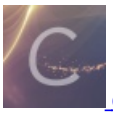
攻防世界Reverse进阶区-elrond32-writeup

原创

y4ung 于 2020-10-01 10:37:54 发布 334 收藏

分类专栏： ctf 文章标签： ctf

本文链接：https://blog.csdn.net/qq_35056292/article/details/108893675



1. 介绍

本题是xctf攻防世界中Reverse的进阶区的题elrond32

题目来源： tinyctf-2014

2. 分析

```
$ file rev300
rev300: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.
so.2, for GNU/Linux 2.6.24, BuildID[sha1]=a93ffe39302e19ef5184a1d86b720b11a7a97941, stripped
$ chmod +x rev300
$ ./rev300
Access denied
```

可以看到main函数需要传入一个字符串参数。

> main函数的参数 `int main(int argc,char *argv[])` 可以参考这篇博客：[C/C++基础知识] main函数的参数argc和argv
> 第一个参数 `argc` 表示的是参数的数量(不需要我们指定)，第二个是字符串数组，用来存放指向字符串参数的指针数组，每个元素指向一个参数，空格分隔参数，其长度为argc，数组下标从0开始。
> argv[0] 指向程序运行时的全路径名
> argv[1] 指向程序在DOS命令中执行程序名后的第一个字符串
> argv[2] 指向执行程序名后的第二个字符串
> argv[argc] 为NULL
> 比如： `./rev300 arg1 arg2` ，那么 `argc` 的值为3， `argv[0]` 为./rev300， `argv[1]` 为 arg1， `argv[2]` 为arg2

```
int __cdecl main(int a1, char **a2)
{
  if ( a1 > 1 && sub_8048414(a2[1], 0) )
  {
    puts("Access granted");
    sub_8048538(a2[1]);
  }
  else
  {
    puts("Access denied");
  }
  return 0;
}
```

首先，main函数参数需要输入一个字符串，如此一来，a1的值为2，满足条件。
其次，`sub_8048414(a2[1], 0)` 的返回值必须为True才行。参数是a2[1]，也就是我们输入的字符串，还有另外一个参数0.

进入sub_8048414。发现就是一个switch case 和递归结合的计算过程:

```
[a1], 0  => i
[a1+1], 7 => s
[a1+2], 1 => e
[a1+2], 3 => n
[a1+3], 6 => g
[a1+4], 5 => a
[a1+5], 9 => r
[a1+6], 4 => d
```

下面是IDA反汇编的代码

```
// a2[1], 0
signed int __cdecl sub_8048414(_BYTE *a1, int a2)
{
  signed int result; // eax

  switch ( a2 )
  {
    case 0:
      if ( *a1 == 105 )
        goto LABEL_19;
      result = 0;
      break;
    case 1:
      if ( *a1 == 'e' )
        goto LABEL_19;
      result = 0;
      break;
    case 3:
      if ( *a1 == 'n' )
        goto LABEL_19;
      result = 0;
      break;
    case 4:
      if ( *a1 == 'd' )
        goto LABEL_19;
      result = 0;
      break;
    case 5:
      if ( *a1 == 'a' )
        goto LABEL_19;
      result = 0;
      break;
    case 6:
      if ( *a1 == 'g' )
        goto LABEL_19;
      result = 0;
      break;
    case 7:
      if ( *a1 == 's' )
        goto LABEL_19;
      result = 0;
      break;
    case 9:
      if ( *a1 == 'r' )
LABEL_19:
        result = sub_8048414(a1 + 1, 7 * (a2 + 1) % 11);
      else
        result = 0;
      break;
    default:
      result = 1;
      break;
  }
  return result;
}
```

最终正确的输入为：`isengard`。运行程序，输入，得到flag：`flag{s0me7hing_S0me7hinG_t0lki3n}`

3. 总结

需要对main函数的参数约定了解，一开始已经把字符串逆出来了，但是在main函数参数这边卡了一下。

补充：因为一开始对main函数参数不了解，就用gdb动态调试了。然后发现这个文件符号表被去掉了，找不到main函数。我试了两种方法：

1. 根据IDA里的函数地址去打断点，比如：`b *0x80432fg`
2. `b __libc_start_main`，参考自：gdb无法找到main