BUUCTF Reverse reverse3 WriteUp

原创

PlumpBoy 于 2021-09-10 18:42:24 发布 307 收藏

分类专栏： BUUCTF 逆向题解 文章标签： 系统安全 安全

reverse3-WP

首先扔到IDA里面，解析出主函数如下

```
int __cdecl main_0(int argc, const char **argv, const char **envp)
{
  int v3; // ebx
  int v4; // edi
  int v5; // esi
  int v6; // eax
  const char *v7; // eax
  size_t v8; // eax
  char v10; // [esp+0h] [ebp-188h]
  char v11; // [esp+0h] [ebp-188h]
  signed int j; // [esp+DCh] [ebp-ACh]
  int i; // [esp+E8h] [ebp-A0h]
  signed int v14; // [esp+E8h] [ebp-A0h]
  char Destination[108]; // [esp+F4h] [ebp-94h] BYREF
  char Str[28]; // [esp+160h] [ebp-28h] BYREF
  char v17[8]; // [esp+17Ch] [ebp-Ch] BYREF

//初始化为0
  for ( i = 0; i < 100; ++i )
  {
    if ( (unsigned int)i >= 0x64 )
      j____report_rangecheckfailure(v3, v4, v5);
    Destination[i] = 0;
  }

  sub_41132F("please enter the flag:", v10);
  sub_411375("%20s", (char)Str);
  v6 = j_strlen(Str);
  v7 = (const char *)sub_4110BE((int)Str, v6, (int)v17);//关键点

  strncpy(Destination, v7, 0x28u);
  v14 = j_strlen(Destination);

  for ( j = 0; j < v14; ++j )
    Destination[j] += j;//每个成员向后移动j位
  v8 = j_strlen(Destination);
  if ( !strncmp(Destination, Str2, v8) )//与str2进行比较
    sub_41132F("rigth flag!\n", v11);
  else
    sub_41132F("wrong flag!\n", v11);
  return 0;
}
```

通过观察发现sub_4110BE是关键函数，转到函数查看

```
void *__cdecl sub_411AB0(char *a1, unsigned int a2, int *a3)
{
  int v4; // [esp+D4h] [ebp-38h]
  int v5; // [esp+D4h] [ebp-38h]
  int v6; // [esp+D4h] [ebp-38h]
  int v7; // [esp+D4h] [ebp-38h]
  int i; // [esp+E0h] [ebp-2Ch]
  unsigned int v9; // [esp+ECh] [ebp-20h]
  int v10; // [esp+ECh] [ebp-20h]
  int v11; // [esp+ECh] [ebp-20h]
  void *v12; // [esp+F8h] [ebp-14h]
  char *v13; // [esp+104h] [ebp-8h]

  if ( !a1 || !a2 )
    return 0;
```

```
  return 0;
v9 = a2 / 3;
if ( (int)(a2 / 3) % 3 )
  ++v9;
v10 = 4 * v9;
*a3 = v10;
v12 = malloc(v10 + 1);
if ( !v12 )
  return 0;
j_memset(v12, 0, v10 + 1);
v13 = a1;
v11 = a2;
v4 = 0;
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 *)v12 + v4) = aAbcdefghijklmn[(int)(unsigned __int8)byte_41A144[0] >> 2];
      v5 = v4 + 1;
      *((_BYTE *)v12 + v5) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | (16 * (byte_41A144[0] & 3))];
      *((_BYTE *)v12 + ++v5) = aAbcdefghijklmn[64];
      *((_BYTE *)v12 + ++v5) = aAbcdefghijklmn[64];
      v4 = v5 + 1;
      break;
    case 2:
      *((_BYTE *)v12 + v4) = aAbcdefghijklmn[(int)(unsigned __int8)byte_41A144[0] >> 2];
      v6 = v4 + 1;
      *((_BYTE *)v12 + v6) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | (16 * (byte_41A144[0] & 3))];
      *((_BYTE *)v12 + ++v6) = aAbcdefghijklmn[((byte_41A144[2] & 0xC0) >> 6) | (4 * (byte_41A144[1] & 0xF))];
      *((_BYTE *)v12 + ++v6) = aAbcdefghijklmn[64];
      v4 = v6 + 1;
      break;
    case 3:
      *((_BYTE *)v12 + v4) = aAbcdefghijklmn[(int)(unsigned __int8)byte_41A144[0] >> 2];
      v7 = v4 + 1;
      *((_BYTE *)v12 + v7) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | (16 * (byte_41A144[0] & 3))];
      *((_BYTE *)v12 + ++v7) = aAbcdefghijklmn[((byte_41A144[2] & 0xC0) >> 6) | (4 * (byte_41A144[1] & 0xF))];
      *((_BYTE *)v12 + ++v7) = aAbcdefghijklmn[byte_41A144[2] & 0x3F];
      v4 = v7 + 1;
      break;
  }
```

通过观察发现，此处非常像base64加密，查看aAbcdefghijklmn处发现，

```
    db    0
    db    0
mn db 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/='
                            ; DATA XREF: .text:004117E8↑o
                            ; .text:00411827↑o ...
    db 0
```

确定是base64加密，再查看str2的值发现是`e3nifIH9b_C@n@dH`

写出解密脚本

```
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))
```

其中ord函数是将一个字符的ASCII值返回，chr函数是将0~255内的整数转换为对应的ASCII字符，这俩是配对函数。

得出最终flag `flag{i_l0ve_you}`

注：base64加密原理

base64是以每3个字节为一组，共24位，再以6位为一个单位组成新的数据。对于不足3字节的处理：1.不足三字节后面填充0；2.对于编码前的数据产生的6位，如果为0，则索引到的字符为‘A’；因不足3字节而填充的0，用’=’来替代。

bse64编码表是64个可见字符集。

十进制数值	编码字符	十进制数值	编码字符	十进制数值	编码字符	十进制数值	编码字符
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

原字符	A								B								C								D															
hex	0x41								0x42								0x43								0x44																
bin	0	1	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
base64值	16								20								9								3								17								0			
base64字符	Q								U								J								D								R								A			

如何分辨出base64加密：

1.base64加密过后的密文后面可能会存在=

2.通过查看字母表来确定

3.代码中含有0xF、0x3F、0x3等符号