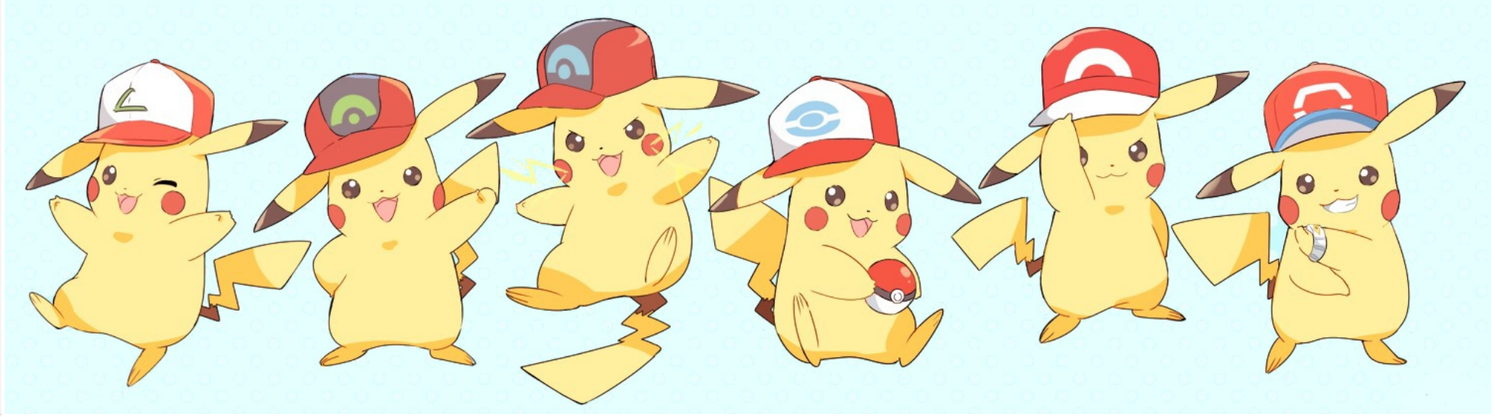


IDA 对 switch 的识别

2020-8-8

security

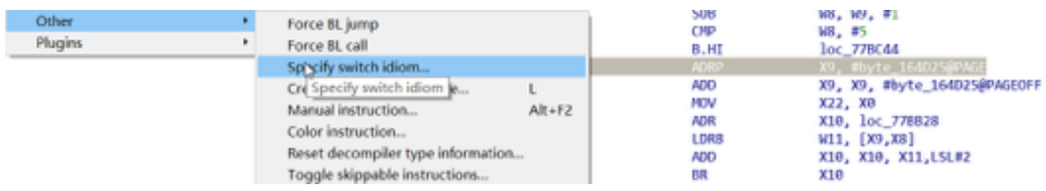
在分析crash问题时，可以下载带有符号信息的动态链接库进行分析，结合源代码可以知道是由于哪条语句或哪个变量引起的。

由于附带有符号信息，所以使用 IDA 的 F5 功能之后和源码对照，可以非常迅速的定位到问题。但是一次分析却卡住了， F5 之后出现了 JUMPOUT 的语句，如下图所示。对照着源代码查看汇编代码，并查了一些资料后，发现代码里有对同一个变量进行了多次 if 判断，并且判断的值接近，所以编译器就采用了“跳转表”的方式加快执行速度（与 switch 机制一样）。

```
if ( (unsigned int)v6 <= 5 )
  JUMPOUT(__CS__, (char *)&loc_77BB28 + 4 * byte_164D25[v6]);
```

switch 修复

查到资料果然 IDA 有相应的处理机制：在跳转表寻址的那条语句上面，点击 Edit - Other - Specify switch idiom 。



在跳出的设置窗口中，可以发现 Address of jump table 和 Start of the switch idiom 的参数已经通过我们指定的指令自动填写了。可以结合 F5 的 JUMPOUT 的语句填写以下参数：

Size of table element ，因为是 char 类型的，所以填写 1 。

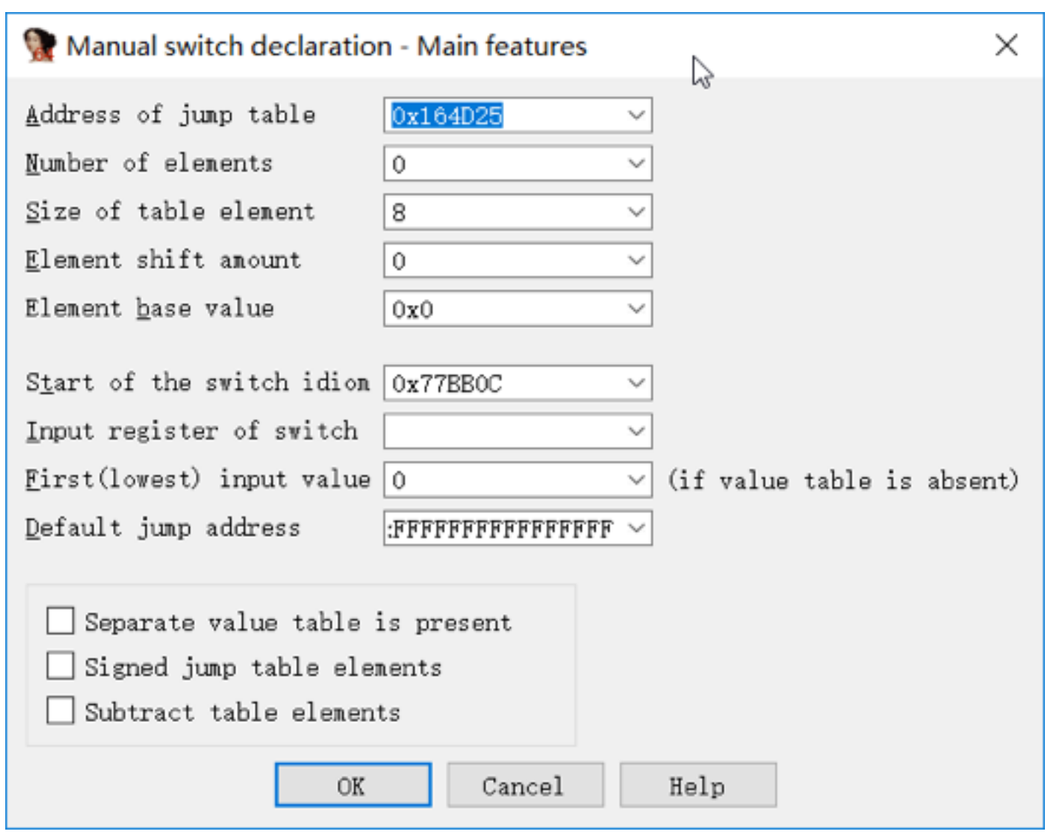
Element shift amount ，因为对查表元素要 *4 ，所以填写 2 。

Element base value ，因为有基地址，这边填写 0x77BB28 。

其余需要填写的参数：

Number of elements ，表元素的个数。结合上下文或源代码，此示例填写 6 。

Input register of switch ，用于 index 的寄存器。结合上下文，此示例填写 x11 。



其余参数还不知道具体的用途，在上述参数填写好后，再按 F5 （有时候我需要重启一下才会生效）就会发现 JUMPOUT 语句消失了，并且反汇编了各条判断分支。

hanhan的学习博客

现在开始写。

路漫漫其修远兮，吾将上下而求索。

搜索

请输入关键字 搜索

点赞排名

简单工厂模式
点赞(0)

[OpenGL快速入门] 3.图元的绘制
点赞(0)

[OpenGL快速入门] 2.搭建OpenGL渲染环境
点赞(0)

[OpenGL快速入门] 1.创建Windows窗口
点赞(0)

面向对象设计原则
点赞(0)