逆向gfx tool第四天

笔记本:	gfx逆向		
创建时间:	2019/11/13 21:53	**更新时间:**	2019/11/18 11:29
作者:	381643188@qq.com		

配置文件生成工具

见<Android 如何使用自定义工具制作gfx工具配置脚本>

又有问题

发现很多fps 90和fps120不能直接生效，怀疑原生工具是修改了其他配置文件。
遍历/sdcard/Android/data/com.tencent.tmgp.pubgmhd目录下的所有文件，使用lastmodifytime进行排序。发现有四个文件被修改了。

```
file =
/sdcard/Android/data/com.tencent.tmgp.pubgmhd/files/UE4Game/ShadowTrackerExtra/Sh
 modify time = 2019-11-13-21:39:31
file =
/sdcard/Android/data/com.tencent.tmgp.pubgmhd/files/UE4Game/ShadowTrackerExtra/Sh
 modify time = 2019-11-13-21:39:31
file =
/sdcard/Android/data/com.tencent.tmgp.pubgmhd/files/UE4Game/ShadowTrackerExtra/Sh
 modify time = 2019-11-13-21:39:31
file =
/sdcard/Android/data/com.tencent.tmgp.pubgmhd/files/UE4Game/ShadowTrackerExtra/Sh
 modify time = 2019-11-13-21:39:31
```

尝试方向

1、jeb有良好的反编译能力，看看能否直接发现。
2、如果修改其他文件的，通过第二天的破解可以知道，所有的路径都是由数字直接生成的，所以还有个方向是使用xposed hook解密路径，这样就可以获取到是哪里进行了修改。

验证后：

1、使用文件对比的方式发现Active.sav被修改了。

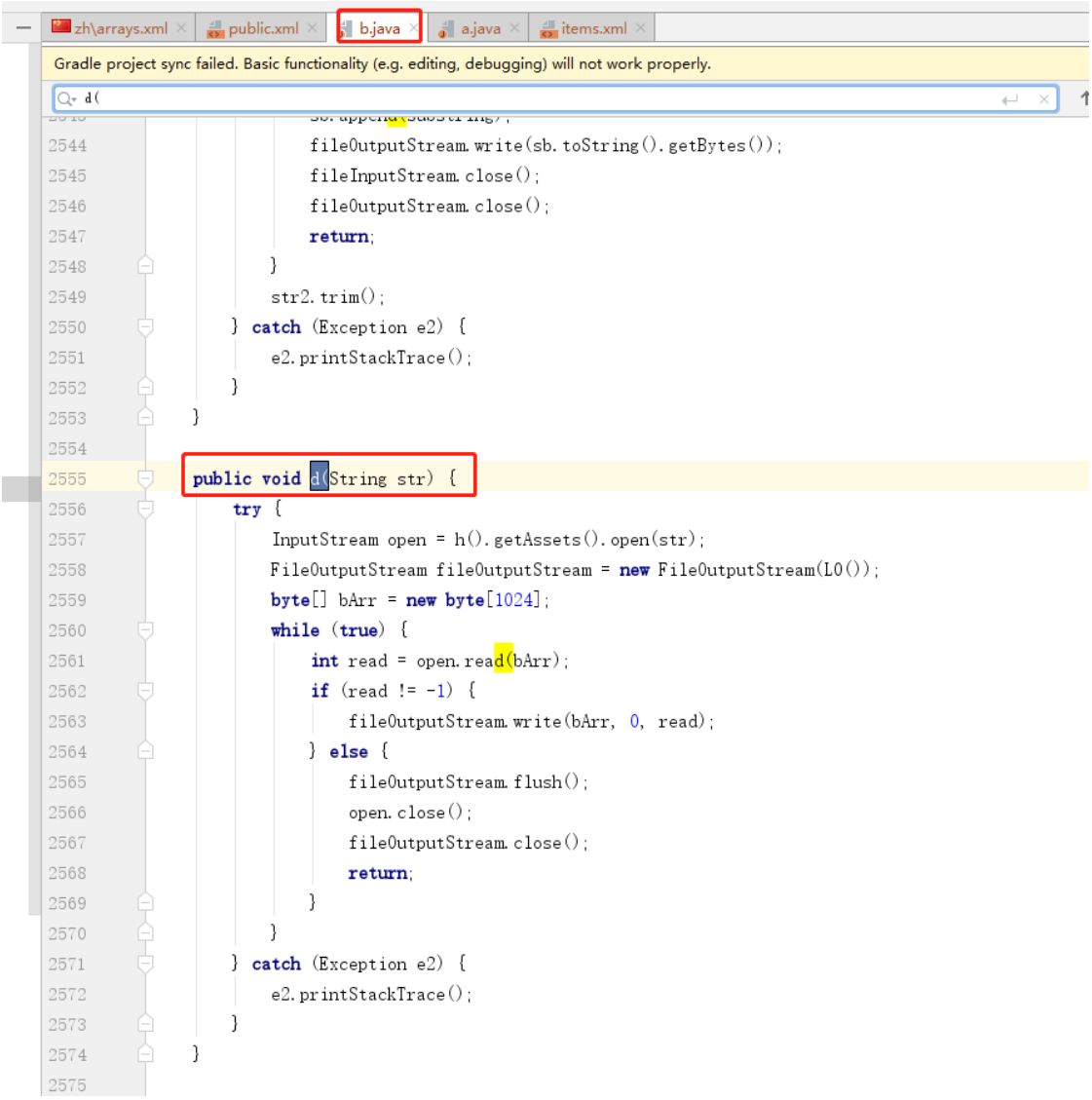
2、通过替换可知，90fps和120fps无法修改确实是由于这个文件导致的。

使用jeb和jadx混合破解

发现gfxtook进行了版本升级，所以我们对apk进行重新破解，老的保存在jadx_base_gradle中。新的保存在jadx_new_version中。

并且数字转文字的秘钥发生了变化。

并且发现确实对Active.sav进行了修改，修改的方式为文件替换。



接下来我们需要确认替换的源文件有哪些。

仔细分析

为了确保不会出其他异常，这里我们对一个函数进行全链路分析，排查逻辑中可能存在的内部联系。
比如，当我们选择第一个选项的时候

```
public void u0() {
......
    if(v5 != 0) {
    int v6 = 7;
    int v7 = 4;
    int v8 = 3;
    int v9 = 2;
    if(v5 == 1) {
```

```
        v5 = this.e1;
        if(v5 != 1 && v5 != v8 && v5 != v7) {
            if(v5 == 5) {
            }
            else if(v5 == v6) {
                this.a(v0, a.a(170));
                this.a(v1, a.a(171));
                this.a(v2, a.a(172));
                this.a(v3, a.a(173));
                v0_1 = 174;
                goto label_206;
            }
            else if(v5 == v9) {
                this.a(v4, a.a(175));
                this.a(v0, a.a(176));
                this.a(v1, a.a(177));
                this.a(v2, a.a(178));
                this.a(v3, a.a(179));
                if(!this.T0.booleanValue()) {
                    v0_1 = 180;
                    goto label_206;
                }
                else {
                    return;
                }
            }
            else {
                return;
            }
        }
        this.a(v4, a.a(164));
        this.a(v0, a.a(165));
        this.a(v1, a.a(166));
        this.a(v2, a.a(167));
        this.a(v3, a.a(168));
        if(this.T0.booleanValue()) {
            return;
        }
        v0_1 = 169;

    }
}
```

其中关键变量e1和版本有关，默认值是1，e的赋值和函数b(int, int)相关，第一个参数为版本序号，如图从左到右，从上到下分别为1-7。

第二个参数固定为1。从而我们来看e1的值。
Global ： 1
CN版本：2
KR版本：3
VN：4
TW：5
Lite版本：7
BETA版本：6。

可以发现，不同版本写入的内容是不一样的。囧，这下工作量大了。

还有个关键变量T0，这个T0会引起函数直接return，导致下边的逻辑走不到了。T0默认走的false。在函数R0()
中被改变。这个值是"保存控件"，在gfxtool的最下方。

当我们选默认分辨率时，Globa、KR，VN，TW。这几个版本使用y6.sav进行，替换，如果"保存控件"被打开了，就不会再更新Active.sav，那这样部分fps设置要失效?

经过验证，确实失效了，但是可以自动手动更改。

分组设置

哪些属性和版本有关呢，这里需要明确一下。
分辨率：不区分版本。
画质：不区分。
帧数：Global ，KR，VN，TW为一组，Lite版本为一组，CN版本为一组。BETA版本没有设置内容。会额外修改一个文件。
抗锯齿：不区分版本。
风格：不区分版本。
渲染质量：会影响阴影，不区分版本。
阴影：会印象阴影距离，移动阴影，不区分版本。
阴影距离：会影响移动阴影，不区分版本。
效果质量：不区分版本。