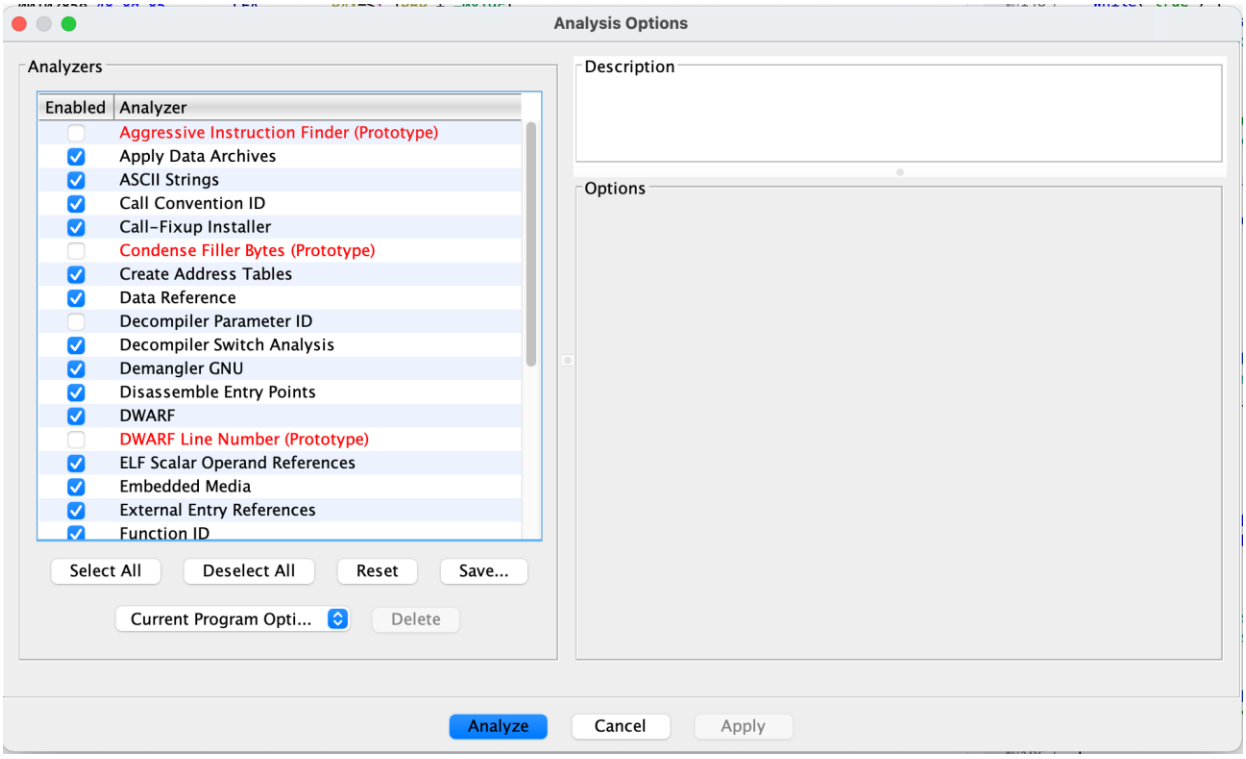
Отчет PZ2

Алексеев Кирилл СКБ172

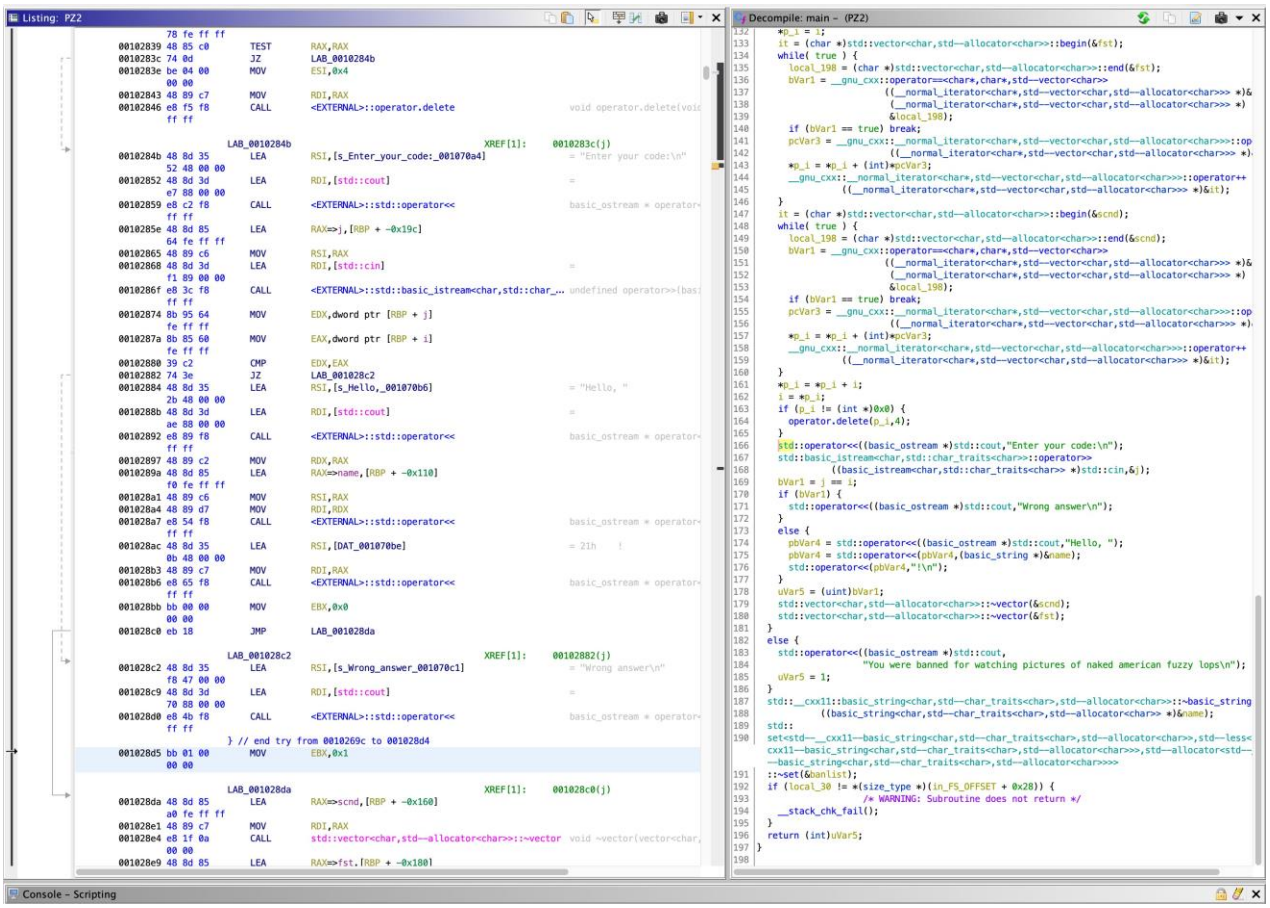
Для дизассемблирования и последующего декомпилирования используем Ghidra

В Ghidra запускаем анализатор



Переходим в функцию main

По полученному декомпилированному коду понятно, что весь алгоритм находится в функции main.



Находим часть кода, которая отвечает за проверку пароля (это понятно просто по строкам).

```
  }
  std::operator<<((basic_ostream *)std::cout,"Enter your code:\n");
  std::basic_istream<char,std::char_traits<char>>::operator>>
            ((basic_istream<char,std::char_traits<char>> *)std::cin,&j);
  bVar1 = j != i;
  if (bVar1) {
    std::operator<<((basic_ostream *)std::cout,"Wrong answer\n");
  }
  else {
    pbVar4 = std::operator<<((basic_ostream *)std::cout,"Hello, ");
    pbVar4 = std::operator<<(pbVar4,(basic_string *)&name);
    std::operator<<(pbVar4,"!\n");
  }
  uVar5 = (uint)bVar1;
  std::vector<char,std--allocator<char>>::~vector(&scnd);
  std::vector<char,std--allocator<char>>::~vector(&fst);
}
```

На этом этапе уже можно пропатчить исходный бинарный файл,

Для этого можно заменить JNZ на JZ, либо же положить в EAX тоже самое, что в EDX.

Так же можем найти список забанных имен и написать патч для его обхода.

```
31  size_type local_30;
32
33  local_30 = *(size_type *)(in_FS_OFFSET + 0x28);
34  std::allocator<char>::allocator();
35                      /* try { // try from 0010234f to 00102353 has its CatchHandler @ 001029ea */
36  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<>
37            ((basic_string<char,std::char_traits<char>,std::allocator<char>> *)&name,"Kirill",
38             (allocator<char> *)&stack0xffffffffffffe55);
39  std::allocator<char>::allocator();
40                      /* try { // try from 00102382 to 00102386 has its CatchHandler @ 001029d6 */
41  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<>
42            (&local_f8,"Alisa",(allocator<char> *)&stack0xffffffffffffe56);
43  std::allocator<char>::allocator();
44                      /* try { // try from 001023b5 to 001023b9 has its CatchHandler @ 001029c2 */
45  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<>
46            (&local_d8,"Renat",(allocator<char> *)&stack0xffffffffffffe57);
47  std::allocator<char>::allocator();
48                      /* try { // try from 001023e8 to 001023ec has its CatchHandler @ 001029ae */
49  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<>
50            (&local_b8,"Yura",(allocator<char> *)&i);
51  std::allocator<char>::allocator();
52                      /* try { // try from 0010241e to 00102422 has its CatchHandler @ 0010299a */
53  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<>
54            (&local_98,"Denis",(allocator<char> *)&j);
55  std::allocator<char>::allocator();
56                      /* try { // try from 00102454 to 00102458 has its CatchHandler @ 00102986 */
57  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<>
58            (&local_78,"Alex",(allocator<char> *)&it);
59  std::allocator<char>::allocator();
60                      /* try { // try from 0010248a to 0010248e has its CatchHandler @ 00102972 */
61  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<>
62            (&local_58,"kolyavv",(allocator<char> *)&local_198);
63  std::allocator<std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>>::
64  allocator((allocator<std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>>
65             *)&scnd);
66                      /* try { // try from 001024db to 001024df has its CatchHandler @ 00102930 */
67  std::
```

Находим проверку на бан лист

```
            ((basic_string<char,std::char_traits<char>,std::allocator<char>> *)&name),
                    /* try { // try from 001025b3 to 001025e7 has its CatchHandler @ 00102a2a */
  std::operator>>((basic_istream *)std::cin,(basic_string *)&name);
  bVar1 = std::
          set<std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>,std::les
          <std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>>,std::alloc
          tor<std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>>>
          ::contains(&banlist,(key_type *)&name);
  if (bVar1 == false) {
```

И меняем инструкцию JZ на на JMP

```
                    LAB_001025b3                                    XREF[1]:     001095e8(*)
001025b3 e8 c8 fb       CALL        <EXTERNAL>::std::operator>>                  basic_istream * operator>>(basic..
         ff ff
001025b8 48 8d 95       LEA         RDX=>name,[RBP + -0x110]
         f0 fe ff ff
001025bf 48 8d 85       LEA         RAX=>banlist,[RBP + -0x140]
         c0 fe ff ff
001025c6 48 89 d6       MOV         RSI,RDX
001025c9 48 89 c7       MOV         RDI,RAX
001025cc e8 53 0c       CALL        std::set<std--__cxx11--basic_string<char,std--... bool contains(set<std--__cxx11--..
         00 00
001025d1 84 c0          TEST        AL,AL
001025d3 74 1d          JZ          LAB_001025f2
```

Проверяем пропатченную версию

```
Greetings! Please, enter your name:
Alex
Enter your code:
thisisawrongkeyword
Hello, Alex!
```

Попробуем понять, какой же алгоритм проверки пароля

```
 97  if (bVar1 == false) {
 98    std::vector<char,std--allocator<char>>::vector(&fst);
 99    std::vector<char,std--allocator<char>>::vector(&scnd);
100    i = 0x3f2;
101    j = 0;
102    it = (char *)std::__cxx11::basic_string<char,std--char_traits<char>,std--allocator<char>>::begin
103                           ((basic_string<char,std--char_traits<char>,std--allocator<char>> *)&name)
104    ;
105    while( true ) {
106      local_198 = (char *)std::__cxx11::
107                           basic_string<char,std--char_traits<char>,std--allocator<char>>::end
108                                    ((basic_string<char,std--char_traits<char>,std--allocator<char>>
109                                      *)&name);
110      bVar1 = __gnu_cxx::operator==<char*,char*,std--__cxx11--basic_string<char>>
111                         (&it,(__normal_iterator<char*,std--__cxx11--basic_string<char,std--char_trai
                           ts<char>,std--allocator<char>>>
112                               *)&local_198);
113      if (bVar1 == true) break;
114      if ((j & 1U) == 0) {
115        pcVar2 = __gnu_cxx::
116                 __normal_iterator<char*,std--__cxx11--basic_string<char,std--char_traits<char>,std-
                 -allocator<char>>>
117                 ::operator*(&it);
118        std::vector<char,std--allocator<char>>::push_back(&scnd,pcVar2);
119      }
120      else {
121        pcVar2 = __gnu_cxx::
122                 __normal_iterator<char*,std--__cxx11--basic_string<char,std--char_traits<char>,std-
                 -allocator<char>>>
123                 ::operator*(&it);
124                    /* try { // try from 0010269c to 001028d4 has its CatchHandler @ 00102a07 */
125        std::vector<char,std--allocator<char>>::push_back(&fst,pcVar2);
126      }
127      __gnu_cxx::
128      __normal_iterator<char*,std--__cxx11--basic_string<char,std--char_traits<char>,std--allocator<
         char>>>
129        ::operator++(&it);
130    }
131    p_i = (int *)operator.new(4);
132    *p_i = i;
133    it = (char *)std::vector<char,std--allocator<char>>::begin(&fst);
134    while( true ) {
135      local_198 = (char *)std::vector<char,std--allocator<char>>::end(&fst);
136      bVar1 = __gnu_cxx::operator==<char*,char*,std--vector<char>>
137                         ((__normal_iterator<char*,std--vector<char,std--allocator<char>>> *)&it,
138                          (__normal_iterator<char*,std--vector<char,std--allocator<char>>> *)
139                          &local_198);
140      if (bVar1 == true) break;
141      pcVar3 = __gnu_cxx::__normal_iterator<char*,std--vector<char,std--allocator<char>>>::operator*
142                          ((__normal_iterator<char*,std--vector<char,std--allocator<char>>> *)&it);
143      *p_i = *p_i + (int)*pcVar3;
144      __gnu_cxx::__normal_iterator<char*,std--vector<char,std--allocator<char>>>::operator++
145                ((__normal_iterator<char*,std--vector<char,std--allocator<char>>> *)&it);
146    }
147    it = (char *)std::vector<char,std--allocator<char>>::begin(&scnd);
148    while( true ) {
149      local_198 = (char *)std::vector<char,std--allocator<char>>::end(&scnd);
150      bVar1 = __gnu_cxx::operator==<char*,char*,std--vector<char>>
151                         ((__normal_iterator<char*,std--vector<char,std--allocator<char>>> *)&it,
152                          (__normal_iterator<char*,std--vector<char,std--allocator<char>>> *)
153                          &local_198);
154      if (bVar1 == true) break;
155      pcVar3 = __gnu_cxx::__normal_iterator<char*,std--vector<char,std--allocator<char>>>::operator*
156                          ((__normal_iterator<char*,std--vector<char,std--allocator<char>>> *)&it);
157      *p_i = *p_i + (int)*pcVar3;
158      __gnu_cxx::__normal_iterator<char*,std--vector<char,std--allocator<char>>>::operator++
159                ((__normal_iterator<char*,std--vector<char,std--allocator<char>>> *)&it);
160    }
161    *p_i = *p_i + i;
162    i = *p_i;
163    if (p_i != (int *)0x0) {
164      operator.delete(p_i,4);
165    }
166    std::operator<<((basic_ostream *)std::cout,"Enter your code:\n");
167    std::basic_istream<char,std::char_traits<char>>::operator>>
168              ((basic_istream<char,std::char_traits<char>> *)std::cin,&j);
```

Видим 3 цикла.
((j & 1U) == 0) для любого j даст true, поэтому в итоге вектор fst будет пустой (видимо, с целью абфускации). Таким образом scnd заполняется введеным именем.

Соответсвтенно второй цикл с обработкой fst можно пропустить
В третьем цикле на каждой итерации в p_i добавляется acsii значение

После цикла видим еще, где i = 0x3f2 (в десятиричной это 1010)

p_i = *p_i + i

Начальное значение p_i так же 1010 (p_i = i;)

Таким образом, можем легко написать keygen на питоне
sum(list(map(lambda x: ord(x), name))) + 1010 + 1010

```
>>> name = "Igoryan"
>>> sum(list(map(lambda x: ord(x), name))) + 1010 + 1010
2749
```

Проверяем:

```
Greetings! Please, enter your name:
Igoryan
Enter your code:
2749
Hello, Igoryan!
```