Jarvis OJ-Reverse题目Writeup

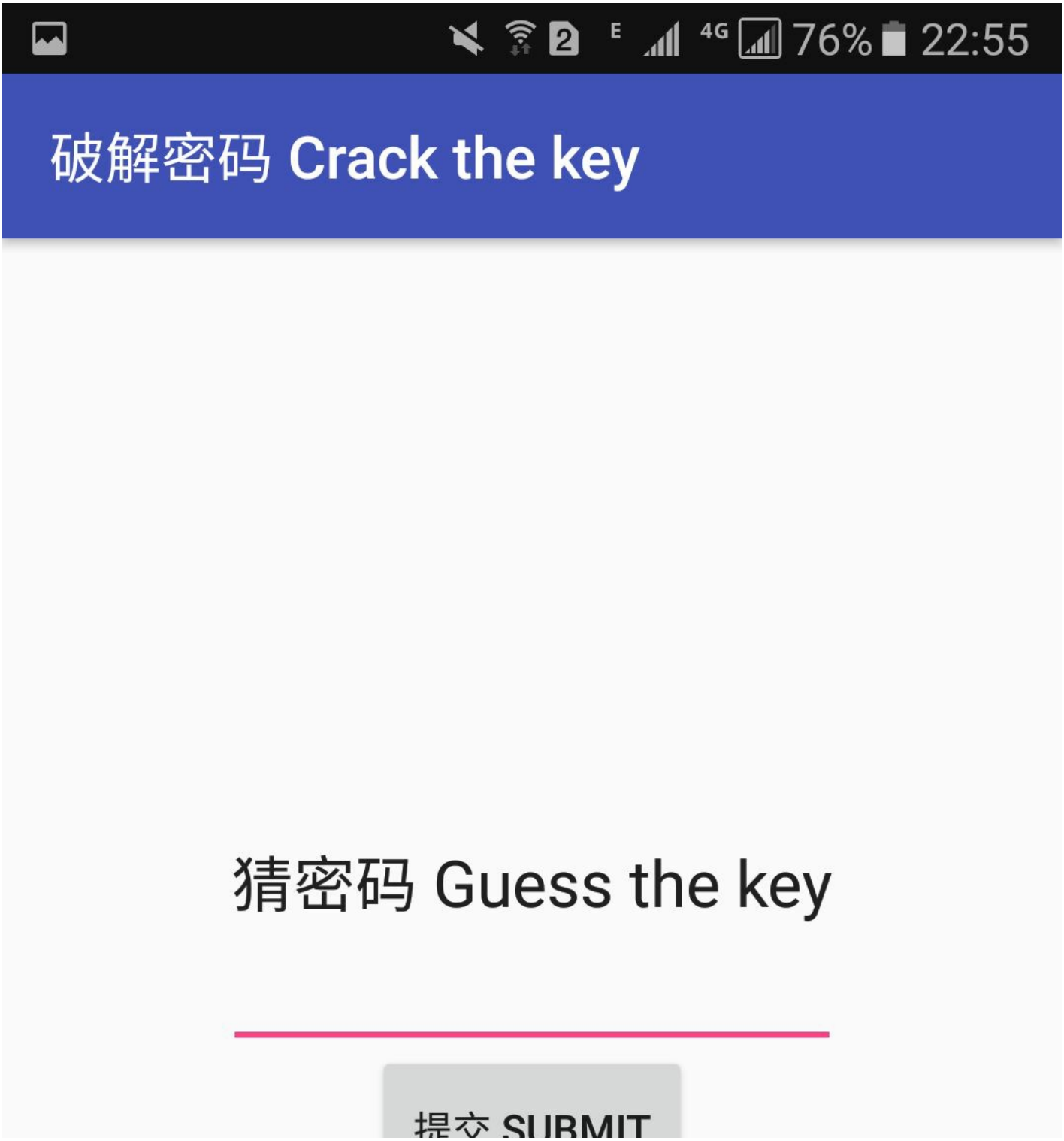
转载

weixin_30611509 于 2019-01-31 23:03:00 发布 88 收藏
文章标签： python java c/c++
原文链接：http://www.cnblogs.com/Briddle-ch/p/10344416.html

做一道更一道吧233333

DD-Android Easy

下载apk，先安装一下试试吧……



猜测是输入正确的内容后给flag吧

将后缀改成zip，解压，用dex2jar处理classes.dex，然后用jd-gui打开，可以看到，该apk中只有一个FlagActivity，一部分一部分来看

```
protected void onCreate(Bundle paramBundle)
  {
    super.onCreate(paramBundle);
    setContentView(2130968602);
    this.n = ((TextView)findViewById(2131427413));
    this.o = ((TextView)findViewById(2131427415));
  }
```

onCreate()等同于什么都没做

```
public void onClickTest(View paramView)
  {
    if (this.n.getText().toString().equals(i()))
    {
      this.o.setText(2131099685);
      return;
    }
    this.o.setText(2131099683);
  }
```

onClickTest()函数的功能非常简单：n指代的应该是输入框 ，o指代的应该是结果提示框，那么这个函数做的事情就是把输入的内容和i()函数的返回值比对，那大概是一样才行吧

所以最后我们来看这个i()函数

```
private static final byte[] p = { -40, -62, 107, 66, -126, 103, -56, 77, 122, -107, -24, -127, 72, -63, -98, 64, -24, -5, -49, -26, 79, -70, -26, -81, 120, 25, 111, -100, -23, -9, 122, -35, 66, -50, -116, 3, -72, 102, -45, -85, 0, 126, -34, 62, 83, -34, 48, -111, 61, -9, -51, 114, 20, 81, -126, -18, 27, -115, -76, -116, -48, -118, -10, -102, -106, 113, -104, 98, -109, 74, 48, 47, -100, -88, 121, 22, -63, -32, -20, -41, -27, -20, -118, 100, -76, 70, -49, -39, -27, -106, -13, -108, 115, -87, -1, -22, -53, 21, -100, 124, -95, -40, 62, -69, 29, 56, -53, 85, -48, 25, 37, -78, 11, -110, -24, -120, -82, 6, -94, -101 };
private static final byte[] q = { -57, -90, 53, -71, -117, 98, 62, 98, 101, -96, 36, 110, 77, -83, -121, 2, -48, 94, -106, -56, -49, -80, -1, 83, 75, 66, -44, 74, 2, -36, -42, -103, 6, -115, -40, 69, -107, 85, -78, -49, 54, 78, -26, 15, 98, -70, 8, -90, 94, -61, -84, 64, 112, 51, -29, -34, 126, -21, -126, -71, -31, -24, -60, -2, -81, 66, -84, 85, -91, 10, 84, 70, -8, -63, 26, 126, -76, -104, -123, -71, -126, -62, -23, 11, -39, 70, 14, 59, -101, -39, -124, 91, -109, 102, -49, 21, 105, 0, 37, -128, -57, 117, 110, -115, -86, 56, 25, -46, -55, 7, -125, 109, 76, 104, -15, 82, -53, 18, -28, -24 };

private String i()
  {
    int j = 0;
    byte[] arrayOfByte1 = new byte[p.length];
    int i = 0;
    while (i < arrayOfByte1.length)
    {
      arrayOfByte1[i] = ((byte)(p[i] ^ q[i]));
      i += 1;
    }
    int k = arrayOfByte1[0];
    i = 0;
    while (arrayOfByte1[(k + i)] != 0)
      i += 1;
    byte[] arrayOfByte2 = new byte[i];
    while (j < i)
    {
      arrayOfByte2[j] = arrayOfByte1[(k + j)];
      j += 1;
    }
    return new String(arrayOfByte2);
  }
```

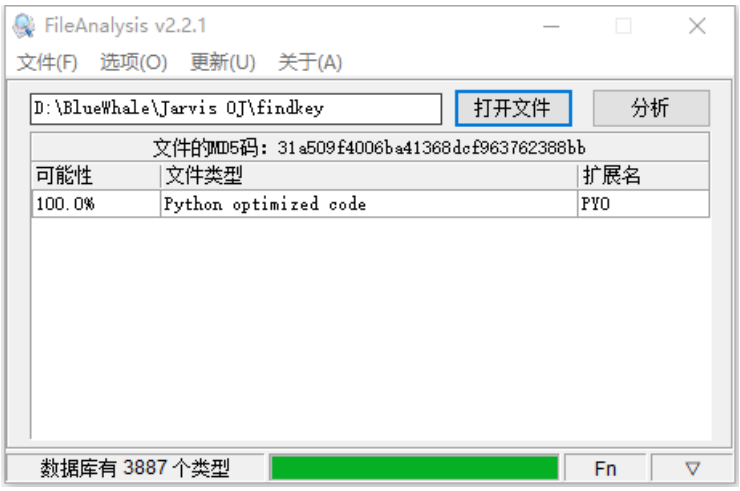
可以看到这个函数不要输入，且输出结果固定……好的果然是AndroidEasy……写个Java跑一下i()函数的结果就可以了……

```
public class DDCTF_EASY {
  private static final byte[] p = { -40, -62, 107, 66, -126, 103, -56, 77, 122, -107, -24, -127, 72, -63,
-98, 64, -24, -5, -49, -26, 79, -70, -26, -81, 120, 25, 111, -100, -23, -9, 122, -35, 66, -50, -116, 3, -72,
102, -45, -85, 0, 126, -34, 62, 83, -34, 48, -111, 61, -9, -51, 114, 20, 81, -126, -18, 27, -115, -76, -116,
-48, -118, -10, -102, -106, 113, -104, 98, -109, 74, 48, 47, -100, -88, 121, 22, -63, -32, -20, -41, -27, -
20, -118, 100, -76, 70, -49, -39, -27, -106, -13, -108, 115, -87, -1, -22, -53, 21, -100, 124, -95, -40, 62,
-69, 29, 56, -53, 85, -48, 25, 37, -78, 11, -110, -24, -120, -82, 6, -94, -101 };
  private static final byte[] q = { -57, -90, 53, -71, -117, 98, 62, 98, 101, -96, 36, 110, 77, -83, -
121, 2, -48, 94, -106, -56, -49, -80, -1, 83, 75, 66, -44, 74, 2, -36, -42, -103, 6, -115, -40, 69, -107,
85, -78, -49, 54, 78, -26, 15, 98, -70, 8, -90, 94, -61, -84, 64, 112, 51, -29, -34, 126, -21, -126, -71, -
31, -24, -60, -2, -81, 66, -84, 85, -91, 10, 84, 70, -8, -63, 26, 126, -76, -104, -123, -71, -126, -62, -23,
11, -39, 70, 14, 59, -101, -39, -124, 91, -109, 102, -49, 21, 105, 0, 37, -128, -57, 117, 110, -115, -86,
56, 25, -46, -55, 7, -125, 109, 76, 104, -15, 82, -53, 18, -28, -24 };

  private String i()
  {
    int j = 0;
    byte[] arrayOfByte1 = new byte[p.length];
    int i = 0;
    while (i < arrayOfByte1.length)
    {
      arrayOfByte1[i] = ((byte)(p[i] ^ q[i]));
      i += 1;
    }
    int k = arrayOfByte1[0];
    i = 0;
    while (arrayOfByte1[(k + i)] != 0)
      i += 1;
    byte[] arrayOfByte2 = new byte[i];
    while (j < i)
    {
      arrayOfByte2[j] = arrayOfByte1[(k + j)];
      j += 1;
    }
    return new String(arrayOfByte2);
  }


    public static void main(String args[]) {
        DDCTF_EASY t = new DDCTF_EASY();
        System.out.println(t.i());
    }
}
```

输出结果（即为本题flag）：DDCTF-3ad60811d87c4a2dba0ef651b2d93476@didichuxing.com

FindKey

把文件下载下来，发现是个没有后缀的东西，看在它是Reverse题目的份上，用IDA打开它……然后IDA告诉我这是一个binary file，于是用FileAnalysis尝试着查一下这是个什么文件



这还是头一次看到一个可能性100%的哈哈哈哈，那它就一定是一个pyo文件了哈哈，查了一番得知pyo和pyc都是py编译出来的中间产物，只不过pyo是pyc的编译优化版本，这道题的flag是我们输入的key，那我们必然要查看它的源代码，查阅一番找到了一个软件Easy Python Decompiler，下载地址：https://sourceforge.net/projects/easypythondecompiler/

使用这个软件逆向之后查看这个python的源代码（省略掉了在这些代码之前的两百多行的数组定义）

```
flag = raw_input('Input your Key:').strip()
if len(flag) != 17:
    print 'Wrong Key!!'
    sys.exit(1)
flag = flag[::-1]
for i in range(0, len(flag)):
    if ord(flag[i]) + pwda[i] & 255 != lookup[i + pwdb[i]]:
        print 'Wrong Key!!'
        sys.exit(1)

print 'Congratulations!!'
```

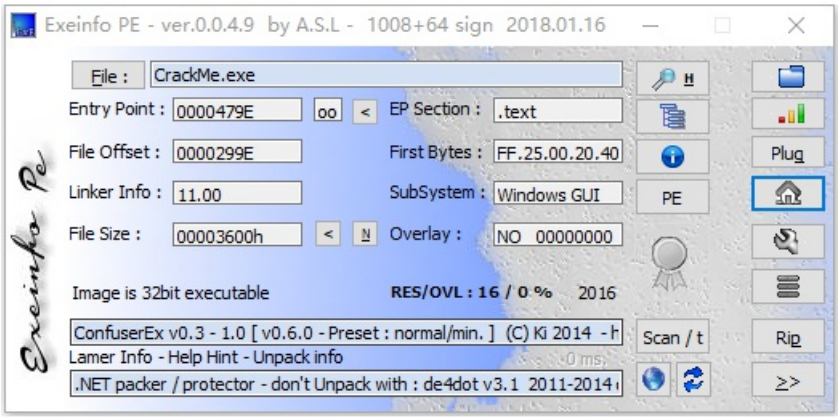
好像也没什么逆向的难度……flag是一个长度为17的字符串，倒序之后每一位的值依次等于lookup[i+pwdb[i]]-pwda[i]&255，写个脚本吧（前面两百多行的数组定义需要复制过来）

```
1 flag = []
2 for i in range(0, 17):
3     flag.append(chr(lookup[i + pwdb[i]] - pwda[i] & 255))
4 flag = flag[::-1]
5 print(''.join(flag))
```

跑一下这个py脚本得到flag：PCTF{PyC_Cr4ck3r}

Classical Crackme

把exe下下来用exeinfope跑一下，得到这样的结果



得到的结论是这是一个加了壳的.net的文件，提示用de4dot来去壳，工具下载地址：https://ci.appveyor.com/project/0xd4d/de4dot/build/artifacts （是那个net35那个压缩包）

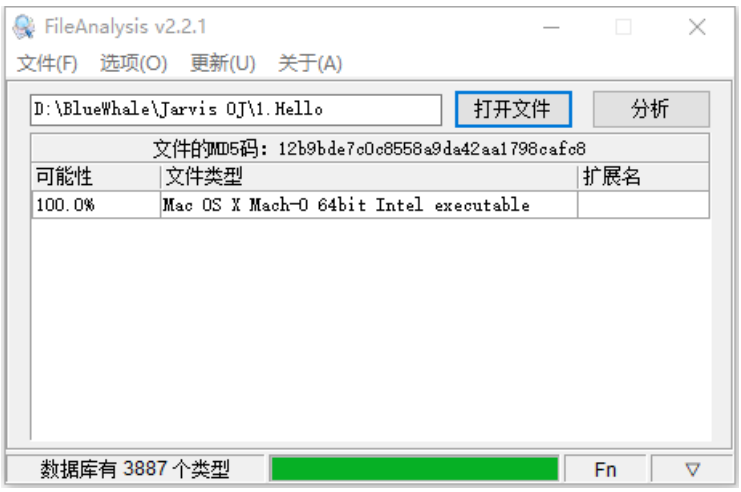
用de4dot去壳，得到了一个CrackMe-cleaned.exe，用Reflector来查看它，观察源代码（主要是button的点击事件）

```
private void button1_Click(object sender, EventArgs e)
{
    string s = this.textBox1.Text.ToString();
    string str2 = Convert.ToBase64String(Encoding.Default.GetBytes(s));
    string str3 = "UENURntFYTV5X0RvX05ldF9DcjRjazNyfQ==";
    if (str2 == str3)
    {
        MessageBox.Show("注册成功！", "提示", MessageBoxButtons.OK);
    }
    else
    {
        MessageBox.Show("注册失败！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Hand);
    }
}
```

只要我们输入的注册码经过base64转码之后和str3相等就可以了，把str3用base64解码一下，就得到了此题flag：PCTF{Ea5y_Do_Net_Cr4ck3r}

DD-Hello

下载下来发现是个不明后缀的文件，用FileAnalysis跑一下



这是一个Mac的应用程序，好像在工具选择上就只能选择IDA了，用IDA来查看这个文件，可以看到start和sub_100000C90两个函数没什么实质含义，但是sub_100000CE0函数有一些可以深挖的内容，该函数对应的IDA代码如下

```
int sub_100000CE0()
{
  int result; // eax
  signed int v1; // [rsp+1Ch] [rbp-14h]
  int v2; // [rsp+24h] [rbp-Ch]

  v2 = ((unsigned __int64)((char *)start - (char *)sub_100000C90) >> 2) ^ byte_100001040[0];
  result = sub_100000DE0();
  if ( !(result & 1) )
  {
    v1 = 0;
    while ( v1 < 55 )
    {
      byte_100001040[v1] -= 2;
      byte_100001040[v1] ^= v2;
      ++v1;
      ++v2;
    }
    result = printf("\nFinal output is %s\n", &byte_100001040[1]);
  }
  return result;
}
```

第一次赋值的result应该是没有实质作用的，不参与后面运算，那么接下来要解决的问题就是v2的初始值是多少？注意这里的start和sub_100000C90的后面都是没有带括号的，换句话说，这两者都不是函数的返回值，而是函数的地址，在IDA中查阅两者的地址

Function name	Segment	Start	Length	Locals	Arguments	R	F	L	S	B	T	=
sub_100000C90	__text	0000000100000C90	0000001F	00000018	00000000	R	.	.	.	B	.	.
start	__text	0000000100000CB0	0000002D	00000018	00000000	R	.	.	.	B	.	.
sub_100000CE0	__text	0000000100000CE0	000000FD	00000038	00000000	R	.	.	.	B	.	.
sub_100000DE0	__text	0000000100000DE0	0000011B	000002D8	00000000	R	.	.	.	B	.	.

得知两个函数的地址的差值为0xCE0-0xC90=32，其余数据都是固定的（还需要去IDA中查一下byte_100001040数组元素的值），写一个c语言跑一下就行（不会使用IDA调试），代码如下：

```
#include <stdio.h>

int main() {
    char c[57] =
{0x41,0x10,0x11,0x11,0x1B,0x0A,0x64,0x67,0x6A,0x68,0x62,0x68,0x6E,0x67,0x68,0x6B,0x62,0x3D,0x65,0x6A,0x6A,0x3D,0x68,4,5,8,3,2,2,0x55,8,0x5D,

0x61,0x55,0x0A,0x5F,0x0D,0x5D,0x61,0x32,0x17,0x1D,0x19,0x1F,0x18,0x20,4,2,0x12,0x16,0x1E,0x54,0x20,0x13,0x14,0,0};
    int v1 = 0;
    int v2 = ((unsigned)(32) >> 2) ^ c[0];
    while ( v1 < 55 )
    {
      c[v1] -= 2;
      c[v1] ^= v2;
      ++v1;
      ++v2;
    }
    int result = printf("\nFinal output is %s\n", &c[1]);
    return result;
}
```

运行，得到的结果为Final output is DDCTF-5943293119a845e9bbdbde5a369c1f50@didichuxing.com，后半段内容为该题flag

Baby's Crack（Basic）

用cmd命令行跑一下程序，可以得知是一个加密程序，那么我们的任务应该就是把flag.enc解密

exe下载下来用exeinfope跑一下，发现没有什么特殊的，是个64位的软件，用IDA打开，找到main函数，main函数中这一段是对明文字符串做处理的

```
while ( feof(File) == 0 )
    {
      v7 = fgetc(File);
      if ( v7 != -1 && v7 )
      {
        if ( v7 > 47 && v7 <= 96 )
        {
          v7 += 53;
        }
        else if ( v7 <= 46 )
        {
          v7 += v7 % 11;
        }
        else
        {
          v7 = 61 * (v7 / 61);
        }
        fputc(v7, v5);
      }
    }
```

分类讨论一下各部分的取值

v7≤46　　v7 += v7%11　　result≤56

47<v7≤96　　v7 += 53　　100<result≤149

v7>96　　61*(v7/61)　　result=61 or 122（受到char的最大大小限制，受到v7>96的条件限制）

看一下flag.enc文件中的内容jeihjiiklwjnk{ljj{kflghhj{ilk{k{kij{ihlgkfkhkwhhjgly，根据上面的讨论，可以看出，flag.enc中所有的内容都是经历了+53得到的，那就……好办了

```
#include <stdio.h>
#include <iostream>
#include <string.h>
#include <string>
using namespace std;
int main() {
    char a[] = "jeihjiiklwjnk{ljj{kflghhj{ilk{k{kij{ihlgkfkhkwhhjgly";
    int length = strlen(a);
    for(int i = 0; i < length; i++) {
        printf("%c", a[i]-53);
    }
    printf("\n%d", length);
    return 0;
}
```

解出来得到的原文内容是504354467B596F755F6172335F476F6F645F437261636B33527D，共计52位

把这一段十六进制转字符串，得到本题flag：PCTF{You_ar3_Good_Crack3R}

Easy Crackme（Basic）

文件下载下来没有后缀，FileAnalysis跑一下，是个ELF文件，用IDA打开，找到主函数

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rdi
  char v5; // [rsp+0h] [rbp-38h]
  char v6; // [rsp+1h] [rbp-37h]
  char v7; // [rsp+2h] [rbp-36h]
  char v8; // [rsp+3h] [rbp-35h]
  char v9; // [rsp+4h] [rbp-34h]
  char v10; // [rsp+5h] [rbp-33h]
  unsigned __int8 v11; // [rsp+10h] [rbp-28h]
  _BYTE v12[7]; // [rsp+11h] [rbp-27h]

  v5 = -85;
  v6 = -35;
  v7 = 51;
  v8 = 84;
  v9 = 53;
  v10 = -17;
  printf((unsigned __int64)"Input your password:");
  _isoc99_scanf((unsigned __int64)"%s");
  if ( strlen((const char *)&v11) == 26 )
  {
    v3 = 0LL;
    if ( (v11 ^ 0xAB) == list1 )
    {
      while ( (v12[v3] ^ (unsigned __int8)*(&v5 + ((signed int)v3 + 1) % 6)) == a[v3] )
      {
        if ( ++v3 == 25 )
        {
          printf((unsigned __int64)"Congratulations!");
          return 0;
        }
      }
    }
  }
  printf((unsigned __int64)"Password Wrong!! Please try again.");
  return 0;
}
```

分析该段程序和栈内存，可以知道我们输入字符串的长度应该为26，且第一位为大写字母P，之后的25位满足一个异或关系式，逐位暴力v12即可，脚本如下

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main() {
    int v3, t;
    int a[31] =
{0x9E,0x67,0x12,0x4E,0x9D,0x98,0xAB,0,6,0x46,0x8A,0xF4,0xB4,6,0x0B,0x43,0xDC,0xD9,0xA4,0x6C,0x31,0x74,0x9C,0
xD2,0xA0,0,0,0,0,0,0};
    int v5[6] = {-85,-35,51,84,53,-17};
    for(v3 = 0; v3 < 25; v3++) {
        int temp = v5[(v3+1)%6];
        printf("a xor t =%d %c\n", (temp^a[v3]), (temp^a[v3]));
        for(t = 0; t < 127; t++) {
            if(t==(temp^a[v3])) {
                printf("%c", t);
            }
        }
    }
    return 0;
}
```

问题在于……结果并不对……有很多v3的值在整个将t从0遍历到127的过程中都没能找到符合if条件的值，百思不得其解，在和某thomount讨论之后，进行了一番尝试，最终找到了问题所在，正确的解题脚本如下：

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main() {
    int v3, t;
    int a[31] =
{0x9E,0x67,0x12,0x4E,0x9D,0x98,0xAB,0,6,0x46,0x8A,0xF4,0xB4,6,0x0B,0x43,0xDC,0xD9,0xA4,0x6C,0x31,0x74,0x9C,0
xD2,0xA0,0,0,0,0,0,0};
    int v5[6] = {-85,-35,51,84,53,-17};
    for(v3 = 0; v3 < 25; v3++) {
        int temp = v5[(v3+1)%6];
        printf("%c", temp ^ a[v3]);
    }
    return 0;
}
```

采用直接异或的方式来求值，最后得到的字符串如下：CTF{r3v3Rse_i5_v3ry_eAsy}，前面再拼上P，得到本题flag：PCTF{r3v3Rse_i5_v3ry_eAsy}

然后再来解释为什么第一个程序得不到正确的结果：

我们已知b,c两者的值，且a按照%c输出能得到一个字符，还知道a^b=c，那么a的值当然可以用b^c来求

但如果你遍历0到127（ascii码的范围），去测试何时a^b=c，是可能得不到正确的值，原因在于a本身可以是一个很大很大的数字（比如说127+256n），还可以是负数（比如说-51），但是这样的数依然可以用%c输出出一个字符来（输出的结果为该数字二进制的最后7位所代表的的真值）……所以面对这样的异或方程，正确的解法只有用b^c，而不能正向去枚举来寻找a的值，因为无法界定a的取值范围是多少。

举一个例子

int a = -205;　　printf("%c", a);　　这样的语句输出结果会是字符3，原因是-205的二进制表示最后8位为00110011，这样的数按照一个8位整数来读是51，而51对应的ascii码的字符为‘3’，所以输出结果是一个字符3

转载于:https://www.cnblogs.com/Briddle-ch/p/10344416.html