MethodID 本身存在的问题

一、前言

路漫漫其修远兮，吾将上下而求索。

二、描述

GetMethodID 获取一个方法的 *jmethodID*

```
jmethodID GetMethodID(JNIEnv *env, jclass clazz,const char *name, const char
*sig);
```

jmethodID 到底是什么?

> 方法 ID 是指向内部运行时数据结构的指针。

在ART虚拟机中，jmethodID就是 *ArtMethod**，在Davlik下是 *Method** ，先不管 *ArtMethod* 或者 *Method* 是个什么，*jmethodID* 不就是内存里一个地址嘛。

那么Unidbg中jmethodID是什么呢——是方法签名这个字符串本身的hashCode。

```
int getMethodID(String methodName, String args) {
    String signature = getClassName() + "->" + methodName + args;
    int hash = signature.hashCode();
    if (log.isDebugEnabled()) {
        log.debug("getMethodID signature=" + signature + ", hash=0x" +
Long.toHexString(hash));
    }
    if (vm.jni == null || vm.jni.acceptMethod(this, signature, false)) {
        if (!methodMap.containsKey(hash)) {
            methodMap.put(hash, new DvmMethod(this, methodName, args, false));
        }
        return hash;
    } else {
        return 0;
    }
}
```

以 *Context* 类的 *getPackageName* 方法为例，Unidbg中的方法签名是 **android/content/Context->getPackageName()Ljava/lang/String;** 这个字符串的 *hashCode* 即为 *Unidbg* 中该方法的 *jmethodID* 。

> String类中的hashCode计算方法是以31为权，每一位为字符的ASCII值进行运算，用自然溢出来等效取模。
>
> 哈希计算公式：s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1]

因此我们可以验证获取的 *jmethodID* 是不是这么构造出来的，如果是，那大概率是 *Unidbg* 的环境。

除此之外，*ArtMethod* 作为一个数据结构，一般四字节自然对齐，但 *hashCode* 显然没有这个要求，这又是一个差异点。

三、尾声

无。