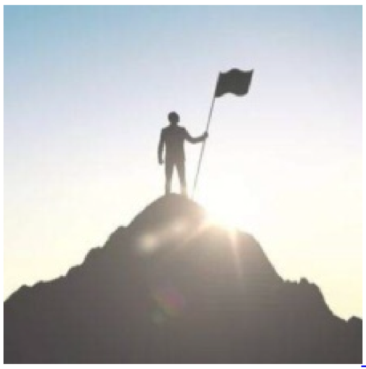
BUUCTF Reverse reverse3

原创

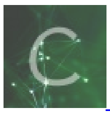
A_dmins 于 2019-07-21 22:09:47 发布 3608 收藏 4

分类专栏： CTF题 BUUCTF 一天一道CTF


CTF题 同时被 3 个专栏收录

115 篇文章 11 订阅

订阅专栏


BUUCTF

24 篇文章 2 订阅

订阅专栏

一天一道CTF

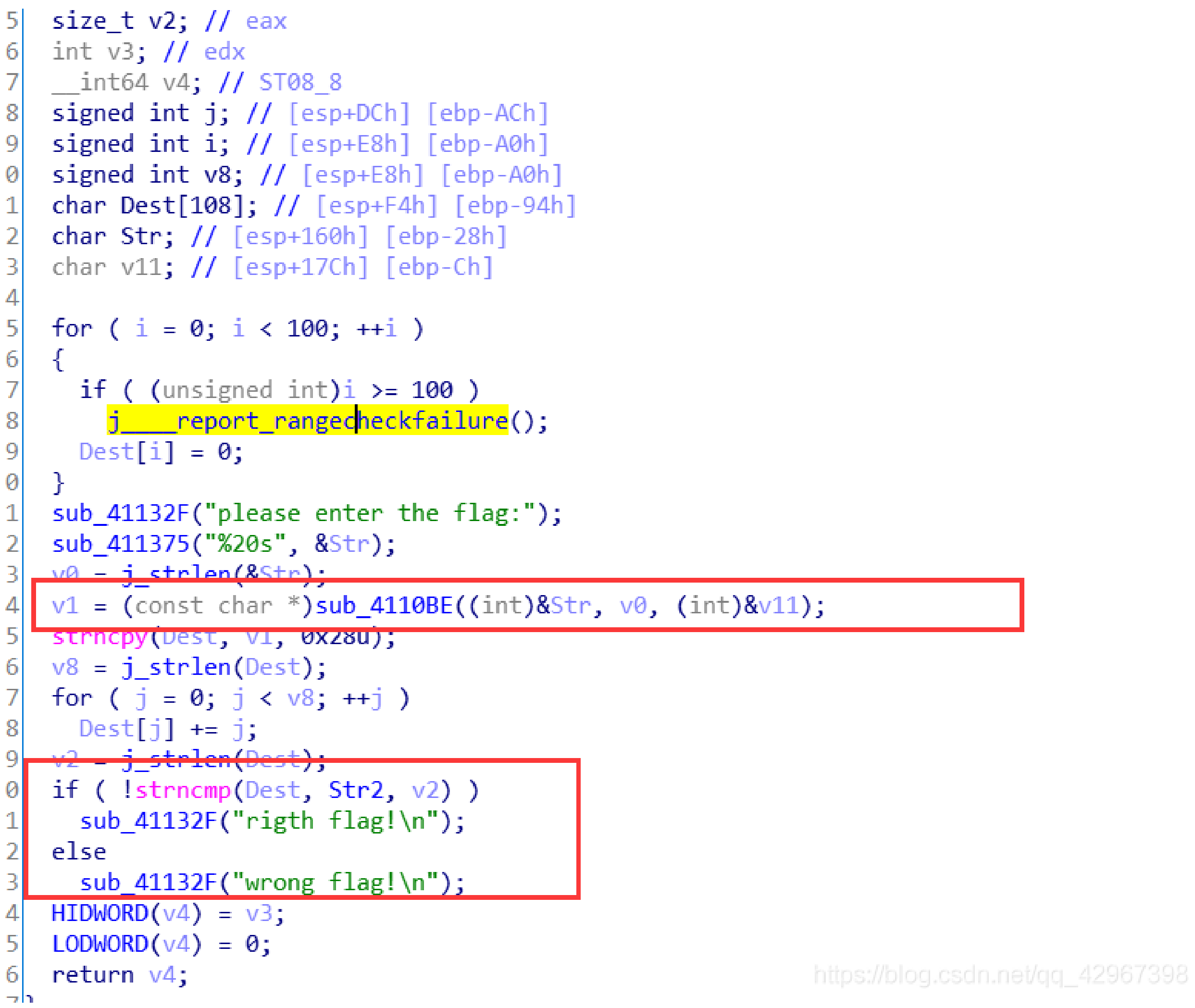
52 篇文章 5 订阅

订阅专栏

BUUCTF Reverse reverse3

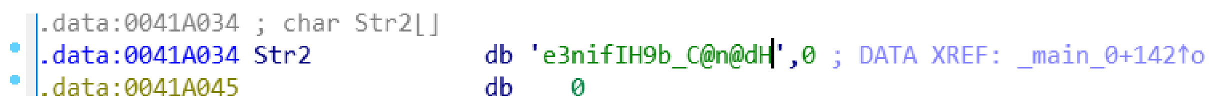
一天一道**CTF**题目，能多不能少

下载文件，无壳，直接使用ida（32）打开，找到主函数：



```
  size_t v2; // eax
  int v3; // edx
  __int64 v4; // ST08_8
  signed int j; // [esp+DCh] [ebp-ACh]
  signed int i; // [esp+E8h] [ebp-A0h]
  signed int v8; // [esp+E8h] [ebp-A0h]
  char Dest[108]; // [esp+F4h] [ebp-94h]
  char Str; // [esp+160h] [ebp-28h]
  char v11; // [esp+17Ch] [ebp-Ch]

  for ( i = 0; i < 100; ++i )
  {
    if ( (unsigned int)i >= 100 )
      j____report_rangecheckfailure();
    Dest[i] = 0;
  }
  sub_41132F("please enter the flag:");
  sub_411375("%20s", &Str);
  v0 = j_strlen(&Str);
  v1 = (const char *)sub_4110BE((int)&Str, v0, (int)&v11);
  strncpy(Dest, v1, 0x28u);
  v8 = j_strlen(Dest);
  for ( j = 0; j < v8; ++j )
    Dest[j] += j;
  v2 = j_strlen(Dest);
  if ( !strncmp(Dest, Str2, v2) )
    sub_41132F("rigth flag!\n");
  else
    sub_41132F("wrong flag!\n");
  HIDWORD(v4) = v3;
  LODWORD(v4) = 0;
  return v4;
```

看上去简单易懂，输入一个字符串然后经过sub_4110BE函数进行加密
然后再通过一个for循环进行变换，然后与str进行比较
直接查看Str2的字符串：



```
.data:0041A034 ; char Str2[]
.data:0041A034 Str2            db 'e3nifIH9b_C@n@dH',0 ; DATA XREF: _main_0+142↑o
.data:0041A045                 db    0
```

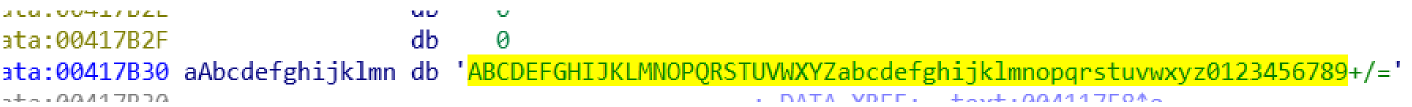
可以~继续查看加密的函数：

```
while ( v11 > 0 )
  {
    byte_41A144[2] = 0;
    byte_41A144[1] = 0;
    byte_41A144[0] = 0;
    for ( i = 0; i < 3 && v11 >= 1; ++i )
    {
      byte_41A144[i] = *v13;
      --v11;
      ++v13;
    }
    if ( !i )
      break;
    switch ( i )
    {
      case 1:
        *((_BYTE *)Dst + v7) = aAbcdefghijklmn[(signed int)(unsigned __int8)byte_41A144[0] >> 2];
        v4 = v7 + 1;
        *((_BYTE *)Dst + v4++) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | 16 * (byte_41A144[0] & 3)];
        *((_BYTE *)Dst + v4++) = aAbcdefghijklmn[64];
        *((_BYTE *)Dst + v4) = aAbcdefghijklmn[64];
        v7 = v4 + 1;
        break;
      case 2:
        *((_BYTE *)Dst + v7) = aAbcdefghijklmn[(signed int)(unsigned __int8)byte_41A144[0] >> 2];
        v5 = v7 + 1;
        *((_BYTE *)Dst + v5++) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | 16 * (byte_41A144[0] & 3)];
        *((_BYTE *)Dst + v5++) = aAbcdefghijklmn[((byte_41A144[2] & 0xC0) >> 6) | 4 * (byte_41A144[1] & 0xF)];
        *((_BYTE *)Dst + v5) = aAbcdefghijklmn[64];
        v7 = v5 + 1;
        break;
      case 3:
        *((_BYTE *)Dst + v7) = aAbcdefghijklmn[(signed int)(unsigned __int8)byte_41A144[0] >> 2];
        v6 = v7 + 1;
        *((_BYTE *)Dst + v6++) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | 16 * (byte_41A144[0] & 3)];
        *((_BYTE *)Dst + v6++) = aAbcdefghijklmn[((byte_41A144[2] & 0xC0) >> 6) | 4 * (byte_41A144[1] & 0xF)];
        *((_BYTE *)Dst + v6) = aAbcdefghijklmn[byte_41A144[2] & 0x3F];
        v7 = v6 + 1;
        break;
    }
  }
  *((_BYTE *)Dst + v7) = 0;
```

这一段看上去挺像base64加密的函数的，由3个字符变成4个字符

还有移位啥的~~

查看一下aAbcdefghijklmn这个变量:



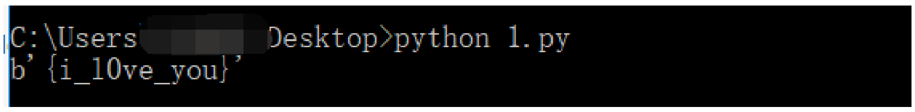
```
ata:00417B2F                 db    0
ata:00417B30 aAbcdefghijklmn db 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/='
```

那应该是base64加密了~

直接编写解题脚本：

```
import base64

s = "e3nifIH9b_C@n@dH"

x = ""

for i in range(0,len(s)):
 x += chr(ord(s[i]) - i)

print(base64.b64decode(x))
```

得到：



```
C:\Users[redacted]Desktop>python 1.py
b'{i_l0ve_you}'
```

得到fla为：`flag{i_l0ve_you}`