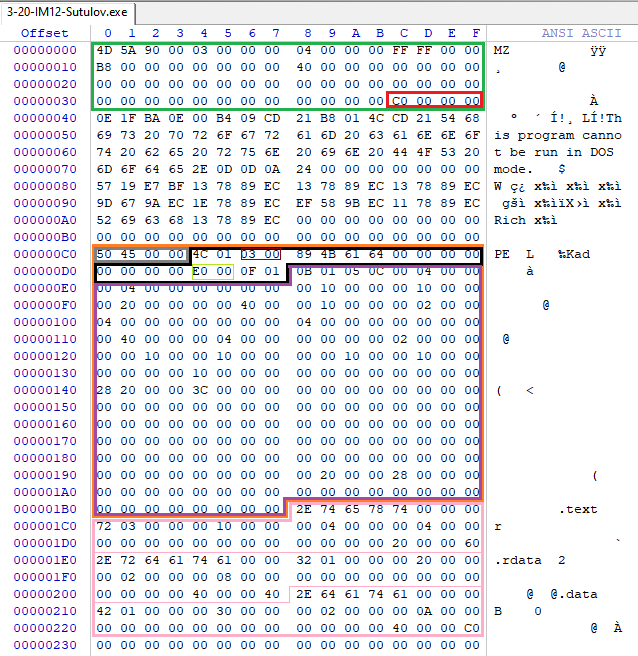
5-6. **Дослідження файлу в WinHex.**

На скріншоті обведено області MS-DOS заголовка (DOS\_HEADER, зеленим кольором), PE заголовка (PE\_HEADER, оранжевим кольором) і таблиці секцій (SECTION\_HEADERS, рожевим кольором).

DOS\_HEADER розпочинається на початку файла, його розмір сягає 64 байт. В його останніх 4 байтах – вказівник на початок PE\_HEADER (обведено червоним кольором). Враховуючи, що байти записано в зворотному порядку, зсув для початку PE\_HEADER дорівнює 00 00 00 C0, тобто 192 байти.

PE\_HEADER розпочинається з PE Signature (обведено сірим кольором), розміром у 4 байти, в якому записано байтову сигнатуру “PE\0\0”. Після PE Signature іде заголовок PE-файлу (обведено чорним кольором), який має розмір у 20 байтів. Після нього йде Optional Header. Його розмір визначено в заголовку PE-файлу (обведено салатовим кольором), він займає 2 байти, і в нашому випадку, звертаючи увагу на те, що байти записано в зворотньому порядку, його значення сягає 00 E0h, тобто 224 байти.

Одразу після Optional Header іде Section Headers (обведено рожевим кольором). Кількість секцій у ньому записано в Optional Header (ці дані займають 2 байти, на скріншоті обведено коричневим кольором), звертаючи увагу на те, що байти записано в зворотньому порядку, це значення дорівнює 00 03, тобто всього три секції (.text, .rdata, .data), розмір кожної з яких сталий і сягає 40 байтів.



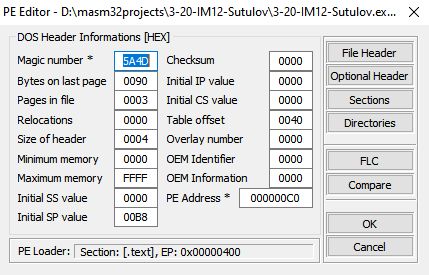
7-8. **Таблиці з параметрами файла.**

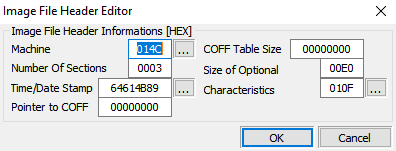
| Розділ 3.3.1: Machine Types | | |
| --- | --- | --- |
| Поле | Значення | Опис |
| Machine type | 01 4С | Тип машини: *Intel 386* або пізніше або інші схожі процесори |

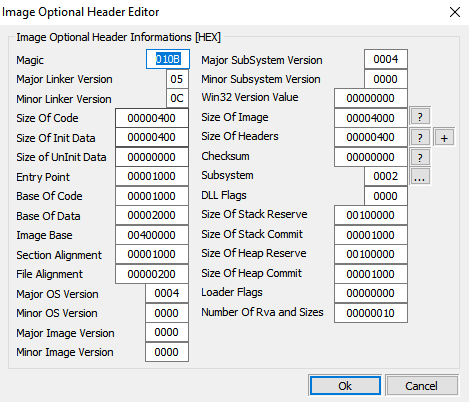
| Розділ 3.4.1: Optional Header Standard Fields (Image Only) | | |
| --- | --- | --- |
| Поле | Значення | Опис |
| Magic | 01 0В | Беззнакове ціле число (unsigned int), що ідентифікує стан файлу.  *0x10B* – ідентифікатор звичайного виконуваного файлу (.exe). |
| MajorLinkerVersion | 05 | Номер основної версії компонувальника |
| MinorLinkerVersion | 0С | Номер другорядної версії компонувальника.  0x0С = 1210 |
| SizeOfCode | 00 00 04 00 | Розмір секції коду або сума всіх розділів коду, якщо їх є декілька |
| SizeOfInitializedData | 00 00 04 00 | Розмір ініціалізованого розділу даних або  сума всіх таких розділів, якщо їх декілька  розділи даних. |
| SizeOfUninitializedData | 00 00 00 00 | Розмір розділу неініціалізованих даних (BSS), або сума всіх таких розділів, якщо є  декілька розділів BSS |
| AddressOfEntryPoint | 00 00 10 00 | Адреса вхідної точки |
| BaseOfCode | 00 00 10 00 | Адреса початку розділу коду |
| BaseOfData | 00 00 20 00 | Адреса початку розділу змінних |

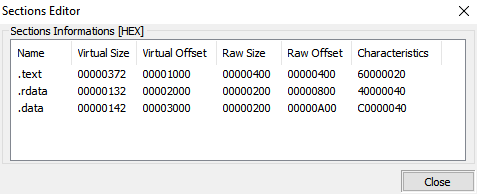
| Розділ 4: Section Table (Section Headers)  Як приклад, по порядку розібрано третю секцію .data | | |
| --- | --- | --- |
| Поле | Значення | Опис |
| Name | .data\0\0\0 | 8-байтовий рядок ASCII із доповненням нульовими значеннями |
| VirtualSize | 00 00 01 42 | Загальний розмір розділу при завантаженні в пам'ять |
| VirtualAddress | 00 00 30 00 | Адреса першого байту розділу |
| SizeOfRawData | 00 00 02 00 | Розмір розділу |
| PointerToRawData | 00 00 0A 00 | Вказівник на першу сторінку розділу в COFF файл |
| PointerToRelocations | 00 00 00 00 | Вказівник файлу на початок записів переміщення для розділу (0 для .exe) |
| PointerToLinenumbers | 00 00 00 00 | Вказівник файлу на початок записів номерів рядків для розділу |
| NumberOfRelocations | 00 00 | Кількість релокаційних входжень в розділ (0 для .ехе) |
| NumberOfLinenumbers | 00 00 | Кількість номерних записів розділу |
| Characteristics | C0 00 00 40 | Прапори, що описують характеристики розділу |

9. **Провести дослідження того ж файлу за допомогою меню "PE Editor"**, як можна переконатися, дані з таблиць та дослідження з PE Tools співпадають:

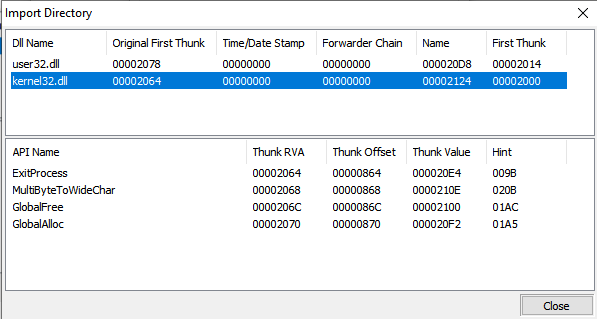
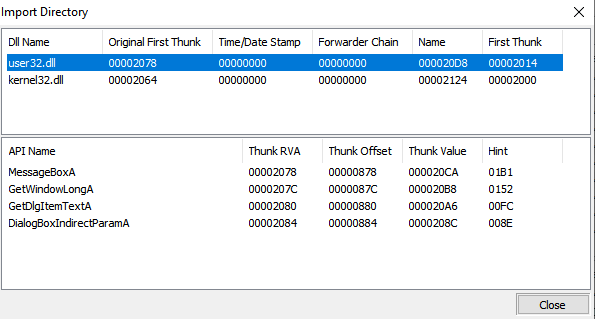






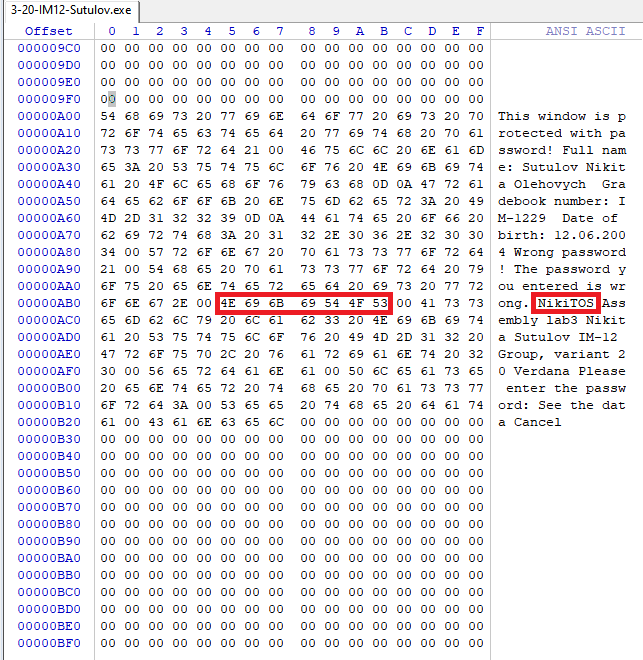


10. **Дослідити таблицю імпорту (Import Directory) даного файлу і визначити, які саме функції використовуються з бібліотек, що підключаються.**



На скріншотах видно, що імпортуються файли бібліотек user32.dll та kernel32.dll. Також видно й функції (по 4 штуки) з цих бібліотек, які використовуються.

11. **Знайти в тексті файлу по зсуву, узятому з побудованої таблиці, секцію з даними і переконатися, що текст оригінала пароля, що міститься в тексті програми, може бути легко виявлений за допомогою HEX-редактора.**



Як видно зі скріншота, за допомогою HEX-редактора у файлі програми можна досить легко знайти оригінал пароля, що є не дуже добре в досвіді реальної розробки.

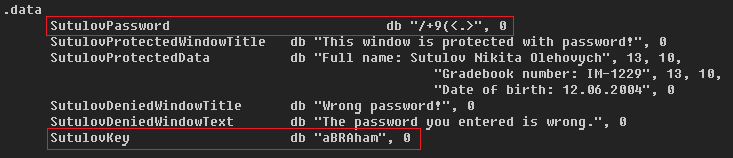
12. **Виконати шифрування пароля за допомогою функції XOR, знову скомпілювати EXE -файл і переконатися, що тепер вони не виявляються явним чином в тексті виконуваного EXE - файлу.**

Якщо зашифрувати наш пароль “NikiTOS” ключем “aBRAham”, отримаємо наступне:

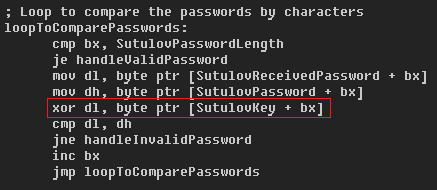
| ASCII | Binary | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NikiTOS | 01001110 | 01101001 | 01101011 | 01101001 | 01010100 | 01001111 | 01010011 |
| aBRAham | 01100001 | 01000010 | 01010010 | 01000001 | 01101000 | 01100001 | 01101101 |
| XOR | | | | | | | |
| /+9(<.> | 00101111 | 00101011 | 00111001 | 00101000 | 00111100 | 00101110 | 00111110 |

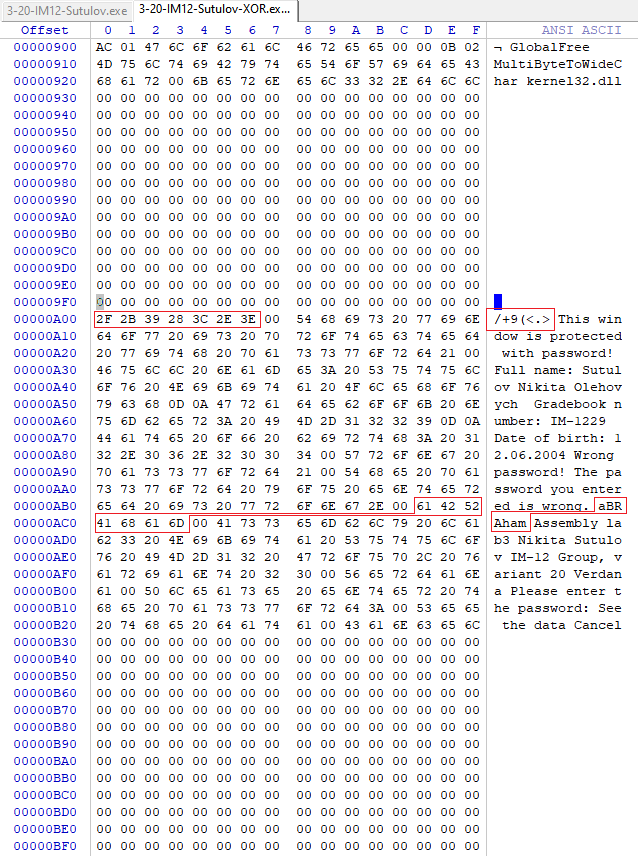
Таким чином, наш пароль у зашифрованому вигляді – “/+9(<.>”.

На скріншоті показано фрагмент коду, де я визначаю зашифрований пароль та ключ для шифрування вводу користувача для порівняння.



Далі у коді програми я додав рядок, який робить операцію xor з кожним байтом введеного користувачем пароля:



Тепер якщо подивитися вміст файлу програми з шифруванням, побачимо, що в ній немає явно записаного пароля, а є лише зашифрований пароль та ключ, завдяки чому визначити справжній пароль буде набагато складніше:  


13. **Висновки**.

Під час виконання даної лабораторної роботи я отримав нові знання щодо часткового аналізу exe-файлів, зокрема:

* Встановлення положення (початок файла) та розміру (64 байти) DOS-заголовка.
* Виявлення початку PE-заголовка (зсув для нього вказано в 4 останніх байтах DOS-заголовка), його компонентів та їх розмірів, якщо вони фіксовані, або визначення місцезнаходження цих розмірів, якщо вони змінні. Також вивчення розміру (40 байт \* кількість секцій) та розташування (одразу після PE-заголовка) Section Headers.
* Встановлення початку секцій у пам'яті для використання в певних програмах (у моєму випадку WinHex та PE Tools), наприклад, для пошуку місцезнаходження пароля у програмі, яка вимагає введення пароля для отримання даних. У результаті порівняння даних, знайдених мною власноруч у файлі програми за допомогою WinHex з даними, які я отримав з PE Tools, я роблю висновок, що я правильно визначив місцезнаходження та значення полів заголовків файла, оскільки досліджені власноруч дані співпадають із даними, отриманими з PE Tools.
* Крім того, ознайомлення з методом XOR-шифрування даних. Незважаючи на те, що цей метод не є надійним, оскільки зашифрований пароль та ключ можуть бути знайдені у файлі, і пароль можна розшифрувати (хоча це займе певний час) і пароль не буде прямо вказаний. Тож у результаті порівняння двох програм (без застосування шифрування та з ним) я впевнився, що застосування шифрування є більш безпечною практикою, оскільки без нього пароль просто можна знайти у виконуваному файлі програми, а з шифруванням справжній пароль дізнатися буде складніше, оскільки явно у файлі зберігатися буде лише зашифрований пароль. Однак, крім зашифрованого пароля також у файлі програми можна знайти й ключ для розшифрування (оскільки шифрування методом XOR є операцією, яка працює як у бік шифрування, так і розшифрування), тому в реальній практиці цей метод не набагато надійніший за відсутність шифрування. Натомість для шифрування на практиці існують більш надійні методи, наприклад, із використанням закритого ключа.