

Gikdbg 是 Android 平台的 32 位汇编级调试器，不管是运行 dalvik 虚拟机还是运行本地代码的 art 均可以使用 gikdbg.art 进行程序的二进制调试分析。不同之处在于 dalvik 虚拟机的运行时只能调试 so 动态库，而 art 运行时不仅能调试 so 动态库，还能调试系统镜像 oat，可执行程序 dex 这样的文件。

环境：pc = windows7 x64 phone=Android 4.03（需要 root 权限）

另：需要注意的是，art 的调试环境目前只适配了 4.4.2，android L 系统也只能调试 so。

官网：<http://gikir.com/>

下载页面：<http://gikir.com/product.php>

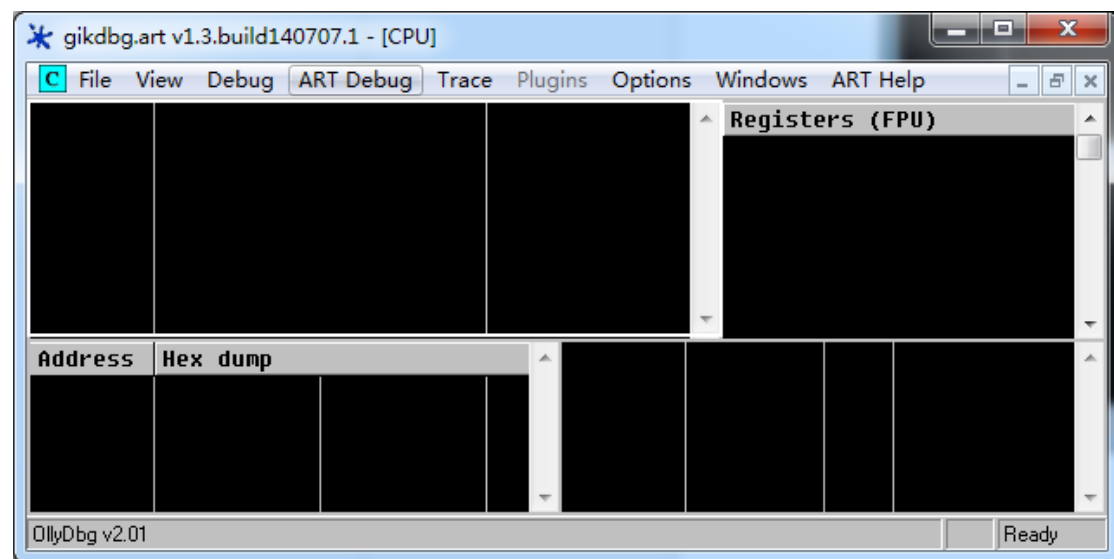
Android 相关的官方例子：<http://gikir.com/blog/?cat=12>

第一步：下载，去官网下载最新版本的 ART 版本的 gikdbg（介绍说以后会统一 ART 和 IOS 这两个版本）

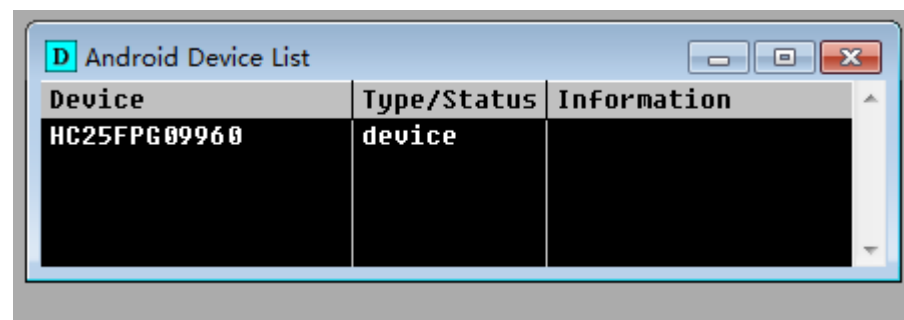
第二步：将压缩包解压，首次运行 gikdbg 的时候会提示你需要安装 gikdbg.apk 服务程序，点击确定后会自动安装（也可以使用 adb install 命令将 gikdbg 根目录 artserver 文件夹下的 gikdbg.apk 安装入手机，在 gikdbg 根目录 artserver 文件夹下有 adb）

第三步：手机运行 gikdbg.apk 并点击 start（手机使用 USB 与 PC 相连）

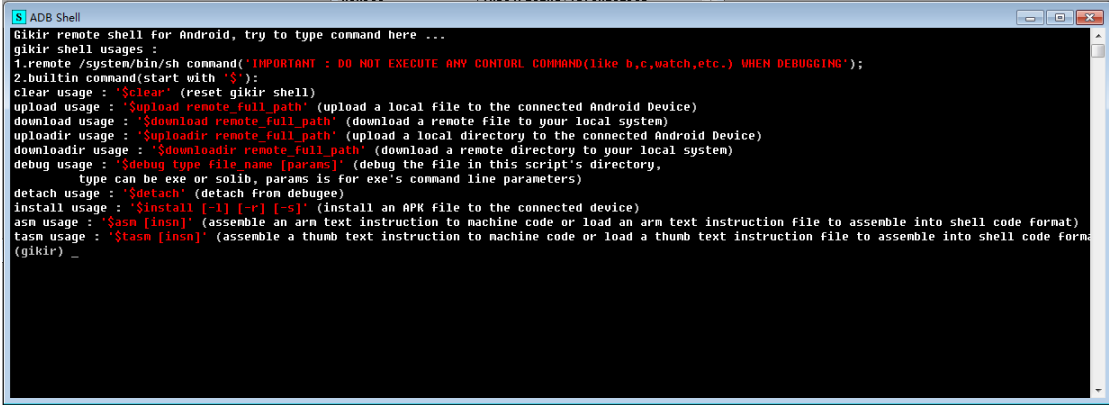
第四步：PC 运行 gikdbg.art.exe，界面如下图：



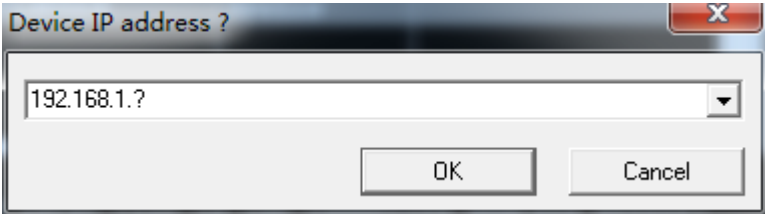
点击 ART Debug->Server->ADB Device 出现下图界面：



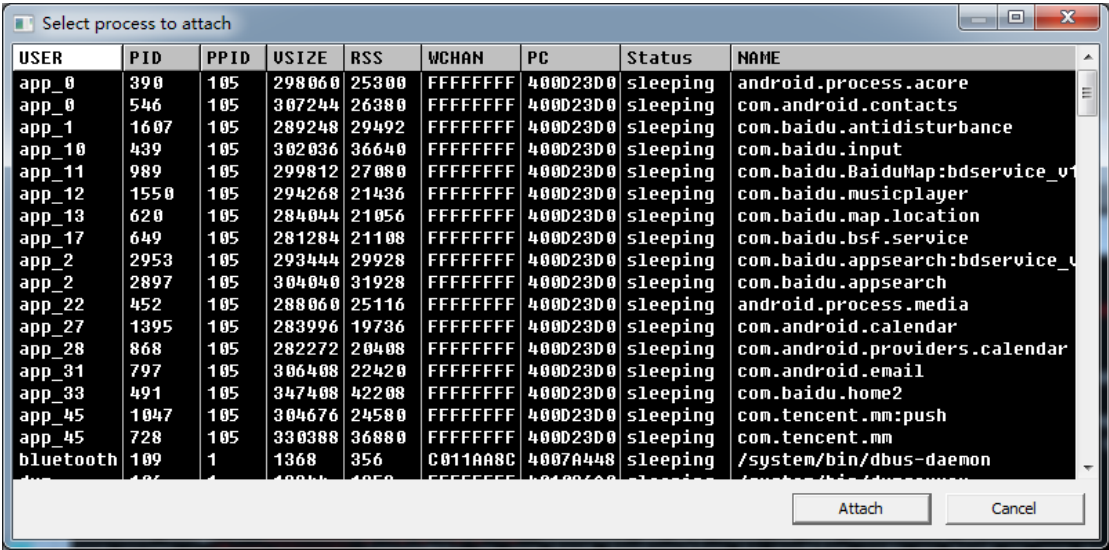
双击选择要调试的设备（也可右键选择 login），出现下图界面表示连接成功。（如果没有出现要选择的设备，可以右键选择 Refresh，如果还没有，请确认连接是否成功）



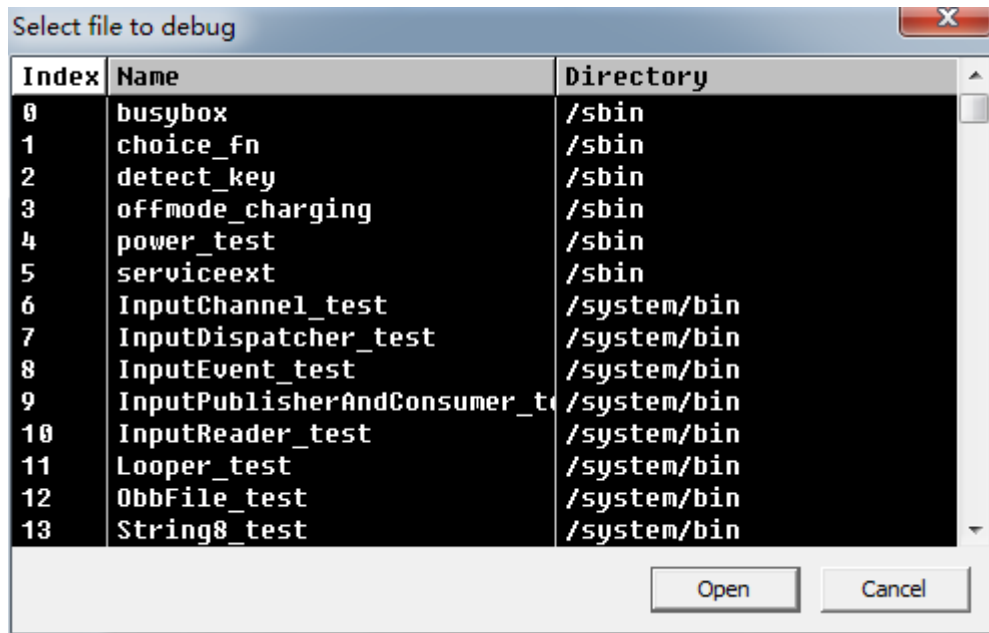
gikdbg 还支持 wifi 调试，点击 ART Debug->Server->Login(WIFI) 出现如下弹出框，输入手机 ip 即可（wifi 调试的时候需要运行手机端的 gikdbg 才行）



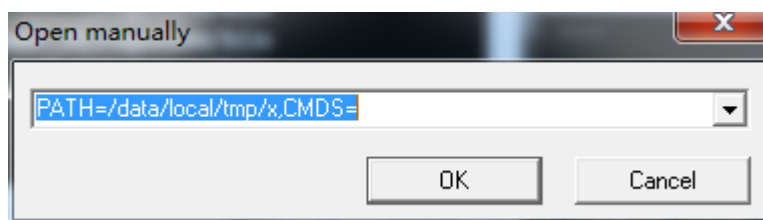
第五步：点击 ART Debug->File->Attach（调试已经运行的进程）



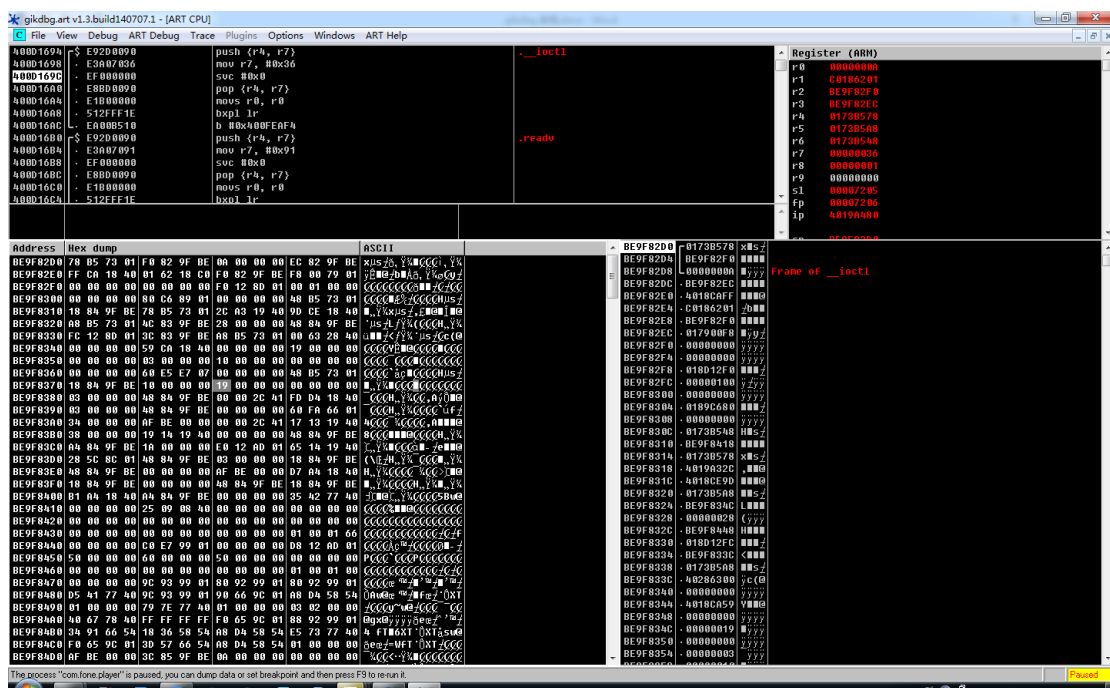
或者 ART Debug->File->Open（调试本地的 bin 程序）



如果在程序列表找不到要调试的文件，可以在执行程序列表右键的 OpenEx 菜单，手动填入程序路径和命令行参数。



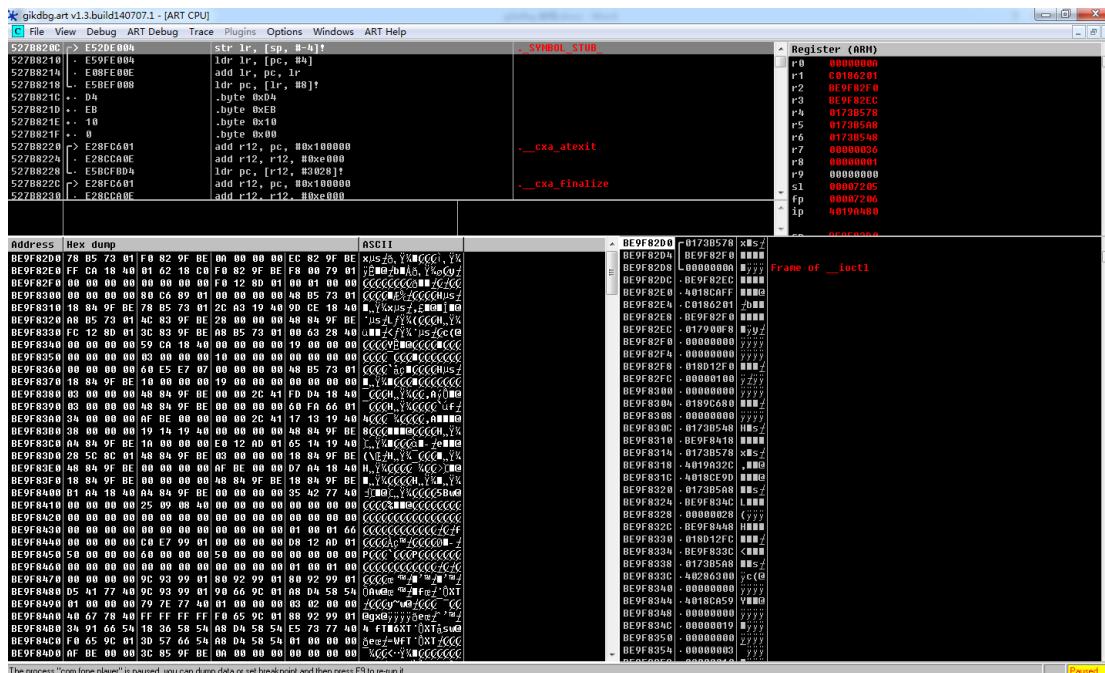
双击选择以后，出现如下界面，开始调试



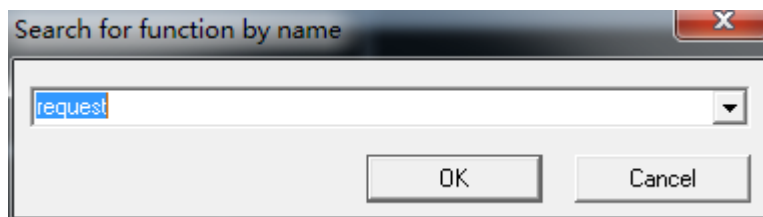
第六步：在 so 库中设置断点，选择 ART Debug->View->Module，出现如下视图

Index	Name	Base	Path
1	app_process	00008000	/system/bin/app_process
2	libutils.so	40006000	/system/lib/libutils.so
3	libexpat.so	40036000	/system/lib/libexpat.so
4	libwpa_client.so	4004A000	/system/lib/libwpa_client.so
5	libhostapd_client.so	4004D000	/system/lib/libhostapd_client.so
6	liblog.so	40059000	/system/lib/liblog.so
7	libnativehelper.so	4005D000	/system/lib/libnativehelper.so
8	libgabi++_so	40093000	/system/lib/libgabi++.so
9	libssl.so	40097000	/system/lib/libssl.so
10	libc.so	400C5000	/system/lib/libc.so
11	libm.so	40125000	/system/lib/libm.so
12	libcutils.so	40130000	/system/lib/libcutils.so
13	libz.so	4015A000	/system/lib/libz.so
14	libbinder.so	40172000	/system/lib/libbinder.so
15	libstlport.so	40198000	/system/lib/libstlport.so

双击选择你想要调试的 so 文件，主界面自动跳转到该 so 的内存入口处



使用 Ctrl+F 进行 export 函数名的搜索



并按 F2 设置断点

527BC390	E92D4830	push {r4, r5, r11, lr}	.jni_fn_core_service_request
527BC394	E28D000C	add r11, sp, #0xc	
527BC398	E24D0801	sub sp, sp, #0x10000	
527BC39C	E24D0D0B	sub sp, sp, #0x2c0	
527BC3A0	E59FCFA8	ldr r12, [pc, #4008]	
527BC3A4	E24B400C	sub r4, r11, #0xc	
527BC3A8	E784000C	str r0, [r4, r12]	
527BC3AC	E59F0E30	ldr r0, [pc, #3632]	
527BC3B0	E24BC00C	sub r12, r11, #0xc	
527BC3B4	E78C1000	str r1, [r12, r0]	
527BC3B8	E59F1F50	ldr r1, [pc, #3920]	
527BC3BC	E24B000C	sub r0, r11, #0xc	
527BC3C0	E7802001	str r2, [r0, r11]	

按 F9 运行至断点处（不要点击 Debug 菜单下的选项，使用快捷键 F9->run; F8->step over; F7->step into; F3->ADB Shell; Alt+F3->GDB Shell; F4->运行到当前选中行; F5->CPU 窗口, F12-> 暂停程序）

附：

ART Debug->View->ELF Data 以及 ART Debug->View->ELF 都可以解析 Android 下的 so 和 bin 文件（ELF 格式）。两者的显示方式不同，Data 是以原属二进制为主，Code 是以汇编代码和 ELF 结构体代码为主。

CPU 数据窗口的数据如何导出到本地或者复制到粘贴版？

答：在该窗口右键菜单有一个 Dump 选项，执行它之后会弹出一个询问 Dump 字节数的对话框，假如输入的字节数为 size，该窗口的基址为 addr，它会将 addr 之后的 size 个字节 dump 到 \$(GKDBG)/idump/name-xxxx.xxxx.dump, \$(GKDBG.ART)/artdump/name-xxxx.xxxx.dump 目录并打开一个新的数据窗口，在这个数据窗口你可以进行各种形式的复制操作。

比如你要 dump 一个 dex 内存文件，只需要做如下操作即可：

- 1) 在数据窗口执行 Ctrl+G，输入 dex 的首地址 addr；
- 2) 在数据窗口执行右键 Dump 菜单，输入 dex 的字节数 size；
- 3) 在 \$(GKDBG.ART)/artdump 中找到 app_process.start.end.dump 文件，其中 start=addr,end=addr+size；