题目名称	Where_is_the_key
题目类型	Reverse
题目描述	This exam question is a real malicious program sample. The behavior of the malicious sample itself will affect the system stability. Please be sure to run it in a virtual machine. To prevent malicious requests, the root domain name of the C2 address has been replaced with a number in the form of 0123. The flag is the hexadecimal lowercase key that encrypts the C2 address. The compressed package decompression password: infected
解题思路	该样本由 Formbook 恶意程序修改而来。由于代码量较为庞大，后期业务逻辑代码复杂，可结合已有报告进行分析。

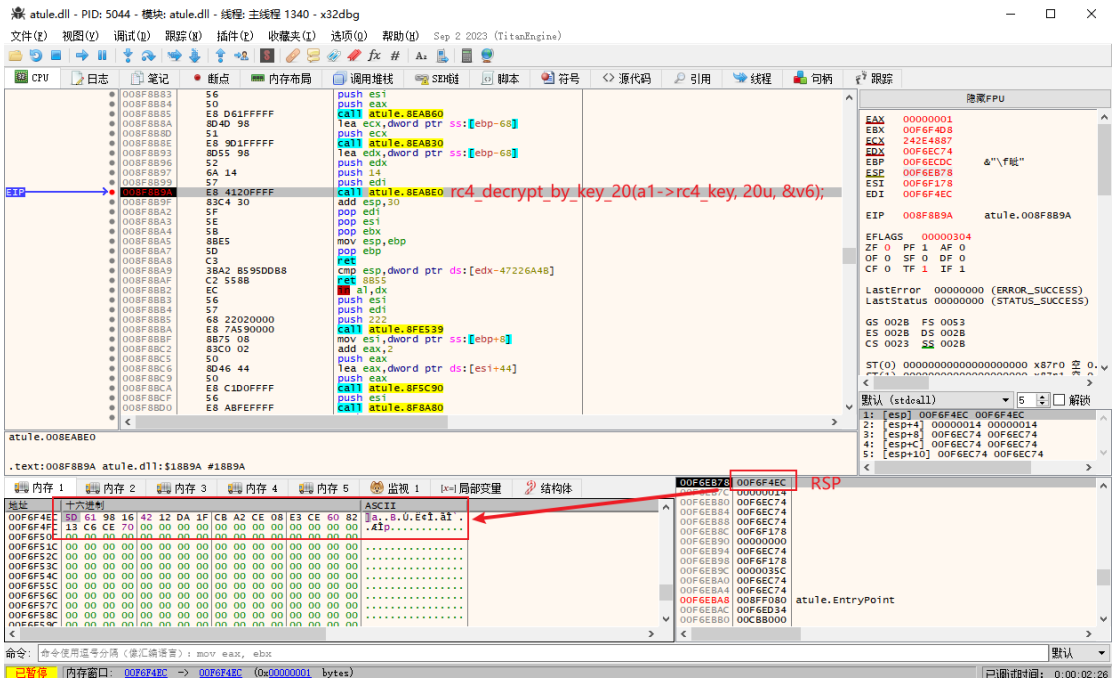
通过逆向分析可以发现，用于加密绝大部分数据的 RC4 密钥在函数 sub_418A80 里被初始化。

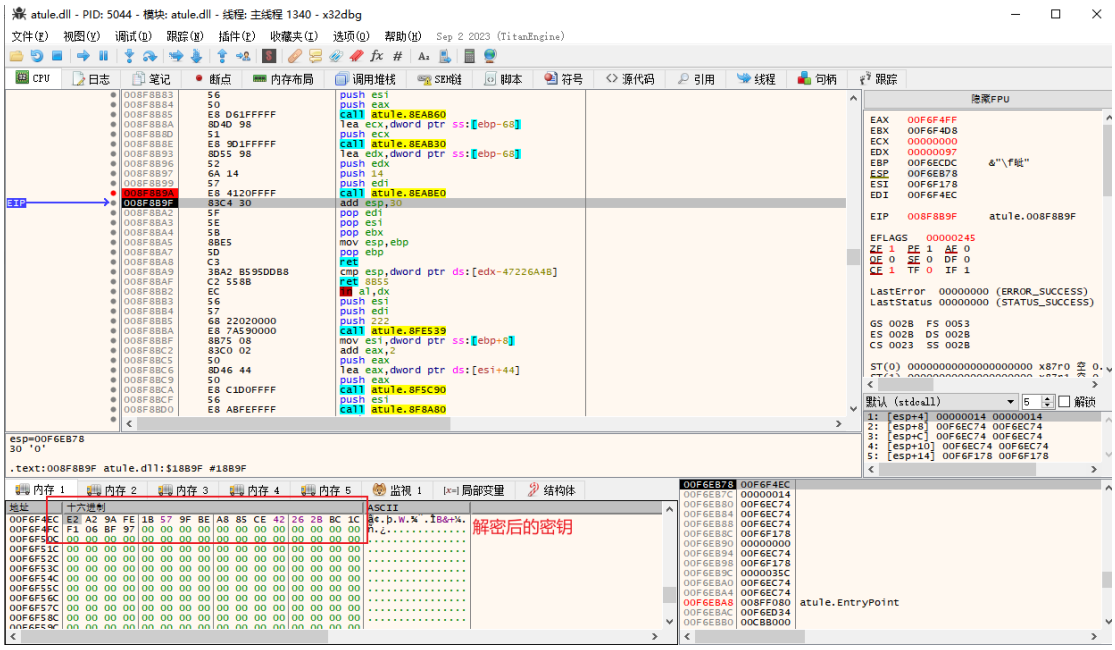
```
memset(v5, 0, 189);
v1 = sub_A2E9AD();
load_text_seg_data(v5, (v1 + 2), 187u);
v2 = sub_A2EAE1();
load_text_seg_data(a1->DWORD_table, (v2 + 2), 860u);
v3 = sub_A2E50D();
load_text_seg_data(a1->rc4_key, (v3 + 2), 20u);
sha1_init(&v6);
sha1_update(&v6, v5, 187);
sha1_final(&v6);
memcpy(a1->sub_41E9B4_sha1, &v6, sizeof(a1->sub_41E9B4_sha1));
rc4_decrypt_by_key_20(a1->DWORD_table, 0x35Cu, a1->sub_41E9B4_sha1);
sha1_init(&v6);
sha1_update(&v6, a1->rc4_key, 20);
sha1_final(&v6);
rc4_decrypt_by_key_20(a1->DWORD_table, 0x35Cu, &v6);
sha1_init(&v6);
sha1_update(&v6, a1->DWORD_table, 860);
sha1_final(&v6);
return rc4_decrypt_by_key_20(a1->rc4_key, 20u, &v6);
```

这个密钥在 sub_40B030 中被用来使用解密字符串。

```
memset(v12, 0, 2413);
memset(v14, 0, sizeof(v14));
v4 = sub_41D58C();
load_text_seg_data(v12, (v4 + 2), 0x96Bu);
v5 = a3;
for ( i = v12; v5; i += *i + 1 )              // 获取第i个字符串
  --v5;
v7 = (i + 1);
rc4_decrypt_by_key_20(i + 1, *i, a1->rc4_key);// 解密字符串
if ( a4 == 4 )                                // 后续的字符串处理 在此题中无需理会
{
  v8 = strlen(v7);
  return memcpy(a2, v7, v8 + 1);
}
else
{
  char_to_wchar(v14, v7);
  if ( a4 )
  {
    v11 = wcstr_len(v14);
    return memcpy(a2, v14, 2 * v11 + 2);
  }
  else
  {
    memset(v13, 0, sizeof(v13));
    wchar_to_unicode_string(&name, v14, 0x104u);
    wchar_to_unicode_string(&value, v13, 0x104u);
    result = RtlQueryEnvironmentVariable_U(a1, 0, &name, &value);
    if ( result >= 0 )
    {
      v10 = wcstr_len(value.Buffer);
      return memcpy(a2, value.Buffer, 2 * v10 + 2);
    }
  }
}
return result;
```

程序的绝大部分反调试在函数 sub_409C80 中运行，但是密钥的初始化在这之前就完成了，因此只要在 sub_418A80 函数中下断点即可获得解密密钥。





得出 flag 为 flag{e2a29afe1b579fbea885ce42262bbc1cf106bf97}。

当然也可以根据样本解密流程，纯静态的解密 RC4 密钥和加密的数据。

纯静态解题脚本：

```
import struct
import types
import hashlib
from Crypto.Cipher import ARC4

with open("atule.dll","rb") as f:
    data = f.read()
```

```
v14 = 0
v15 = 0
#from sub_415C90
def c66(src,offset,dst):
    global v14
    match src[offset+1]:
        case 0x6A:
            dst += struct.pack("<I",v15)[:2]
            v14 = src[offset+2]
            return offset+3
        case 0x68:
            dst += src[offset+2:offset+4]
            return offset+4
        case 0xB8:
            dst += src[offset+2:offset+4]
            return offset+4
        case _:
            return offset+1
def c69(src,offset,dst):
    dst += src[offset+2:offset+8]
    return offset+10
def c6a(src,offset,dst):
    global v15
    v = src[offset+1]
    if v&0x80 != 0:
        v |=  0xFFFFFF00
    v15 = v
    dst += struct.pack("<I",v15)
    return offset+2
def c6b(src,offset,dst):
    global v14
    v14 = src[offset+6]
    dst += src[offset+2:offset+6]
    dst += struct.pack("<I",v14)[:2]
    return offset+7
def c80(src,offset,dst):
    if src[offset+1] == 5:
        dst += src[offset+2:offset+6]
        return offset+7
    else:
        dst += src[offset+2:offset+3]
        return offset+3
def c83(src,offset,dst):
    global v15
    v = src[offset+2]
    if (v&0x80) != 0 :
        v |= 0xFFFFFF00
    v15 = v
    dst += struct.pack("<I",v15)
    return offset+3
def cC0(src,offset,dst):
    dst += src[offset+2:offset+6]
    return offset+7
def cC1(src,offset,dst):
    dst += src[offset+2:offset+6]
    return offset+7
def cFF(src,offset,dst):
    if src[offset+1] == 0x35:
```

```
        dst += src[offset+2:offset+6]
        return offset+6
    return offset
drop_copy_table = {
    0x00:(2,4),0x01:(2,4),0x02:(2,4),0x03:(2,4),0x04:(1,1),0x05:(1,4),
    0x08:(2,4),0x09:(2,4),0x0a:(2,4),0x0b:(2,4),0x0c:(1,1),0x0d:(1,4),          0x0f:(6,0),
    0x10:(2,4),0x11:(2,4),0x12:(2,4),0x13:(2,4),0x14:(1,1),0x15:(1,4),
    0x18:(2,4),0x19:(2,4),0x1a:(2,4),0x1b:(2,4),0x1c:(1,1),0x1d:(1,4),
    0x20:(2,4),0x21:(2,4),0x22:(2,4),0x23:(2,4),0x24:(1,1),0x25:(1,4),
    0x28:(2,4),0x29:(2,4),0x2a:(2,4),0x2b:(2,4),0x2c:(1,1),0x2d:(1,4),
    0x30:(2,4),0x31:(2,4),0x32:(2,4),0x33:(2,4),0x34:(1,1),0x35:(1,4),
    0x38:(2,4),0x39:(2,4),0x3a:(2,4),0x3b:(2,4),0x3c:(1,1),0x3d:(1,4),
    0x40:(0,1),0x41:(0,1),0x42:(0,1),0x43:(0,1),0x44:(0,1),0x45:(0,1),0x46:(0,1),0x47:(0,1),
    0x48:(0,1),0x49:(0,1),0x4a:(0,1),0x4b:(0,1),0x4c:(0,1),0x4d:(0,1),0x4e:(0,1),0x4f:(0,1),
    0x50:(0,1),0x51:(0,1),0x52:(0,1),0x53:(0,1),0x54:(0,1),0x55:(0,1),0x56:(0,1),0x57:(0,1),
    0x58:(0,1),0x59:(0,1),0x5a:(0,1),0x5b:(0,1),0x5c:(0,1),0x5d:(0,1),0x5e:(0,1),0x5f:(0,1),
                                                        0x66:c66,
                                                        0x68:(1,4),0x69:c69,
                                                        0x6a:c6a,  0x6b:c6b,
    0x70:(2,0),0x71:(2,0),0x72:(2,0),0x73:(2,0),0x74:(2,0),0x75:(2,0),0x76:(2,0),0x77:(2,0),
    0x78:(2,0),0x79:(2,0),0x7a:(2,0),0x7b:(2,0),0x7c:(2,0),0x7d:(2,0),0x7e:(2,0),0x7f:(2,0),
    0x80:c80  ,0x81:(2,4),          0x83:c83  ,0x84:(2,4),0x85:(2,4),0x86:(2,4),0x87:(2,4),
    0x88:(2,4),0x89:(2,4),0x8a:(2,4),0x8b:(2,4),          0x8d:(2,4),          0x8f:(2,4),
    0x90:(0,1),
    0xa0:(1,4),0xa1:(1,4),0xa2:(1,4),0xa3:(1,4),0xa4:(0,1),0xa5:(0,1),0xa6:(0,1),0xa7:(0,1),
    0xa8:(1,1),0xa9:(1,4),0xaa:(0,1),0xab:(0,1),0xac:(0,1),0xad:(0,1),0xae:(0,1),0xaf:(0,1),
    0xb0:(1,1),0xb1:(1,1),0xb2:(1,1),0xb3:(1,1),0xb4:(1,1),0xb5:(1,1),0xb6:(1,1),0xb7:(1,1),
    0xb8:(1,4),0xb9:(1,4),0xba:(1,4),0xbb:(1,4),0xbc:(1,4),0xbd:(1,4),0xbe:(1,4),0xbf:(1,4),
    0xc0:cC0,  0xc1:cC1,  0xc2:(1,2),0xc3:(0,1),
    0xd0:(2,4),0xd1:(2,4),
    0xe8:(5,0),0xe9:(5,0),          0xeb:(2,0),
                        0xf2:(0,1),                                 0xf6:(2,1),0xf7:(2,4),
                                                                            0xff:cFF
}
def load_data(arr,offset,data_length):
    assert(arr[offset] == 0x55 and arr[offset+1] == 0x8B)
    offset  += 3
    data = []
    while len(data)<data_length:
        opcode = arr[offset]
        operator = drop_copy_table[opcode]
        if isinstance(operator,types.FunctionType):
            #print(hex(len(data))," ",hex(opcode),"opcode")
            offset = operator(arr,offset,data)
        else:
            drop,copy = operator
            #print(hex(len(data))," ",hex(opcode)," ",arr[offset+drop:offset+drop+copy].hex())
            data += arr[offset+drop:offset+drop+copy]
            offset += drop+copy
    return data

data_1E9AD = load_data(data,0x1E9AD+5+2,0xBB)
data_1E9AD = bytes(data_1E9AD)

data_1EAE1 = load_data(data,0x1EAE1+5+2,0x35C)
data_1EAE1 = bytes(data_1EAE1)

data_1E50D = load_data(data,0x1E50D+5+2,20)
```

```
data_1E50D = bytes(data_1E50D)

def rc4_decrypt(enc,key):
    enc = list(enc)
    for i in range(len(enc) - 1, 0, -1):
        enc[i-1] = (enc[i-1]-enc[i])&0xFF
    for i in range(0, len(enc) -1):
        enc[i] = (enc[i]-enc[i+1])&0xFF
    enc = bytes(enc)
    arc4 = ARC4.new(key)
    enc = arc4.decrypt(enc)
    enc = list(enc)
    for i in range(len(enc) - 1, 0, -1):
        enc[i-1] = (enc[i-1]-enc[i])&0xFF
    for i in range(0, len(enc) -1):
        enc[i] = (enc[i]-enc[i+1])&0xFF
    return bytes(enc)

data_1E9AD_sha1 = hashlib.sha1(data_1E9AD).digest()
data_1E9AD_sha1 = struct.unpack(">5I",data_1E9AD_sha1)
data_1E9AD_sha1 = struct.pack("<5I",*data_1E9AD_sha1)
data_1EAE1_decrypt = rc4_decrypt(data_1EAE1,data_1E9AD_sha1)
data_1E50D_sha1 = hashlib.sha1(data_1E50D).digest()
data_1E50D_sha1 = struct.unpack(">5I",data_1E50D_sha1)
data_1E50D_sha1 = struct.pack("<5I",*data_1E50D_sha1)
data_1EAE1_decrypt_decrypt = rc4_decrypt(data_1EAE1_decrypt,data_1E50D_sha1)
data_1EAE1_decrypt_decrypt_sha1 = hashlib.sha1(data_1EAE1_decrypt_decrypt).digest()
data_1EAE1_decrypt_decrypt_sha1 = struct.unpack(">5I",data_1EAE1_decrypt_decrypt_sha1)
data_1EAE1_decrypt_decrypt_sha1 = struct.pack("<5I",*data_1EAE1_decrypt_decrypt_sha1)
data_1E50D_decrypt = rc4_decrypt(data_1E50D,data_1EAE1_decrypt_decrypt_sha1)

data_1D58C = load_data(data,0x1D58C+5+2,0x96B)
data_1D58C = bytes(data_1D58C)

print("decrypt key:",data_1E50D_decrypt.hex())
idx = 0
while len(data_1D58C):
    l = data_1D58C[0]
    data_1D58C = data_1D58C[1:]
    d = data_1D58C[:l]
    data_1D58C = data_1D58C[l:]
    print(idx," %02x"%l," ",rc4_decrypt(d,data_1E50D_decrypt).decode())
    idx += 1
```