

**Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО  
Факультет безопасности информационных технологий**

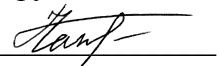
**Дисциплина:**

«Технологии и методы программирования»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

**Выполнил:**

Чапасов Пётр Константинович  
студент группы N33491

\_\_\_\_\_  
  
(подпись)

**Проверил:**

Ищенко Алексей Петрович

\_\_\_\_\_  
(подпись)

Санкт-Петербург

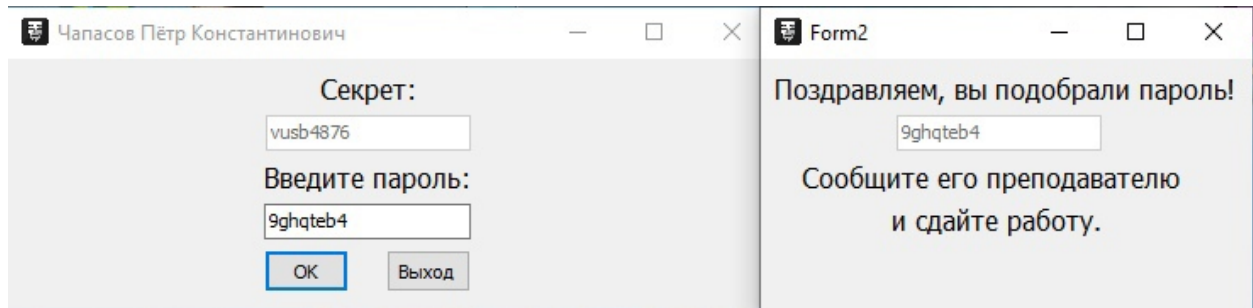
2022 г

### Задание:

Провести реверс инжиниринг над программой, выданной по варианту, и найти пароль.

### Результат выполнения:

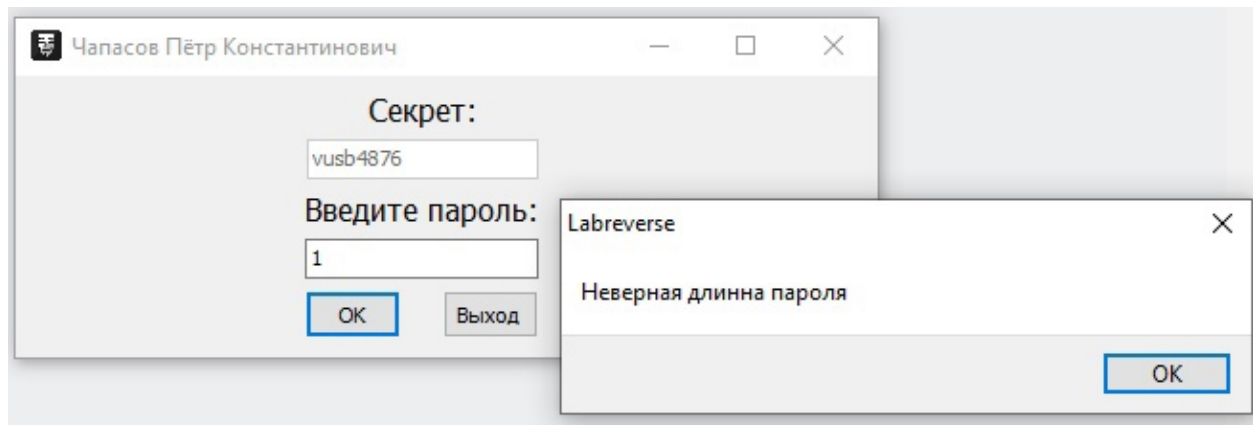
Пароль: 9ghqteb4



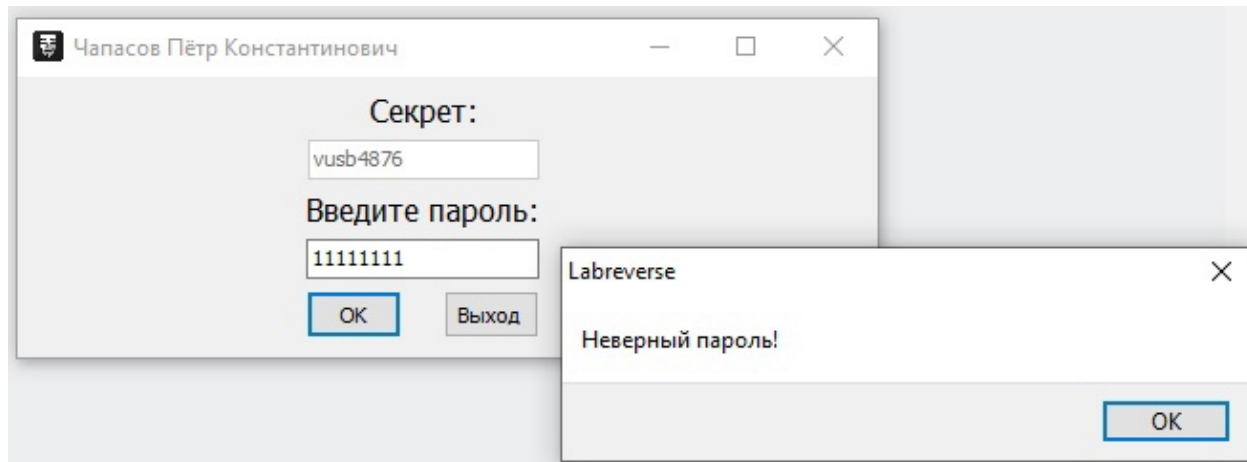
### Ход работы:

Запускаем программу и смотрим функционал программы.

Исходя из сообщения подбираем нужную длину пароля. Она равна 8.

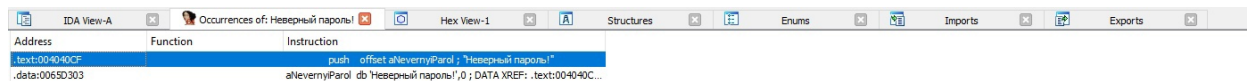


При неверном вводе пароля получаем надпись "Неверный пароль!"

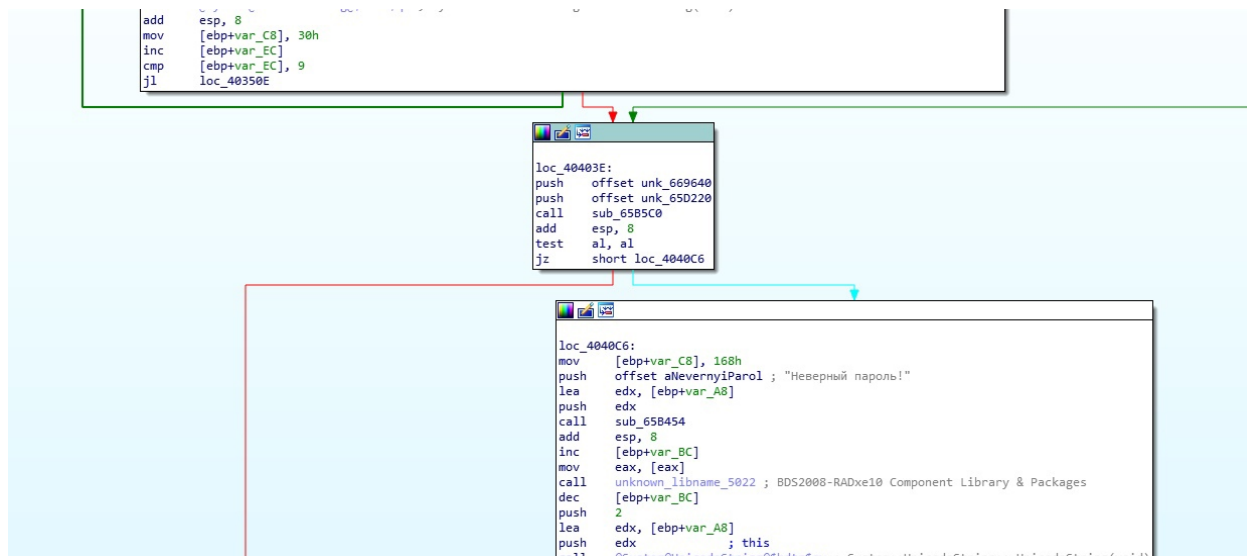


Теперь используя Ida Pro дизассемблируем наш exe-файл и найдём эту надпись в коде программы.

Результат поиска выдаёт два места в коде. Первый раз текст встречается в какой-то функции, второй - её инициализация. Переходим к функции.



Здесь мы видим две переменные unk\_669640 и unk\_65D220 и операцию над ними, предположительно сравнения, в результате которой выдается или надпись “Неверный пароль!”, или Form2.



Переведем ассемблерный код в более понятный вид, нажав F5. Теперь мы точно убедились, что наше предположение верно, осталось понять, какая переменная - наш пароль, а какая отвечает за верный.

```

if ( (unsigned __int8)sub_65B5C0(&unk_65D220, &unk_669640) )
{
    unknown_libname_482(&v67);
    Vcl::Controls::TControl::GetText(*(Vcl::Controls::TControl **)(v59 + 964));
    sub_52E684(*(Vcl::Controls::TControl **)(*(DWORD *)off_668E58 + 964));
    System::UnicodeString::~UnicodeString((System::UnicodeString *)&v67);
    Vcl::Forms::TCustomForm::Show(*(Vcl::Forms::TCustomForm **)off_668E58);
}
else
{
    v43 = (DWORD *)sub_65B454(&v66, laNevernyiParol);
    unknown_libname_5022(*v43);
    System::UnicodeString::~UnicodeString((System::UnicodeString *)&v66);
    if ( ++dword_65D228 > 2 )
    {
        dword_65D228 = 0;
    }
}

```

Для этого мы запускаем дебаггер и смотрим что хранится в обеих переменных (Я ввел пароль 11111111). Видим что и там и там хранится указатель, на строку. В unk\_669640 - 259EEE4, в unk\_65D220 - 259F574.

```

.data:00669640 unk_669640 db 0E4h ; d ; DATA XREF: sub_40304C+14A↑to
.data:00669640 ; sub_40330C:loc_40403E↑to
.data:00669641 db 0EEh ; o
.data:00669642 db 59h ; Y
.data:00669643 db 2
.data:0065D220 unk_65D220 db 74h ; t ; DATA XREF: sub_402FA4+3E↑to
.data:0065D220 ; sub_403000+15↑to ...
.data:0065D221 db 0F5h ; x
.data:0065D222 db 59h ; Y
.data:0065D223 db 2

```

Переходим по этим адресам и смотрим что там хранится. В unk\_669640 - [0x4F,0x72,0x7B,0x73,0x40,0x3D,0x35,0x2], в unk\_65D220 - [0x47,0x44,0x42,0x33,0x5,0x9,0x6,0x7].

```

debug047:025CEEE4 db 4Fh ; 0
debug047:025CEEE5 db 0
debug047:025CEEE6 db 72h ; r
debug047:025CEEE7 db 0
debug047:025CEEE8 db 7Bh ; {
debug047:025CEEE9 db 0
debug047:025CEEEA db 73h ; s
debug047:025CEEEB db 0
debug047:025CEEEC db 40h ; @
debug047:025CEEED db 0
debug047:025CEEEE db 3Dh ; =
debug047:025CEEEF db 0
debug047:025CEEF0 db 35h ; 5
debug047:025CEEF1 db 0
debug047:025CEEF2 db 2

```

```

debug046:0259F574 db 47h ; G
debug046:0259F575 db 0
debug046:0259F576 db 44h ; D
debug046:0259F577 db 0
debug046:0259F578 db 42h ; B
debug046:0259F579 db 0
debug046:0259F57A db 33h ; 3
debug046:0259F57B db 0
debug046:0259F57C db 5
debug046:0259F57D db 0
debug046:0259F57E db 9
debug046:0259F57F db 0
debug046:0259F580 db 6
debug046:0259F581 db 0
debug046:0259F582 db 7

```

Запустим программу ещё раз с другим паролем (Я ввел 00000000), чтобы проверить, какая строка изменится. Видим, что изменился unk\_65D220. Из этого следует, что unk\_65D220 - наш пароль, но над ним проведены какие-то операции, так как каждый символ отличается. Unk\_669640 - это зашифрованный пароль, так как он не изменился.

```

debug042:0272F59C db 46h ; F
debug042:0272F59D db 0
debug042:0272F59E db 45h ; E
debug042:0272F59F db 0
debug042:0272F5A0 db 43h ; C
debug042:0272F5A1 db 0
debug042:0272F5A2 db 32h ; 2
debug042:0272F5A3 db 0
debug042:0272F5A4 db 4
debug042:0272F5A5 db 0
debug042:0272F5A6 db 8
debug042:0272F5A7 db 0
debug042:0272F5A8 db 7
debug042:0272F5A9 db 0
debug042:0272F5AA db 6

```

Теперь разберемся, какие операции производятся над введенным паролем. Видим что посимвольно из пароля вычитают 48 и приводят к бинарному виду алгоритмом по определению.

```

v57 = v4 - 48;
if ( v57 )
{
    for ( ; v57 != 1; v57 /= 2 )
    {
        sub_404330(&v95, v57 % 2);
        v5 = unknown_libname_482(&v94);
        System::UnicodeString::operator+(v5, &v102, &v95);
        sub_65B524(&v102, &v94);
        System::UnicodeString::~UnicodeString((System::UnicodeString *)&v94);
        System::UnicodeString::~UnicodeString((System::UnicodeString *)&v95);
    }
}

```

По сути здесь происходит простое вычитание 48 и bin(x)

Аналогичный набор действий производится с другой такой же строкой. Следовательно это может быть или зашифрованный пароль или секрет, данный нам в задании.

---

```

v53 = v8 - 48;
if ( v53 )
{
    for ( ; v53 != 1; v53 /= 2 )
    {
        sub_404330(&v87, v53 % 2);
        v9 = unknown_libname_482(&v86);
        System::UnicodeString::operator+(v9, &v104, &v87);
        sub_65B524(&v104, &v86);
        System::UnicodeString::~UnicodeString((System::UnicodeString *)&v86);
        System::UnicodeString::~UnicodeString((System::UnicodeString *)&v87);
    }
}

```

Дальше по коду видим приведение к типу int и хог между собой. Делаем вывод, что второй строкой был секрет, так как мы впоследствии сравниваем полученные значения с зашифрованным паролем.



```

v18 = (__int16 *)sub_404308((System::UnicodeString *)&v100, 1);
v19 = (_DWORD *)sub_404370(&v77, *v18);
v20 = sub_4244FC(*v19) << 7;
v21 = (__int16 *)sub_404308((System::UnicodeString *)&v100, 2);
v22 = (_DWORD *)sub_404370(&v76, *v21);
v23 = (sub_4244FC(*v22) << 6) + v20;
v24 = (__int16 *)sub_404308((System::UnicodeString *)&v100, 3);
v25 = (_DWORD *)sub_404370(&v75, *v24);
v26 = 32 * sub_4244FC(*v25) + v23;
v27 = (__int16 *)sub_404308((System::UnicodeString *)&v100, 4);
v28 = (_DWORD *)sub_404370(&v74, *v27);
v29 = 16 * sub_4244FC(*v28) + v26;
v30 = (__int16 *)sub_404308((System::UnicodeString *)&v100, 5);
v31 = (_DWORD *)sub_404370(&v73, *v30);
v32 = 8 * sub_4244FC(*v31) + v29;
v33 = (__int16 *)sub_404308((System::UnicodeString *)&v100, 6);
v34 = (_DWORD *)sub_404370(&v72, *v33);
v35 = 4 * sub_4244FC(*v34) + v32;
v36 = (__int16 *)sub_404308((System::UnicodeString *)&v100, 7);
v37 = (_DWORD *)sub_404370(&v71, *v36);
v38 = 2 * sub_4244FC(*v37) + v35;
v39 = (__int16 *)sub_404308((System::UnicodeString *)&v100, 8);
v40 = (_DWORD *)sub_404370(&v70, *v39);
v41 = sub_4244FC(*v40) + v38;
System::UnicodeString::~UnicodeString((System::UnicodeString *)&v70);
System::UnicodeString::~UnicodeString((System::UnicodeString *)&v71);
System::UnicodeString::~UnicodeString((System::UnicodeString *)&v72);
System::UnicodeString::~UnicodeString((System::UnicodeString *)&v73);
System::UnicodeString::~UnicodeString((System::UnicodeString *)&v74);
System::UnicodeString::~UnicodeString((System::UnicodeString *)&v75);
System::UnicodeString::~UnicodeString((System::UnicodeString *)&v76);
System::UnicodeString::~UnicodeString((System::UnicodeString *)&v77);
sub_4043BC(&v69, v41);

```

Теперь мы имеем представление об алгоритме и можем сделать вывод, что над каждой буквой производится следующие операции:

$$(\text{password}[i] - 48) \wedge (\text{secret}[i] - 48) == \text{encrypted\_password}[i]$$

Зная секрет и зашифрованный пароль, можем попытаться восстановить верный пароль. Я использовал скрипт на Python3.10:

```

encrypted_password = [0x4F, 0x72, 0x7B, 0x73, 0x40, 0x3D, 0x35, 0x2]
secret = [ord(x) for x in list("vusb4876")]
for i in range(8):
    print(chr(48 + int(encrypted_password[i] ^ (secret[i] - 48))), end = "")

```

В результате мы получаем следующий пароль: **9ghqteb4**. Пробуем его. Он оказывается верным.

## Вывод:

Выполнив эту лабораторную работу, я научился основам реверс инжиниринга. В работе я использовал интерактивный дизассемблер Ida Pro и встроенный Windows дебаггер. Я более подробно разобрался в ассемблере и научился, хоть и с помощью визуализации Ida, но понимать логику программы в ассемблерном виде.