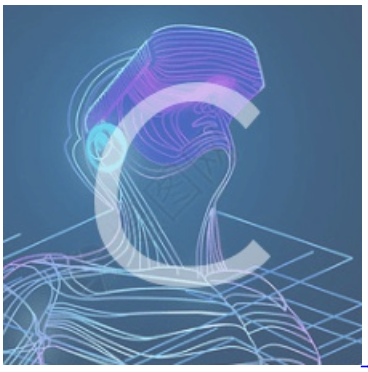
BUUCTF_SimpleRev

原创

ZYen12138 于 2020-10-20 15:44:34 发布 216 收藏

分类专栏： # BUUCTF CTF

本文链接：https://blog.csdn.net/weixin_46009088/article/details/109180960



BUUCTF 同时被 2 个专栏收录

17 篇文章 2 订阅

订阅专栏

CTF

17 篇文章 0 订阅

订阅专栏

BUUCTF_SimpleRev

同样，用IDA载入找到main函数，如图：

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  char v4; // [rsp+Fh] [rbp-1h]

  while ( 1 )
  {
    while ( 1 )
    {
      printf("Welcome to CTF game!\nPlease input d/D to start or input q/Q to quit this program: ", argv, envp);
      v4 = getchar();
      if ( v4 != 'd' && v4 != 'D' )
        break;
      Decry();
    }
    if ( v4 == 'q' || v4 == 'Q' )
      Exit();
    puts("Input fault format!");
    v3 = getchar();
    putchar(v3);
  }
}
```


浏览一下代码发现，v4读取一个输入如果是 d 或者 D 就进入**Decry**函数，如果是 q 或 Q 则退出程序，那么**Decry**就是个关键函数，我们跟进，如图：

```
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);          // killshadow
strcpy(key, key1);
strcat(key, src);                              // ADSFKNDCLS
v2 = 0;
v3 = 0;
getchar();
v5 = strlen(key);                              // 10
for ( i = 0; i < v5; ++i )                     // 大写字符转为小写字符
{
  if ( key[v3 % v5] > 64 && key[v3 % v5] <= 90 )
    key[i] = key[v3 % v5] + 32;
  ++v3;
}
printf("Please input your flag:", src);
while ( 1 )
{
  v1 = getchar();                              // 读取输入
  if ( v1 == '\n' )                            // 遇到换行符就break出去
    break;
  if ( v1 == ' ' )                             // 遇到空字符v2加1
  {
    ++v2;
  }
  else
  {
    if ( v1 <= 96 || v1 > 122 )
    {
      if ( v1 > 64 && v1 <= 90 )               // 如果是大写字符
        str2[v2] = (v1 - 39 - key[v3++ % v5] + 97) % 26 + 97;
    }
    else
    {
      str2[v2] = (v1 - 39 - key[v3++ % v5] + 'a') % 26 + 'a';
    }
    if ( !(v3 % v5) )
      putchar(' ');
    ++v2;
  }
}
```

```
  if ( !strcmp(text, str2) )
    puts("Congratulation!\n");
  else
    puts("Try again!\n");
  return __readfsqword(0x28u) ^ v12;
}
```

发现两个关键的字符串处理：

1.

```
      text = join(key3, (const char *)&v9);         // killshadow
```

2.

```
  strcpy(key, key1);
  strcat(key, src);                             // ADSFKNDCLS
```

先看text这个字符串，点进join查看，如图：

```
 1 char *__fastcall join(const char *a1, const char *a2)
 2 {
 3   size_t v2; // rbx
 4   size_t v3; // rax
 5   char *dest; // [rsp+18h] [rbp-18h]
 6
 7   v2 = strlen(a1);                              // 5
 8   v3 = strlen(a2);                              // 5
 9   dest = (char *)malloc(v2 + v3 + 1);           // 11
10   if ( !dest )
11     exit(1);
12   strcpy(dest, a1);
13   strcat(dest, a2);
14   return dest;
15 }
```


这就是简单的将 a1 和 a2 拼接，回到 Decry 查看传入的参数一个是 key3 ，我们点进去看是 *kills* ,另一个是v9，**这里要注意了！！！** IDA扫描出来的字符串是反序输出的，所以字符串v9是*hadow*，而不是 wodah ，将两个拼接就得到了 killshadow 这个字符串，key这个字符串同理可得ADSFKNDCLS，往下看，发现一个把字符串转为小写的语句，那么 key 就变成了 adsfkndcls .

```
      for ( i = 0; i < v5; ++i )                    // 大写字符转为小写字符
      {
        if ( key[v3 % v5] > 64 && key[v3 % v5] <= 90 )
          key[i] = key[v3 % v5] + 32;
        ++v3;
      }
```

再往下看就是一个改变输入的字符串的语句，完事后，就用改变后的字符和 text 进行比较，最后如果正确就输出 Congratulation！，如图：

```
while ( 1 )
{
  v1 = getchar();                               // 读取输入
  if ( v1 == '\n' )                             // 遇到换行符就break出去
    break;
  if ( v1 == ' ' )                              // 遇到空字符v2加1
  {
    ++v2;
  }
  else
  {
    if ( v1 <= 96 || v1 > 122 )                 // 感觉在混淆视听
    {
      if ( v1 > 64 && v1 <= 90 )                // 如果是大写字符
        str2[v2] = (v1 - 39 - key[v3++ % v5] + 97) % 26 + 97;
    }
    else
    {
      str2[v2] = (v1 - 39 - key[v3++ % v5] + 'a') % 26 + 'a';
    }
    if ( !(v3 % v5) )
      putchar(' ');
    ++v2;
  }
}
if ( !strcmp(text, str2) )
  puts("Congratulation!\n");
```


那我们只能通过爆破进行算法逆向，最终得到字符串，(v1我们真的不知道是啥)，用python写出脚本,如下：

```python
key = 'adsfkndcls'
text = 'killshadow'
v5 = len(text)
flag = ''
for i in range(len(text)):
      for j in range(65,91): #仅遍历大写字母
         if ord(text[i]) == (j - 39 - ord(key[i % v5]) + 97) % 26 + 97:
             flag += chr(j)

print(flag)
```

运行得到结果！！！成功了！！！

```
KLDQCUDFZO
```