用高级语言实现指令自修改

窦路
alexdou@e28.com

[摘要] 所谓指令自修改即程序在运行时产生后面即将执行的指令的技术。自修改指令在诸如软件加密，程序装载等方面均有应用。本文举例说明如何用高级语言实现针对 C++程序的自修改指令。

1. 基本方法

在输入法所依赖的开发库中，中文成语的封装库对其所支持语言采用了缺省为简体中文的硬编码方式，因此无法通过 C++方法调用来设置期望支持的语言，例如繁体中文。而相关的负责工程师暂时没有精力来增加这个设置语言的接口，于是我们考虑是否可以通过运行时修改指令的方式来实现在现有二进制库的基础上动态设置所期望支持的语言。

通过分析源代码及反汇编代码发现

```
                        ; CODE XREF: _ZN18YLPhraseCandidates10GetPhrasesEPt+20↑j
BL      _ZN18YLPhraseCandidates17ClearResultBufferEv ; PIC mode

MOV     R4, R6
ADD     R1, R5, #0x22C
STRH    R4, [R1]
LDR     R0, [R5,#0x228] ; <suspicious>
MOV     LR, #0x100 ; <suspicious>
ADD     R2, R5, #0x214
MOV     R1, R6
MOV     R3, R7
STR     LR, [SP,#0x1C+var_1C]
STR     R4, [SP,#0x1C+var_18]
BL      PPFRAS_getword  ; PIC mode
```

在 ***YLPhraseCandidates::GetPhrases()*** 方法中，***mov*** 指令来设置了语言 ID。

通过查阅 ARM 指令格式和分析二进制库的十六进制数据了解到，***mov*** 指令的最低 8 位为立即数，本例中，立即数为语言的 ID。也就是说，只要能设置该指令的低 8 位，就能设置所期望的语言 ID。

指令在内存中的地址可以通过计算其与函数首地址的偏移量来确定，而函数地址可以通过其函数名作为符号来获取。本例中，指令偏移量为 ***0x3a0-0x370***。只要知道了指令的地址，就可以通过简单地赋值操作来修改它。

基于此，动态修改指令的流程可以用如下伪代码表示：

```
void dynamicInstruction() {
```