[8] Find a valid flag

1. GetDlgItemText. 어떻게 correct가 뜨는지

- 올바른 flag를 입력해야 하므로, **GetDlgItemText**로 ID가 1005(0x3ed)인 것을 찾는다. (3 or 4번에서 ID 알아냈었음)

```
Decompile: FUN_140008e60 - (KU-RE{FinaL}.exe)
27  UCHAR local_3c8 [80];
28  undefined local_378 [64];
29  BYTE local_338 [256];
30  WCHAR local_238 [264];
31  ulonglong local_28;
32
33  local_28 = DAT_14002d080 ^ (ulonglong)auStackY_4c8;
34  puVar5 = &DAT_14001ff68;
35  puVar6 = local_378;
36  for (lVar4 = 0x34; lVar4 != 0; lVar4 = lVar4 + -1) {
37    *puVar6 = *puVar5;
38    puVar5 = puVar5 + 1;
39    puVar6 = puVar6 + 1;
40  }
41  pcVar7 = local_408;
42  for (lVar4 = 0x40; lVar4 != 0; lVar4 = lVar4 + -1) {
43    *pcVar7 = '\0';
44    pcVar7 = pcVar7 + 1;
45  }
46  GetDlgItemTextA(*(HWND *)(param_1 + 8),0x3ed,local_408,0x40);
```

- 이렇게 `local_408`에 flag를 받고,

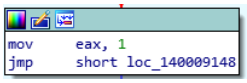
▼ FUN_140008d40이 FUN_140008e60을 부른다. 이 함수의 반환값이 중요한 역할을 함

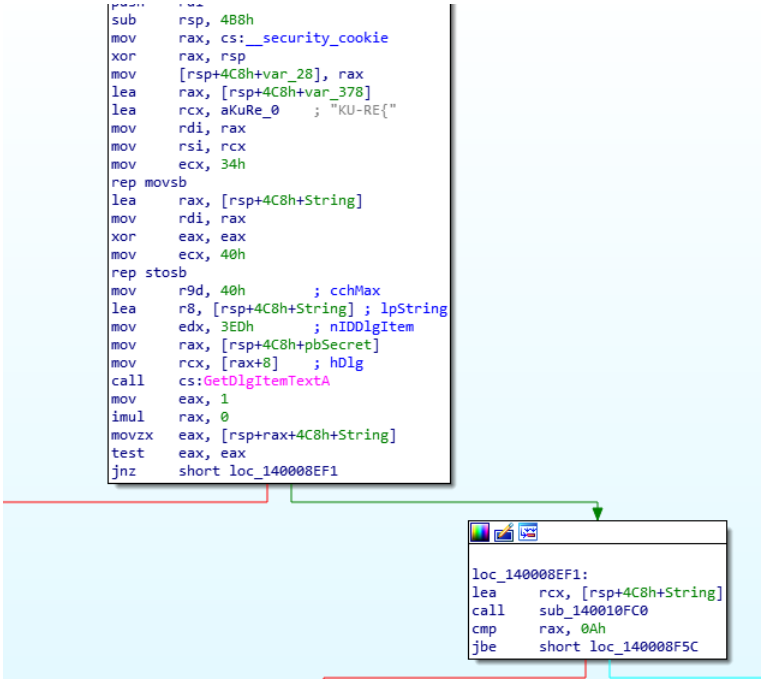
```
Decompile: FUN_140008d40 - (KU-RE{FinaL}.exe)
5   int iVar1;
6   HWND *ppHVar2;
7   longlong lVar3;
8   undefined1 *puVar4;
9   undefined *puVar5;
10  undefined auStack_88 [32];
11  longlong local_68;
12  undefined8 local_60;
13  undefined8 local_58;
14  undefined local_50 [48];
15  ulonglong local_20;
16
17  local_20 = DAT_14002d080 ^ (ulonglong)auStack_88;
18  local_68 = param_1;
19  ppHVar2 = (HWND *)FUN_14000a5d0((HWND *)(param_1 + 8),&local_60,0x3eb);
20  FUN_14000a570(ppHVar2,0);
21  puVar4 = &DAT_14001ff00;
22  puVar5 = local_50;
23  for (lVar3 = 0x2e; lVar3 != 0; lVar3 = lVar3 + -1) {
24    *puVar5 = *puVar4;
25    puVar4 = puVar4 + 1;
26    puVar5 = puVar5 + 1;
27  }
28  Sleep(0x3777);
29  FUN_140009170();
30  iVar1 = FUN_140008e60(local_68);
31  if (iVar1 == 0) {
32    FUN_14000a6c0((HWND *)(local_68 + 8),0x3ee,L"Incorrect...");
33  }
34  else {
35    FUN_14000a6c0((HWND *)(local_68 + 8),0x3ee,L"Correct!");
36  }
37  FUN_1400094b0();
38  ppHVar2 = (HWND *)FUN_14000a5d0((HWND *)(local_68 + 8),&local_58,0x3eb);
39  FUN_14000a570(ppHVar2,1);
40  FUN_14000b800(local_20 ^ (ulonglong)auStack_88);
41  return;
42}
```

- flag 길이가 0xb보다 작으면, 49번째줄 if문이 실행된다.

FUN_140008e60의 마지막 부분

- correct가 실행되려면 eax가 0이 아니어야 하는데 그 부분은 지금 하나의 길만 가능하다.

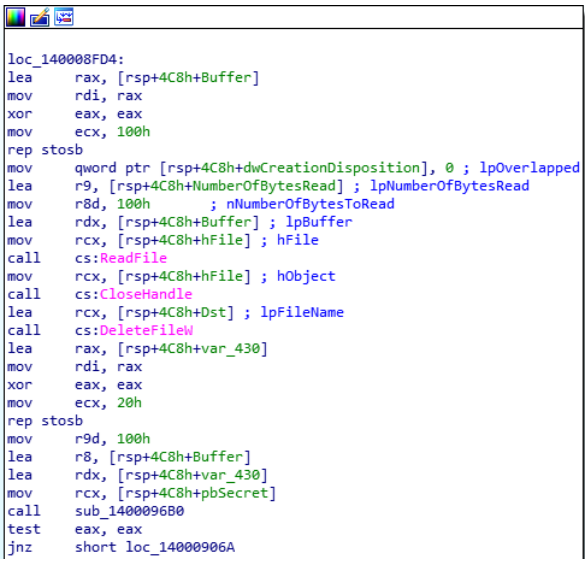




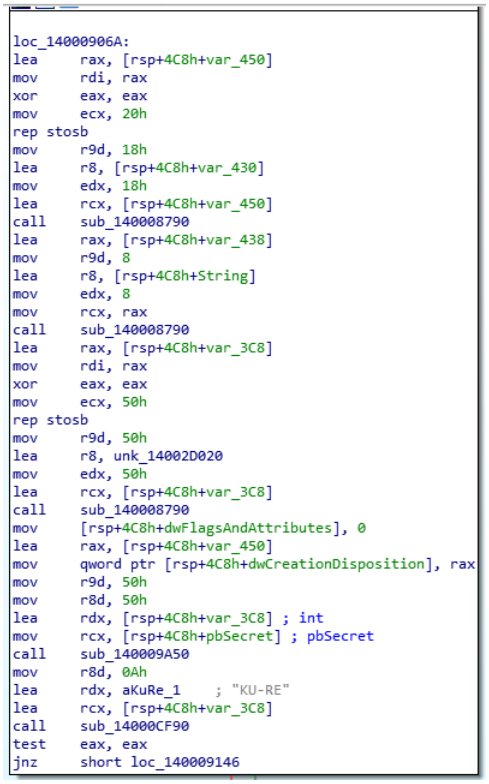
- 입력받은 flag의 길이가0xA보다 작거나 같으면 아래로

```
loc_140008F5C:
lea     rax, [rsp+4C8h+Dst]
mov     rdi, rax
xor     eax, eax
mov     ecx, 208h
rep stosb
mov     r8d, 104h       ; nSize
lea     rdx, [rsp+4C8h+Dst] ; lpDst
lea     rcx, Src        ; "%TEMP%\\KU-RE{k2y}"
call    cs:ExpandEnvironmentStringsW
mov     [rsp+4C8h+hTemplateFile], 0 ; hTemplateFile
mov     [rsp+4C8h+dwFlagsAndAttributes], 80h ; dwFlagsAndAttributes
mov     [rsp+4C8h+dwCreationDisposition], 3 ; dwCreationDisposition
xor     r9d, r9d        ; lpSecurityAttributes
mov     r8d, 1          ; dwShareMode
mov     edx, 80000000h  ; dwDesiredAccess
lea     rcx, [rsp+4C8h+Dst] ; lpFileName
call    cs:CreateFileW
mov     [rsp+4C8h+hFile], rax
cmp     [rsp+4C8h+hFile], 0FFFFFFFFFFFFFFFFh
jnz     short loc_140008FD4
```

- hFile이 0xffff가 아니므로 아래로 넘어간다.



```
loc_140008FD4:
lea     rax, [rsp+4C8h+Buffer]
mov     rdi, rax
xor     eax, eax
mov     ecx, 100h
rep stosb
mov     qword ptr [rsp+4C8h+dwCreationDisposition], 0 ; lpOverlapped
lea     r9, [rsp+4C8h+NumberOfBytesRead] ; lpNumberOfBytesRead
mov     r8d, 100h       ; nNumberOfBytesToRead
lea     rdx, [rsp+4C8h+Buffer] ; lpBuffer
mov     rcx, [rsp+4C8h+hFile] ; hFile
call    cs:ReadFile
mov     rcx, [rsp+4C8h+hFile] ; hObject
call    cs:CloseHandle
lea     rcx, [rsp+4C8h+Dst] ; lpFileName
call    cs:DeleteFileW
lea     rax, [rsp+4C8h+var_430]
mov     rdi, rax
xor     eax, eax
mov     ecx, 20h
rep stosb
mov     r9d, 100h
lea     r8, [rsp+4C8h+Buffer]
lea     rdx, [rsp+4C8h+var_430]
mov     rcx, [rsp+4C8h+pbSecret]
call    sub_1400096B0
test    eax, eax
jnz     short loc_14000906A
```



```
loc_14000906A:
lea     rax, [rsp+4C8h+var_450]
mov     rdi, rax
xor     eax, eax
mov     ecx, 20h
rep stosb
mov     r9d, 18h
lea     r8, [rsp+4C8h+var_430]
mov     edx, 18h
lea     rcx, [rsp+4C8h+var_450]
call    sub_140008790
lea     rax, [rsp+4C8h+var_438]
mov     r9d, 8
lea     r8, [rsp+4C8h+String]
mov     edx, 8
mov     rcx, rax
call    sub_140008790
lea     rax, [rsp+4C8h+var_3C8]
mov     rdi, rax
xor     eax, eax
mov     ecx, 50h
rep stosb
mov     r9d, 50h
lea     r8, unk_14002D020
mov     edx, 50h
lea     rcx, [rsp+4C8h+var_3C8]
call    sub_140008790
mov     [rsp+4C8h+dwFlagsAndAttributes], 0
lea     rax, [rsp+4C8h+var_450]
mov     qword ptr [rsp+4C8h+dwCreationDisposition], rax
mov     r9d, 50h
mov     r8d, 50h
lea     rdx, [rsp+4C8h+var_3C8] ; int
mov     rcx, [rsp+4C8h+pbSecret] ; pbSecret
call    sub_140009A50
mov     r8d, 0Ah
lea     rdx, aKuRe_1    ; "KU-RE"
lea     rcx, [rsp+4C8h+var_3C8]
call    sub_14000CF90
test    eax, eax
jnz     short loc_140009146
```

- jnz가 맞으면 안되니까, eax는 0이 되어야 한다.
- sub_14000cf90은 memcmp함수이다. 따라서 memcmp(local_3c8, “KU-RE”, 10) 결과 eax=0이 되어야 한다.
- 즉, flag의 길이는 0xb보다 작고(조건1), hFile(KU-RE{k2y}를 CreateFile)에서 오류가 없었으며(조건2), FUN_1400096b0의 반환값이 0이 아니면(조건3) 기본 조건이 성립하게 된다. (local_408은 flag값)

FUN_140008e60

```
void FUN_140008e60(longlong param_1)

{
  size_t sVar1;
  LPCWSTR *ppWVar2;
  HANDLE hFile;
  undefined8 uVar3;
  longlong lVar4;
  undefined1 *puVar5;
  undefined *puVar6;
  char *pcVar7;
  WCHAR *pWVar8;
  BYTE *pBVar9;
  UCHAR *pUVar10;
  undefined auStackY_4c8 [32];
  DWORD local_480 [2];
  ulonglong *local_478;
  undefined8 *local_470;
  LPCWSTR *local_468;
  ulonglong local_460;
  undefined8 local_458;
  UCHAR local_450 [24];
  undefined local_438 [8];
  BYTE local_430 [40];
  char local_408 [64];
  UCHAR local_3c8 [80];
  undefined local_378 [64];
  BYTE local_338 [256];
  WCHAR local_238 [264];
  ulonglong local_28;

  local_28 = DAT_14002d080 ^ (ulonglong)auStackY_4c8;
  puVar5 = &DAT_14001ff68;
  puVar6 = local_378;
  for (lVar4 = 0x34; lVar4 != 0; lVar4 = lVar4 + -1) {
    *puVar6 = *puVar5;
    puVar5 = puVar5 + 1;
    puVar6 = puVar6 + 1;
  }
  pcVar7 = local_408;
  for (lVar4 = 0x40; lVar4 != 0; lVar4 = lVar4 + -1) {
    *pcVar7 = '\0';
```

```
    pcVar7 = pcVar7 + 1;
  }
  GetDlgItemTextA(*(HWND *)(param_1 + 8),0x3ed,local_408,0x40);
  if (local_408[0] != '\0') {
    sVar1 = strlen(local_408);
    if (sVar1 < 0xb) {
      pWVar8 = local_238;
      for (lVar4 = 0x208; lVar4 != 0; lVar4 = lVar4 + -1) {
        *(undefined *)pWVar8 = 0;
        pWVar8 = (WCHAR *)((longlong)pWVar8 + 1);
      }
      ExpandEnvironmentStringsW(L"%TEMP%\\KU-RE{k2y}",local_238,0x104);
      hFile = CreateFileW(local_238,0x80000000,1,(LPSECURITY_ATTRIBUTES)0x0,3,0x80,(HANDLE)0x0);
      if (hFile != (HANDLE)0xffffffffffffffff) {
        pBVar9 = local_338;
        for (lVar4 = 0x100; lVar4 != 0; lVar4 = lVar4 + -1) {
          *pBVar9 = '\0';
          pBVar9 = pBVar9 + 1;
        }
        ReadFile(hFile,local_338,0x100,local_480,(LPOVERLAPPED)0x0);
        CloseHandle(hFile);
        DeleteFileW(local_238);
        pBVar9 = local_430;
        for (lVar4 = 0x20; lVar4 != 0; lVar4 = lVar4 + -1) {
          *pBVar9 = '\0';
          pBVar9 = pBVar9 + 1;
        }
        uVar3 = FUN_1400096b0(param_1,local_430,local_338,0x100);
        if ((int)uVar3 != 0) {
          pUVar10 = local_450;
          for (lVar4 = 0x20; lVar4 != 0; lVar4 = lVar4 + -1) {
            *pUVar10 = '\0';
            pUVar10 = pUVar10 + 1;
          }
          memcpy_s(local_450,0x18,local_430,0x18);
          memcpy_s(local_438,8,local_408,8);
          pUVar10 = local_3c8;
          for (lVar4 = 0x50; lVar4 != 0; lVar4 = lVar4 + -1) {
            *pUVar10 = '\0';
            pUVar10 = pUVar10 + 1;
          }
          memcpy_s(local_3c8,0x50,&DAT_14002d020,0x50);
          FUN_140009a50(param_1,local_3c8,0x50,0x50,local_450,0);
          memcmp(local_3c8,L"KU-RE",10);
        }
      }
    }
    else {
      local_478 = &local_460;
      local_470 = &local_458;
      local_468 = (LPCWSTR *)FUN_140009e80(local_478,0x80);
      ppWVar2 = (LPCWSTR *)FUN_140002160(local_470,L"Try again. It\'s too long. :(");
      FUN_140009ef0((HWND)0x0,*ppWVar2,*local_468,0x20);
    }
  }
  FUN_14000b800(local_28 ^ (ulonglong)auStackY_4c8);
  return;
}
```

- 조건 1은 내가 하기 나름
- 조건 2는 그냥 통과
- 조건 3은 이제 알아보면 된다.
- 조건을 맞추면 궁극적으로 초록색으로 된 memcmp의 결과가 같음이어야한다. 즉, local_3c8이 가리키는 값이 KU-RE여야 한다.
- 위의 `FUN_140009a50`으로 `local_3c8`이 만들어져서 KU-RE와 비교될 것 같다.

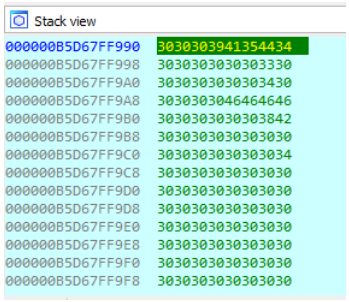
FUN_1400096b0

- `FUN_140008e60`에서 이 함수를 부르고 있다.

  ```
  uVars = FUN_1400096b0(param_1, local_430, local_338, 0x100); //
  ```

  - `local_430`은 이름이 pbsecret이고,
  - `local_338`은 buffer로 r8이 가리키는 곳을 보면 아래와 같다.

    위에서 Readfile으로 local_338에 KU-RE{k2y}의 0x100만큼 가져오는 것을 봤다. 아래 있는 스택부분이 KU-RE{k2y}의 앞부분 0x100만큼이다.




```
undefined8 FUN_1400096b0(undefined8 param_1,BYTE *param_2,BYTE *param_3,DWORD param_4)
//param2=localk_430(pbsecret), param3=local_338, param4=0x100
{
  BOOL BVar1;
  undefined8 uVar2;
  DWORD local_28 [2];
  HCRYPTPROV local_20;
  HCRYPTHASH local_18 [3];

  BVar1 = CryptAcquireContextW
                    (&local_20,(LPCWSTR)0x0,L"Microsoft Enhanced RSA and AES Cryptographic Provider"
                     ,0x18,0);
  if ((BVar1 == 0) &&
     (BVar1 = CryptAcquireContextW
                        (&local_20,(LPCWSTR)0x0,
                         L"Microsoft Enhanced RSA and AES Cryptographic Provider",0x18,8),
      BVar1 == 0)) {
    return 0;
  }
  BVar1 = CryptCreateHash(local_20,0x800c,0,0,local_18);
  if (BVar1 == 0) {
    uVar2 = 0;
  }
  else {
    BVar1 = CryptHashData(local_18[0],param_3,param_4,0);
    if (BVar1 == 0) {
      uVar2 = 0;
    }
    else {
      local_28[0] = 0x20;
      CryptGetHashParam(local_18[0],2,param_2,local_28,0);
      CryptReleaseContext(local_20,0);
      uVar2 = 1;
    }
  }
  return uVar2;
}
```

- **FUN_1400096b0**을 거치면, `param_2(local_430, pbsecret)`에 `param_3(local_338)`에 대한 해쉬값을 저장하는 것을 알 수 있다.

```
uVar3 = FUN_1400096b0(param_1, local_430, local_338. 0x100);
```

- CypyHashData로 `param_3` 0x100에 대한 핸들을 `local_18`에 주고, CryptHashParam으로 `param_2`에 해쉬값을 넣는다.

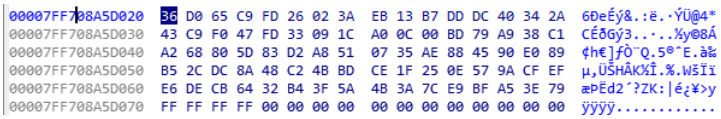
다시 FUN_140008e60으로

- FUN_1400096b0에서 local_430(pbsecret)에 해쉬값을 저장했다.
- 그리고, local_430의 0x18만큼 `local_450`에 복사한다. local_450=해쉬값의 0x18바이트

  local_408의 8만큼 `local_438`에 복사한다. local_438=입력받은 flag의 8바이트. 따라서 flag는 8글자로 보인다.

  DAT_14002d020의 0x50만큼 `local_3c8`에 복사한다.

  일단 `DAT_14002d020`을 들어가보면 아래와 같다.



- 그리고 FUN_140009a50을 불러서 `local_3c8`을 얻는다.

FUN_140009a50

이 함수는 아래 코드와같이 불려진다.

```
FUN_140009a50(param_1, local_3c8, 0x50, 0x50, local_450, 0);
//local_3c8 = DAT_14002d020의 0x50만큼
//local_450 = 해쉬값의 0x18만큼
```

- 이 함수에 암호화와 관련된 함수들이 잔뜩 있다.
- flag는 Bcrypt를 이용해서 구하는 것으로 추정된다.
- 결론적으로 param_2가 KU-RE가 되게 해야한다.

```
void FUN_140009a50(undefined8 param_1,PUCHAR param_2,ULONG param_3,undefined8 param_4,PUCHAR param_5
                  ,int param_6)

{
  NTSTATUS NVar1;
  undefined auStackY_98 [32];
  ULONG local_44;
  ULONG local_40 [2];
  BCRYPT_KEY_HANDLE local_38;
  BCRYPT_HANDLE local_30;
  UCHAR local_28;
  undefined local_27;
  undefined local_26;
  undefined local_25;
  undefined local_24;
  undefined local_23;
  undefined local_22;
  undefined local_21;
  undefined local_20;
  undefined local_1f;
  undefined local_1e;
  undefined local_1d;
  undefined local_1c;
  undefined local_1b;
  undefined local_1a;
  undefined local_19;
  ulonglong local_18;

  local_18 = DAT_14002d080 ^ (ulonglong)auStackY_98;
  local_30 = (BCRYPT_HANDLE)0x0;
  local_38 = (BCRYPT_KEY_HANDLE)0x0;
  local_28 = 0xd5;
  local_27 = 0xbb;
  local_26 = 0x5f;
  local_25 = 0x7f;
  local_24 = 0x49;
  local_23 = 0xcd;
  local_22 = 0x4d;
  local_21 = 0x3a;
  local_20 = 0xc5;
  local_1f = 0x70;
  local_1e = 0xaf;
  local_1d = 0xf1;
  local_1c = 0xdf;
  local_1b = 0xee;
  local_1a = 0xd8;
  local_19 = 0xda;
  NVar1 = BCryptOpenAlgorithmProvider(&local_30,L"AES",(LPCWSTR)0x0,0);
  if (NVar1 == 0) {
    BCryptSetProperty(local_30,L"ChainingMode",(PUCHAR)L"ChainingModeCBC",0x20,0);
    BCryptGenerateSymmetricKey(local_30,&local_38,(PUCHAR)0x0,0,param_5,0x20,0);
    //local_38은 phkey, param_5(해쉬값의 0x18)는 키를 만들 재료(포인터, pbsecret)
    if (local_38 != (BCRYPT_KEY_HANDLE)0x0) {
      if (param_6 == 0) {
        local_40[0] = 0;
        BCryptDecrypt(local_38,param_2,param_3,(void *)0x0,&local_28,0x10,(PUCHAR)0x0,0,local_40,1);
        BCryptDecrypt(local_38,param_2,param_3,(void *)0x0,&local_28,0x10,param_2,local_40[0],
                      local_40,1);
        //Buf1을 phkey로 복호화한다.
      }
      else {
        local_44 = 0;
        BCryptEncrypt(local_38,param_2,param_3,(void *)0x0,&local_28,0x10,(PUCHAR)0x0,0,&local_44,1)
        ;
        BCryptEncrypt(local_38,param_2,param_3,(void *)0x0,&local_28,0x10,param_2,local_44,&local_44
                      ,1);
      }
    }
    if (local_38 != (BCRYPT_KEY_HANDLE)0x0) {
      BCryptDestroyKey(local_38);
    }
    if (local_30 != (BCRYPT_ALG_HANDLE)0x0) {
      BCryptCloseAlgorithmProvider(local_30,0);
    }
  }
  FUN_14000b800(local_18 ^ (ulonglong)auStackY_98);
  return;
}
```

- 위 Bcrypt는 AES-CBC 알고리즘을 사용한다.

- **BcryptGeneraateSymmetricKey**로 `param_5`를 이용해 phkey `local_38`을 생성한다.
- param_6=0이기 때문에 BcryptDecrypt만 실행한다.
- `param_2`를 `phkey(local_38)`을 사용해 decrypt를 하면 `param_2(local_3c8)`에 복호화된 값이 들어간다. 이 값이 KU-RE이면 된다.
- 그러면… `param_2`는 정해진 값이기 때문에, phkey에 따라 복호화 결과가 달라진다.
- 하지만 phkey는 분명히 입력받은 flag값을 이용해서 만들어질텐데, 입력으로는 param_5인 local_450으로 만들어지는지 모르겠다…… 한참을 고민하다가 왜그런지 알게됐다.

  flag 입력값은 FUN_140008e60에서 local_408이고, copy되어서 local_438에도 담겨있었다. 그런데 FUN_140009a50에 들어가는 인자로는 local_450밖에 없었다.

  그리고 다시 보면, FUN_140009a50에서 키를 만들때 param_5(local_450) 20바이트를 사용했다. local_450은 0x18바이트인데 왜 0x20바이트나 사용하나 했더니만. local_438이 local_450의 바로 옆에 붙어있던 것이었다.

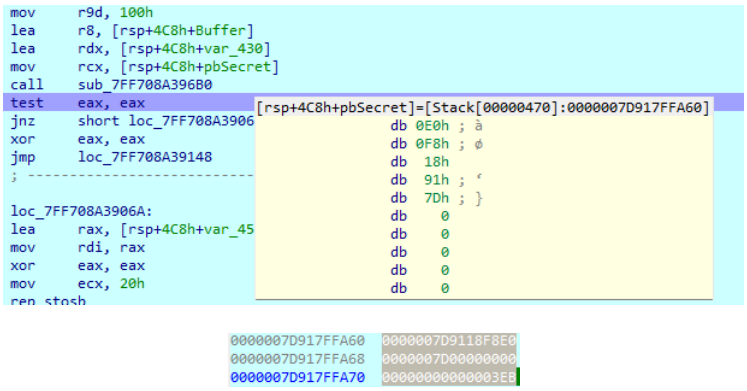
  그래서 local_450 0x20만큼 사용한다는 것은, local_430+local_408을 쓴다는 것이다.

  즉, 키를 만드는데 이용되는 값은 `해쉬값0x18+입력한flag0x8`인 것이다.
- 결론적으로 local_338(고정)에 대한 해쉬값의 0x18바이트를 가져오고, flag 입력값 0x8바이트를 가져와서 phkey를 만든다.

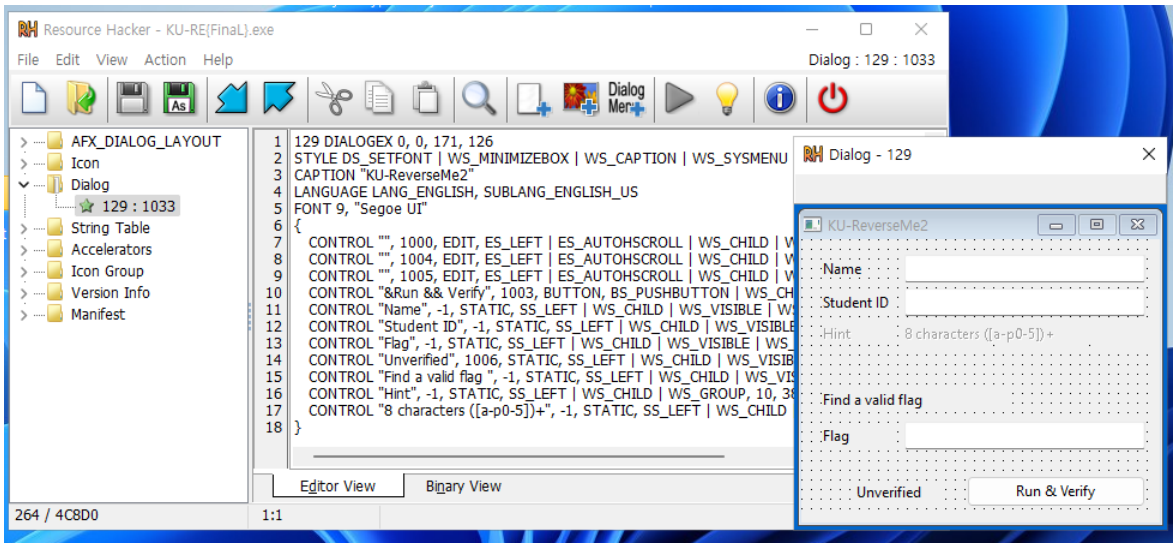
  그리고, local_3c8(상수)을 phkey로 복호화하면 KU-RE가 나온다.

```
local_3c8(복호화할 값)
36 D0 65 C9 FD 26 02 3A  EB 13 B7 DD DC 40 34 2A
43 C9 F0 47 FD 33 09 1C  A0 0C 00 BD 79 A9 38 C1
A2 68 80 5D 83 D2 A8 51  07 35 AE 88 45 90 E0 89
B5 2C DC 8A 48 C2 4B BD  CE 1F 25 0E 57 9A CF EF
E6 DE CB 64 32 B4 3F 5A  4B 3A 7C E9 BF A5 3E 79
```

- 이제, 이런 phkey를 생성하면 된다…



- 그리고 추가로 Resource Hacker에 있는 힌트를 통해 a-p 0-5로 이루어진 flag라는 것을 알게 되었다.




```
from Crypto.Cipher import AES
import hashlib import sha256
```

```
with open('KU-RE{k2y}', 'rb') as f:
  key1=f.read()[0:255]
key2 = sha256(buffer).digest()[0:23]
bufs = bytes({0x36, 0xD0, 0x65, 0xC9, 0xFD, 0x26, 0x02, 0x3A, 0xEB, 0x13, 0xB7, 0xDD, 0xDC, 0x40, 0x34, 0x2A, 0x43, 0xC9, 0xF0, 0x47, 0xFD, 0x33, 0x0
flagrange = "abcdefghijklmnop012345"

for a in flagrange:
  for b in flagrange:
    for c in flagrange:
      for d in flagrange:
        for e in flagrange:
          flag = "2" + a + b + c + d + e + "1" + "3"
          phkey = key1 + flag.encode()
          aes = AES.new(phkey, AES.MODE_CBC, iv=iv)
          if(aes.decrypt(bufs) == "KU-RE")
            print(flag)
```

- flag는 2로 시작하고 13으로 끝나는 8자리이다. 그래서 bruteforce로 복호화 결과가 KU-RE인 flag를 찾고 싶었다.
- 하지만 코드를 크리스마스이브 아침부터 저녁까지 작성해도 계속 오류가 났다.. 그래서 flag를 찾지 못했다.

KU-RE{k2y}의 앞 0x100바이트를 해쉬화하고 0x18바이트를 뽑는다. 그리고, 거기에 내가 입력한 flag 8바이트를 더한다.

flag 가운데 5바이트는 불분명하기 때문에 for문으로 돌리고서, 그 flag를 통한 키를 이용해서 bufs를 복호화한다. 이 값이 KU-RE이면 그 flag가 정답인 것이다.

이렇게 구하는 법은 잘 아니까……점수 조금이라도 주세요….교수님….정말 어려웠습니다ㅠㅠㅠㅠㅠㅠㅠㅠㅠㅠㅠㅠ