ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования
Национальный исследовательский университет
«Высшая школа экономики»

Московский институт электроники и математики

Департамент прикладной математики

Кафедра «Компьютерная безопасность»

Отчет
о домашней работе по предмету
«Защита программ и данных»

Выполнил:
Студент группы СКБ181
Винников Н.В.
Проверил:
Жуков А.К.

Задача

Реализовать программу на языке C\C++, которая должна включать следующую последовательность действий:

1. Запрос какой-либо информации от пользователя (логин, номер телефона, номер какого-либо документа, ключ любого вида и т.д.)
2. Произведение манипуляций по разработанному алгоритму с введенной пользователем информацией для генерации ответа на ключевой вопрос
3. Запрос ключа от пользователя и сравнение его со сгенерированным ключом на этапе 2.
4. Выдача результата по принципу:

сгенерированный и алгоритмом и введенный пользователем
ключи совпадают – TRUE->поздравление или доступ к какой
либо информации
Не совпадают – ошибка
В приложении к программе должен быть описан алгоритм работы.

Далее переписать программу с учетом следующих условий:

1. Внутри программы должен быть какой-либо алгоритм преобразования данных для генерации ключевой информации. Например, сложение по модулю кодов букв из таблицы ASCII, Подсчет количества букв в логине пользователя и умножение его на текущий год, запросы системного времени или к каким-либо заранее определённым файлам.
2. Должны быть реализованы меры защиты от отладки: специальные функции языка, искусственное усложнение кода, директивы препроцессора, условия сборки компилятора, упаковщики и т. д. (см. лекция)
3. В приложении должны быть описаны механизмы, применяющиеся для защиты от анализа

Далее провести сравнительный анализ двух реализаций в дизассемблерах и описать в чём наблюдается усложнение во второй программе с точки зрения анализа ПО.

Описать в отчете с привидением сравнительных скриншотов с объяснениями

Результатом выполнения задания должны быть две разработанные программы в виде исполняемых файлов и исходных кодов, make/cmake
файла сборки (или инструкции по сборке в случае применения дополнительных средств или специальных скриптов.). Документация к программе с описанием алгоритма работы, описанием входных и выходных данных и применяемых в программе мер защиты, а также отчёта о сравнительном анализе в дизассемблере двух реализаций.

Алгоритм работы программы до антиотладки

1. Программа запрашивает у пользователя Логин (Введите логин:).
2. Программа генерирует ключ на основе логина. Алгоритм генерации ключа следующий: проходим по каждому символу из переменной логина пользователя и получаем соответствующий ASCII-код символа, который преобразуется в строку с помощью функции to_string и добавляется в стркоу generated_key, которая и формирует ключ.
3. Далее программа запрашивает у пользователя Ключ (Введите ключ:) и сравнивает его с сгенерированным.

4. Выводится результат. Если сгенерированный и введенный ключи совпадают, то выводится сообщение TRUE - поздравляем!, иначе - FALSE - ошибка!.

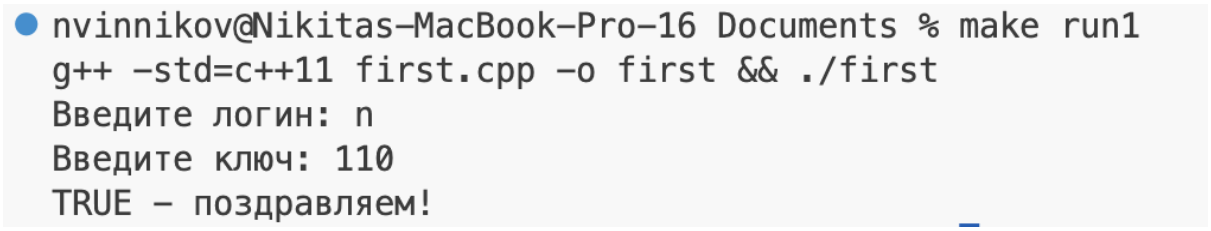
Исходный код программы:

Приложение 1.

Сборка и запуск программы:

```
g++ -std=c++11 first.cpp -o first && ./first

Введите логин: n
Введите ключ: 110
TRUE - поздравляем!
```



Алгоритм работы программы после антиотладки

1. Программа запрашивает у пользователя Логин (Введите логин:).
2. Преобразование Логина для генерации ключевой информации. Реверс запись Логина.
3. Программа генерирует ключ на основе логина. Алгоритм генерации ключа следующий: подсчет длины Логина и запись в login_size, далее проходим по каждому символу из переменной логина пользователя и получаем соответствующий ASCII-код символа, который умножаем на длину логина и преобразуем в строку с помощью функции to_string, далее добавляем в строку generated_key, которая и формирует ключ.
4. Далее программа запрашивает у пользователя Ключ (Введите ключ:) и сравнивает его с сгенерированным.
5. Выводится результат. Если сгенерированный и введенный ключи совпадают, то выводится сообщение TRUE - поздравляем!, иначе - FALSE - ошибка!.

Исходный код программы:

Приложение 2.

Сборка и запуск программы:

```
g++ -w -std=c++11 second.cpp -o second && strip second && upx -qqq second && ./second

Введите логин: n
Введите ключ: 110
TRUE - поздравляем!
```

```
nvinnikov@Nikitas-MacBook-Pro-16 Documents % make run2
g++ -w -std=c++11 second.cpp -o second && strip second && ./second
Введите логин: qwert
Введите ключ: 213213213
FALSE - ошибка!
```

Меры защиты от анализа

Реализованы следующие меры защиты:

1. Специальные функции языка - Ptrace(), IsDebuggerPresent(), DebugBreak().
2. Искусственное усложнение структуры программы - вызов функции через указатель на нее (косвенный вызов).
3. Защита от отладки с помощью директив препроцессора - _DEBUG.
4. Условия сборки - ключ оптимизации кода -O2.
5. Утилита strip, удаляет отладочную информацию.
6. Упаковщик upx.

Сравнительный анализ

Исследую программы через Ghidra.

до реализации методов защиты	**после реализации методов защиты**
Функции и переменные в них находятся в том виде, в котором они указаны в коде. (рис. 1)	Названия функции и переменных в них отображаются в зашифрованном виде. (рис. 2)
Можно увидеть вызов функции generate_key(), которая находится после ввода ключа. (рис. 3)	Названия функции и переменных в них отображаются в зашифрованном виде. (рис. 4)
В функции generate_key() видно, что на вход передается один параметр. (рис. 5) Аналитик, запустив программу в режиме дебага, поймет, что параметр и есть логин пользователя.	Названия функции и переменных в них отображаются в зашифрованном виде. (рис. 4)
Граф (рис. 6)	Граф усложнен. (рис. 7)
Поиск переменных, текста и функций. Дизассемблер может найти искомый объект. (рис. 8)	Дизассемблер не может найти искомый объект. (рис. 9) (рис. 10)

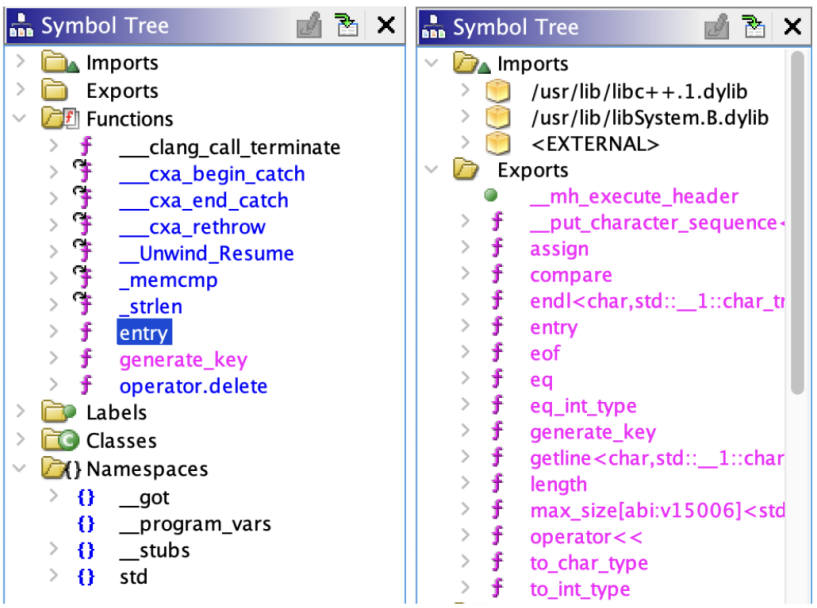

Symbol Tree
Imports
Exports
Functions
___clang_call_terminate
___cxa_begin_catch
___cxa_end_catch
___cxa_rethrow
__Unwind_Resume
_memcmp
_strlen
entry
generate_key
operator.delete
Labels
Classes
Namespaces
__got
__program_vars
__stubs
std
Symbol Tree
Imports
/usr/lib/libc++.1.dylib
/usr/lib/libSystem.B.dylib
<EXTERNAL>
Exports
__mh_execute_header
__put_character_sequence
assign
compare
endl<char,std::__1::char_tr
entry
eof
eq
eq_int_type
generate_key
getline<char,std::__1::char
length
max_size[abi:v15006]<std
operator<<
to_char_type
to_int_type

рис. 1

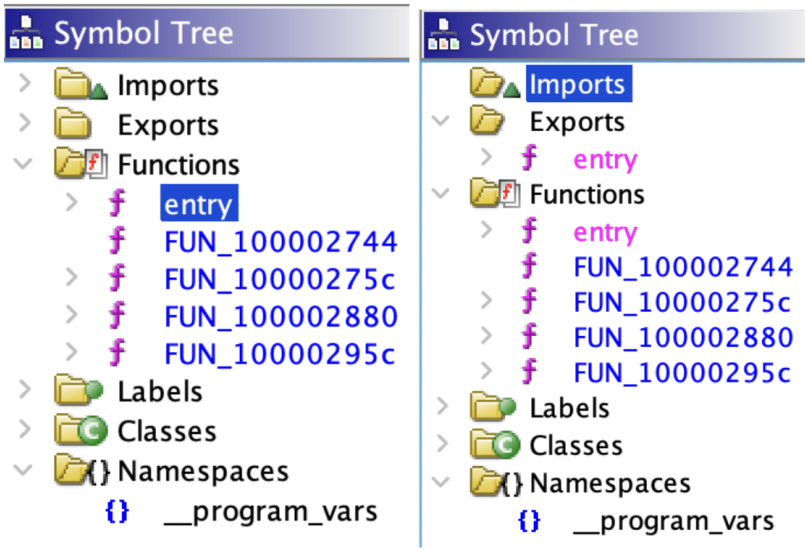

Symbol Tree
Imports
Exports
Functions
entry
FUN_100002744
FUN_10000275c
FUN_100002880
FUN_10000295c
Labels
Classes
Namespaces
__program_vars
Symbol Tree
Imports
Exports
entry
Functions
entry
FUN_100002744
FUN_10000275c
FUN_100002880
FUN_10000295c
Labels
Classes
Namespaces
__program_vars

рис. 2

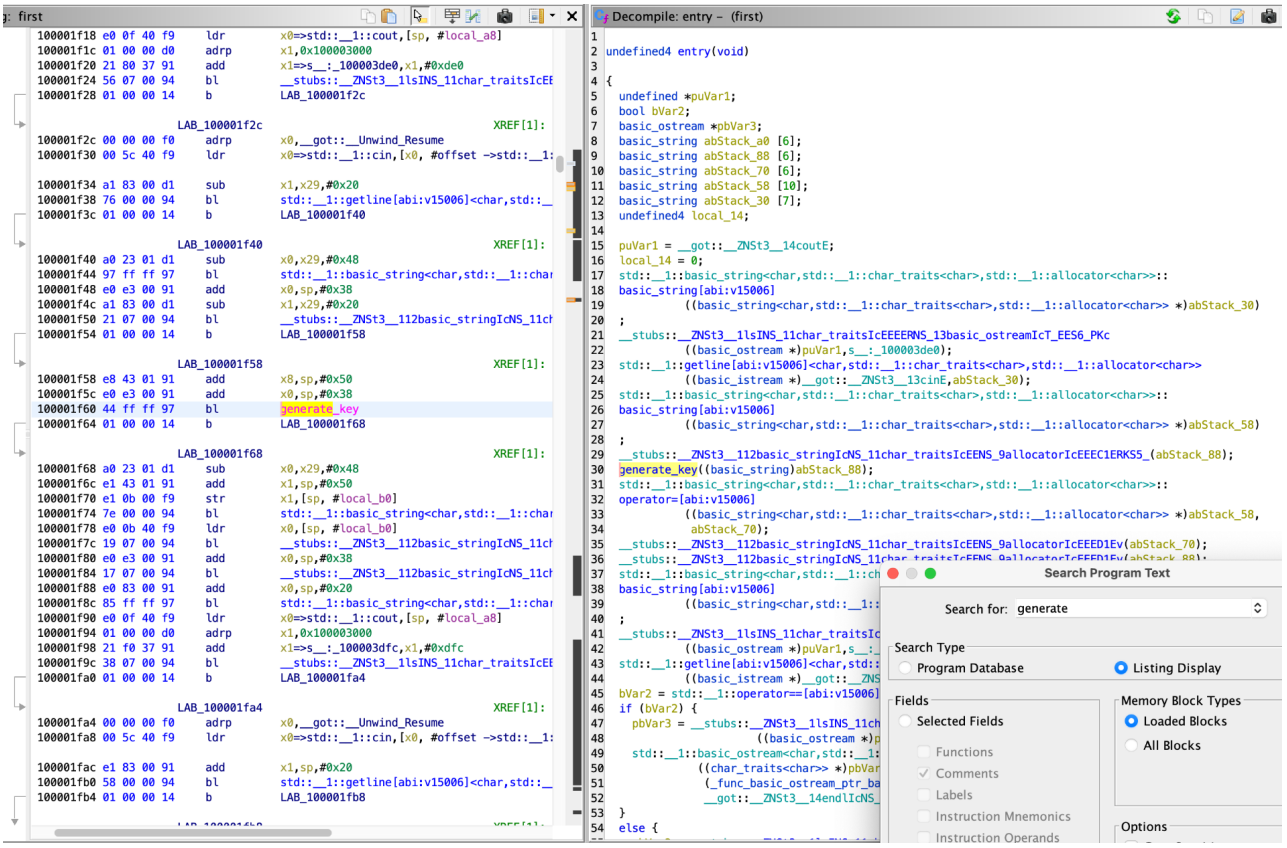

first
Decompile: entry - (first)
Search Program Text
Search for: generate
Search Type
Program Database
Listing Display
Fields
Selected Fields
Functions
Comments
Labels
Instruction Mnemonics
Instruction Operands
Memory Block Types
Loaded Blocks
All Blocks
Options

рис. 3

Decompile: entry - (second_packed)

```
1
2  long entry(void)
3
4  {
5    undefined in_CY;
6    byte bVar1;
7    byte bVar2;
8    undefined *puVar3;
9    byte *pbVar4;
10   byte *pbVar5;
11   uint uVar6;
12   int iVar7;
13   undefined *puVar8;
14   undefined4 *in_x3;
15   ulong uVar9;
16   ulong uVar10;
17   undefined auVar11 [16];
18   undefined auVar12 [12];
19   undefined in_stack_00000000;
20   undefined *local_20;
21   undefined *puStack_18;
22   undefined4 *local_10;
23
24   auVar11 = FUN_10000295c();
25   puVar8 = SUB168(auVar11,0);
26   local_20 = puVar8 + (SUB168(auVar11 >> 0x40,0) & 0xffffffff);
27   uVar10 = 0xffffffff;
28   puStack_18 = (undefined *)register0x00000008;
29   local_10 = in_x3;
30   do {
31     while (puVar3 = (undefined *)FUN_100002744(puVar8), (bool)in_CY) {
32       puVar8 = puVar3 + 1;
33       *(undefined *)register0x00000008 = *puVar3;
34       register0x00000008 = (BADSPACEBASE *)((long)register0x00000008 + 1);
35     }
36     auVar12 = FUN_10000275c();
37     uVar6 = SUB124(auVar12 >> 0x40,0);
38     pbVar4 = SUB128(auVar12,0);
39     bVar1 = 2 < uVar6;
40     pbVar5 = pbVar4;
41     if ((bool)bVar1) {
42       pbVar5 = pbVar4 + 1;
43       uVar6 = ~((uint)*pbVar4 | (uVar6 - 3) * 0x100);
44       uVar10 = (ulong)uVar6;
```

рис. 4

Decompile: entry - (first)

```
1
2  undefined4 entry(void)
3
4  {
5    undefined *puVar1;
6    bool bVar2;
7    basic_ostream *pbVar3;
8    basic_string abStack_a0 [6];
9    basic_string abStack_88 [6];
10   basic_string abStack_70 [6];
11   basic_string abStack_58 [10];
12   basic_string abStack_30 [7];
13   undefined4 local_14;
14
15   puVar1 = __got::__ZNSt3__14coutE;
16   local_14 = 0;
17   std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::
18   basic_string[abi:v15006]
19             ((basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>> *)abStack_30)
20   ;
21   __stubs::__ZNSt3__1lsINS_11char_traitsIcEEEERNS_13basic_ostreamIcT_EES6_PKc
22             ((basic_ostream *)puVar1,s__:_100003de0);
23   std::__1::getline[abi:v15006]<char,std::__1::char_traits<char>,std::__1::allocator<char>>
24             ((basic_istream *)__got::__ZNSt3__13cinE,abStack_30);
25   std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::
26   basic_string[abi:v15006]
27             ((basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>> *)abStack_58)
28   ;
29   __stubs::__ZNSt3__112basic_stringIcNS_11char_traitsIcEENS_9allocatorIcEEEC1ERKS5_(abStack_88);
30   generate_key((basic_string)abStack_88);
31   std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::
32   operator=[abi:v15006]
33             ((basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>> *)abStack_58,
34              abStack_70);
35   __stubs::__ZNSt3__112basic_stringIcNS_11char_traitsIcEENS_9allocatorIcEEED1Ev(abStack_70);
36   __stubs::__ZNSt3__112basic_stringIcNS_11char_traitsIcEENS_9allocatorIcEEED1Ev(abStack_88);
37   std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::
38   basic_string[abi:v15006]
39             ((basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>> *)abStack_a0)
40   ;
41   __stubs::__ZNSt3__1lsINS_11char_traitsIcEEEERNS_13basic_ostreamIcT_EES6_PKc
42             ((basic_ostream *)puVar1,s__:_100003dfc);
43   std::__1::getline[abi:v15006]<char,std::__1::char_traits<char>,std::__1::allocator<char>>
44             ((basic_istream *)__got::__ZNSt3__13cinE,abStack_a0);
45   bVar2 = std::__1::operator==[abi:v15006]<std::__1::allocator<char>>(abStack_a0,abStack_58);
46   if (bVar2) {
47     pbVar3 = __stubs::__ZNSt3__1lsINS_11char_traitsIcEEEERNS_13basic_ostreamIcT_EES6_PKc
48                        ((basic_ostream *)puVar1,s_TRUE_-_!_100003e16);
49     std::__1::basic_ostream<char,std::__1::char_traits<char>>::operator<<[abi:v15006]
50               ((char_traits<char>> *)pbVar3,
51                (_func_basic_ostream_ptr_basic_ostream_ptr *)
52                __got::__ZNSt3__14endlIcNS_11char_traitsIcEEEERNS_13basic_ostreamIT_T0_EES7_);
53   }
54   else {
```

рис. 5

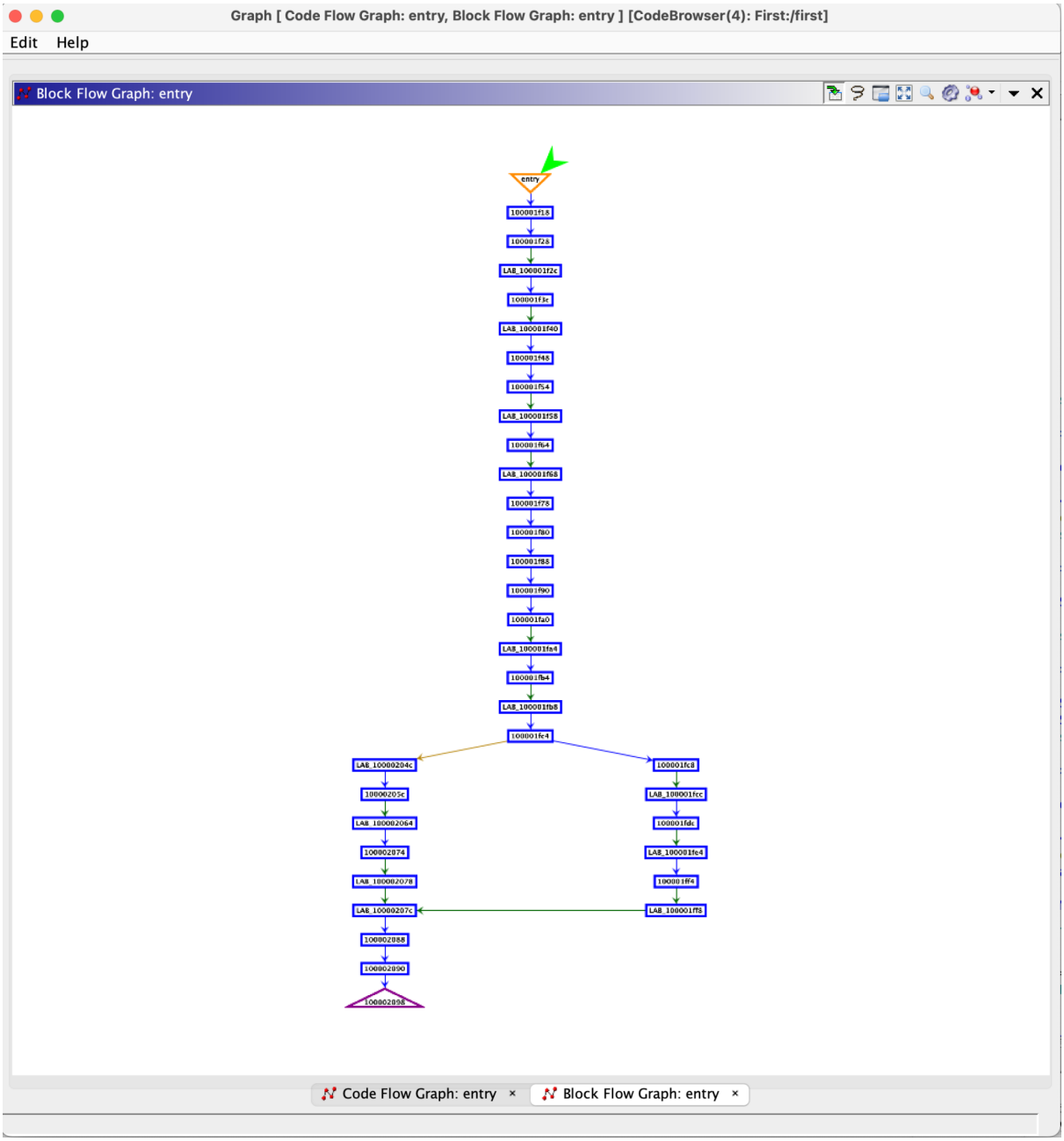

Graph [Code Flow Graph: entry, Block Flow Graph: entry] [CodeBrowser(4): First:/first]
Edit
Help
Block Flow Graph: entry
entry
100001f18
100001f28
LAB_100001f2c
100001f3c
LAB_100001f40
100001f48
100001f54
LAB_100001f58
100001f64
LAB_100001f68
100001f78
100001f80
100001f85
100001f90
100001fa0
LAB_100001fa4
100001fb4
LAB_100001fb8
100001fc4
LAB_10000204c
10000205c
LAB_100002064
100002074
LAB_10000207f
LAB_10000207c
100002088
100002090
100002098
100001fc8
LAB_100001fcc
100001fdc
LAB_100001fe4
100001ff4
LAB_100001ff8
Code Flow Graph: entry
Block Flow Graph: entry

рис. 6

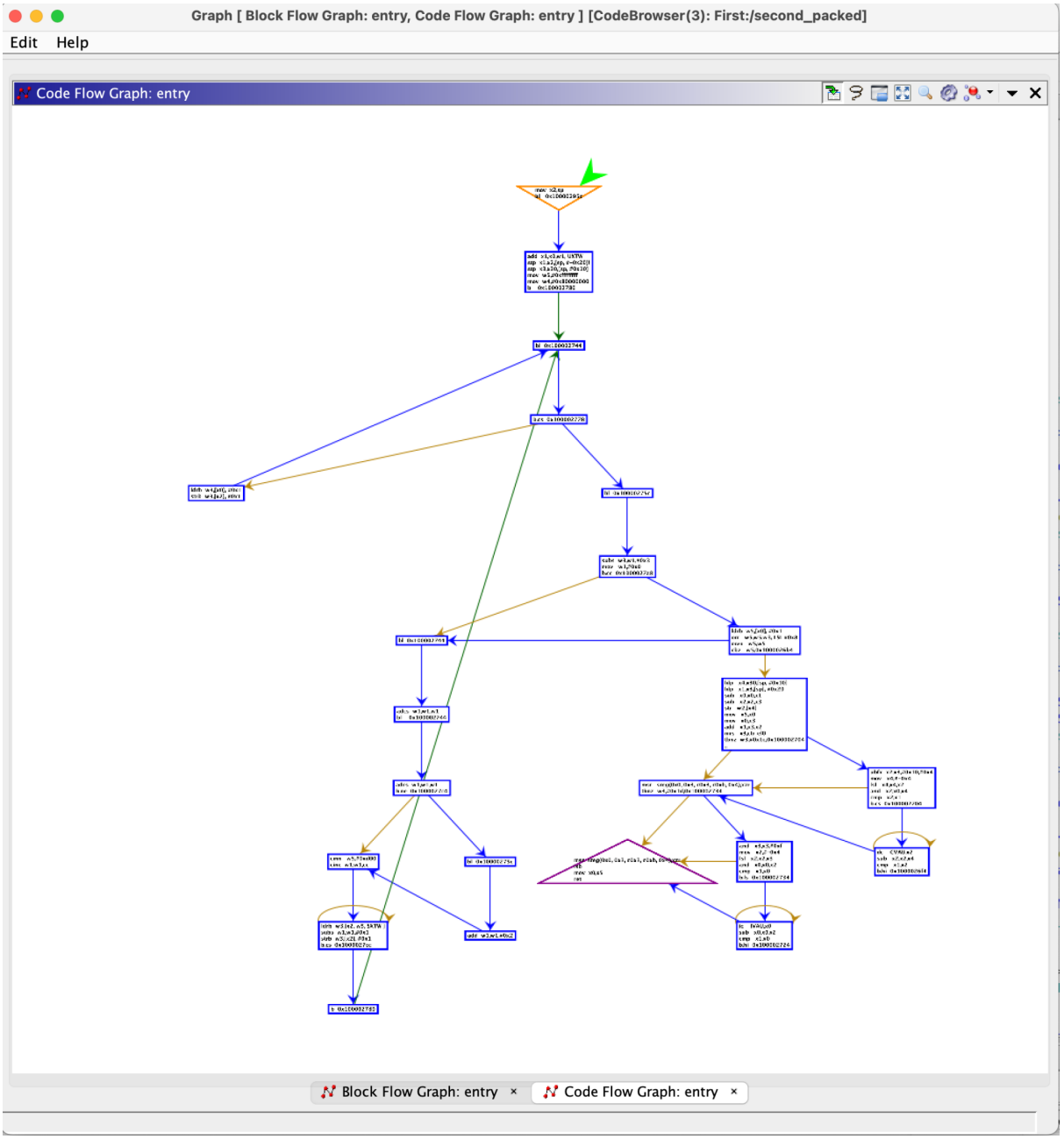

Graph [Block Flow Graph: entry, Code Flow Graph: entry] [CodeBrowser(3): First:/second_packed]
Edit
Help
Code Flow Graph: entry
Block Flow Graph: entry
Code Flow Graph: entry

рис. 7

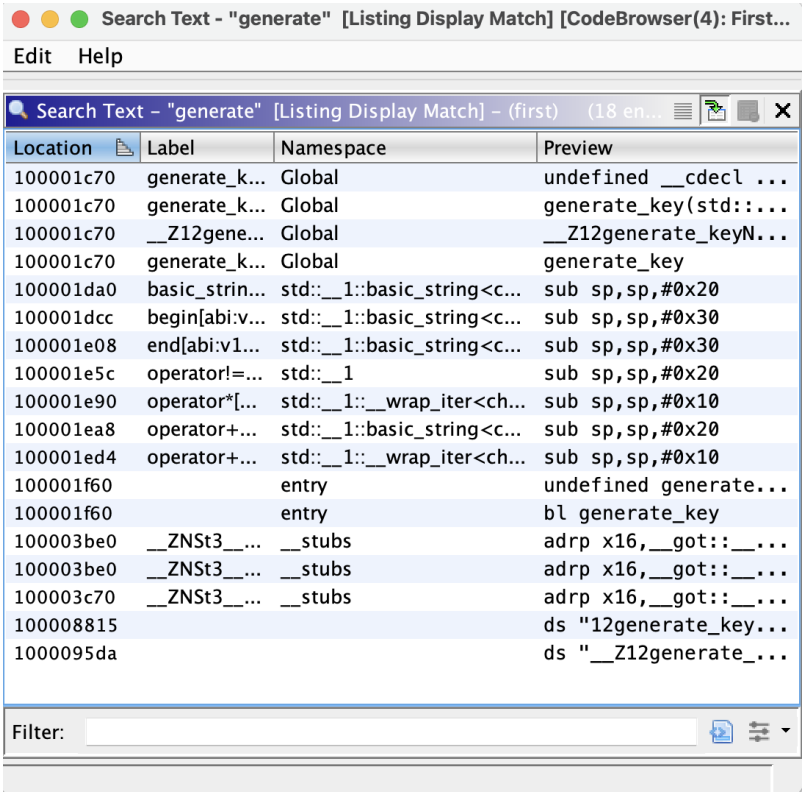

Search Text - "generate" [Listing Display Match] [CodeBrowser(4): First...
Edit Help
Search Text - "generate" [Listing Display Match] - (first) (18 en...
Location Label Namespace Preview
100001c70 generate_k... Global undefined __cdecl ...
100001c70 generate_k... Global generate_key(std::...
100001c70 __Z12gene... Global __Z12generate_keyN...
100001c70 generate_k... Global generate_key
100001da0 basic_strin... std::__1::basic_string<c... sub sp,sp,#0x20
100001dcc begin[abi:v... std::__1::basic_string<c... sub sp,sp,#0x30
100001e08 end[abi:v1... std::__1::basic_string<c... sub sp,sp,#0x30
100001e5c operator!=... std::__1 sub sp,sp,#0x20
100001e90 operator*[... std::__1::__wrap_iter<ch... sub sp,sp,#0x10
100001ea8 operator+... std::__1::basic_string<c... sub sp,sp,#0x20
100001ed4 operator+... std::__1::__wrap_iter<ch... sub sp,sp,#0x10
100001f60 entry undefined generate...
100001f60 entry bl generate_key
100003be0 __ZNSt3__... __stubs adrp x16,__got::__...
100003be0 __ZNSt3__... __stubs adrp x16,__got::__...
100003c70 __ZNSt3__... __stubs adrp x16,__got::__...
100008815 ds "12generate_key...
1000095da ds "__Z12generate_...
Filter:

рис. 8

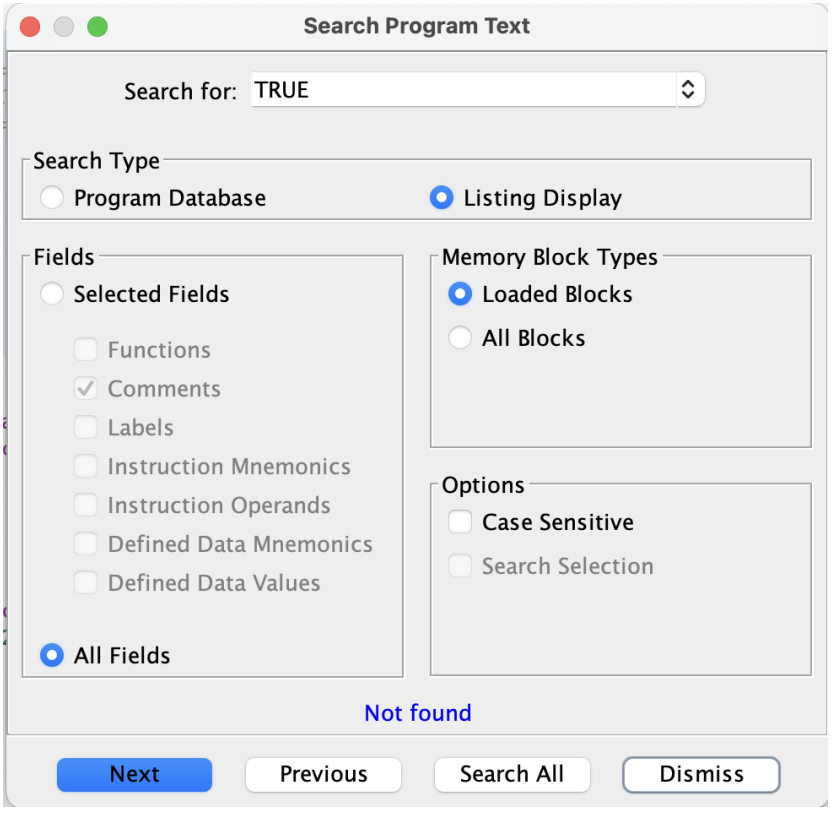

Search Program Text
Search for: TRUE
Search Type
Program Database
Listing Display
Fields
Selected Fields
Functions
Comments
Labels
Instruction Mnemonics
Instruction Operands
Defined Data Mnemonics
Defined Data Values
All Fields
Memory Block Types
Loaded Blocks
All Blocks
Options
Case Sensitive
Search Selection
Not found
Next
Previous
Search All
Dismiss

рис. 9

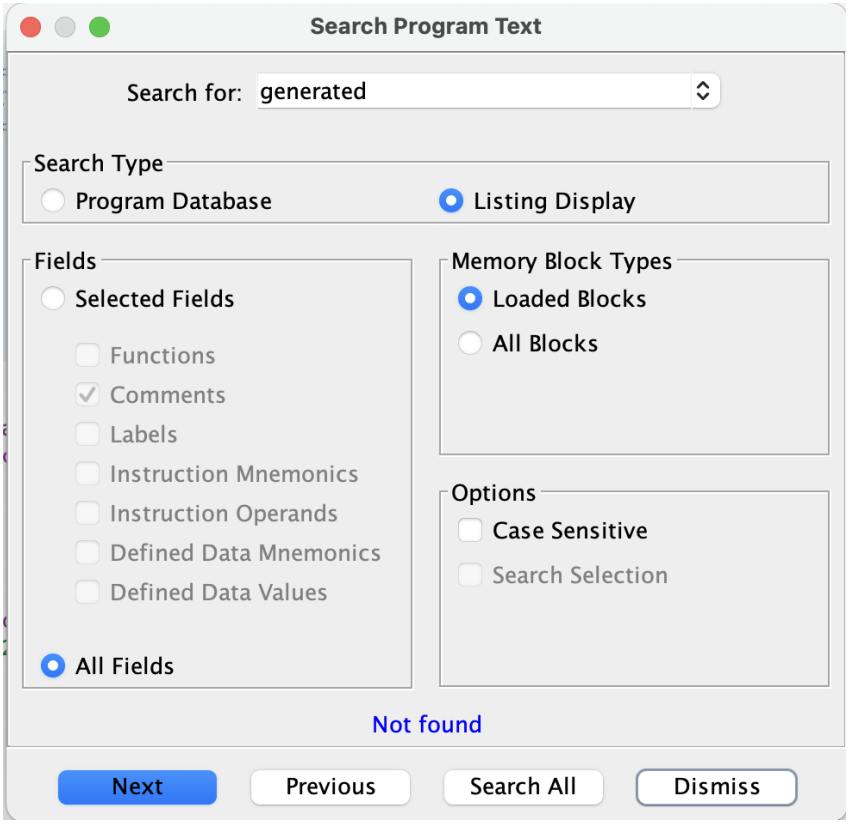

Search Program Text
Search for: generated
Search Type
Program Database
Listing Display
Fields
Selected Fields
Functions
Comments
Labels
Instruction Mnemonics
Instruction Operands
Defined Data Mnemonics
Defined Data Values
All Fields
Memory Block Types
Loaded Blocks
All Blocks
Options
Case Sensitive
Search Selection
Not found
Next
Previous
Search All
Dismiss

рис. 10

Decompile: generate_key - (first)

```
1
2  /* WARNING: Removing unreachable block (ram,0x000100001d70) */
3  /* generate_key(std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char>
4     >) */
5
6  void generate_key(basic_string param_1)
7
8  {
9    bool bVar1;
10   byte *pbVar2;
11   basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>> *in_x8;
12   basic_string abStack_58 [6];
13   uint local_40;
14   byte local_39;
15   undefined8 local_38;
16   undefined8 local_30;
17   basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>> *local_28;
18   undefined local_19;
19
20   local_28 = (basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>> *)
21              (ulong)param_1;
22   local_19 = 0;
23   std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::
24   basic_string[abi:v15006](in_x8);
25   local_30 = std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::
26              begin[abi:v15006](local_28);
27   local_38 = std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::
28              end[abi:v15006](local_28);
29   while (bVar1 = std::__1::operator!=[abi:v15006]<char*>
30                            ((__wrap_iter *)&local_30,(__wrap_iter *)&local_38), bVar1) {
31     pbVar2 = (byte *)std::__1::__wrap_iter<char*>::operator*[abi:v15006]
32                                ((__wrap_iter<char*> *)&local_30);
33     local_39 = *pbVar2;
34     local_40 = (uint)local_39;
35     __stubs::__ZNSt3__19to_stringEi(local_40);
36     std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::
37     operator+=[abi:v15006](in_x8,abStack_58);
38     __stubs::__ZNSt3__112basic_stringIcNS_11char_traitsIcEENS_9allocatorIcEEED1Ev(abStack_58);
39     std::__1::__wrap_iter<char*>::operator++[abi:v15006]((__wrap_iter<char*> *)&local_30);
40   }
41   return;
42 }
43
```

Decompile: FUN_100002744 - (second_packed)

```
1
2  long FUN_100002744(long param_1)
3
4  {
5    uint in_w4;
6
7    if ((in_w4 & 0x7fffffff) != 0) {
8      return param_1;
9    }
10   return param_1 + 4;
11 }
12
```

Decompile: FUN_10000275c - (second_packed)

```
1
2  void FUN_10000275c(undefined8 param_1)
3
4  {
5    byte in_CY;
6    bool bVar1;
7    uint uVar2;
8    undefined8 uVar3;
9    code *UNRECOVERED_JUMPTABLE;
10   undefined auVar4 [16];
11   undefined auVar5 [12];
12
13   uVar3 = 1;
14   do {
15     auVar5 = FUN_100002744(param_1,uVar3);
16     uVar2 = SUB124(auVar5 >> 0x40,0);
17     bVar1 = CARRY4(uVar2,uVar2) || CARRY4(uVar2 * 2,(uint)in_CY);
18     auVar4 = FUN_100002744(SUB128(auVar5,0),uVar2 * 2 + (uint)in_CY);
19     uVar3 = SUB168(auVar4 >> 0x40,0);
20     param_1 = SUB168(auVar4,0);
21     in_CY = bVar1;
22   } while (!bVar1);
23                     /* WARNING: Could not recover jumptable at 0x000100002774. Too many branches *
24                     /* WARNING: Treating indirect jump as call */
25   (*UNRECOVERED_JUMPTABLE)();
26   return;
27 }
28
```

Decompile: FUN_100002880 - (second_packed)

```
1
2  void FUN_100002880(void)
3
4  {
5    ulong uVar1;
6    code *UNRECOVERED_JUMPTABLE;
7    ulong uVar2;
8    long lVar3;
9    undefined8 in_x6;
10   long unaff_x22;
11   code *unaff_x28;
12   uint *unaff_x30;
13   undefined8 param_11;
14   long param_12;
15   long param_13;
16   ulong local_8;
17
18   param_12 = (long)unaff_x30 + ((ulong)*unaff_x30 - unaff_x22);
19   CallSupervisor(0);
20   param_11 = 0;
21   CallSupervisor(0);
22   uVar1 = (long)unaff_x28 + -param_13;
23   uVar2 = uVar1 & 0xfffffffffffff000;
24   lVar3 = param_12 - uVar2;
25   local_8 = (ulong)*unaff_x30;
26   UNRECOVERED_JUMPTABLE = (code *)((long)unaff_x30 + -param_13);
27   (*unaff_x28)(unaff_x30 + 3,unaff_x30[1],UNRECOVERED_JUMPTABLE,&local_8,unaff_x30[2],0,in_x6,
28                unaff_x30[1],uVar1);
29   CallSupervisor(0);
30                     /* WARNING: Could not recover jumptable at 0x000100002944. Too many branches *
31                     /* WARNING: Treating indirect jump as call */
32   (*UNRECOVERED_JUMPTABLE)(uVar2,lVar3,5);
33   return;
34 }
35
```

```
Decompile: FUN_10000295c - (second_packed)

void FUN_10000295c(undefined8 param_1,undefined8 param_2,char **param_3)

{
  code *UNRECOVERED_JUMPTABLE_00;
  char *pcVar1;
  char *pcVar2;
  char **ppcVar3;
  char **ppcVar4;
  int iVar5;

  do {
    pcVar1 = *param_3;
    ppcVar3 = param_3 + 1;
    param_3 = param_3 + 1;
  } while (pcVar1 != (char *)0x0);
  do {
    pcVar1 = *ppcVar3;
    ppcVar4 = ppcVar3 + 1;
    ppcVar3 = ppcVar3 + 1;
  } while (pcVar1 != (char *)0x0);
  do {
    ppcVar3 = ppcVar4 + 1;
    if (*ppcVar4 == (char *)0x0) {
                    /* WARNING: Treating indirect jump as call */
      UNRECOVERED_JUMPTABLE_00 = (code *)SoftwareBreakpoint(0,0x10000295c);
      (*UNRECOVERED_JUMPTABLE_00)();
      return;
    }
    iVar5 = 0x10;
    pcVar1 = *ppcVar4;
    pcVar2 = "executable_path=";
    while( true ) {
      iVar5 = iVar5 + -1;
      ppcVar4 = ppcVar3;
      if (*pcVar1 != *pcVar2) break;
      pcVar1 = pcVar1 + 1;
      pcVar2 = pcVar2 + 1;
      if (iVar5 == 0) {
        CallSupervisor(0);
        FUN_100002880(0x2278,entry);
                    /* WARNING: Treating indirect jump as call */
        UNRECOVERED_JUMPTABLE_00 = (code *)UndefinedInstructionException(0xaa0,0x1000029d0);
        (*UNRECOVERED_JUMPTABLE_00)();
        return;
      }
    }
  } while( true );
}
```

Приложение 1. https://github.com/nvinnikov/software_security/blob/master/first.cpp
first.cpp Код до реализации методов защиты от отладки

```
#include <iostream>
#include <string>
#include <cstdlib>

std::string generate_key(std::string login){
```

```
    std::string generated_key;
    for (char c : login) {
        int ascii_code = static_cast<int>(c);
        generated_key += std::to_string(ascii_code);
    }
    return generated_key;
}


int main() {
    std::string login;
    std::cout << "Введите логин: ";
    std::getline(std::cin, login);

    // Алгоритм генерации ключа
    std::string generated_key;
    generated_key = generate_key(login);

    std::string user_key;
    std::cout << "Введите ключ: ";
    std::getline(std::cin, user_key);

    if (user_key == generated_key) {
        std::cout << "TRUE - поздравляем!" << std::endl;
    } else {
        std::cout << "FALSE - ошибка!" << std::endl;
    }

    return 0;
}
```

Приложение 2. https://github.com/nvinnikov/software_security/blob/master/second.cpp
second.cpp Код после реализации методов защиты от отладки

```
#include <iostream>
#include <string>
#include <cstdlib>

#ifdef __unix__
#include <sys/ptrace.h>
```

```
#include <unistd.h>
#endif

#ifdef _WIN32
#include <Windows.h>
#endif

using namespace std;

bool is_debugger_present() {

// Защита от отладки с помощью специальных функций языка
#ifdef __linux__
   // Проверка наличия отладчика в Linux
   if (ptrace(PTRACE_TRACEME, 0, 1, 0) == -1) {
       return true;
   }
   ptrace(PTRACE_DETACH, 0, 1, 0);
#endif

// Защита от отладки с помощью специальных функций языка
#ifdef _WIN32
   // Проверка наличия отладчика в Windows
   if (IsDebuggerPresent()) {
       //навязывание отладчику ложных точек останова
       srand(time(nullptr));
       int num_breakpoints = rand() % 10 + 1; // от 1 до 10
       for (int i = 0; i < num_breakpoints; ++i) {
           DebugBreak();
   }
       return true;
   }
#endif

// Защита от отладки с помощью директив препроцессора
#ifdef _DEBUG
   return true;
#endif

return false;
}
```

```
//функция генерации ключа
string generate_key(string login){
   string generated_key;
   //длина логина - подсчет количества букв в логине (1)
   int login_size = login.length();
   for (char c : login) {

       int ascii_code = static_cast<int>(c);

       //умножение аски кода на длина логина - преобразование генерации
ключевой информации (1)
       generated_key += to_string(ascii_code*login_size);
   }
   return generated_key;
}

//функция для генерации ключа-обманки
string generate_key_new(string login){
   string generated_key;
   //рандом
   generated_key = to_string(rand() % 1000 + 1);
   return generated_key;
}

int main() {

   string login;
   cout << "Введите логин: ";
   getline(cin, login);

   // Реверс login - преобразование данных для генерации ключевой информации
(1)
   reverse(login.begin(), login.end());

   // Алгоритм генерации ключа
   string generated_key;

   if (is_debugger_present()){
           // Вызов функции через указатель на нее (косвенный вызов) (2)
           string (*pgenerate_key_new)(string) = generate_key_new;
```

```
            generated_key = pgenerate_key_new(login);
            return 0;
    } else {
            // Вызов функции через указатель на нее (косвенный вызов) (2)
            string (*pgenerate_key)(string) = generate_key;
            generated_key = pgenerate_key(login);
    }



    string user_key;
    cout << "Введите ключ: ";
    getline(cin, user_key);

    if (user_key == generated_key) {
        cout << "TRUE - поздравляем!" << endl;
    } else {
        cout << "FALSE - ошибка!" << endl;
    }

    return 0;
}
```

Приложение 3.
Makefile

```
run1:first
    g++ -std=c++11 first.cpp -o first && ./first
run2:second
    g++ -w -std=c++11 second.cpp -o second && strip second && upx -qqq second
&& ./second
```