# re学习笔记（6）BUUCTF-re-SimpleRev

原创

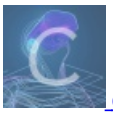
Forgo7ten 于 2019-11-12 19:46:10 发布 1663 收藏 3

分类专栏： ctf小白成长ing # reverse 文章标签： BUUCTF re SimpleRev CTF



本文链接：https://blog.csdn.net/Palmer9/article/details/103036953





ctf小白成长ing 同时被 2 个专栏收录

112 篇文章 6 订阅

订阅专栏

reverse

113 篇文章 6 订阅

订阅专栏

**新手一枚，如有错误（不足）请指正，谢谢！！**

题目链接：BUUCTF-re-SimpleRev

载入IDA_64

进入main函数查看

发现没关键代码，进入Decry()函数查看

IDA View-A | Pseudocode-A | Hex View-1 | Structures | Enums | Impor

```
1  int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
2  {
3    int v3; // eax
4    char v4; // [rsp+Fh] [rbp-1h]
5
6    while ( 1 )
7    {
8      while ( 1 )
9      {
10       printf("Welcome to CTF game!\nPlease input d/D to start or input q/Q to quit this program: ", argv, envp);
11       v4 = getchar();
12       if ( v4 != 'd' && v4 != 'D' )
13         break;                                  // 如果输入的不为d或D则推出本次循环
14       Decry();
15     }
16     if ( v4 == 'q' || v4 == 'Q' )
17       Exit();                                   // 如果输入为Q，q则退出
18                                                 // 若不为，则打印输出错误格式
19                                                 // 并进行下一次循环
20     puts("Input fault format!");
21     v3 = getchar();
22     putchar(v3);
23   }
24 }
```



```
unsigned __int64 Decry()
{
  char v1; // [rsp+Fh] [rbp-51h]
  int v2; // [rsp+10h] [rbp-50h]
  int v3; // [rsp+14h] [rbp-4Ch]
  int i; // [rsp+18h] [rbp-48h]
```

```
int v5; // [rsp+1Ch] [rbp-44h]
char src[8]; // [rsp+20h] [rbp-40h]
__int64 v7; // [rsp+28h] [rbp-38h]
int v8; // [rsp+30h] [rbp-30h]
__int64 v9; // [rsp+40h] [rbp-20h]
__int64 v10; // [rsp+48h] [rbp-18h]
int v11; // [rsp+50h] [rbp-10h]
unsigned __int64 v12; // [rsp+58h] [rbp-8h]

v12 = __readfsqword(0x28u);
*(_QWORD *)src = 'SLCDN';
v7 = 0LL;
v8 = 0;
v9 = 'wodah';
v10 = 0LL;
v11 = 0;
text = join(key3, (const char *)&v9);          // 让text等于key3+v9
                                               // key3 = "kills"
                                               // v9   = "hadow"     //因为小端序存储
                                               //
                                               // 则text = "killshadow"
strcpy(key, key1);                             // 将key1复制给key
                                               // key = "ADSFK"
strcat(key, src);                              // 将src处的字符串拼接到key后
                                               // key = "ADSFKNDCLS"
v2 = 0;
v3 = 0;
getchar();                                     // 获取输入（清空缓冲区？）
v5 = strlen(key);                              // v5 = key的长度   v5 = 10
for ( i = 0; i < v5; ++i )
{
  if ( key[v3 % v5] > 64 && key[v3 % v5] <= 90 )// if(key[v3]>64 && key[v3]<=90)
                                               // key[v3] = key[v3]+32
                                               // : 将大写字母转换成小写字母
    key[i] = key[v3 % v5] + 32;                // key = "adsfkndcls"
  ++v3;
}
printf("Please input your flag:", src);
while ( 1 )
{
  v1 = getchar();
  if ( v1 == '\n' )                            // 如果输入的为换行符，则退出
    break;
  if ( v1 == ' ' )
  {
    ++v2;                                      // 如果输入的为空格，则v2加一
  }
  else
  {
    if ( v1 <= 96 || v1 > 122 )                // 如果输入的v1不为小写字母
    {
      if ( v1 > 64 && v1 <= 90 )               // 如果v1为大写字母
        str2[v2] = (v1 - 39 - key[v3++ % v5] + 97) % 26 + 97;// 对str2[v2]进行处理（v2为0每次加1）
                                               // str1[v2] = (v1-key[v3]+58)%26 + 97
                                               // 变换后str1[v2]存放小写字母
    }
    else
    {                                          // 如果输入的值v1为小写字母
      str2[v2] = (v1 - 39 - key[v3++ % v5] + 97) % 26 + 97;// 做同样处理
```

```
    }                                       // 如果不为大小写字母，则不进行处理
    if ( !(v3 % v5) )                       // 如果循环到key的最后一位
      putchar(' ');                         // 打印处一个空格
    ++v2;
  }
}
if ( !strcmp(text, str2) )                  // 如果text和str2存储的相同，则成功
                                            // text = "killshadow"
  puts("Congratulation!\n");
else
  puts("Try again!\n");
return __readfsqword(0x28u) ^ v12;
}
```

其中还有一个自定义的join函数

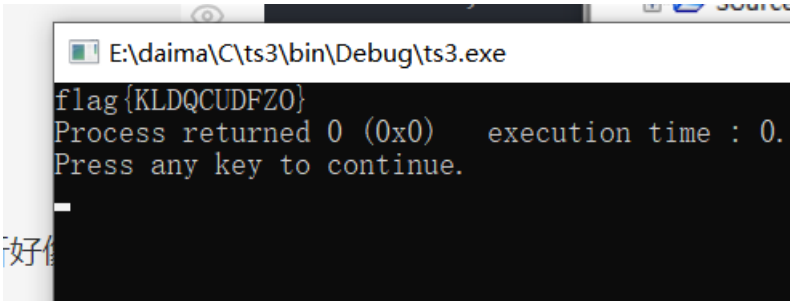
```
1  // 传入两个指针
2  // a1 = key3
3  // a2 = v9
4  char *__fastcall join(const char *a1, const char *a2)
5  {
6    size_t v2; // rbx
7    size_t v3; // rax
8    char *dest; // [rsp+18h] [rbp-18h]
9
10   v2 = strlen(a1);                       // v2 为key3的长度
11   v3 = strlen(a2);                       // v3为v9的长度
12   dest = (char *)malloc(v2 + v3 + 1);    // 动态分配一个能存下key3和v9的空间dest
13   if ( !dest )
14     exit(1);
15   strcpy(dest, a1);                      // 将a1赋值给dest
16   strcat(dest, a2);                      // 把a2拼接到dest后
17   return dest;                           // 也就是返回的是key3和v9的拼接
18 }
```



然后开始写脚本代码

```
#include <stdio.h>
int main(void)
{
    char text[] = "killshadow";
    char key[] = "adsfkndcls";
    int key_=strlen(key),text_ = strlen(text),i,j;
    char str2[15] = {0};
    for(i=0;i<text_;i++)
        for(j=0;;j++)
        {
            str2[i] = text[i] - 97 + 26*j + 39 -97 + key[i];
            if(str2[i]>64 && str2[i]<91)
                break;
        }
    printf("flag{%s}",str2);
    return 0;
}
```

其中对26取余的逆运算让我很懵，，，只好用这种笨方法了，，经过分析好像flag大小写都可？因为运算是一样的……
最后得出flag





## 往期回顾

小白学习笔记（0） CG-CTF-re-3 py交易
小白学习笔记（1） BUUCTF-re xor
小白学习笔记（2）BUUCTF-re-新年快乐
小白学习笔记（3） CG-CT re ReadAsm2
小白学习笔记（4）BUUCTF-re-reverse_1
小白学习笔记（5）BUUCTF-re-内涵软件