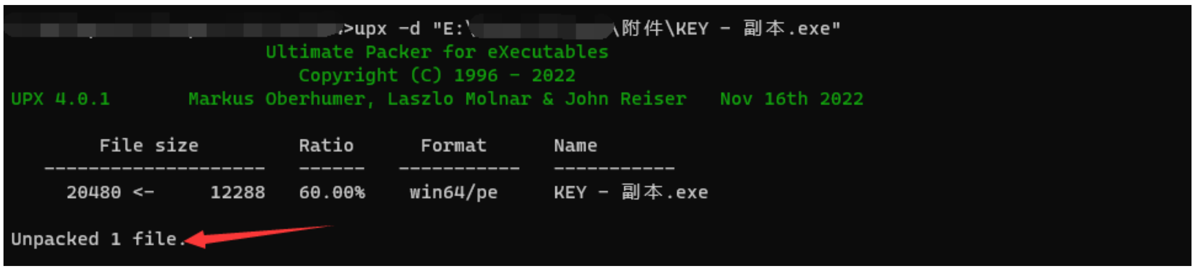
首先用4.01的upx脱壳

直接找到upx官方下载一个脱壳的exe然后upx -d



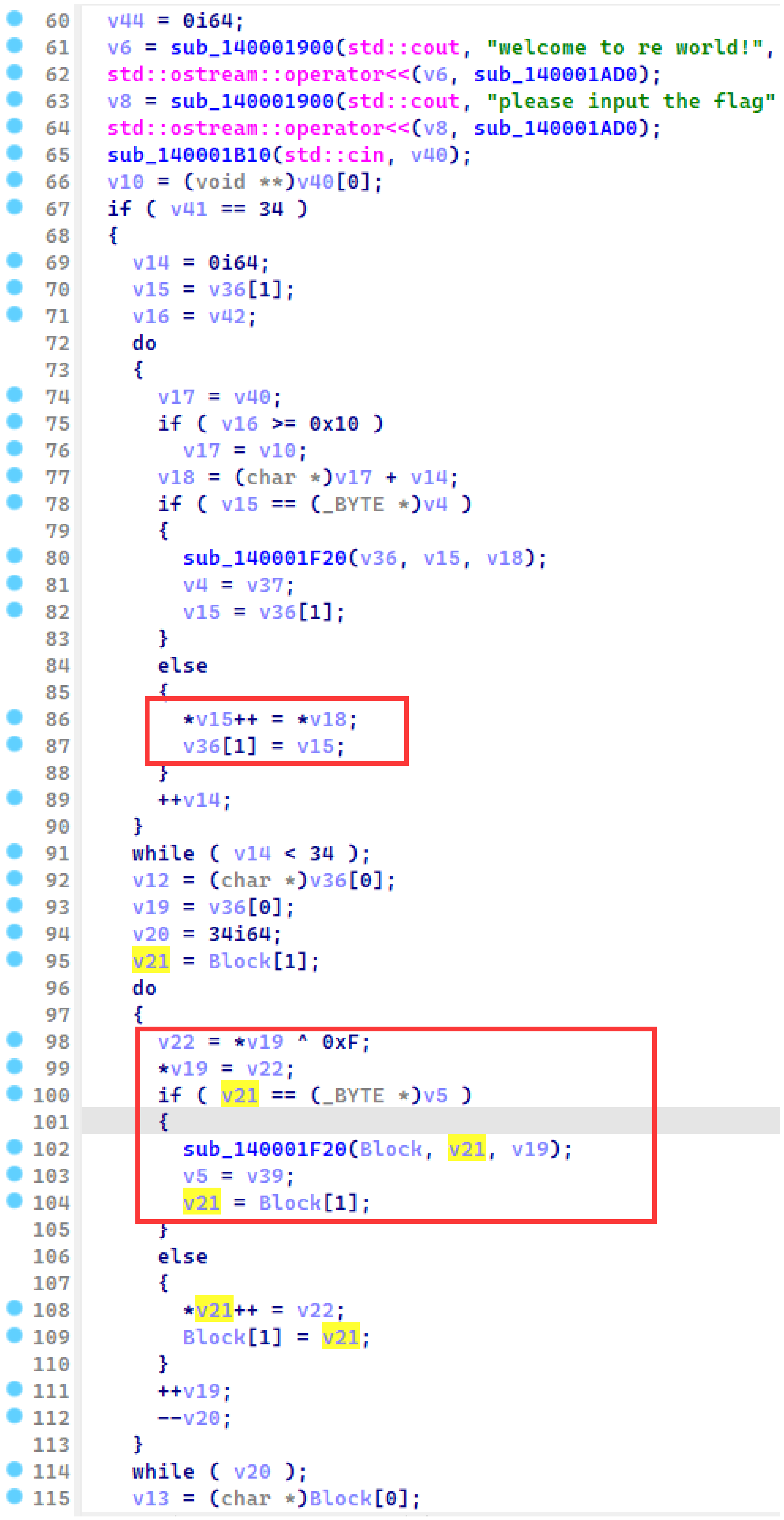
```
[redacted]>upx -d "E:\[redacted]\附件\KEY - 副本.exe"
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2022
UPX 4.0.1       Markus Oberhumer, Laszlo Molnar & John Reiser   Nov 16th 2022

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     20480 <-     12288   60.00%    win64/pe     KEY - 副本.exe

Unpacked 1 file.
```

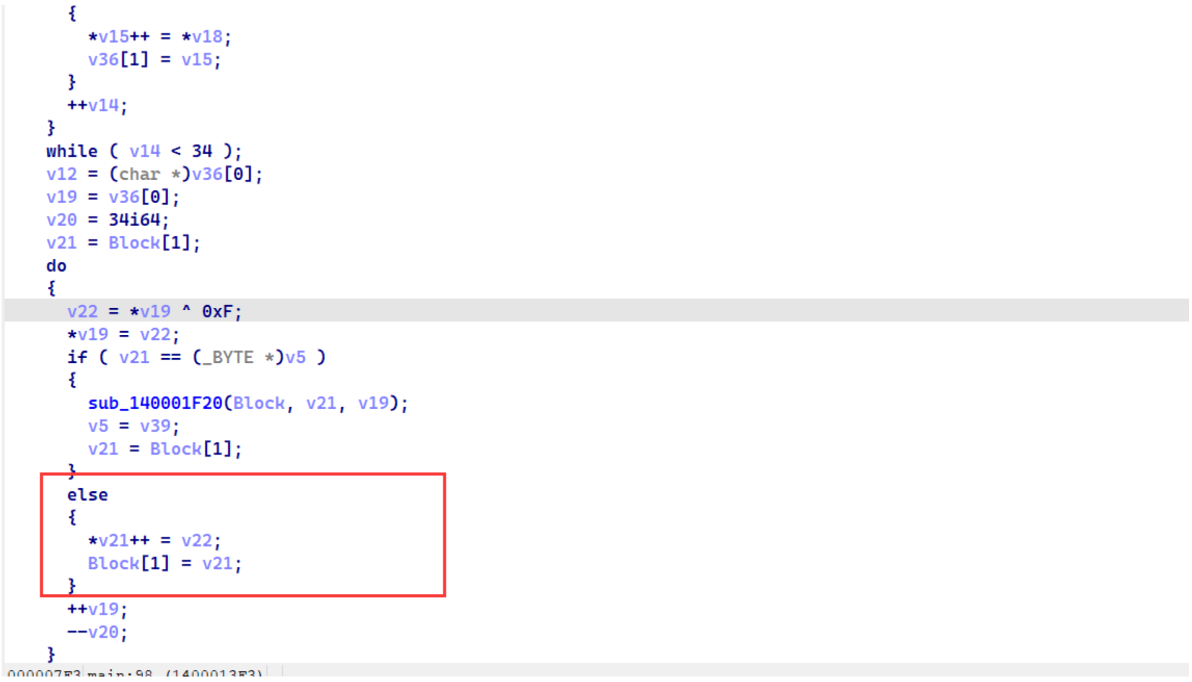
这里对应的是将输入push进vector里

然后将脱壳后程序丢入IDA里面分析



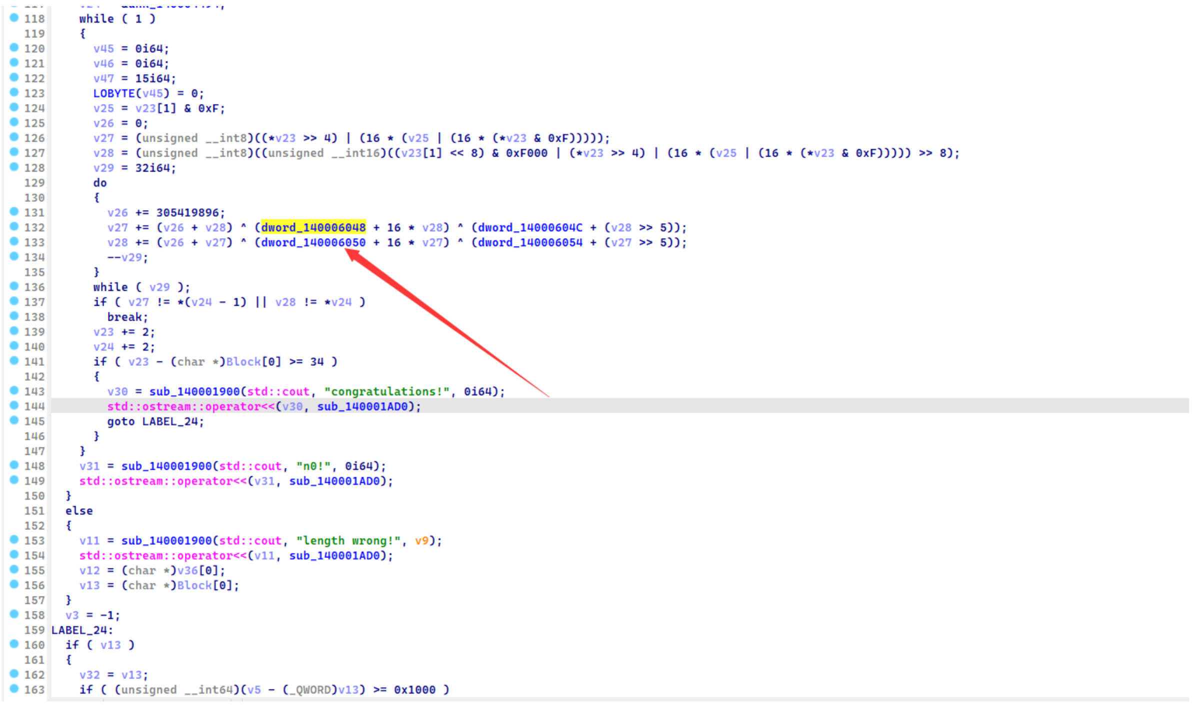
```
v44 = 0i64;
v6 = sub_140001900(std::cout, "welcome to re world!",
std::ostream::operator<<(v6, sub_140001AD0);
v8 = sub_140001900(std::cout, "please input the flag"
std::ostream::operator<<(v8, sub_140001AD0);
sub_140001B10(std::cin, v40);
v10 = (void **)v40[0];
if ( v41 == 34 )
{
  v14 = 0i64;
  v15 = v36[1];
  v16 = v42;
  do
  {
    v17 = v40;
    if ( v16 >= 0x10 )
      v17 = v10;
    v18 = (char *)v17 + v14;
    if ( v15 == (_BYTE *)v4 )
    {
      sub_140001F20(v36, v15, v18);
      v4 = v37;
      v15 = v36[1];
    }
    else
    {
      *v15++ = *v18;
      v36[1] = v15;
    }
    ++v14;
  }
  while ( v14 < 34 );
  v12 = (char *)v36[0];
  v19 = v36[0];
  v20 = 34i64;
  v21 = Block[1];
  do
  {
    v22 = *v19 ^ 0xF;
    *v19 = v22;
    if ( v21 == (_BYTE *)v5 )
    {
      sub_140001F20(Block, v21, v19);
      v5 = v39;
      v21 = Block[1];
    }
    else
    {
      *v21++ = v22;
      Block[1] = v21;
    }
    ++v19;
    --v20;
  }
  while ( v20 );
  v13 = (char *)Block[0];
```

然后就是简单的逐位异或



```
    {
      *v15++ = *v18;
      v36[1] = v15;
    }
    ++v14;
  }
  while ( v14 < 34 );
  v12 = (char *)v36[0];
  v19 = v36[0];
  v20 = 34i64;
  v21 = Block[1];
  do
  {
    v22 = *v19 ^ 0xF;
    *v19 = v22;
    if ( v21 == (_BYTE *)v5 )
    {
      sub_140001F20(Block, v21, v19);
      v5 = v39;
      v21 = Block[1];
    }
    else
    {
      *v21++ = v22;
      Block[1] = v21;
    }
    ++v19;
    --v20;
  }
000007F3 main:98 (1400013F3)
```

这里的逻辑是将异或后的字符串两个两个一组先位运算打乱位的顺序，然后丢进tea里面加密。加密后的结果和已知密文进行比较。每次比较两位。



```
    while ( 1 )
    {
      v45 = 0i64;
      v46 = 0i64;
      v47 = 15i64;
      LOBYTE(v45) = 0;
      v25 = v23[1] & 0xF;
      v26 = 0;
      v27 = (unsigned __int8)((*v23 >> 4) | (16 * (v25 | (16 * (*v23 & 0xF)))));
      v28 = (unsigned __int8)((unsigned __int16)((v23[1] << 8) & 0xF000 | (*v23 >> 4) | (16 * (v25 | (16 * (*v23 & 0xF))))) >> 8);
      v29 = 32i64;
      do
      {
        v26 += 305419896;
        v27 += (v26 + v28) ^ (dword_140006048 + 16 * v28) ^ (dword_14000604C + (v28 >> 5));
        v28 += (v26 + v27) ^ (dword_140006050 + 16 * v27) ^ (dword_140006054 + (v27 >> 5));
        --v29;
      }
      while ( v29 );
      if ( v27 != *(v24 - 1) || v28 != *v24 )
        break;
      v23 += 2;
      v24 += 2;
      if ( v23 - (char *)Block[0] >= 34 )
      {
        v30 = sub_140001900(std::cout, "congratulations!", 0i64);
        std::ostream::operator<<(v30, sub_140001AD0);
        goto LABEL_24;
      }
    }
    v31 = sub_140001900(std::cout, "n0!", 0i64);
    std::ostream::operator<<(v31, sub_140001AD0);
  }
  else
  {
    v11 = sub_140001900(std::cout, "length wrong!", v9);
    std::ostream::operator<<(v11, sub_140001AD0);
    v12 = (char *)v36[0];
    v13 = (char *)Block[0];
  }
  v3 = -1;
LABEL_24:
  if ( v13 )
  {
    v32 = v13;
    if ( (unsigned __int64)(v5 - (_QWORD)v13) >= 0x1000 )
```

exp

```python
def decrypt(v, k):
    v0 = v[0]
    v1 = v[1]
    x = (0x12345678*32)&0xffffffff
    delta = 0x12345678
    k0 = k[0]
    k1 = k[1]
    k2 = k[2]
    k3 = k[3]
    for i in range(32):
        v1 -= ((v0 << 4) + k2) ^ (v0 + x) ^ ((v0 >> 5) + k3)
        v1 = v1 & 0xFFFFFFFF
        v0 -= ((v1 << 4) + k0) ^ (v1 + x) ^ ((v1 >> 5) + k1)
        v0 = v0 & 0xFFFFFFFF
        x -= delta
        x = x & 0xFFFFFFFF
    v[0] = v0
    v[1] = v1
    return v
if __name__ == '__main__':
    plain =
[0x53d5c338,0x3a3bd468,0x639dfa0,0x4b21ae83,0xa22978f3,0x9a503149,0x6245d5a2,0xe
b1b3894,0xe91c7431,0xefa82ff8,0x84102a18,0x6276bf7a,0xac1d4eaf,0x1545a345,0x7e14
f1c3,0x961a6041,0xf2864e31,0xc7e0537f,0xf3e2e4c6,0xa9baf698,0xfe39dc26,0x5238dcf
7,0xb40dd177,0x9a13445,0xb1ab02bb,0x5c88e313,0x1f49d959,0x662e6383,0x2e842449,0x
bdd7200c,0xf2864e31,0xc7e0537f,0x9ea28242,0xb22a138c]
    key = [0x61656574,0x79656b5f,0x5f73695f,0x65726568]
    f2 = []
    for i in range(len(plain)//2):
        temp = plain[:2]
        decrypted = decrypt(temp, key)
        f2.append(decrypted[0])
        f2.append(decrypted[1])
        plain = plain[2:]
    f3 = []
    for i in range(len(f2)//2):
        f3.append(((f2[2*i]&0xf)<<4)|((f2[i*2+1]&0xf)))
        f3.append(((f2[2*i])>>4)|(((f2[2*i+1])>>4)<<4))
    for i in f3:
        print(chr((i^15)&0xff),end="")

# MOCSCTF{no_cpp_re_but_you_made_it}
```