Машинно-зависимые языки программирования

Лабораторная работа №11

"Низкоуровневое программирование под Windows/Linux. Дизассемблирование. Реверс-инжиниринг"

Справочная информация

Реверс-инжиниринг (обратная разработка) — исследование готовой программы с целью понять принцип работы, поиска недокументированных возможностей или внесения изменений.

Правообладатели могут запретить проведение обратной разработки и использование её результатов, чтобы исключить нарушения их исключительных прав по закону об авторском праве и патентному законодательству.

Средства дизассемблирования и отладки под Windows

Устаревшие, но иногда используемые дизассемблеры и отладчики: W32Dasm, SoftICE, OllyDbg и форки.

Актуальные на сегодняшний день средства: x64dbg, WinDbg (Microsoft), IDA Pro. Далее примеры будут основаны на x64dbg (https://x64dbg.com/) (программа будет скомпилирована под x86-64) и на IDA Freeware (программа будет скомпилирована под x86) (https://www.hex-ravs.com/products/ida/support/download_freeware/).

Пример исследуемой программы

```
#include <iostream>
int main()
{
    char a[100];
    std::cout << "Enter password: ";
    std::cin >> a;
    if (strcmp(a, "1234") == 0) {
        std::cout << "access granted";
    }
    else {
        std::cout << "access denied";
    }
}</pre>
```

В рамках исследования требуется определить строку, которую требуется ввести в программе, чтобы она показала сообщение "access granted". Будем считать, что исходный код программы исследователю неизвестен.

Исследование полученной программы в x64dbg

Для этого исследования программа была скомпилирована с помощью MinGW с дополнительным заголовком #include <cstring>.

После загрузки программы в отладчике требуется выполнить программу по шагам с остановками на автоматически созданных точках останова вплоть до запуска ввода с клавиатуры с помощью пункта Выполнить (F9):

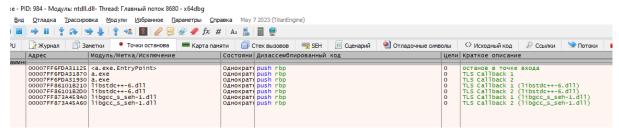


Рис. 1. Автоматически добавленные точки останова

Далее, чтобы программа не завершилась, можно поставить точку останова при обращении к нужной ячейке памяти. Для этого можно обратиться к разделу "Карта памяти" и перейти к области "Инициализированные данные только для чтения":

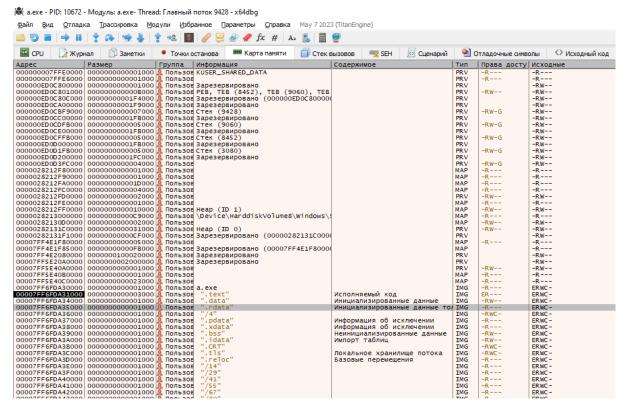


Рис. 2. Карта памяти в x64dbg

Далее имеется возможность установить точку останова на ячейке памяти при условии обращения к ней:

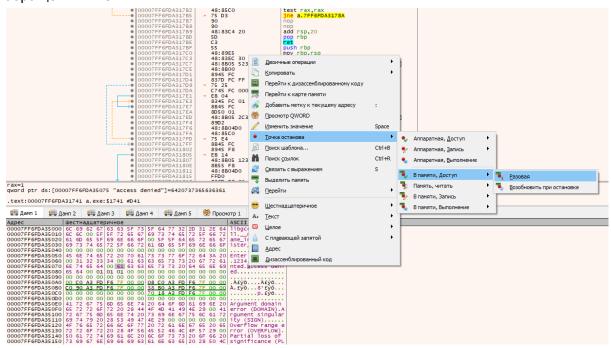


Рис. 3. Установка точки останова на доступ к переменной



Рис. 4. Сравнение введённой строки zzz и строки 1234

Далее с помощью действия Шаг с обходом (F8) убеждаемся, что именно это сравнение влияет на вывод одного сообщения из двух, а значит, 1234 является паролем:

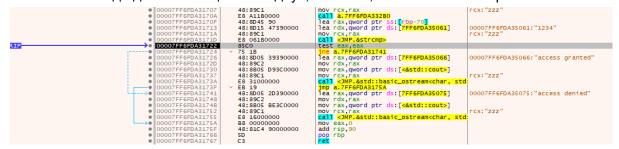


Рис. 5. Вывод одного или другого сообщения на экран

Исследование полученной программы в IDA Freeware

Для этого исследования программа была скомпилирована с помощью Visual Studio 2019 в конфигурации Release (для сокращения размера исполняемого файла) под x86.

```
; Attributes: bp-based frame fuzzy-sp
           ; int __cdecl main(int argc, const char **argv, const char **envp) main proc near
           var_70= byte ptr -70h
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
            push
                          ebp esp esp, 9FFFFFF8h esp, 70h eax, _security_cookie eax, esp [esp+70h+var_4], eax ex, ds: _imp_?cout@std@@3V?$basic_ostream@0U?$char_traits@0@std@@@1@A ; std::basic_ostream<char_std::char_traits<char>> std::cout edx, offset aEnterPassword; "Enter password: " std_operator__std_char_traits_char__ex, ds:_imp_?cin@std@@3V?$basic_istream@0U?$char_traits@0@std@@@1@A ; std::basic_istream<char,std::char_traits<char>> std::cin eax, [esp+70h+var_70] eax
           mov
and
sub
mov
xor
mov
mov
call
           mov
lea
           push
call
add
lea
                           eax
std__Istream_extract_into_buffer_char_std__char_traits_char__
                          std__Istream_extract_into
esp, 4
ecx, [esp+70h+var_70]
eax, offset al234; "1234"
ax, ax
                                                                                                                             loc_401190:
mov dl, [ecx]
cmp dl, [eax]
jnz short loc_401180
                                                                                                               <u></u>
                                                                                                               test
jz
                                                                                                                              dl, dl
short loc_4011A0
                                                                                                                                   <u>...</u> 🚄 📜
                                                                                                                                                   dl, [ecx+1]
dl, [eax+1]
short loc_401180
                                                                                                                                                              Т
                                                                                                                            add
add
test
jnz
                                                                                                                                            eax, 2
dl, dl
short loc_401190
                                                                                            <u></u>
                                                                                             loc 4011AC:
                                                                                                                                                  loc 401180:
                                                                                                            eax, eax
short loc_401185
                                                                                                                                                   sbb
                                                                                                                                                                 eax, eax
eax, 1
<u></u>
loc_4011B5:
               ecx, ds:_imp_?cout@std@@3V?$basic_ostream@OU?$char_traits@D@std@@@1@A ; std::basic_ostream<char,std::char_traits<char>> std::cout edx, offset aAccessGranted ; "access granted" eax, eax short loc_4011C9
test
                                                                                    <mark>∭ ⊿</mark> 
mov
                                                                                                    edx, offset aAccessDenied ;
                                                                                                                                     √ √
                                                                                          II 🚄
                                                                                                         std_operator__std_char_traits_char
ecx, [esp+70h+var_4]
eax, eax
ecx, eax
__security_check_cookie
                                                                                          call
                                                                                          xor
call
                                                                                                         __securit
esp, ebp
ebp
                                                                                          mov empop el retn main endp
```

Рис. 1 Начальный вид IDA Freeware после загрузки программы

После загрузки исполняемого модуля в IDA Freeware видим общий вид структуры программы на вкладке IDA View-A (рис. 1). Прямо в нём доступна возможность получения дополнительной информации об элементах программы, а также перехода к местам, где они объявлены.

Также можно перейти в режим просмотра дизассемблированного текста ехе-файла, например, с помощью двойного клика по имени метки a1234.

В общем-то, видя эту метку и строку db '1234', 0, на которую она указывает, можно проверить, является ли эта строка искомой, и завершить исследование, но попробуем изучить программу по порядку.

Изучая дизассемблированный код сверху вниз, можно обнаружить главную функцию:

```
.text:00401150 ; ======== S U B R O U T I N E
______
.text:00401150
.text:00401150 ; Attributes: bp-based frame fuzzy-sp
.text:00401150
.text:00401150 ; int cdecl main(int argc, const char **argv,
const char **envp)
.text:00401150 main
                              proc near
                                                  ; CODE XREF:
scrt common main seh+F5↓p
.text:00401150
.text:00401150 var 70
                             = byte ptr -70h
.text:00401150 var 4
                             = dword ptr -4
.text:00401150 argc
                             = dword ptr 8
.text:00401150 argv
                             = dword ptr 0Ch
.text:00401150 envp
                              = dword ptr 10h
.text:00401150
                              push ebp
.text:00401150
.text:00401151
                              mov ebp, esp
                              and esp, OFFFFFF8h
.text:00401153
.text:00401156
                              sub esp, 70h
.text:00401159
                              mov eax, __security_cookie
.text:0040115E
                              xor eax, esp
.text:00401160
                              mov [esp+70h+var 4], eax
.text:00401164
                              mov ecx,
ds: imp ?cout@std@@3V?$basic ostream@DU?$char traits@D@std@@@1@A
; std::basic_ostream<char,std::char_traits<char>> std::cout
.text:0040116A
                              mov edx, offset aEnterPassword;
"Enter password: "
.text:0040116F
                              call
std operator std char traits char
.text:00401174
                              mov
                                   ecx,
ds: imp ?cin@std@@3V?$basic istream@DU?$char traits@D@std@@@1@A ;
std::basic istream<char,std::char traits<char>> std::cin
.text:0040117A
                              lea eax, [esp+70h+var 70]
.text:0040117D
                              push eax
```

```
.text:0040117E
                               call
std__Istream_extract_into_buffer_char_std__char_traits_char___
.text:00401183
                               add esp, 4
                               lea ecx, [esp+70h+var 70]
.text:00401186
.text:00401189
                               mov eax, offset a1234; "1234"
.text:0040118E
                               xchg ax, ax
.text:00401190
.text:00401190 loc 401190:
                                                   ; CODE XREF:
main+5A↓j
.text:00401190
                         mov dl, [ecx]
                          cmp dl, [eax]
.text:00401192
                         jnz short loc 4011B0
.text:00401194
.text:00401196
                         test dl, dl
                              short loc 4011AC
.text:00401198
                         jz
.text:0040119A
                         mov dl, [ecx+1]
.text:0040119D
                         cmp dl, [eax+1]
                         jnz short loc 4011B0
.text:004011A0
.text:004011A2
                         add ecx, 2
.text:004011A5
                         add eax, 2
                         test dl, dl
.text:004011A8
.text:004011AA
                         jnz short loc 401190
.text:004011AC
.text:004011AC loc 4011AC:
                                                   ; CODE XREF:
main+48↑j
.text:004011AC
                               xor eax, eax
                               jmp short loc 4011B5
.text:004011AE
.text:004011B0 ;
______
_____
.text:004011B0
.text:004011B0 loc 4011B0:
                                                   ; CODE XREF:
main+44↑j
.text:004011B0
                                                   ; main+50↑j
.text:004011B0
                               sbb eax, eax
.text:004011B2
                               or eax, 1
.text:004011B5
.text:004011B5 loc 4011B5:
                                                   ; CODE XREF:
main+5E↑j
.text:004011B5
                              mov ecx,
ds: imp ?cout@std@@3V?$basic ostream@DU?$char traits@D@std@@@1@A
; std::basic_ostream<char,std::char_traits<char>> std::cout
.text:004011BB
                              mov edx, offset aAccessGranted;
"access granted"
.text:004011C0
                               test eax, eax
.text:004011C2
                               jz short loc 4011C9
```

```
.text:004011C4
                               mov edx, offset aAccessDenied;
"access denied"
.text:004011C9
.text:004011C9 loc 4011C9:
                                                     ; CODE XREF:
main+72↑j
.text:004011C9
                               call
std operator std char traits char
.text:004011CE
                               mov ecx, [esp+70h+var 4]
.text:004011D2
                               xor eax, eax
.text:004011D4
                               xor ecx, esp
.text:004011D6
                               call __security_check_cookie
.text:004011DB
                               mov esp, ebp
.text:004011DD
                               pop ebp
.text:004011DE
                               retn
.text:004011DE main
                               endp
```

Жирным шрифтом выделены ключевые места программы: вывод приглашения, ввод строки и цикл сравнения строк. Видим, что сравниваются строки по смещениям [ecx] и [eax], одна из них - полученная с ввода, другая обозначена меткой a1234. Остаётся её проверить и убедиться, что задача решена.

Практическое задание

С помощью x64dbg, IDA Freeware или других дизассемблеров/отладчиков определить пароль, необходимый для получения сообщения "congrats you cracked the password" в прикреплённой программе (https://crackmes.one/crackme/5fe8258333c5d4264e590114).