2021湖湘杯 Hideit Writeup

原创

1mmorta1 于 2021-11-15 14:39:59 发布 2959 收藏 2

分类专栏： reverse 文章标签： 安全

本文链接：https://blog.csdn.net/qq_41866334/article/details/121334376

2021湖湘杯 Hideit

> AAA : immortal

动态调试

直接x64dbg动调找到了关键的加密代码分别是xxtea 和 chacha20，直接动调从中拿出各种参数然后写代码进行解密。其实一开始没看出来chacha20，找到了这篇文章 逆向中常见的Hash算法和对称加密算法的分析 .

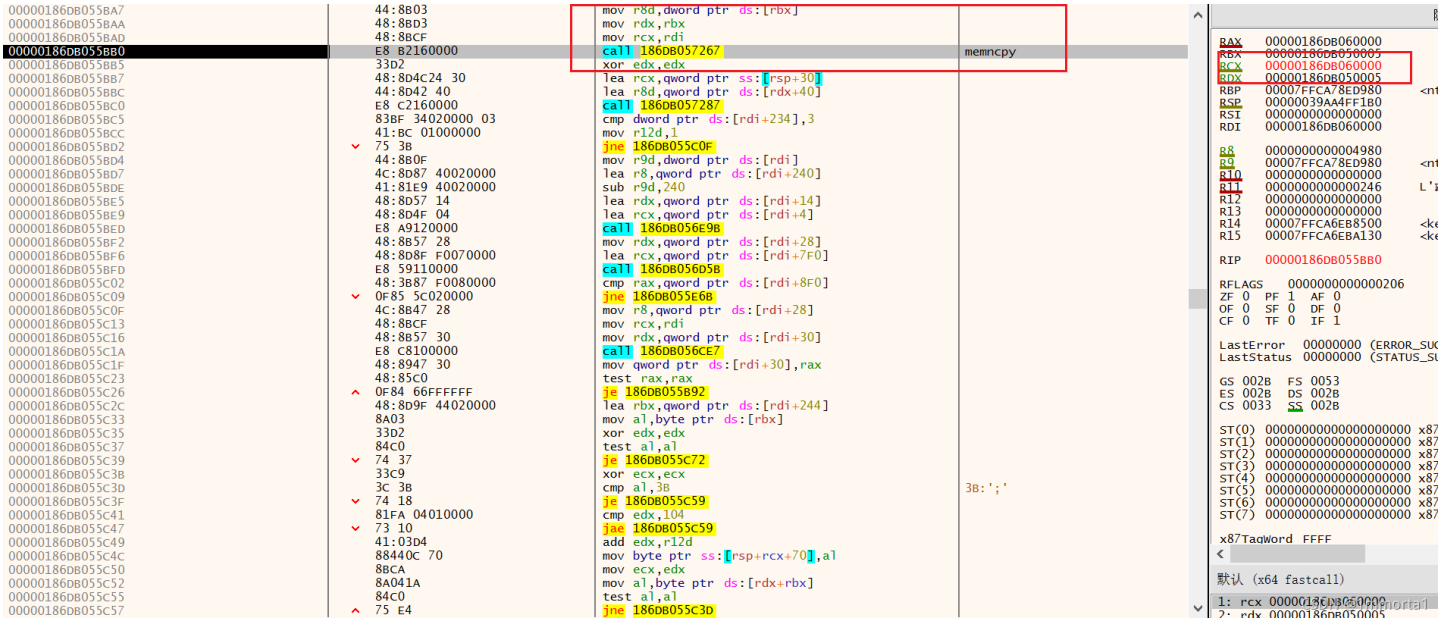
```
000001F0D9E41D30   41:81C3 B979379E      add r11d,9E3779B9
000001F0D9E41D37   42:8D148D 00000000    lea edx,qword ptr ds:[r9*4]
000001F0D9E41D3F   45:8BC3               mov r8d,r11d
000001F0D9E41D42   8BC7                  mov eax,edi
000001F0D9E41D44   C1E8 05               shr eax,5
000001F0D9E41D47   41:8BC9               mov ecx,r9d
000001F0D9E41D4A   33D0                  xor edx,eax
000001F0D9E41D4C   C1E9 03               shr ecx,3
000001F0D9E41D4F   41:C1E8 02            shr r8d,2
000001F0D9E41D53   8BC7                  mov eax,edi
000001F0D9E41D55   C1E0 04               shl eax,4
000001F0D9E41D58   41:83E0 03            and r8d,3
000001F0D9E41D5C   33C8                  xor ecx,eax
000001F0D9E41D5E   41:8BC1               mov eax,r9d
000001F0D9E41D61   03D1                  add edx,ecx
000001F0D9E41D63   41:33C3               xor eax,r11d
000001F0D9E41D66   42:8B4C84 60          mov ecx,dword ptr ss:[rsp+r8*4+60]
000001F0D9E41D6B   49:83F0 01            xor r8,1
000001F0D9E41D6F   33CF                  xor ecx,edi
000001F0D9E41D71   03C8                  add ecx,eax
000001F0D9E41D73   33D1                  xor edx,ecx                          ecx是key
000001F0D9E41D75   44:03D2               add r10d,edx
000001F0D9E41D78   41:8BC2               mov eax,r10d
000001F0D9E41D7B   41:8BCA               mov ecx,r10d
000001F0D9E41D7E   C1E8 05               shr eax,5
000001F0D9E41D81   C1E9 03               shr ecx,3
000001F0D9E41D84   42:8D1495 00000000    lea edx,qword ptr ds:[r10*4]
000001F0D9E41D8C   33D0                  xor edx,eax
000001F0D9E41D8E   41:8BC2               mov eax,r10d
000001F0D9E41D91   C1E0 04               shl eax,4
000001F0D9E41D94   33C8                  xor ecx,eax
000001F0D9E41D96   41:8BC2               mov eax,r10d
000001F0D9E41D99   03D1                  add edx,ecx
000001F0D9E41D9B   41:33C3               xor eax,r11d
000001F0D9E41D9E   42:8B4C84 60          mov ecx,dword ptr ss:[rsp+r8*4+60]
000001F0D9E41DA3   41:33CA               xor ecx,r10d
000001F0D9E41DA6   03C8                  add ecx,eax
000001F0D9E41DA8   33D1                  xor edx,ecx
000001F0D9E41DAA   44:03CA               add r9d,edx
000001F0D9E41DAD   41:8BF9               mov edi,r9d
000001F0D9E41DB0   83C6 FF               add esi,FFFFFFFF
000001F0D9E41DB3   0F85 77FFFFFF         jne 1F0D9E41D30
000001F0D9E41DB9   41:81FA 1BBE3011      cmp r10d,1130BE1B
000001F0D9E41DC0   0F85 B8000000         jne 1F0D9E41E7E
000001F0D9E41DC6   41:81F9 43747463      cmp r9d,63747443
```



```
000001F0D9E41DB9   41:81FA 1BBE3011      cmp r10d,1130BE1B
000001F0D9E41DC0   0F85 B8000000         jne 1F0D9E41E7E
000001F0D9E41DC6   41:81F9 43747463      cmp r9d,63747443
000001F0D9E41DCD   0F85 AB000000         jne 1F0D9E41E7E
000001F0D9E41DD3   0FB64424 42           movzx eax,byte ptr ss:[rsp+42]
000001F0D9E41DD8   49:8BD6               mov rdx,r14                          r14:"0N3@aYI_M3l0dy_KurOml_W_Su
000001F0D9E41DDB   0FB64C24 43           movzx ecx,byte ptr ss:[rsp+43]
000001F0D9E41DE0   C1E1 08               shl ecx,8
000001F0D9E41DE3   0BC8                  or ecx,eax
000001F0D9E41DE5   48:895D D0            mov qword ptr ss:[rbp-30],rbx
000001F0D9E41DE9   0FB64424 41           movzx eax,byte ptr ss:[rsp+41]
000001F0D9E41DEE   C1E1 08               shl ecx,8
000001F0D9E41DF1   0BC8                  or ecx,eax
000001F0D9E41DF3   0FB64424 40           movzx eax,byte ptr ss:[rsp+40]
000001F0D9E41DF8   C1E1 08               shl ecx,8
000001F0D9E41DFB   0BC8                  or ecx,eax
000001F0D9E41DFD   0FB64424 46           movzx eax,byte ptr ss:[rsp+46]
000001F0D9E41E02   894D D8               mov dword ptr ss:[rbp-28],ecx
000001F0D9E41E05   0FB64C24 47           movzx ecx,byte ptr ss:[rsp+47]
000001F0D9E41E0A   C1E1 08               shl ecx,8
000001F0D9E41E0D   0BC8                  or ecx,eax
000001F0D9E41E0F   0FB64424 45           movzx eax,byte ptr ss:[rsp+45]
000001F0D9E41E14   C1E1 08               shl ecx,8
000001F0D9E41E17   0BC8                  or ecx,eax
000001F0D9E41E19   0FB64424 44           movzx eax,byte ptr ss:[rsp+44]
000001F0D9E41E1E   C1E1 08               shl ecx,8
000001F0D9E41E21   0BC8                  or ecx,eax
000001F0D9E41E23   894D DC               mov dword ptr ss:[rbp-24],ecx
000001F0D9E41E26   48:8D4D A0            lea rcx,qword ptr ss:[rbp-60]
000001F0D9E41E2A   E8 D1F1FFFF           call 1F0D9E41000
000001F0D9E41E2F   44:8D4E 20            lea r9d,qword ptr ds:[rsi+20]
000001F0D9E41E33   4C:8D45 E0            lea r8,qword ptr ss:[rbp-20]
000001F0D9E41E37   48:8D4D A0            lea rcx,qword ptr ss:[rbp-60]
000001F0D9E41E3B   48:8D5424 70          lea rdx,qword ptr ss:[rsp+70]
000001F0D9E41E40   E8 0BF3FFFF           call 1F0D9E41150
000001F0D9E41E45   48:8D0D 84130000      lea rcx,qword ptr ds:[1F0D9E431D0]
000001F0D9E41E4C   0F1F40 00             nop dword ptr ds:[rax],eax
000001F0D9E41E50   0FB6441D E0           movzx eax,byte ptr ss:[rbp+rbx-20]
000001F0D9E41E55   38040B                cmp byte ptr ds:[rbx+rcx],al         cmp string
000001F0D9E41E58   75 24                 jne 1F0D9E41E7E
000001F0D9E41E5A   48:FFC3               inc rbx
000001F0D9E41E5D   48:83FB 20            cmp rbx,20                           20:' '
000001F0D9E41E61   7C ED                 jl 1F0D9E41E50
000001F0D9E41E63   48:8D0D E6130000      lea rcx,qword ptr ds:[1F0D9E43250]   000001F0D9E43250:"You find last
000001F0D9E41E6A   FF15 88120000         call qword ptr ds:[<&puts>]
```


解密dll

后来在赛后和队友交流发现这题其实在数据段放了一个加密的dll文件，在解密段可以x64dbg dump出dll的代码。断点下在memncpy上，解密结束后就可以dump出对应的地址，删去pe头之前的内容就是一个dll文件。放在ida内即可反编译。




```
_DWORD *v26; // [rsp+1C0h] [rbp+8h] BYREF
char v27; // [rsp+1C8h] [rbp+10h] BYREF
char v28; // [rsp+1D0h] [rbp+18h] BYREF

v2 = (__int64 (__fastcall *)(_QWORD, __int64, __int64, __int64))sub_7FF71ED59D27(
                                                                  (__int64)a1,
                                                                  *((_QWORD *)a1 + 9),
                                                                  *((_QWORD *)a1 + 5));
v3 = (void (__fastcall *)(char *, _QWORD, __int64))sub_7FF71ED59D27(
                                                     (__int64)a1,
                                                     *((_QWORD *)a1 + 10),
                                                     *((_QWORD *)a1 + 5));
v4 = sub_7FF71ED59D27((__int64)a1, *((_QWORD *)a1 + 51), *((_QWORD *)a1 + 5));
v5 = (void (__fastcall *)(_QWORD))v4;
if ( !v2 || !v3 || !v4 )
  return 0xFFFFFFFFi64;
v6 = (char *)v2(0i64, *a1, 12288i64, 4i64);
v7 = v6;
if ( !v6 )
{
  if ( a1[140] == 2 )
    v5(0i64);
  return 0xFFFFFFFFi64;
}
memncpy(v6, (__int64)a1, *a1);
memset(v24, 0, sizeof(v24));
if ( *((_DWORD *)v7 + 141) != 3
  || (sub_7FF71ED59EDB(v7 + 4, v7 + 20, v7 + 576, (unsigned int)(*(_DWORD *)v7 - 576)),
      sub_7FF71ED59D9B(v7 + 2032, *((_QWORD *)v7 + 5)) == *((_QWORD *)v7 + 286)) )
{
  v9 = sub_7FF71ED59D27((__int64)v7, *((_QWORD *)v7 + 6), *((_QWORD *)v7 + 5));
  *((_QWORD *)v7 + 6) = v9;
  if ( !v9 )
    return 0xFFFFFFFFi64;
  v10 = v7 + 580;
  while ( 1 )
  {
```

00007DCC sub_7FF71ED58B4F:49 (7FF71ED58BCC)


```
if ( !RegOpenKeyW(HKEY_LOCAL_MACHINE, L"SOFTWARE\\Classes\\.txt", &phkResult) )
{
  Type = 0;
  memset(Data, 0, 0x208ui64);
  cbData = 66;
  if ( !RegQueryValueExW(phkResult, L"Keys Secret", 0i64, &Type, (LPBYTE)Data, &cbData) )
```

```
    WideCharToMultiByte(0, 0, Data, -1, MultiByteStr, 260, 0i64, 0i64);
}
puts("First secret here:");
v10 = 0i64;
v11 = 0;
sub_180001B50("%s", (const char *)&v10);
v12 = 0i64;
strcpy((char *)&v12, (const char *)&v10);
v13[0] = 114;
v13[1] = 514;
v13[2] = 19;
v13[3] = 19;
memset(v21, 0, sizeof(v21));
v3 = HIDWORD(v12);
v4 = 32;
v5 = v12;
v6 = HIDWORD(v12);
v7 = 0;
do
{
  v7 -= 1640531527;
  v8 = (v7 >> 2) & 3;
  v5 += ((v7 ^ v3) + (v6 ^ v13[v8])) ^ (((16 * v6) ^ (v3 >> 3)) + ((v6 >> 5) ^ (4 * v3)));
  v3 += ((v7 ^ v5) + (v5 ^ v13[v8 ^ 1])) ^ (((16 * v5) ^ (v5 >> 3)) + ((v5 >> 5) ^ (4 * v5)));
  v6 = v3;
  --v4;
}
while ( v4 );
if ( v5 == 288407067 && v3 == 1668576323 )
{
  v18 = 0i64;
  v19 = (unsigned __int8)v10 | ((BYTE1(v10) | (WORD1(v10) << 8)) << 8);
  v20 = BYTE4(v10) | ((BYTE5(v10) | (HIWORD(v10) << 8)) << 8);
  sub_180001000(v17, (unsigned __int8 *)a1);
  sub_180001150(v17, MultiByteStr, v21, 32i64);
  while ( byte_1800031D0[v2] == v21[v2] )
  {
    if ( ++v2 >= 32 )
```

000010E3 sub_180001BB0:46 (180001CE3)