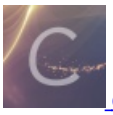
攻防世界Reverse进阶区-re1-100-writeup

原创

y4ung 于 2020-09-24 23:17:42 发布 395 收藏 1

分类专栏： ctf 文章标签： ctf

本文链接：https://blog.csdn.net/qq_35056292/article/details/108785694



1. 介绍

本题是xctf攻防世界中Reverse的进阶区的题re1-100

2. 分析

```
$ file RE100
RE100: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), dynamically linked, interpreter /lib64/ld-linux
-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=94dd59e952c54304bd14f282695ae62c4f6cbe55, not stripped
$ chmod +x RE100
$ ./RE100
Input key : 123
Wrong !!!

Input key : 213
Wrong !!!

Input key : ^C[1]  + 24423 interrupt  ./RE100
```

找到`Input key :`被引用的地方是在main函数中。

首先是调用write函数将用户输入写到pParentWrite里，然后再读出来做了一些校验。

```
30      {
31        printf("Input key : ", argva);
32        memset(bufWrite, 0, 200uLL);
33        gets(bufWrite, 0LL);                    // 用户输入保存在bufWrite中
34        v4 = strlen(bufWrite);
35        v5 = write(pParentWrite[1], bufWrite, v4);// 将用户输入写入到pParentWrite里
36        if ( v5 != strlen(bufWrite) )
37          printf("parent - partial/failed write", bufWrite);
38        do
39        {
40          memset(bufParentRead, 0, 200uLL);
41          numReada = read(pParentRead[0], bufParentRead, 0xC8uLL);
42          v6 = bCheckPtrace || checkDebuggerProcessRunning();
43          if ( v6 )
44          {
45            puts("Wrong !!!\n");
46          }
47          else if ( !checkStringIsNumber(bufParentRead) )
48          {
49            puts("Wrong !!!\n");
50          }
51          else
52          {
53            if ( atoi(bufParentRead) )
54            {
55              puts("True");
56              if ( close(pParentWrite[1]) == -1 )
57                exit(1);
58              exit(0);
59            }
60            puts("Wrong !!!\n");
61          }
00008688 main:52 (408688)
<
```


接下来在while循环中，首先将用户输入从pParentWrite中读出来，放到bufParentRead中。然后做了一些校验：

- bufParentRead必须以 `{` 开始
- bufParentRead长度为42

> strlen函数计算指定的字符串s 的长度，不包括结束字符"\0"

- bufParentRead索引为[1]~[10]的内容为：`53fc275d81`
- 最后一个字符为`}`
- bufParentRead索引为[31]~[40]的内容为: `4938ae4efd`
- bufParentRead必须与 `{daf29f59034938ae4efd53fc275d81053ed5be8c}` 相等

显然，最后一个条件就不满足前面的 `bufParentRead索引为[1]~[10]的内容为：53fc275d81` 这个条件。奇怪怎么会自相矛盾呢？

突然看到在94行这个if判断前面，还有个if判断：`if ( !confuseKey(bufParentRead, 42) )`，会不会是在这个函数里修改了bufParentRead的内容呢？跟进去看看

```
68    while ( 1 )
69    {
70      memset(bufParentRead, 0, 0xC8uLL);
71      numRead = read(pParentWrite[0], bufParentRead, 200uLL);// 从pParentWrite中读出来，放到bufParentRead中
72      if ( numRead == -1 )
73        break;
74      if ( numRead )
75      {
76        if ( childCheckDebugResult() )
77        {
78          responseFalse();
79        }
80        else if ( bufParentRead[0] == '{' )     // 用户输入以{开始
81        {
82          if ( strlen(bufParentRead) == 42 )    // 用户输入长度为42，strlen函数计算指定的字符串s 的长度，不包括结束字符"\0"
83          {
84            if ( !strncmp(&bufParentRead[1], "53fc275d81", 10uLL) )// [1]~[10]: 53fc275d81
85            {
86              if ( bufParentRead[strlen(bufParentRead) - 1] == '}' )// 最后一个字符为}
87              {
88                if ( !strncmp(&bufParentRead[31], "4938ae4efd", 10uLL) )// [31]~[40]开始为4938ae4efd
89                {
90                  if ( !confuseKey(bufParentRead, 42) )
91                  {
92                    responseFalse();
93                  }
94                  else if ( !strncmp(bufParentRead, "{daf29f59034938ae4efd53fc275d81053ed5be8c}", 42uLL) )// 明明bufParentRead只要
95                                                    // 等于{daf29f59034938ae4efd53fc275d81053ed5be8c}就正确了，
96                                                    // 但是提交以后并不是flag，说明可能{daf29f59034938ae4efd53fc275d81053ed5be8c}是从用户输入变换过来的，
97                                                    // 往前看只有在第90行的confuseKey比较可疑了
98                  {
99                    responseTrue();
100                 }
101                 else
00008688 main:73 (408688)
```

在confuseKey函数中，bufParentRead作为参数，在函数中是szKey。

首先初始化了四个数组szPart1szPart4，然后将szKey不考虑首字符和最后一个字符的情况下，每10个字符为一组，分别赋值给szPart1szPart4。

然后将szKey清空，用0填充，再讲szPart1~4乱序重新拼接会szKey。即szKey='{' + szPart1 + szPart2 + szPart3 + szPart4 + '}'

```
34  strncpy(szPart1, szKey + 1, 10uLL);           // 把szKey掐头去尾（不考虑{和}），每10个字符分为1组，分别赋值给szPart1~szPart4
35  strncpy(szPart2, szKey + 11, 10uLL);
36  strncpy(szPart3, szKey + 21, 10uLL);
37  strncpy(szPart4, szKey + 31, 10uLL);
38  memset(szKey, 0, iKeyLength);
39  *szKey = '{';
40  strcat(szKey, szPart3);                       // 重新拼接，
41  strcat(szKey, szPart4);
42  strcat(szKey, szPart1);
43  strcat(szKey, szPart2);
44  szKey[41] = '}';
45  return 1;
46}
```

ok，那么我们只要重新排序即可，最终得到正确的输入为：`{53fc275d81053ed5be8cdaf29f59034938ae4efd}`

```
key = "daf29f59034938ae4efd53fc275d81053ed5be8c"
k3 = key[0:10]
k4 = key[10:20]
k1 = key[20:30]
k2 = key[30:40]

res = "{" + k1+k2+k3+k4 + "}"
print(res)  # {53fc275d81053ed5be8cdaf29f59034938ae4efd}
```

执行程序测试一下：

```
$ ./RE100
Input key : {53fc275d81053ed5be8cdaf29f59034938ae4efd}
True
```

需要注意的是，提交时候的flag是`53fc275d81053ed5be8cdaf29f59034938ae4efd`。 我一开始以为flag是`{53fc275d81053ed5be8cdaf29f59034938ae4efd}`，提交了一直显示我算错了emmm

ps：题目底下评论区好多都在说这个flag的格式的hhhh