

Машинно-зависимые языки программирования

Лабораторная работа №9

“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 к 2020 году, но иногда используемые дизассемблеры и отладчики:

W32Dasm, SoftICE, OllyDbg и форки.

Развивающиеся средства: WinDbg (Microsoft), IDA Pro.

Далее пример будет основан на IDA Freeware

(https://www.hex-rays.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";
    }
}
```

В примере эта программа скомпилирована с помощью Visual Studio 2019 в конфигурации Release (для сокращения размера исполняемого файла) под x86.

Исследование полученной программы

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.

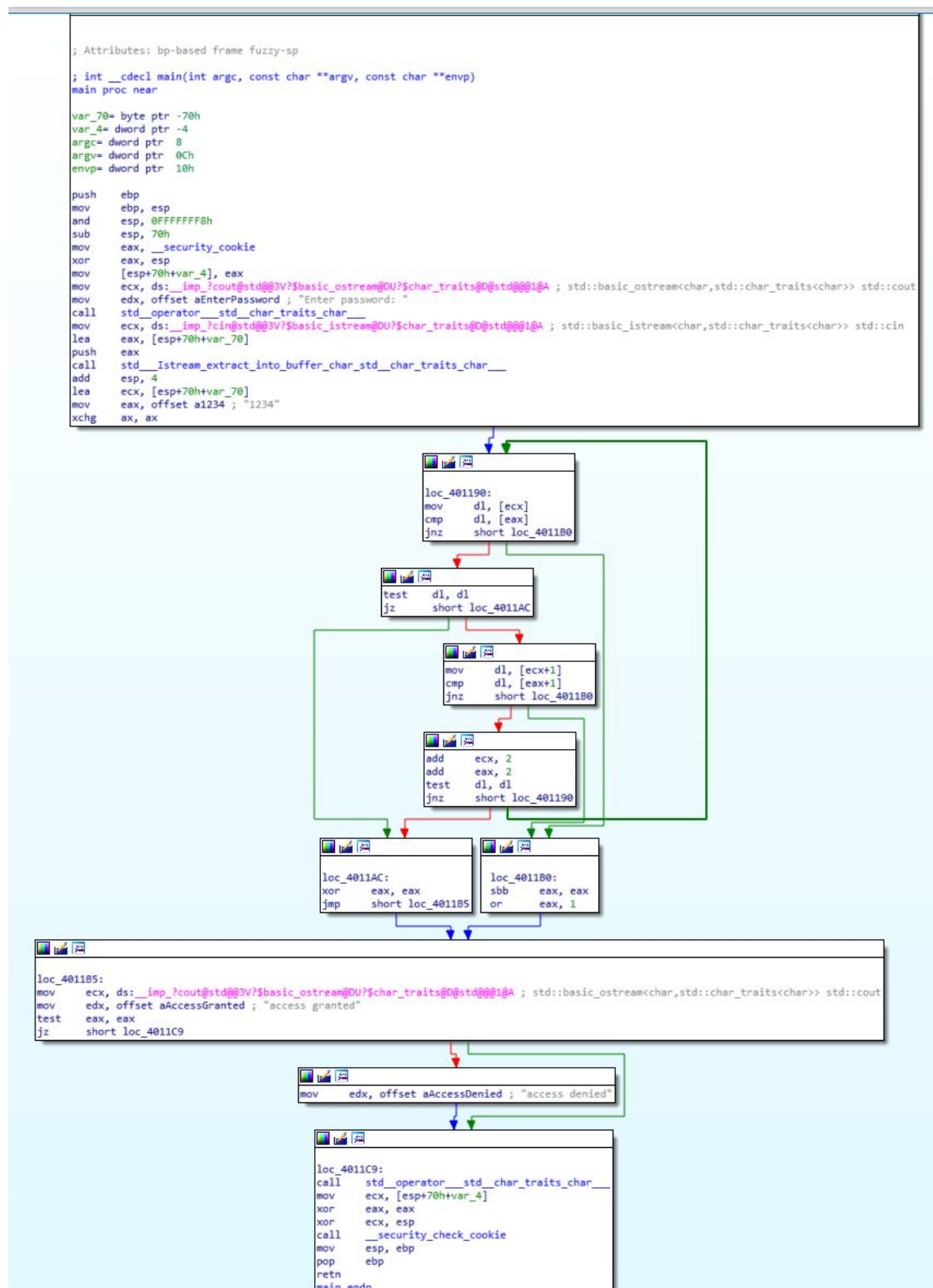


Рис. 1 Начальный вид IDA Freeware после загрузки программы

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
.text:00401150                push ebp
.text:00401151                mov  ebp, esp
.text:00401153                and  esp, 0FFFFFFF8h
.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
```

```

.text:00401186          lea  ecx, [esp+70h+var_70]
.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]
.text:00401192          cmp  dl, [eax]
.text:00401194          jnz  short loc_4011B0
.text:00401196          test dl, dl
.text:00401198          jz   short loc_4011AC
.text:0040119A          mov  dl, [ecx+1]
.text:0040119D          cmp  dl, [eax+1]
.text:004011A0          jnz  short loc_4011B0
.text:004011A2          add  ecx, 2
.text:004011A5          add  eax, 2
.text:004011A8          test dl, dl
.text:004011AA          jnz  short loc_401190
.text:004011AC
.text:004011AC loc_4011AC:                                ; CODE XREF:
main+48↑j
.text:004011AC          xor  eax, eax
.text:004011AE          jmp  short loc_4011B5
.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

Требуется провести дизассемблирование какого-либо исполняемого файла, например, самостоятельно созданного .exe, и изучить особенности его устройства/получить какую-либо информацию, доступную только при реверс-инжиниринге.