REVERSE-PRACTICE-BUUCTF-7

原创

P1umH0 于 2021-02-26 23:04:17 发布 55 收藏 1

分类专栏： Reverse-BUUCTF

本文链接：https://blog.csdn.net/weixin_45582916/article/details/114155806

Reverse-BUUCTF 专栏收录该内容

32 篇文章 3 订阅

订阅专栏

REVERSE-PRACTICE-BUUCTF-7

Youngter-drive

exe程序，运行后提示输入flag，有upx壳，脱壳后ida分析

main函数中获取输入并拷贝，开启了两个线程分别运行StartAddress和sub_41119F两个函数，sub_411190函数验证输入，输入即为flag

```
int main_0()
{
  HANDLE v1; // [esp+D0h] [ebp-14h]
  HANDLE hObject; // [esp+DCh] [ebp-8h]

  sub_4110FF();                                 // 获取输入
  ::hObject = CreateMutexW(0, 0, 0);
  j_strcpy(input_copy, input);                  // 拷贝输入
  hObject = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)StartAddress, 0, 0, 0);
  v1 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_41119F, 0, 0, 0);
  CloseHandle(hObject);
  CloseHandle(v1);
  while ( input_index != -1 )
    ;
  sub_411190();                                 // check
  CloseHandle(::hObject);
  return 0;
}
```

分析StartAddress函数，input_index的初始值为29，当input_index的值大于-1时，调用sub_41112C函数，然后input_index值减1，休眠100ms

```
IDA View-A | Pseudocode-A | Stack of _main_0 | Hex View-1
1 void __stdcall StartAddress_0(int a1)
2 {
3   while ( 1 )
4   {
5     WaitForSingleObject(hObject, 0xFFFFFFFF);
6     if ( input_index > -1 )                    // input_index初始值等于29
7     {
8       sub_41112C((int)input, input_index);
9       --input_index;
10      Sleep(100u);
11    }
12    ReleaseMutex(hObject);
13  }
14 }
```


分析sub_41112C函数（反编译该函数需要先平衡栈），首先验证输入均为英文字母，然后对大小写区分，输入的内容的ascii码减去38或96的结果，作为下标，在table数组里取值，再赋给原来的位置，即对flag内容进行变换，实际上是大小写转换

```
1  char *__cdecl sub_411940(int input, int input_index)
2  {
3    char *result; // eax
4    char v3; // [esp+D3h] [ebp-5h]
5
6    v3 = *(_BYTE *)(input_index + input);
7    if ( (v3 < 97 || v3 > 122) && (v3 < 65 || v3 > 90) )// 保证输入是英文字母
8      exit(0);
9    if ( v3 < 97 || v3 > 122 )                 // 输入是大写字母
10   {
11     result = table[0];
12     *(_BYTE *)(input_index + input) = table[0][*(char *)(input_index + input) - 38];
13   }
14   else                                       // 输入是小写字母
15   {
16     result = table[0];
17     *(_BYTE *)(input_index + input) = table[0][*(char *)(input_index + input) - 96];
18   }
19   return result;
20 }
```


回到main函数，再分析sub_41119F函数
该函数先休眠100ms，再对input_index减1，没有对flag内容的变换

```
IDA View-A | Pseudocode-A | Stack of _mai
1  void __stdcall sub_411B10(int a1)
2  {
3    while ( 1 )
4    {
5      WaitForSingleObject(hObject, 0xFFFFFFFF);
6      if ( input_index > -1 )
7      {
8        Sleep(0x64u);
9        --input_index;
10     }
11     ReleaseMutex(hObject);
12   }
```

```
}
```


于是可以知道，input_index初始值为29，在第一个线程作为下标对flag内容变换后减1，在第二个线程直接减1，相当于减2，值为27，再回到第一个线程……于是第一个线程仅在input_index为奇数时对flag内容进行变换
写逆脚本，由于input_index初始值为29，说明flag的内容长度为30，但是在check（sub_411190）函数中，只比较了29个字符，试出来最后一位是“E”提交成功

```
res="T0iZiZt0rYaToUwPnToBs0a0apsyS"
table="QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm"
flag=""
for i in range(len(res)):
    if i%2==1:
        if res[i].isupper():
            flag+=chr(table.find(res[i])+96)
        else:
            flag+=chr(table.find(res[i])+38)
    else:
        flag+=res[i]
print(flag)
```

```
test1
D:\python27-x64\python2.exe D:/Python/pycharm/pycfile/test1.py
ThisisthreadofwindowshahaIsES
```


[ACTF新生赛2020]rome

exe程序，运行后提示输入，输入错误直接退出，无壳，ida分析
主逻辑在func函数中
第一个红框是对flag的内容变换，分大小写，原来是大写字母的变换后仍然是大写字母，原来是小写字母的变换后仍然是小写字母，如果flag的内容中某个字符为其他字符，则不进行变换
第二个红框是循环比较，验证flag的内容

```
scanf("%s", &input);
result = input;
if ( input == 'A' )
{
  result = v6;
  if ( v6 == 'C' )
  {
    result = v7;
    if ( v7 == 'T' )
    {
      result = v8;
      if ( v8 == 'F' )
      {
        result = v9;
        if ( v9 == '{' )
        {
          result = v14;
          if ( v14 == '}' )
          {
            v1 = v10;
            v2 = v11;
            v3 = v12;
            v4 = v13;
            for ( i = 0; i <= 15; ++i )
            {
```

```
    if ( *((_BYTE *)&v1 + i) > 64 && *((_BYTE *)&v1 + i) <= 90 )// 大写字母
      *((_BYTE *)&v1 + i) = (*((char *)&v1 + i) - 51) % 26 + 65;
    if ( *((_BYTE *)&v1 + i) > 96 && *((_BYTE *)&v1 + i) <= 122 )// 小写字母
      *((_BYTE *)&v1 + i) = (*((char *)&v1 + i) - 79) % 26 + 97;
  }
  for ( i = 0; i <= 15; ++i )       // check
  {
    result = (unsigned __int8)*(&v15 + i);
    if ( *((_BYTE *)&v1 + i) != (_BYTE)result )
      return result;
  }
```


写脚本即可得到flag

```
res=[81,115,119,51,115,106,95,108,122,52,95,85,106,119,64,108]
upper_table="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
lower_table="abcdefghijklmnopqrstuvwxyz"
flag=""
for i in range(len(res)):
    if chr(res[i]).isupper():
        for c in upper_table:
            if res[i]==(ord(c)- 51) % 26 + 65:
                flag+=c
    elif chr(res[i]).islower():
        for c in lower_table:
            if res[i]==(ord(c)- 79) % 26 + 97:
                flag+=c
    else:
        flag+=chr(res[i])
print(flag)
```

test1

```
D:\python27-x64\python2.exe D:/Python/pycharm/pycfile/test1.py
Cae3ar_th4_Gre@t
```


[FlareOn4]login

html文件，打开后提示输入flag并点击验证，右键->查看网页源代码
逻辑清晰，获取输入，输入变换，验证输入
重要的是flag.replace部分的内容
首先判断flag内容中某个字符是大写还是小写字母，如果是大写字母，>=左边取90，如果是小写字母，>=左边取122，再判断该字符在26个字母表中的位置，如果该字符在字母表的前半部分，则替换为字母表后半部分对应位置的字符，同理，如果该字符在字母表的后半部分，则替换为字母表后半部分对应位置的字符，字符的大小写不变

```
<!DOCTYPE Html />
<html>
    <head>
        <title>FLARE On 2017</title>
    </head>
    <body>
        <input type="text" name="flag" id="flag" value="Enter the flag" />
        <input type="button" id="prompt" value="Click to check the flag" />
        <script type="text/javascript">
            document.getElementById("prompt").onclick = function () {
                var flag = document.getElementById("flag").value;
                var rotFlag = flag.replace(/[a-zA-Z]/g, function(c){return String.fromCharCode((c <= "Z" ? 90 : 122) >= (c = c.charCodeAt(0) + 13) ? c : c - 26);});
                if ("PyvragFvqrYbtvafNerRnfl@syner-ba.pbz" == rotFlag) {
                    alert("Correct flag!");
                } else {
                    alert("Incorrect flag, rot again");
                }
            }
        </script>
    </body>
</html>
```


写逆脚本即可得到flag

```
res="PyvragFvqrYbtvafNerRnfl@syner-ba.pbz"
flag=""
for i in range(len(res)):
    if res[i].isalnum():
        if res[i]>='A' and res[i] <='M' or res[i]>='a' and res[i] <='m':
            flag+=chr(ord(res[i])+13)
        elif res[i]>='N' and res[i] <='Z' or res[i]>='n' and res[i] <='z':
            flag += chr(ord(res[i]) - 13)
    else:
        flag+=res[i]
print(flag)
```

```
test1
D:\python27-x64\python2.exe D:/Python/pycharm/pycfile/test1.py
ClientSideLoginsAreEasy@flare-on.com
```


[SUCTF2019]SignIn

elf文件，无壳，ida分析
main函数逻辑清晰，先获取输入，然后调用sub_96A函数分割输入，例如输入字符串的第一个字符为“f”，其十六进制ascii码为0x66，经过sub_96A函数，结果存储到v9，有v9[0]=0x6，v9[1]=0x6，然后v9数组转成整形作为RSA的明文m，接下来就是RSA加密和验证

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char n; // [rsp+0h] [rbp-4A0h]
  char e; // [rsp+10h] [rbp-490h]
  char m; // [rsp+20h] [rbp-480h]
  char cipher; // [rsp+30h] [rbp-470h]
  char input; // [rsp+40h] [rbp-460h]
  char v9; // [rsp+B0h] [rbp-3F0h]
  unsigned __int64 v10; // [rsp+498h] [rbp-8h]

```

```
11  v10 = __readfsqword(0x28u);
12  puts("[sign in]");
13  printf("[input your flag]: ", a2);
14  __isoc99_scanf("%99s", &input);
15  sub_96A(&input, (__int64)&v9);                // split input
16  __gmpz_init_set_str(
17    (__int64)&cipher,
18    (__int64)"ad939ff59f6e70bcbfad406f2494993757eee98b91bc244184a377520d06fc35",
19    16LL);
20  __gmpz_init_set_str((__int64)&m, (__int64)&v9, 16LL);
21  __gmpz_init_set_str(
22    (__int64)&n,
23    (__int64)"10346103590081691412139010129904904441395040517371217043416168653987816098454 9",
24    10LL);
25  __gmpz_init_set_str((__int64)&e, (__int64)"65537", 10LL);
26  __gmpz_powm(&m, &m, &e, &n);
27  if ( (unsigned int)__gmpz_cmp(&m, &cipher) )
28    puts("GG!");
29  else
30    puts("TTTTTTTTTql!");
31  return 0LL;
32}
```


在线网站或者yafu分解模数n

写脚本即可得到flag

```python
import gmpy2
from Crypto.Util.number import long_to_bytes
cipher=int('ad939ff59f6e70bcbfad406f2494993757eee98b91bc244184a377520d06fc35',16)
n=10346103590081691412139010129904904441395040517371217043416168653987816098454 9
e=65537
p=3666691020029668568766056698370142294 19
q=2821645874595121248442451139505933482 71
phin=(p-1)*(q-1)
d=gmpy2.invert(e,phin)
m=gmpy2.powmod(cipher,d,n)
print(long_to_bytes(m))
```

```
test1
D:\python27-x64\python2.exe D:/Python/pycharm/pycfile/test1.py
suctf{Pwn_@_hundred_years}

进程已结束,退出代码0
```