目录

Frida调试经验总结

- 最新版本：`0.8.0`
- 更新时间：`20250525`

简介

总结Frida调试期间遇到的各种问题和解决方案，以及各种经验和心得，帮助读者更好地理解和使用Frida进行调试。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

HonKit源码

- crifan/frida_debug_summary: Frida调试经验总结

如何使用此HonKit源码去生成发布为电子书

详见：crifan/honkit_template: demo how to use crifan honkit template and demo

在线浏览

- Frida调试经验总结 book.crifan.org
- Frida调试经验总结 crifan.github.io

离线下载阅读

- Frida调试经验总结 PDF
- Frida调试经验总结 ePub
- Frida调试经验总结 Mobi

版权和用途说明

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

其他

作者的其他电子书

本人 `crifan` 还写了其他 `150+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

关于作者

关于作者更多介绍，详见：

关于CrifanLi李茂 – 在路上

Frida调试概述

之前已整理出：

- Frida教程：逆向调试利器：Frida
  - 相关工具类：Frida逆向实例和工具函数

现在对于，Frida调试（iOS和Android）期间，遇到的各种**经验心得**和**常见问题**等内容，单独整理成册，供参考。

经验心得

此处整理frida中的一些经验和心得。

frida

通过函数地址去hook函数

- 已知：函数的（二进制内偏移量）地址
- 想要：去hook拦截函数
- 举例说明
  - akd中函数 `___lldb_unnamed_symbol2575$$akd` 的二进制内偏移量是：`0xa0460`
    - 思路：动态计算加上模块后的真实函数地址，再去hook
    - 核心代码：

```
console.log("dynamicDebug/frida/scripts/fridaStalker_akdSymbol2575.js");
// var akdSymbol2575_functionAddress = 0x1000a0460;
var akdSymbol2575_functionAddress = 0xa0460;
// arm64 akd: ___lldb_unnamed_symbol2575$$akd
const moduleName = "akd";
const moduleBaseAddress = Module.findBaseAddress(moduleName);
console.log("moduleName=", moduleName, "moduleBaseAddress=", moduleBaseAddre
ss);
const functionRealAddress = moduleBaseAddress.add(akdSymbol2575_functionAddr
ess);
console.log("functionRealAddress=", functionRealAddress);
Interceptor.attach(functionRealAddress, {
    onEnter: function(args) {
        var arg0 = args[0]
        var arg1 = args[1]
        var arg2 = args[2]
        console.log("arg0=" + arg0 + ", arg1=" + arg1 + ", arg2=" + arg2);
        var curTid = Process.getCurrentThreadId();
        console.log("curTid=", curTid);
...
```

    - 输出

```
dynamicDebug/frida/scripts/fridaStalker_akdSymbol2575.js
moduleName= akd moduleBaseAddress= 0x102b40000
functionRealAddress= 0x102be0460
arg0 0xfffffffffffffffe, arg1=0x16d346838, arg2 0x16d346838
curTid= 35847
...
```

    - 完整代码和输出，详见
      - ___lldb_unnamed_symbol2575$$akd · Frida逆向实例和工具函数

Stalker

实际内存地址和Stalker输出地址不匹配不一致

某次Stalker调试的log是：

```
===== dynamicDebug/frida/scripts/fridaStalker_akdSymbol2575.js =====
funcRelativeStartAddr=656480, functionSize=9416, funcRelativeEndAddr=665896
moduleName=akd, moduleBaseAddress=0x10054c000
funcRealStartAddr=0x1005ec460, funcRealEndAddr=0x1005ec4609416
[iPhone::PID::13098 ]-> ----- arg0=0xfffffffffffffffe, arg1=0x16fe26838, arg2=0x16fe268
38, arg3=0xfffffffffffffffe

curTid=26635

+++ into iterator=
[0x1071dbcd8] b #0x1071dbce8
+++ into iterator=
[0x1071dbce8] str wzr, [x19, #0x90]
[0x1071dbcec] ldr x0, [x19, #0xa0]
[0x1071dbcf0] str xzr, [x19, #0xa0]
[0x1071dbcf4] cbz x0, #0x1071dbcfc
+++ into iterator=
[0x1071dbcf8] bl #0x1070dfef8
+++ into iterator=
[0x1071dbcfc] ldr x0, [x19, #0x98]
[0x1071dbd00] str xzr, [x19, #0x98]
[0x1071dbd04] cbz x0, #0x1071dbd14
...
```

其中指令地址看起来不匹配=不一致：

- 指令地址：0x1071dbcd8
- 函数二进制内偏移量：0xa0460
  - 此处akd模块基地址：0x10054c000
    - 所以函数的实际真实起始地址：0x1005ec460

后来才明白是：

Frida的Stalker中，看到2个函数地址不匹配：

- 函数实际内存起始地址：`0x1005ec460`
  - 0x1005exxxx 之类的地址
- Stalker中运行期间的函数地址：`0x1071dbcd8`
  - 0x1071dxxxx 之类的地址

的原因：

Stalker内部的hook指令的原理就是：把当前代码拷贝一份，加上hook代码及其他逻辑

而此处的：拷贝一份，到别处，别的一段内存地址空间，此处就是指的是：

- 0x1071dbcd8
  - 0x1071dxxxx 之类的地址

所以，原始函数代码和被Stalker去hook的代码，两个地址是不同的，是可以理解的。

Stalker的transform中Instruction的属性

Frida的Stalker的transform中Instruction指令，有哪些属性，参考官网：

https://frida.re/docs/javascript-api/#instruction

得知有如下属性：

- `address`
- `next`
- `size`
- `mnemonic`
- `opStr`
- `operands`
- `regsAccessed`
- `regsRead`
- `regsWritten`
- `groups`
- `toString()`
- `toJSON()`

注：但是没有（其实希望也有的） `bytes` = `opcode` 属性。

而这些属性的来源是： `Capstone`

- Capstone
  - Java接口
    - capstone/Capstone.java at next · capstone-engine/capstone · GitHub
  - Python接口
    - capstone/**init**.py at next · capstone-engine/capstone · GitHub

举例

如之前示例代码：

___lldb_unnamed_symbol2575$$akd · Frida逆向实例和工具函数

中的，但是注释掉的代码：

（注：当时没加 `regsAccessed` 和 `toJSON()` ）

```
    // console.log("instruction: address=" + instruction.address
    //     + ",next=" + instruction.next()
    //     + ",size=" + instruction.size
    //     + ",mnemonic=" + instruction.mnemonic
    //     + ",opStr=" + instruction.opStr
    //     + ",operands=" + JSON.stringify(instruction.operands)
    //     + ",regsAccessed=" + JSON.stringify(instruction.regsAccessed)
    //     + ",regsRead=" + JSON.stringify(instruction.regsRead)
    //     + ",regsWritten=" + JSON.stringify(instruction.regsWritten)
```

```
    //     + ",groups=" + JSON.stringify(instruction.groups)
    //     + ",toString()=" + instruction.toString()
    //     + ",toJSON()=" + instruction.toJSON()
    // );
```

取消注释后，可以输出log：

```
instruction: address 0x10f4ecef4,next 0x4,size 4,mnemonic ldr,opStr x0, #0x10f4ecf78,op
erands=[{"type":"reg","value":"x0","access":"w"},{"type":"imm","value":"4551790456","ac
cess":"r"}],regsRead=[],regsWritten=[],groups=[],toString()=ldr x0, #0x10f4ecf78
[0x10f4ecef4] ldr x0, #0x10f4ecf78

instruction: address 0x10f4ecef8,next 0x4,size 4,mnemonic bl,opStr #0x1091a500c,operand
s=[{"type":"imm","value":"4447686668","access":"r"}],regsRead=[],regsWritten=["lr"],gro
ups=["call","jump","branch_relative"],toString()=bl #0x1091a500c
[0x10f4ecef8] bl #0x1091a500c
```

Stalker中去判断是否是函数的代码指令的逻辑

- 需求：Frida的Stalker调试期间，想要知道当前所执行的代码，是否属于：真正的要hook的函数的真实代码
- 解决办法

经过调试用如下代码：

```
console.log("===== Frida Stalker hook for arm64 ___lldb_unnamed_symbol2575$$akd =====");

// arm64 akd: ___lldb_unnamed_symbol2575$$akd
// var funcRelativeStartAddr = 0x1000a0460;
var funcRelativeStartAddr = 0xa0460;
var functionSize = 0x24C8; // 9416 == 0x24C8
var funcRelativeEndAddr = funcRelativeStartAddr + functionSize;
console.log("funcRelativeStartAddr=" + funcRelativeStartAddr + ", functionSize=" + func
tionSize + ", funcRelativeEndAddr=" + funcRelativeEndAddr);
const moduleName = "akd";
const moduleBaseAddress = Module.findBaseAddress(moduleName);
console.log("moduleName=" + moduleName + ", moduleBaseAddress=" + moduleBaseAddress);
// console.log("moduleName=%s, moduleBaseAddress=%p", moduleName, moduleBaseAddress);
const funcRealStartAddr = moduleBaseAddress.add(funcRelativeStartAddr);
// var funcRealEndAddr = funcRealStartAddr + functionSize;
const funcRealEndAddr = funcRealStartAddr.add(functionSize);
console.log("funcRealStartAddr=" + funcRealStartAddr + ", funcRealEndAddr=" + funcRealE
ndAddr);
var curTid = null;
Interceptor.attach(funcRealStartAddr, {
    onEnter: function(args) {
        var arg0 = args[0]
        var arg1 = args[1]
        var arg2 = args[2]
        var arg3 = args[3]
        console.log("----- arg0=" + arg0 + ", arg1=" + arg1 + ", arg2=" + arg2 + ", arg
3=" + arg3);
        var curTid = Process.getCurrentThreadId();
        console.log("curTid=", curTid);
        Stalker.follow(curTid, {
            events: {
                call: true, // CALL instructions: yes please
                ret: false, // RET instructions
                exec: false, // all instructions: not recommended as it's
                block: false, // block executed: coarse execution trace
                compile: false // block compiled: useful for coverage
            },
            // transform: (iterator: StalkerArm64Iterator) => {
            transform: function (iterator) {
                var instruction = iterator.next();
                const startAddress = instruction.address;
                console.log("+++ into iterator: startAddress=" + startAddress);
                // const isAppCode = startAddress.compare(funcRealStartAddr) >= 0 && st
artAddress.compare(funcRealEndAddr) === -1;
                // const isAppCode = (startAddress.compare(funcRealStartAddr) >= 0) &&
```

```
(startAddress.compare(funcRealEndAddr) < 0);
                const gt_realStartAddr = startAddress.compare(funcRealStartAddr) >= 0
                const lt_realEndAddr = startAddress.compare(funcRealEndAddr) < 0
                const isAppCode = gt_realStartAddr && lt_realEndAddr
                console.log("isAppCode=" + isAppCode + ", gt_realStartAddr=" + gt_realS
tartAddr + ", lt_realEndAddr=" + lt_realEndAddr);
                do {
                    if (isAppCode) {
                        var curRealAddr = instruction.address;
                        var curOffset = curRealAddr.sub(funcRealStartAddr)
                        var curOffsetInt = curOffset.toInt32()
                        var instructionStr = instruction.toString()
                        console.log("\t" + curRealAddr + " <+" + curOffsetInt + ">: " +
 instructionStr);
                    }
                    iterator.keep();
                } while ((instruction = iterator.next()) !== null);
            }
        })
    },
    onLeave: function(retval) {
        console.log("retval:", new ObjC.Object(retval))
        if (curTid != null) {
            Stalker.unfollow(curTid);
            console.log("Stalker.unfollow curTid=", curTid)
        }
    }
});
```

可以输出：

```
===== Frida Stalker hook for arm64 ___lldb_unnamed_symbol2575$$akd =====
funcRelativeStartAddr=656480, functionSize=9416, funcRelativeEndAddr=665896
moduleName=akd, moduleBaseAddress=0x10237c000
funcRealStartAddr=0x10241c460, funcRealEndAddr=0x10241e928
[iPhone::akd ]-> ----- arg0=0xfffffffffffffffe, arg1=0x16dcae838, arg2=0x16dcae838, arg3
=0xfffffffffffffffe
curTid=15635
+++ into iterator: startAddress=0x1089dbcd8
isAppCode=false, gt_realStartAddr=true, lt_realEndAddr=false
+++ into iterator: startAddress=0x1089dbce8
...
isAppCode=true, gt_realStartAddr=true, lt_realEndAddr=true, arg3=0xfffffffffffffffe
    0x10241c470 +16 : stp x22, x21, [sp, #0xc0]
    0x10241c474 +20 : stp x20, x19, [sp, #0xd0]
    0x10241c478 +24 : stp x29, x30, [sp, #0xe0]
    0x10241c47c +28 : add x29, sp, #0xe0
...
```

算是实现了，通过计算，最后用 `isAppCode` 这个变量去判断是否是当前代码

上述代码的所属的完整的示例代码，详见：

___lldb_unnamed_symbol2575$$akd · Frida逆向实例和工具函数

Stalker中优化hook到指令时的log显示格式

- 需求：Frida的Stalker中，希望输出的log日志 -》 尽量模拟之前Xcode的汇编代码的形式
- 核心代码：

```
var curRealAddr = instruction.address;
var curOffsetHexPtr = curRealAddr.sub(funcRealStartAddr)
var curOffsetInt = curOffsetHexPtr.toInt32()
var instructionStr = instruction.toString()
console.log("\t" + curRealAddr + " <+" + curOffsetInt + ">: " + instructionStr);
```

- 输出效果

```
+++ into iterator: startAddress=0x104b48470
  0x104b48470 <+16> : stp x22, x21, [sp, #0xc0]
  0x104b48474 <+20> : stp x20, x19, [sp, #0xd0]
  0x104b48478 <+24> : stp x29, x30, [sp, #0xe0]
  0x104b4847c <+28> : add x29, sp, #0xe0
  0x104b48480 <+32> : nop
  0x104b48484 <+36> : ldr x8, #0x104b9c7d8
```

  - 注：完整代码和输出，详见：___lldb_unnamed_symbol2575$$akd · Frida逆向实例和工具函数

frida-trace

hook指定的多个二进制库文件

想要去用frida-trace追踪的iOS的ObjC的库=Framework=二进制：

- AppleAccount
- AppleAccountUI
- Accounts
- AccountsDaemon
- AuthKitUI
- AuthKit
- UserManagement

最后是用：

- 去include 多个 module

```
frida-trace -U -F com.apple.Preferences -I "AppleAccount*" -I "UserManagement" -I "Accounts*" -I "AuthKit*"
```

可以运行，但是结果：

- 只找到少数几个函数，不是我们要的
  - -》这么多库，一共才有107个函数
  - -》其中AppleAccount只有7个函数，其中就有上面的_AALogSystem仅仅是日志的函数，而不是我们要的：很多其他账号登录相关的函数
  - -》正常情况下，应该会有非常多的函数才对

注：

也可以换另外写法：

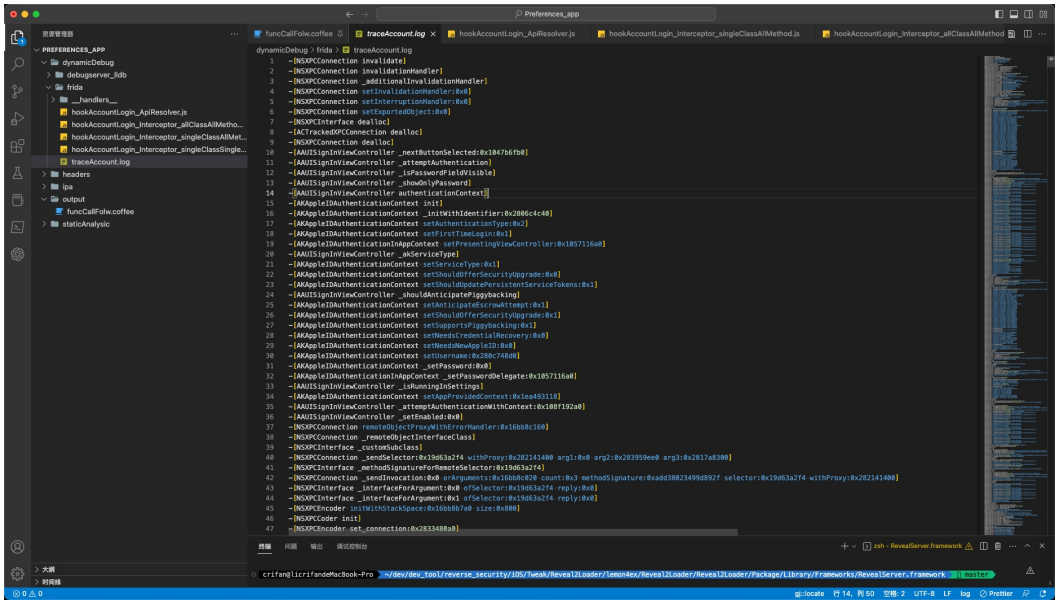
```
frida-trace -U -F com.apple.Preferences -i "AppleAccount\!" -i "AppleAccountUI\!"  -i "Accounts\!" -i "AccountsDaemon\!" -i "AuthKit\!" -i "AuthKitUI\!" -i "UserManagement\!"
```

不过结果都一样：hook的函数都很少，不是我们想要的。

输出日志到文件中

- 解决办法：加上 `-o` 参数，去指定输出log到文件
  - 命令

```
frida-trace -U -F com.apple.Preferences -o traceAccount.log -m "*[AA* *]" -m "*[AK* *]" -m "*[AS* *]" -m "*[NSXPC* *]" -M "-[ASDBundle copyWithZone:]" -M "-[ASDInstallationEvent copyWithZone:]" -M "-[NSXPCEncoder _encodeArrayOfObjects:forKey:]" -M "-[NSXPCEncoder _encodeUnkeyedObject:]" -M "-[NSXPCEncoder _replaceObject:]" -M "-[NSXPCEncoder _checkObject:]" -M "-[NSXPCEncoder _encodeObject:]" -M "-[NSXPCConnection replacementObjectForEncoder:object:]"
```

  - 缺点
    - 之前=输出日志到终端：带颜色和缩进 -> 利于查看函数调用关系
    - 
    - 现在=输出日志到文件：丢失了颜色，更主要是缩进也丢失了 -> 非常不利于查看函数调用关系
    - 

Mac中想要保留所有的frida-trace的带缩进的日志

如上所述，想要：保留所有的 `frida-trace` 的日志，但如果用输出到日志文件，却又丢失缩进（和颜色）

另外有个规避办法：

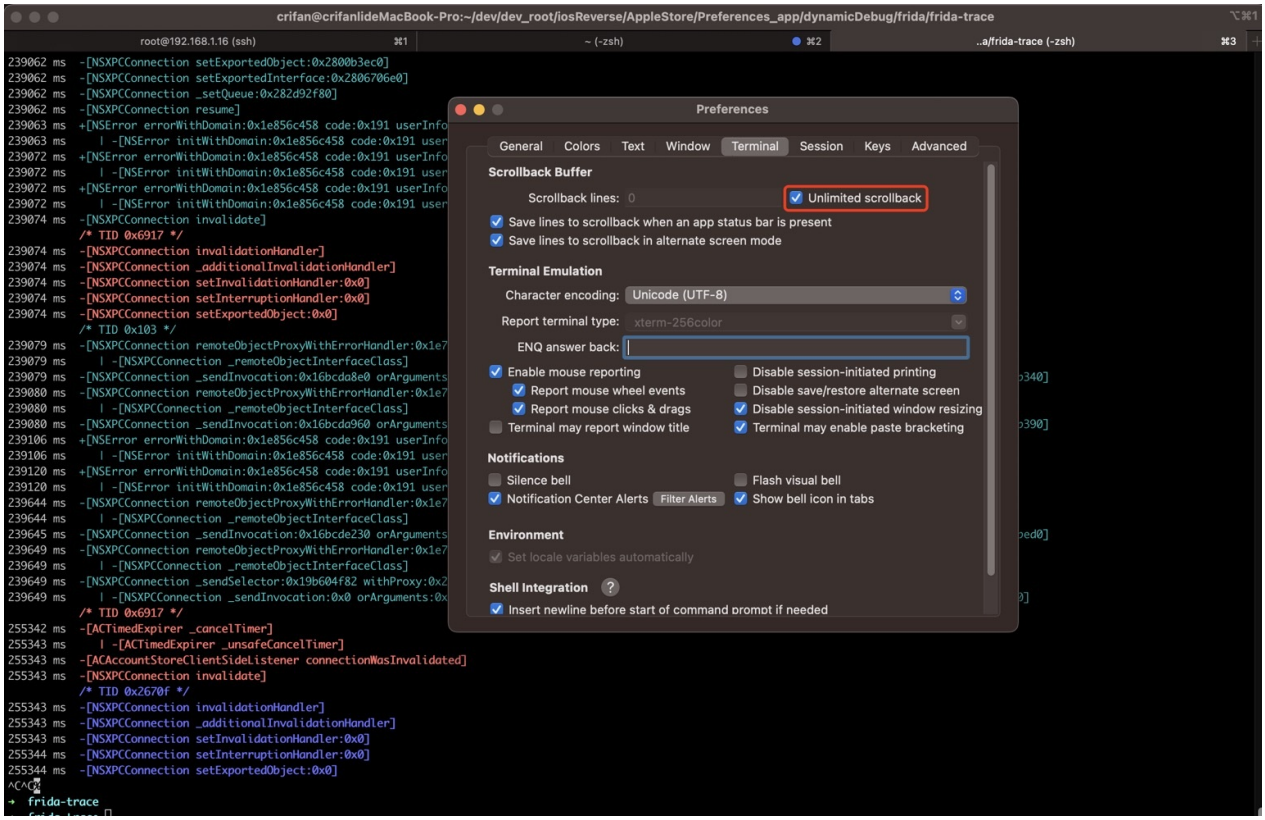
修改当前终端的最大显示行数 -> 就可以避免输出日志行数太多，之前内容被冲掉，看不到的问题了

比如：

- `Mac` 中 `iTerm2` 中，右键当前tab页顶部-》右键-》 `Edit Session` -> `Terminal` -> `Scrollback Buffer` -> `Scrollback line` ：改为足够大的数值，比如 `5000`

  ◦

甚至如果log日志特别长，那么可以考虑：设置为无限行数都保留 == `Unlimited scrollback`



这样就完全不用担心日志行数太多，前面的日志被冲掉，看不到的问题了。

修改函数hook的js去打印参数值

对于之前的需求：frida-trace时，打印其中特定的某个ObjC函数的参数

之前不知道如何解决，后来参考这里，突然想到：

倒是可以借助其所说的，对于frida-trace自动为每个类的函数，所生成的js文件：

- 位置：`__handlers__/{ClassName}/{FunctionName}.js`

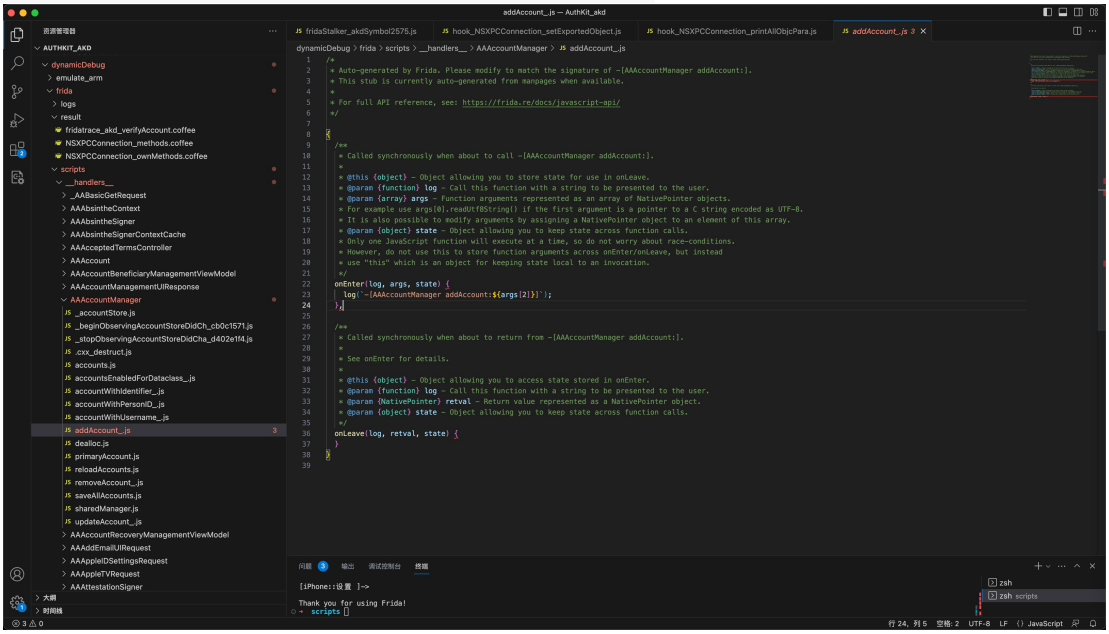
去修改js代码，加上打印对应的`args`的代码，即可打印对应参数值了。

举例

-[AAAccountManager addAccount:]

frida-trace为函数`-[AAAccountManager addAccount:]`自动生成的：

- js文件
  - `/Users/crifan/dev/dev_root/iosReverse/AppleStore/AuthKit_akd/dynamicDebug/frida/scripts/__handlers__/AAAccountManager/addAccount_.js`



- 完整js代码

```
/*
* Auto-generated by Frida. Please modify to match the signature of -[AAAccountManager addAccount:].
* This stub is currently auto-generated from manpages when available.
*
* For full API reference, see: https://frida.re/docs/javascript-api/
*/

{
  /**
```

```
    * Called synchronously when about to call -[AAAccountManager addAccount:].
  *
  * @this {object} - Object allowing you to store state for use in onLeave.
  * @param {function} log - Call this function with a string to be presented to the
 user.
  * @param {array} args - Function arguments represented as an array of NativePoint
er objects.
  * For example use args[0].readUtf8String() if the first argument is a pointer to
a C string encoded as UTF-8.
  * It is also possible to modify arguments by assigning a NativePointer object to
an element of this array.
  * @param {object} state - Object allowing you to keep state across function calls
.
  * Only one JavaScript function will execute at a time, so do not worry about race
-conditions.
  * However, do not use this to store function arguments across onEnter/onLeave, bu
t instead
  * use "this" which is an object for keeping state local to an invocation.
  */
  onEnter(log, args, state) {
    log(`-[AAAccountManager addAccount:${args[2]}]`);
  },

  /**
    * Called synchronously when about to return from -[AAAccountManager addAccount:
].
  *
  * See onEnter for details.
  *
  * @this {object} - Object allowing you to access state stored in onEnter.
  * @param {function} log - Call this function with a string to be presented to the
 user.
  * @param {NativePointer} retval - Return value represented as a NativePointer obj
ect.
  * @param {object} state - Object allowing you to keep state across function calls
.
  */
  onLeave(log, retval, state) {
  }
}
```

可以拷贝把其中的：

```
log(`-[AAAccountManager addAccount:${args[2]}]`);
```

改为：

```
log(`-[AAAccountManager addAccount:${new ObjC.Object(args[2])}]`);
```

就可以：打印出**ObjC对象的信息**了，而不仅仅是：`ptr` = `NativePointer` 的**指针的字符串**

iOS的经验心得

常用iOS函数

Frida中逆向iOS的app期间，常会涉及到，一些相对通用的，常用的函数，整理如下：

- 网络相关
  - 请求=request
    - `NSURLRequest`

```
+[NSURLRequest requestWithURL:]
+[NSURLRequest requestWithURL:cachePolicy:timeoutInterval:]
-[NSURLRequest requestWithURL:cachePolicy:timeoutInterval:]
-[NSURLRequest initWithURL:]
-[NSURLRequest initWithURL:cachePolicy:timeoutInterval:]
```

    - `NSMutableURLRequest`

```
-[NSMutableURLRequest initWithURL:]
-[NSMutableURLRequest setHTTPBody:]
```

    - `NSURL`

```
+[NSURL URLWithString:]
+[NSURL URLWithString:relativeToURL:]
-[NSURL initWithScheme:host:path:]
-[NSURL initWithString:]
-[NSURL initWithString:relativeToURL:]
```

    - `NSURLConnection`

```
+[NSURLConnection sendSynchronousRequest:returningResponse:error:]
+[NSURLConnection sendAsynchronousRequest:queue:completionHandler:]
+[NSURLConnection connectionWithRequest:delegate:]
-[NSURLConnection initWithRequest:delegate:]
-[NSURLConnection start]
-[NSURLConnection cancel]
-[NSURLConnection connection:didReceiveResponse:]
-[NSURLConnection connection:didReceiveData:]
-[NSURLConnection connectionDidFinishLoading:]
-[NSURLConnection connection:didFailWithError:]
```

  - 响应=response
    - `NSHTTPURLResponse`

```
-[NSHTTPURLResponse initWithURL:statusCode:HTTPVersion:headerFields:]
-[NSHTTPURLResponse allHeaderFields]
-[NSHTTPURLResponse statusCode]
```

    - `NSURLResponse`

```
-[NSURLResponse initWithURL:MIMEType:expectedContentLength:textEncodingName]
```

onLeave中如何获取到self类本身的实例变量

- 思路：`onEnter` 时，把self变量赋值保存给 `this.xxx` ， `onLeave` 时去读取即可
- 代码

```
Interceptor.attach(curMethod.implementation, {
  onEnter: function(args) {
      this.argSelfObj = argSelfObj
      ...
  },

  onLeave: function (retval) {
    var retValObj = ObjC.Object(retval);
    var retValObjStr = retValObj.toString();
    console.log("onLeave: retValObj=" + retValObj + ",retValObjStr=" + retValObjStr)

    const curSelfObj = this.argSelfObj
    console.log("curSelfObj=" + curSelfObj)
  }
})
```

- 输出

```
==================== -[NSXPCConnection setExportedObject:] ====================
argSelfObj:  <ACTrackedXPCConnection: 0x154907c50> connection to service named com.
apple.accountsd.accountmanager
onLeave: retValObj=nil,retValObjStr=nil
curSelfObj=<ACTrackedXPCConnection: 0x154907c50> connection to service named com.ap
ple.accountsd.accountmanager
```

Frida调试Android经验心得

hook类的ctor返回空值

对于java代码

```
public final class C666190Q1w<A, B> {
    public final A LIZ;
    public final B LIZIZ;
...
    public C666190Q1w(A a2, B b) {
        this.LIZ = a2;
        this.LIZIZ = b;
    }
}
```

去hook了构造函数，最整代码是：

```
function hookDouyinClass_X_0Q1w(ThrowableCls) {
  /******************** X.0Q1w ********************/


  var clsName_0Q1w = "X.0Q1w"
  // printClassAllMethodsFields(clsName_0Q1w)


  var cls_0Q1w = Java.use(clsName_0Q1w)
  console.log("cls_0Q1w=" + cls_0Q1w)


  // public C666190Q1w(A a2, B b) {
  // call: C666190Q1w<ByteBuffer, Long> c666190Q1w = new C666190Q1w<>(allocate5, Long.v
alueOf(j6))
  var func_0Q1w_ctor = cls_0Q1w.$init
  console.log("func_0Q1w_ctor=" + func_0Q1w_ctor)
  if (func_0Q1w_ctor) {
    func_0Q1w_ctor.implementation = function (a2, b) {
      var funcName = "0Q1w(a2,b)"
      var funcParaDict = {
        "a2": a2,
        "b": b,
      }
      printFunctionCallAndStack(funcName, funcParaDict, ThrowableCls)
      // var new0Q1w = this.$init(a2, b)
      // console.log("new0Q1w=" + new0Q1w)
      this.$init(a2, b)
      console.log("this=" + this)
      console.log("this.LIZ=" + this.LIZ)
      console.log("this.LIZIZ=" + this.LIZIZ)
      // var new0Q1w = this
      // return new0Q1w
      return
    }
  }
```

```
}
```

而其中，之前的写法是：

```
var new0Q1w = this.$init(a2, b)
console.log("new0Q1w=" + new0Q1w)
```

但会输出：`new0Q1w=undefined`

是因为：

- 构造函数：并没有返回值
  - 所以此时的值（始终）是：undefined
  - 所以后续改为：直接return返回
    - 效果也是一样的

而要想要： 去打印，初始还后（执行完毕构造函数后）的对应的实例（的属性值）

则是：

在运行完毕：`this.$init(a2, b)`

后，再去查看属性值：

```
        console.log("this=" + this)
        console.log("this.LIZ=" + this.LIZ)
        console.log("this.LIZIZ=" + this.LIZIZ)
```

即可查看到：（内存中的）实例的相关的值：

```
this: X.0Q1w@e1993e0
this.LIZ: Java.Field{holder: X.0Q1w@e1993e0, fieldType: 2, fieldReturnType: [object Obje
ct], value: java.nio.HeapByteBuffer[pos=12288 lim=12288 cap=12288]}
this.LIZIZ: Java.Field{holder: X.0Q1w@e1993e0, fieldType: 2, fieldReturnType: [object Ob
ject], value: 236556288}
```

即可。

对了，进一步，查看属性的值，是加上：`.value`

```
        console.log("this.LIZ.value=" + this.LIZ.value)
        console.log("this.LIZIZ.value=" + this.LIZIZ.value)
```

输出：

```
this.LIZ.value: java.nio.HeapByteBuffer[pos=12288 lim=12288 cap=12288]
this.LIZIZ.value: 236556288
```

借助崩溃找overload型参数定义

心得概述

如果要hook的函数的，不会写具体的定义

如果是同名的重载的overload类型，则可以：

故意不写具体定义，只写上implementation

然后让frida报错，而告知你多个函数的完整的定义的写法

举例

对于代码：

`sources/com/bytedance/retrofit2/client/Request.java`

```
    public Request(Builder builder) {
...

    public Request(String str, String str2, List<Header> list, TypedOutput typedOutput,
int i, boolean z, int i2, boolean z2, Object obj) {
...

    public Request(String str, String str2, List<Header> list, TypedOutput typedOutput,
RequestBody requestBody, int i, int i2, boolean z, int i3, boolean z2, Object obj, Stri
ng str3, Map<Class<?>, Object> map) {
...
```

这个类，有3个构造函数

想要hook，但是对于类的具体定义

尤其是其中一些类，比如：

- `Builder`
  - 是Request的内部的类
- `List<Header> list`
- `Map<Class<?>, Object> map`

等类型，不知道具体的Java的类型如何写

那么就可以故意写成：

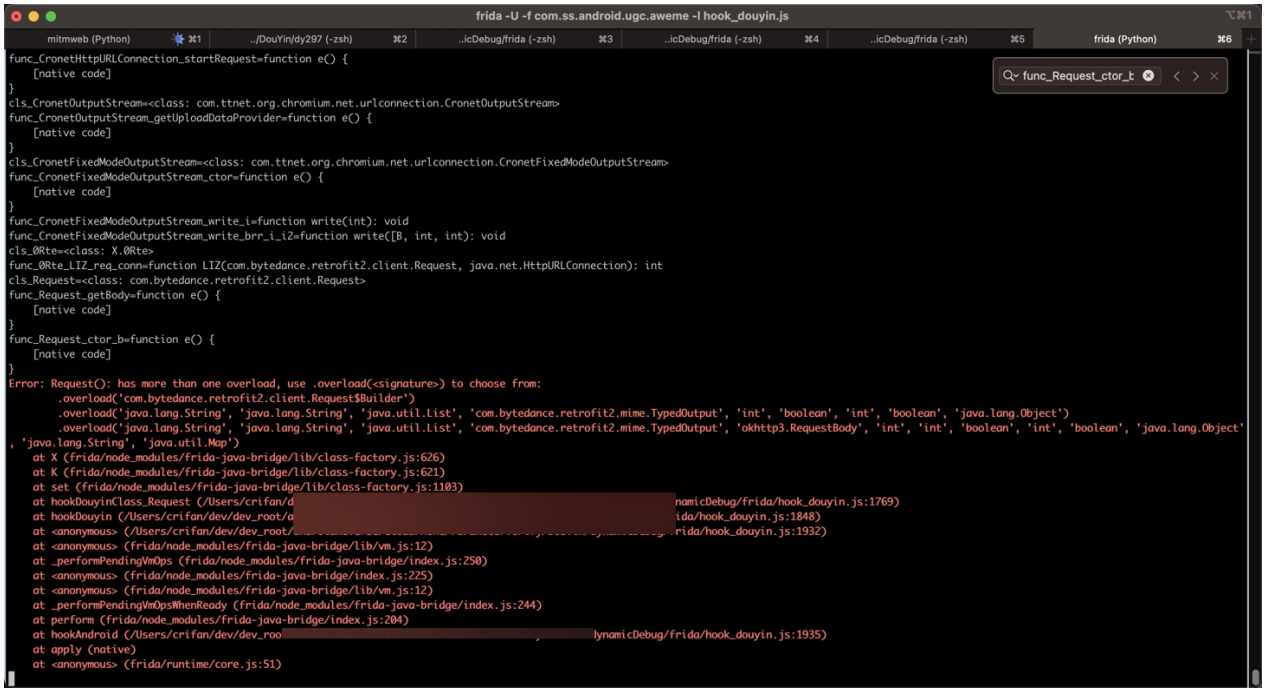
```
var func_Request_ctor_b = cls_Request.$init
...
func_Request_ctor_b.implementation = function (builder) {
```

完整代码：

```
  // public Request(Builder builder) {
  var func_Request_ctor_b = cls_Request.$init
  // var func_Request_ctor_b = cls_Request.$init.overload('com.bytedance.retrofit2.clie
nt.Request$Builder')
  console.log("func_Request_ctor_b=" + func_Request_ctor_b)
  if (func_Request_ctor_b) {
    func_Request_ctor_b.implementation = function (builder) {
      var funcName = "Request(builder)"
      var funcParaDict = {
        "builder": builder,
      }
      if (isWeConcernedUrl(this)){
        FridaUtil.printFunctionCallAndStack(funcName, funcParaDict)
      }
      return this.$init(builder)
    }
  }
```

运行后触发Frida的崩溃报错，说找不到对应overload的函数

```
Error: Request(): has more than one overload, use .overload(<signature>) to choose from
:
    .overload('com.bytedance.retrofit2.client.Request$Builder')
    .overload('java.lang.String', 'java.lang.String', 'java.util.List', 'com.bytedance.
retrofit2.mime.TypedOutput', 'int', 'boolean', 'int', 'boolean', 'java.lang.Object')
    .overload('java.lang.String', 'java.lang.String', 'java.util.List', 'com.bytedance.
retrofit2.mime.TypedOutput', 'okhttp3.RequestBody', 'int', 'int', 'boolean', 'int', 'bo
olean', 'java.lang.Object', 'java.lang.String', 'java.util.Map')
    at X (frida/node_modules/frida-java-bridge/lib/class-factory.js:626)
    at K (frida/node_modules/frida-java-bridge/lib/class-factory.js:621)
    at set (frida/node_modules/frida-java-bridge/lib/class-factory.js:1103)
    at hookyyyClass_Request (/Users/crifan/dev/dev_root/androidReverse/xx/x/yyy/dynamic
Debug/frida/hook_yyy.js:1769)
    at hookyyy (/Users/crifan/dev/dev_root/androidReverse/x/x/yyy/dynamicDebug/frida/ho
ok_yyy.js:1848)
    at anonymous (/Users/crifan/dev/dev_root/androidReverse/x/x/yyy/dynamicDebug/frid
a/hook_yyy.js:1932)
    at anonymous (frida/node_modules/frida-java-bridge/lib/vm.js:12)
    at _performPendingVmOps (frida/node_modules/frida-java-bridge/index.js:250)
    at anonymous (frida/node_modules/frida-java-bridge/index.js:225)
    at anonymous (frida/node_modules/frida-java-bridge/lib/vm.js:12)
    at _performPendingVmOpsWhenReady (frida/node_modules/frida-java-bridge/index.js:244)

    at perform (frida/node_modules/frida-java-bridge/index.js:204)
    at hookAndroid (/Users/crifan/dev/dev_root/androidReverse/x/x/yyy/dynamicDebug/frid
a/hook_yyy.js:1935)
    at apply (native)
    at anonymous (frida/runtime/core.js:51)
```



-》其中列出完全的全部的overload的函数的定义，即：

- `.overload('com.bytedance.retrofit2.client.Request$Builder')`
- `.overload('java.lang.String', 'java.lang.String', 'java.util.List', 'com.bytedance.retrofit2.mime.TypedOutput', 'int', 'boolean', 'int', 'boolean', 'java.lang.Object')`
- `.overload('java.lang.String', 'java.lang.String', 'java.util.List', 'com.bytedance.retrofit2.mime.TypedOutput', 'okhttp3.RequestBody', 'int', 'int', 'boolean', 'int', 'boolean', 'java.lang.Object', 'java.lang.String', 'java.util.Map')`

如此，就可以正常去写hook函数了：

```
  // public Request(Builder builder) {
  // var func_Request_ctor_b = cls_Request.$init
  var func_Request_ctor_b = cls_Request.$init.overload('com.bytedance.retrofit2.client.
Request$Builder')
  console.log("func_Request_ctor_b=" + func_Request_ctor_b)
  if (func_Request_ctor_b) {
    func_Request_ctor_b.implementation = function (builder) {
      var funcName = "Request(builder)"
      var funcParaDict = {
        "builder": builder,
      }
      if (isWeConcernedUrl(this)){
        FridaUtil.printFunctionCallAndStack(funcName, funcParaDict)
      }
      return this.$init(builder)
    }
  }


  // public Request(String str, String str2, List<Header> list, TypedOutput typedOutput
, int i, boolean z, int i2, boolean z2, Object obj) {
  var func_Request_ctor_9para = cls_Request.$init.overload('java.lang.String', 'java.la
```

```
ng.String', 'java.util.List', 'com.bytedance.retrofit2.mime.TypedOutput', 'int', 'boole
an', 'int', 'boolean', 'java.lang.Object')
  console.log("func_Request_ctor_9para=" + func_Request_ctor_9para)
  if (func_Request_ctor_9para) {
    func_Request_ctor_9para.implementation = function (str, str2, list, typedOutput, i,
 z, i2, z2, obj) {
      var funcName = "Request(9 Para)"
      var funcParaDict = {
        "str": str,
        "str2": str2,
        "list": list,
        "typedOutput": typedOutput,
        "i": i,
        "z": z,
        "i2": i2,
        "z2": z2,
        "obj": obj,
      }
      if (isWeConcernedUrl(this)){
        FridaUtil.printFunctionCallAndStack(funcName, funcParaDict)
      }
      return this.$init(str, str2, list, typedOutput, i, z, i2, z2, obj)
    }
  }


  // public Request(String str, String str2, List<Header> list, TypedOutput typedOutput
, RequestBody requestBody, int i, int i2, boolean z, int i3, boolean z2, Object obj, St
ring str3, Map<Class<?>, Object> map) {
    var func_Request_ctor_13para = cls_Request.$init.overload('java.lang.String', 'java
.lang.String', 'java.util.List', 'com.bytedance.retrofit2.mime.TypedOutput', 'okhttp3.R
equestBody', 'int', 'int', 'boolean', 'int', 'boolean', 'java.lang.Object', 'java.lang.
String', 'java.util.Map')
    console.log("func_Request_ctor_13para=" + func_Request_ctor_13para)
    if (func_Request_ctor_13para) {
      func_Request_ctor_13para.implementation = function (str, str2, list, typedOutput,
 requestBody, i, i2, z, i3, z2, obj, str3, map) {
        var funcName = "Request(13 Para)"
        var funcParaDict = {
          "str": str,
          "str2": str2,
          "list": list,
          "typedOutput": typedOutput,
          "requestBody": requestBody,
          "i": i,
          "i2": i2,
          "z": z,
          "i3": i3,
          "z2": z2,
          "obj": obj,
          "str3": str3,
          "map": map,
        }
        if (isWeConcernedUrl(this)){
          FridaUtil.printFunctionCallAndStack(funcName, funcParaDict)
        }
```

```
        return this.$init(str, str2, list, typedOutput, requestBody, i, i2, z, i3, z2,
obj, str3, map)
      }
    }
```

js

Frida对于js支持的不够好

Frida对于js支持的不够好，不够完美：

- console.log不支持参数格式化
- js报错时代码错误行数指示有误
- putCallout中传递有名的独立函数会导致崩溃 -> 改为匿名函数才可以

下面详细解释：

console.log不支持参数的格式化

- 问题
  - Frida中console.log打印参数，去格式化参数：

```
console.log("moduleName=%s, moduleBaseAddress=%p", moduleName, moduleBaseAddress
)
```

    - 输出：

```
moduleName %s, moduleBaseAddress %p akd 0x10015c000
```

- 结论：Frida中js的console.log，不支持参数格式化 == `%s` 、`%d` 、`%o` 、`%p` 等格式化参数无效
- 规避办法
  - 用 `逗号` = `,` 或 `加号` = `+` 去打印参数

```
console.log("moduleName=", moduleName, ", moduleBaseAddress=", moduleBaseAddress
)

console.log("moduleName=" + moduleName + ", moduleBaseAddress=" + moduleBaseAdd
ress)
```

js报错时代码错误行数指示有误

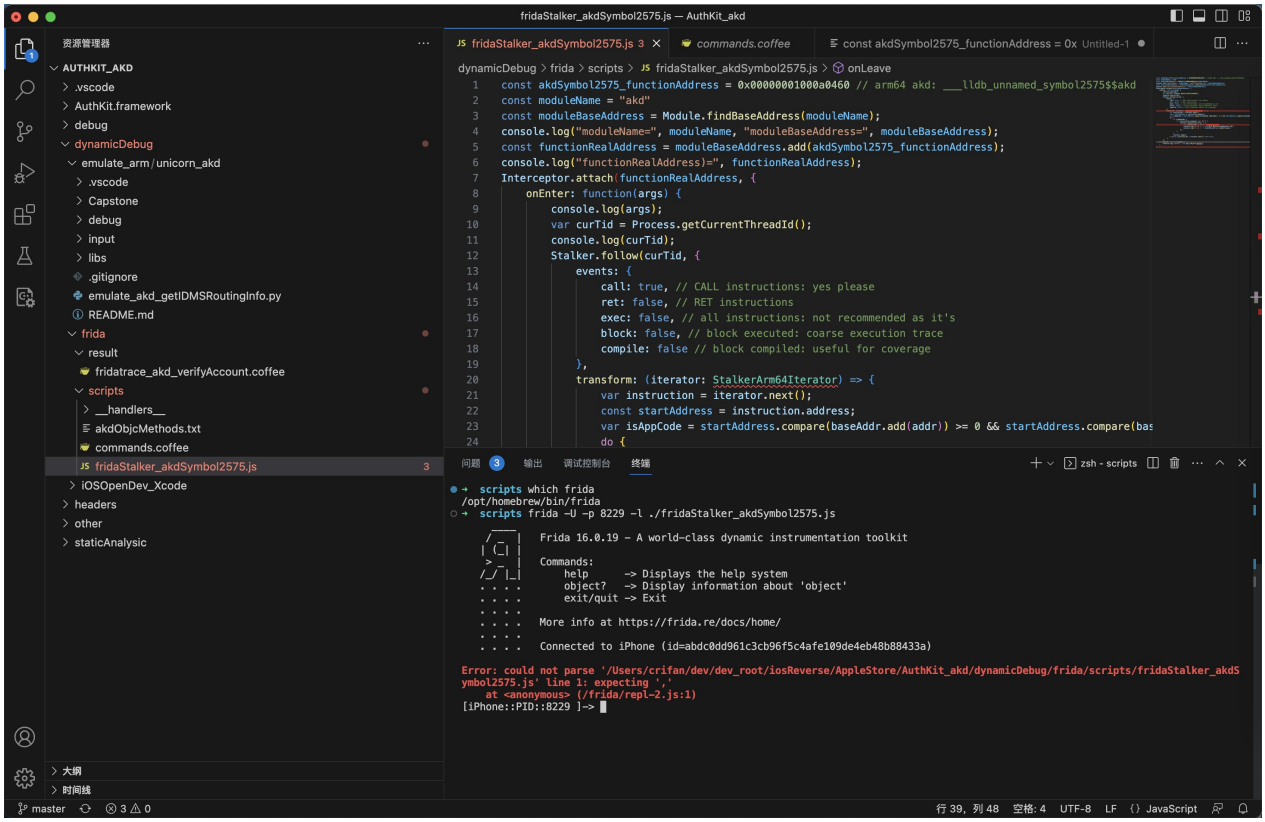
- 问题：

Frida的 `js` 代码文件中，参考别人的 `ts` 的代码，写了：

```
transform: (iterator: StalkerArm64Iterator) => {
    ...
    var arm64Context = ctx as Arm64CpuContext;
```

结果语法报错

```
Error: could not parse 'xxxf/ridaStalker_akdSymbol2575.js' line 1: expecting ','
    at anonymous (/frida/repl-2.js:1)
```



- 解决过程
  - Frida中对js代码报错，始终是第一行代码有问题
  - 后来是，花了很长时间和精力，才定位到真正问题：
- 原因：js代码中，后面的某2行中，先后出现的2个变量，没有定义
  - 2个变量是：`StalkerArm64Iterator` 和 `Arm64CpuContext`
  - 导致的整体js代码无法运行而报错
- 解决办法：注释掉2个未定义的变量（ `StalkerArm64Iterator` 和 `Arm64CpuContext` ）
- 心得：
  - 总之说明，js代码的解析，是Frida整个框架去基于js引擎（v8或其他？）去解析的
  - 所以最终输出的报错信息，不够友好，导致容易让人误判错误原因，从而增加解决问题的难度

putCallout中传递有名的独立函数会导致崩溃

- 问题概述
  - 给putCallout传递函数：
    - 传递有名的独立的函数：会导致崩溃
    - 传递匿名的函数：就可以正常运行
- 详解

Frida的Stalker的transform中去写putCallout代码

注：完整代码详见：___lldb_unnamed_symbol2575$$akd · Frida逆向实例和工具函数

如果把putCallout写成是传入一个**独立的js函数**：

```
transform: function (iterator) {
...
    if (curOffsetInt == 8516) {
        iterator.putCallout(needDebug);
    }
}


...
    function needDebug (context) {
        console.log("into needDebug: context=" + context);
        ...
    }
```

->就会导致此处出现崩溃：

```
...
Process terminated

Thank you for using Frida
Fatal Python error: _enter_buffered_busy: could not acquire lock for <_io.BufferedReade
r name='<stdin>'> at interpreter shutdown, possibly due to daemon threads
Python runtime state: finalizing (tstate=0x000000010308e0c8)

Current thread 0x00000001f4ab4140 (most recent call first):
  <no Python frame>
```

而如果改为：**js匿名名函数**

```
transform: function (iterator) {
...
    if (curOffsetInt == 8516) {
        // iterator.putCallout(needDebug);
        iterator.putCallout((context) => {
            console.log("into needDebug: context=" + context);
        });
    }
}
```

就不会崩溃

很是诡异。暂时不清楚具体原因。

- 心得：Frida中对于js的支持，还是不够完善
  - 容易遇到一些诡异的bug

console.log日志

用JSON.stringify打印对象

既然console.log中无法直接（用 `%o` 去格式化）打印对象，那么可以考虑用 `JSON.stringify` 去转化为JSON再去打印：

举例：

```
console.log("instruction: address=" + instruction.address
    + ",next=" + instruction.next
    + ",size=" + instruction.size
    + ",mnemonic=" + instruction.mnemonic
    + ",opStr=" + instruction.opStr
    + ",operands=" + JSON.stringify(instruction.operands)
    + ",regsRead=" + JSON.stringify(instruction.regsRead)
    + ",regsWritten=" + JSON.stringify(instruction.regsWritten)
    + ",groups=" + JSON.stringify(instruction.groups)
    + ",toString()=" + instruction.toString()
);
```

输出效果：

```
instruction: address 0x1091dbcd8,next 0x4,size 4,mnemonic b,opStr #0x1091dbce8,operands
=[{"type":"imm","value":"4447911144","access":"r"}],regsRead=[],regsWritten=[],groups=[
"jump","branch_relative"],toString()=b #0x1091dbce8
[0x1091dbcd8] b #0x1091dbce8
+++ into iterator=
instruction: address 0x1091dbce8,next 0x4,size 4,mnemonic str,opStr wzr, [x19, #0x90],o
perands=[{"type":"reg","value":"wzr","access":"r"},{"type":"mem","value":{"base":"x19",
"disp":144},"access":"rw"}],regsRead=[],regsWritten=[],groups=[],toString()=str wzr, [x
19, #0x90]
[0x1091dbce8] str wzr, [x19, #0x90]
```

console.log输出日志到文件

- 需求：希望Frida中console.log打印的日志，不是输出到当前终端，而是输出到文件中
  - 目的：方便后续随时查看，方便调试
- 解决办法：

在运行Frida带js的命令行最后加上：

- `> xxx.log 2>&1`
  - 参数说明
    - `> xxx.log`：把普通的=`stdout`日志信息，都输出到`log文件`
    - `2>&1`：把特殊的`stderr`=错误日志信息，输出到终端
- 举例

```
frida -U -p 13098 -l ./fridaStalker_akdSymbol2575.js > stalker_akd1575.log 2 &1
```

frida-server

自己编译frida-server

`iPhone11` 中，已用 `XinaA15` 进行了 `rootless越狱`，然后去用 `frida`：

```
crifan@licrifandeMacBook-Pro ~ frida -U -f com.apple.store.Jolly -l /Users/crifan/dev
/dev_root/iosReverse/AppleStore/dynamicDebug/frida/hookNSFileManager.js
     ____
    / _  |   Frida 16.0.8 - A world-class dynamic instrumentation toolkit
    | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to iPhone (id 00008030-00011C49366B802E)
Failed to attach: missing gProcessInfo
```

结果报错：`Failed to attach missing gProcessInfo`

之后就是尝试解决此问题，最终涉及到：自己去编译frida-server的过程。

此处记录相关内容和心得，供参考。

研究gProcessInfo的来源

之前frida-ios-dump也遇到类似问题

而之前就见过此处的 `missing gProcessInfo`，找到之前的：

- 原因是：属于偶尔的bug
- 解决办法：多试几次

-》此处：继续尝试多次，始终无法规避，始终报错。

研究frida中是否存在导入外部变量gProcessInfo

从报错信息 `Failed to attach: missing gProcessInfo` 中推测：

gProcessInfo是（iOS的app启动阶段涉及到的）dyld中的变量

怀疑是类似于：

- frida import 变量：gProcessInfo
- dyld export 变量：gProcessInfo

这种机制

所以去研究看看：

frida中，是否有import的变量，叫做gProcessInfo

后来确认，`missing gProcessInfo`来自iPhone端的`frida-server`

去导出`frida-server`

```
scp root@192.168.2.12:/var/sbin/frida-server frida-server
```

然后继续静态分析：

```
rabin2 -i frida-server > fridaServer_rabin2_i_imports.txt
rabin2 -E frida-server > fridaServer_rabin2_E_exports.txt
```

没找到`gProcessInfo`

另外找到：

- `dyld/dyldMain.cpp`

```
namespace dyld4 {
...

#if TARGET_OS_OSX
static void* getProcessInfo()
{
    return gProcessInfo;
}
```

而根据：

【记录】dyld相关资料：启动过程

知道了：

- dyld
  - 之前是：dyld2
  - 后来是：dyld3
  - 此处是：dyld4

即：`dyld`有3个版本，`dyld2`、`dyld3`、`dyld4`

最后确认：

不是Frida（的frida-server）引用了外部的变量：

`dyld`源码中的`gProcessInfo`

而是：

frida源码中有gProcessInfo

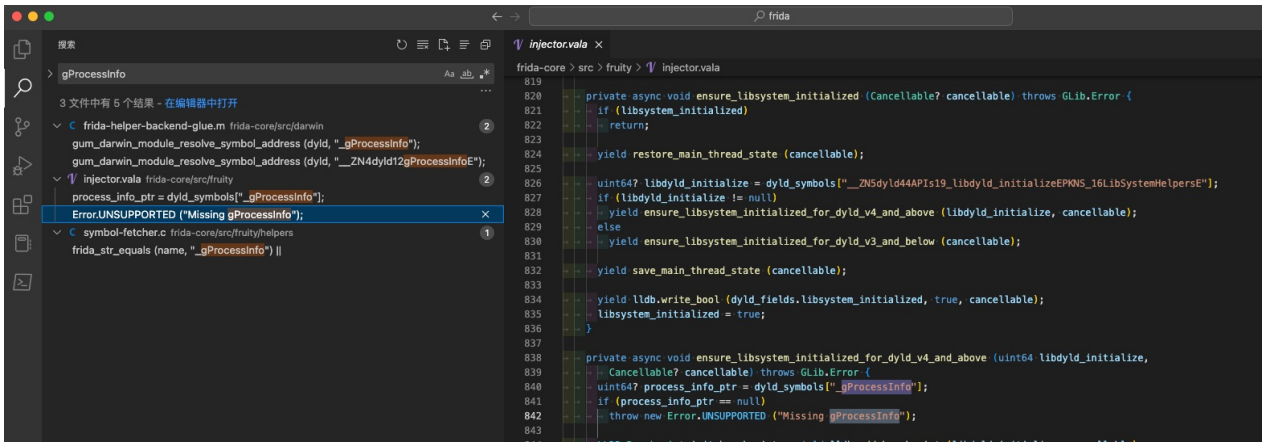
即：

`frida-core/src/fruity/injector.vala`中就有对应代码：

```
ensure_libsystem_initialized_for_dyld_v4_and_above
...
  throw new Error.UNSUPPORTED ("Missing gProcessInfo");
```

Frida源码中找到了：gProcessInfo

`missing gProcessInfo` 相关完整的代码：



- `frida-core/src/fruity/injector.vala`

```
private async void ensure_libsystem_initialized_for_dyld_v4_and_above (uint64 libdyld_initialize,
        Cancellable? cancellable) throws GLib.Error {
    uint64? process_info_ptr = dyld_symbols["_gProcessInfo"];
    if (process_info_ptr == null)
        throw new Error.UNSUPPORTED ("Missing gProcessInfo");
...
```

其他地方也有：

- `frida-core/src/fruity/helpers/symbol-fetcher.c`

```
size_t
frida_fetch_dyld_symbols (char * output_buffer, const void * dyld_load_address)
{
...
  for (i = dyld_dysymtab->ilocalsym; i != dyld_dysymtab->nlocalsym; i++)
  {
    const struct nlist_64 * sym = &symbols[i];
    const char * name = strings + sym->n_un.n_strx;

    if (frida_str_contains (name, "libdyld_initialize") ||
        frida_str_contains (name, "restartWithDyldInCache") ||
        frida_str_equals (name, "_gProcessInfo") ||
        frida_str_contains (name, "launchWithClosure") ||
        frida_str_contains (name, "initializeMainExecutable") ||
        frida_str_contains (name, "registerThreadHelpers") ||
        frida_str_has_prefix (name, "_dlopen") ||
        frida_str_has_prefix (name, "_strcmp") ||
        frida_str_contains (name, "doModInitFunctions") ||
```

```
          frida_str_contains (name, "doGetDOFSections"))
      {
        if (n != 0)
          frida_append_char (&cursor, '\n');

        frida_append_uint64 (&cursor, (guint64_t) (dyld->base + sym->n_value));
        frida_append_char (&cursor, '\t');
        frida_append_string (&cursor, name);

        n++;
      }
    }
```

- `frida-core/src/darwin/frida-helper-backend-glue.m`

```
modern_entry_address = gum_darwin_module_resolve_symbol_address (dyld, "__ZN5dyld44APIs
19_libdyld_initializeEPKNS_16LibSystemHelpersE");
instance->dyld_flavor = (modern_entry_address != 0) ? FRIDA_DYLD_V4_PLUS : FRIDA_DYLD_V
3_MINUS;
if (instance->dyld_flavor == FRIDA_DYLD_V4_PLUS)
{
  instance->modern_entry_address = modern_entry_address;
  legacy_entry_address = 0;


  instance->info_ptr_address = gum_darwin_module_resolve_symbol_address (dyld, "_gProce
ssInfo");
  if (instance->info_ptr_address == 0)
    goto dyld_probe_failed;
}
...
  instance->dlopen_address = gum_darwin_module_resolve_symbol_address (dyld, "_dlopen");

  if (instance->dlopen_address == 0)
    instance->dlopen_address = gum_darwin_module_resolve_symbol_address (dyld, "_dlopen
_internal");
  instance->register_helpers_address = gum_darwin_module_resolve_symbol_address (dyld,
"__ZL21registerThreadHelpersPKN4dyld16LibSystemHelpersE");
  instance->dlerror_clear_address = gum_darwin_module_resolve_symbol_address (dyld, "__
ZL12dlerrorClearv");
  instance->info_address = gum_darwin_module_resolve_symbol_address (dyld, "__ZN4dyld12
gProcessInfoE");
  instance->helpers_ptr_address = gum_darwin_module_resolve_symbol_address (dyld, "__ZN
4dyld17gLibSystemHelpersE");
  instance->do_modinit_strcmp_checks = frida_find_modinit_strcmp_checks (task, dyld);
  ...
```

心得：

其中有很多这种：

编译器编译后的固定的函数名：

- `_dlopen`

- `_dlopen_internal`
- `__ZL21registerThreadHelpersPKN4dyld16LibSystemHelpersE`
- `__ZL12dlerrorClearv`
- `__ZN4dyld12gProcessInfoE`
  - 其中包含：`gProcessInfo`
- `__ZN4dyld17gLibSystemHelpersE`

研究Frida中Missing gProcessInfo出错的逻辑和原因

经过后续了解： 【记录】dyld源码中的gProcessInfo

应该把：

- 只判断是否存在：_gProcessInfo

改为：

- 同时判断多种情况（先后顺序是）
  - `_gProcessInfo`
    - 对应原始代码中：`gProcessInfo`
  - `__ZN5dyld412gProcessInfoE`
    - 对应原始代码中：`dyld4::gProcessInfo`
  - `__ZN4dyld12gProcessInfoE`
    - 对应原始代码中：`dyld::gProcessInfo` 应该就可以了。

研究二进制/usr/lib/dyld中是否包含或导出变量_gProcessInfo

对于iPhone中的dyld：

```
iPhone11-151:~ root# ls -lh /usr/lib/dyld
-rwxr-xr-x 1 root wheel 630K Oct 15  2021 /usr/lib/dyld*
```

用：

```
scp root@192.168.2.12:/usr/lib/dyld dyld
```

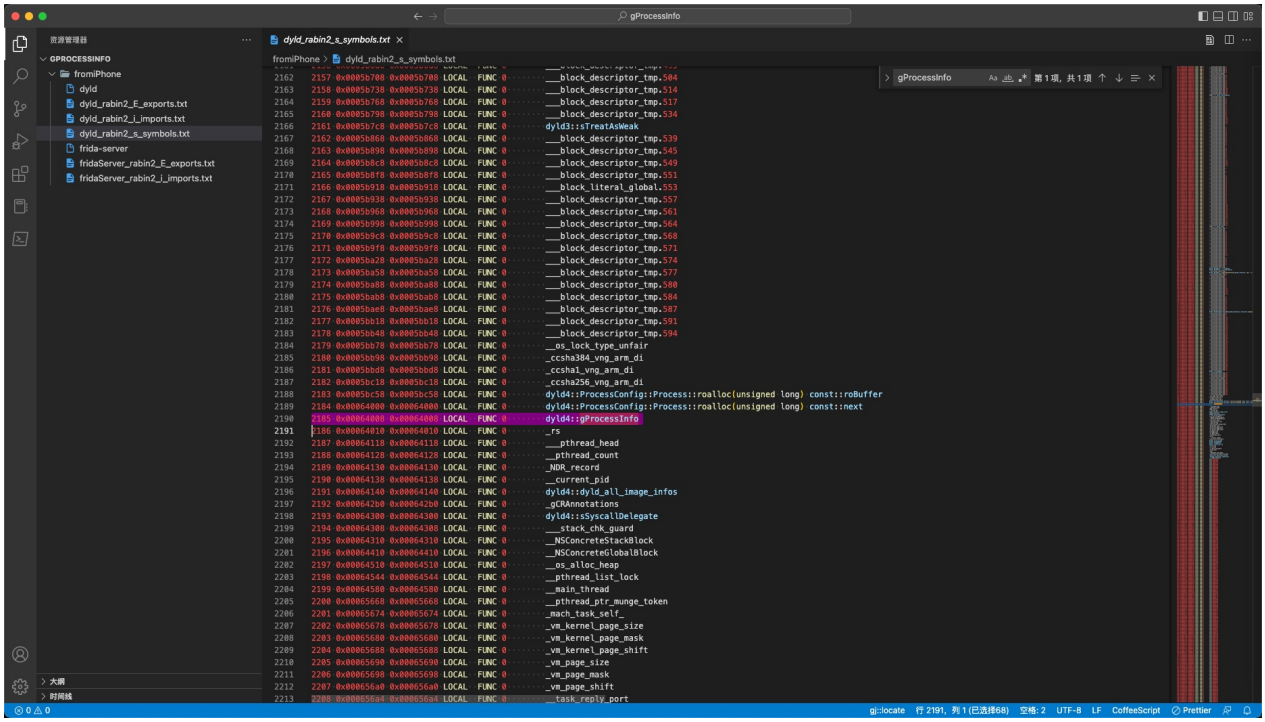
导出后，再从dyld中导出符号：

```
rabin2 -s dyld > dyld_rabin2_s_symbols.txt
```

发现是有的：

- `dyld_rabin2_s_symbols.txt`

```
2185 0x00064008 0x00064008 LOCAL  FUNC 0        dyld4::gProcessInfo
```



另外，突然注意到：

- `fromiPhone/dyld_rabin2_s_symbols.txt`

```
891  0x0002e500 0x0002e500 LOCAL  FUNC 0        dyld4::APIs::_dyld_shared_cache_optimized()
892  0x0002e57c 0x0002e57c LOCAL  FUNC 0        dyld4::APIs::_dyld_register_for_image_loads(void (*)(mach_header const*, char const*, bool))
893  0x0002e680 0x0002e680 LOCAL  FUNC 0        ___ZN5dyld44APIs30_dyld_register_for_image_loadsEPFvPK11mach_headerPKcbE_block_invoke
894  0x0002e76c 0x0002e76c LOCAL  FUNC 0        ___ZN5dyld44APIs30_dyld_register_for_image_loadsEPFvPK11mach_headerPKcbE_block_invoke_2
895  0x0002e7b4 0x0002e7b4 LOCAL  FUNC 0        ___ZN5dyld44APIs35_dyld_register_for_bulk_image_loadsEPFvjPPK11mach_headerPPKcE_block_invoke_2
896  0x0002e7c8 0x0002e7c8 LOCAL  FUNC 0        dyld4::APIs::dyld_shared_cache_file_path()
897  0x0002e810 0x0002e810 LOCAL  FUNC 0        dyld4::APIs::dyld_has_inserted_or_interposing_libraries()
898  0x0002e874 0x0002e874 LOCAL  FUNC 0        dyld4::APIs::dyld_shared_cache_find_iterate_text(unsigned char const*, char const**, void ( block_pointer)(dyld_shared_cache_dylib_text_info const*))
899  0x0002ea8c 0x0002ea8c LOCAL  FUNC 0        dyld4::findCacheInDirAndMap(dyld4::RuntimeState , unsigned char const*, char const*, unsigned long )
900  0x0002eb5c 0x0002eb5c LOCAL  FUNC 0        ___ZN5dyld44APIs35dyld_shared_cache_find_iterate_textEPKhPPKcU13block_pointerFvPK33dyld_shared_cache_dylib_text_infoE_block_invoke.173
901  0x0002ebe0 0x0002ebe0 LOCAL  FUNC 0        dyld4::APIs::dyld_shared_cache_iterate_text(unsigned char const*, void ( block_pointer)(dyld_shared_cache_dylib_text_info const*))
902  0x0002ec60 0x0002ec60 LOCAL  FUNC 0        dyld4::APIs::_dyld_fork_child()
```

即：

此处symbol中，也是有一些：

- ____ZN5dyld44APIs30_dyld_register_for_image_loadsEPFvPK11mach_headerPKcbE_block_invoke

这种编译后的函数名的值的

同时，也有，编译前的，普通的函数名：

- dyld4::APIs::_dyld_register_for_image_loads(void ()*(mach_header const*, char const*, bool))

-》所以突然想到：

估计是，编译前的，普通函数名，是：

此处rabin2，自动帮忙翻译的（因为其懂得编译和反编译函数名 symbol的内在逻辑？）

-》所以去找找：

是否有机会，让rabin2，只输出：

编译后的symbol名字？

这样就能找到，确认：

`dyld4::gProcessInfo`

是不是：

`__ZN5dyld44gProcessInfo`

了

去找找看：

rabin2的其他参数，能输出原始的symbol的？

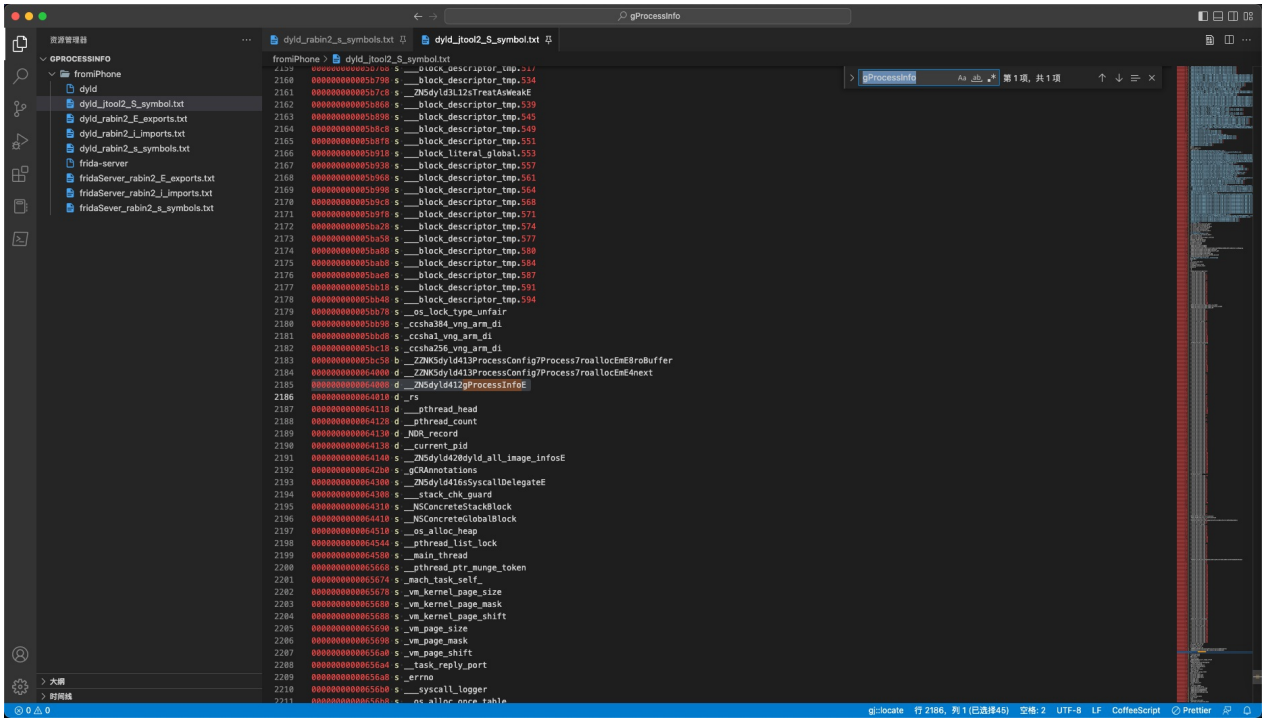
对了，或许也可以用另外的工具：`jtool2`

```
jtool2 -S dyld > dyld_jtool2_S_symbol.txt
```

果然是我们希望的，原始的，编译后的，没有被解析的：`gProcessInfo`

- `dyld_jtool2_S_symbol.txt`

  ```
  0000000000064008 d __ZN5dyld412gProcessInfoE
  ```



所以就是：

- `dyld4::gProcessInfo`
  - 编译生成：`__ZN5dyld412gProcessInfoE`
  - 不是我以为的：`__ZN5dyld44gProcessInfo`

再去dyld源码中，多搜搜：

`namespace dyld4`

看看是否有其他新发现

`namespace dyld4`

- `/Users/crifan/dev/dev_src/ios_reverse/AppleOpenSource/dyld/dyld-dyld-1042.1/dyld/DebuggerSupport.h`

```
namespace dyld4 {
    using lsl::Allocator;
    void addImagesToAllImages(RuntimeState& state, uint32_t infoCount, const dyld_image_info info[], uint32_t initialImageCount);
    void removeImageFromAllImages(const mach_header* loadAddress);
...
}

extern "C" void lldb_image_notifier(enum dyld_image_mode mode, uint32_t infoCount, const dyld_image_info info[]);

extern dyld_all_image_infos*        gProcessInfo;
```

->也还是：`gProcessInfo`

->不是放在`namespace dyld4`中的

- `/Users/crifan/dev/dev_src/ios_reverse/AppleOpenSource/dyld/dyld-dyld-1042.1/dyld/DyldAPIs.cpp`

```
// internal libc.a variable that needs to be reset during fork()
extern mach_port_t mach_task_self_;

using dyld3::MachOFile;
using dyld3::MachOLoaded;

extern const dyld3::MachOLoaded __dso_handle;

...

namespace dyld4 {
...
}
```

-》自己当前是dyld4的namespace，但是也会引用外部变量：

- 其中也有：
  - 没有namespace的：
    - `extern mach_port_t mach_task_self_;`
  - 和另外的namespace的：`dyld3`
    - `extern const dyld3::MachOLoaded __dso_handle;`

然后去：

【未解决】rabin2输出C++的未解析的原始的编译后的函数名mangle name

所以去：

【未解决】C++代码中函数变量编译生成符号symbol的规则

期间去：

【已解决】从C++的编译后的符号symbol得到原始的变量函数名

另外，看看此处的dyld版本：

```
iPhone11-151:~ root# /usr/lib/dyld --version
-sh: /usr/lib/dyld: cannot execute binary file: Exec format error
```

无法查看。

【已解决】iOS 13.3的iPhone7中/usr/lib/dyld的版本和gProcessInfo相关信息

也去研究：dyld源码中的gProcessInfo

至此找到：

- `gProcessInfo`
  - 变量来源：
    - `dyld/dyld-dyld-1042.1/dyld/DebuggerSupport.cpp`

```
struct dyld_all_image_infos* gProcessInfo = &dyld_all_image_infos;
```

以及：

- `struct dyld_all_image_infos`
  - 定义
    - `libdyld/dyld_process_info_internal.h`
      - `struct dyld_all_image_infos_32`
      - `struct dyld_all_image_infos_64`
    - `include/mach-o/dyld_images.h`
      - `struct __attribute__((aligned(16))) dyld_all_image_infos`

具体定义详见：

(1) `include/mach-o/dyld_images.h`

```
// Must be aligned to support atomic updates
// Note sim cannot assume alignment until all host dylds are new enough
#if TARGET_OS_SIMULATOR
struct dyld_all_image_infos
#else
struct __attribute__((aligned(16))) dyld_all_image_infos
#endif
{
    uint32_t                        version;        /* 1 in Mac OS X 10.4 and 10.5 */
    uint32_t                        infoArrayCount;
#if defined(__cplusplus) && (BUILDING_LIBDYLD || BUILDING_DYLD)
    std::atomic<const struct dyld_image_info*>    infoArray;
#else
    const struct dyld_image_info*    infoArray;
#endif
    dyld_image_notifier               notification;
    bool                            processDetachedFromSharedRegion;
    /* the following fields are only in version 2 (Mac OS X 10.6, iPhoneOS 2.0) and lat
er */
    bool                            libSystemInitialized;
    const struct mach_header*        dyldImageLoadAddress;
    /* the following field is only in version 3 (Mac OS X 10.6, iPhoneOS 3.0) and later
 */
    void*                            jitInfo;
    /* the following fields are only in version 5 (Mac OS X 10.6, iPhoneOS 3.0) and lat
er */
    const char*                       dyldVersion;
    const char*                       errorMessage;
    uintptr_t                        terminationFlags;
    /* the following field is only in version 6 (Mac OS X 10.6, iPhoneOS 3.1) and later
 */
    void*                            coreSymbolicationShmPage;
    /* the following field is only in version 7 (Mac OS X 10.6, iPhoneOS 3.1) and later
 */
    uintptr_t                        systemOrderFlag;
    /* the following field is only in version 8 (Mac OS X 10.7, iPhoneOS 3.1) and later
 */
    uintptr_t                        uuidArrayCount;
```

```
    const struct dyld_uuid_info*    uuidArray;        /* only images not in dyld shared
 cache */
    /* the following field is only in version 9 (Mac OS X 10.7, iOS 4.0) and later */
    struct dyld_all_image_infos*    dyldAllImageInfosAddress;
    /* the following field is only in version 10 (Mac OS X 10.7, iOS 4.2) and later */
    uintptr_t                       initialImageCount;
    /* the following field is only in version 11 (Mac OS X 10.7, iOS 4.2) and later */
    uintptr_t                       errorKind;
    const char*                       errorClientOfDylibPath;
    const char*                       errorTargetDylibPath;
    const char*                       errorSymbol;
    /* the following field is only in version 12 (Mac OS X 10.7, iOS 4.3) and later */
    uintptr_t                       sharedCacheSlide;
    /* the following field is only in version 13 (Mac OS X 10.9, iOS 7.0) and later */
    uint8_t                           sharedCacheUUID[16];
    /* the following field is only in version 15 (macOS 10.12, iOS 10.0) and later */
    uintptr_t                       sharedCacheBaseAddress;
#if defined(__cplusplus) && (BUILDING_LIBDYLD || BUILDING_DYLD)
    // We want this to be atomic in libdyld so that we can see updates when we map it s
hared
    std::atomic<uint64_t>           infoArrayChangeTimestamp;
#else
    uint64_t                        infoArrayChangeTimestamp;
#endif
    const char*                       dyldPath;
    mach_port_t                       notifyPorts[DYLD_MAX_PROCESS_INFO_NOTIFY_COUNT];
#if __LP64__
    uintptr_t                       reserved[11-(DYLD_MAX_PROCESS_INFO_NOTIFY_COUNT-2)]
;
#else
    uintptr_t                       reserved[9-DYLD_MAX_PROCESS_INFO_NOTIFY_COUNT];
#endif
    // The following fields were added in version 18 (previously they were reserved pad
ding fields)
    uint64_t                        sharedCacheFSID;
    uint64_t                        sharedCacheFSObjID;
    /* the following field is only in version 16 (macOS 10.13, iOS 11.0) and later */
    uintptr_t                       compact_dyld_image_info_addr;
    size_t                          compact_dyld_image_info_size;
    uint32_t                        platform; // FIXME: really a dyld_platform_t, but t
hose aren't exposed here.


    /* the following field is only in version 17 (macOS 10.16) and later */
    uint32_t                          aotInfoCount;
    const struct dyld_aot_image_info* aotInfoArray;
    uint64_t                          aotInfoArrayChangeTimestamp;
    uintptr_t                         aotSharedCacheBaseAddress;
    uint8_t                           aotSharedCacheUUID[16];
};
```

(2) 还有个分32和64的:

- `struct dyld_all_image_infos_32`

- `struct dyld_all_image_infos_64`

->

- `libdyld/dyld_process_info_internal.h`

```
struct dyld_all_image_infos_32 {
    uint32_t                        version;
    uint32_t                        infoArrayCount;
    std::atomic<uint32_t>           infoArray;
    uint32_t                        notification;
    bool                            processDetachedFromSharedRegion;
    bool                            libSystemInitialized;
    uint32_t                        dyldImageLoadAddress;
    uint32_t                        jitInfo;
    uint32_t                        dyldVersion;
    uint32_t                        errorMessage;
    uint32_t                        terminationFlags;
    uint32_t                        coreSymbolicationShmPage;
    uint32_t                        systemOrderFlag;
    uint32_t                        uuidArrayCount;
    uint32_t                        uuidArray;
    uint32_t                        dyldAllImageInfosAddress;
    uint32_t                        initialImageCount;
    uint32_t                        errorKind;
    uint32_t                        errorClientOfDylibPath;
    uint32_t                        errorTargetDylibPath;
    uint32_t                        errorSymbol;
    uint32_t                        sharedCacheSlide;
    std::array<uint8_t, 16>         sharedCacheUUID;
    uint32_t                        sharedCacheBaseAddress;
    std::atomic<uint64_t>           infoArrayChangeTimestamp;
    uint32_t                        dyldPath;
    uint32_t                        notifyMachPorts[8];
    uint32_t                        reserved;
    uint64_t                        sharedCacheFSID;
    uint64_t                        sharedCacheFSObjID;
    uint32_t                        compact_dyld_image_info_addr;
    uint32_t                        compact_dyld_image_info_size;
    uint32_t                        platform;
    // the aot fields below will not be set in the 32 bit case
    uint32_t                        aotInfoCount;
    std::atomic<uint64_t>           aotInfoArray;
    uint64_t                        aotInfoArrayChangeTimestamp;
    uint64_t                        aotSharedCacheBaseAddress;
    std::array<uint8_t, 16>         aotSharedCacheUUID[16];
};


struct dyld_all_image_infos_64 {
    uint32_t                    version;
    uint32_t                    infoArrayCount;
    std::atomic<uint64_t>       infoArray;
    uint64_t                    notification;
    bool                        processDetachedFromSharedRegion;
    bool                        libSystemInitialized;
```

```
    uint32_t                paddingToMakeTheSizeCorrectOn32bitAndDoesntAffect64b; // NO
T PART OF DYLD_ALL_IMAGE_INFOS!
    uint64_t                dyldImageLoadAddress;
    uint64_t                jitInfo;
    uint64_t                dyldVersion;
    uint64_t                errorMessage;
    uint64_t                terminationFlags;
    uint64_t                coreSymbolicationShmPage;
    uint64_t                systemOrderFlag;
    uint64_t                uuidArrayCount;
    uint64_t                uuidArray;
    uint64_t                dyldAllImageInfosAddress;
    uint64_t                initialImageCount;
    uint64_t                errorKind;
    uint64_t                errorClientOfDylibPath;
    uint64_t                errorTargetDylibPath;
    uint64_t                errorSymbol;
    uint64_t                sharedCacheSlide;
    std::array<uint8_t, 16> sharedCacheUUID;
    uint64_t                sharedCacheBaseAddress;
    std::atomic<uint64_t>   infoArrayChangeTimestamp;
    uint64_t                dyldPath;
    uint32_t                notifyMachPorts[8];
    uint64_t                reserved[7];
    uint64_t                sharedCacheFSID;
    uint64_t                sharedCacheFSObjID;
    uint64_t                compact_dyld_image_info_addr;
    uint64_t                compact_dyld_image_info_size;
    uint32_t                platform;
    uint32_t                aotInfoCount;
    std::atomic<uint64_t>   aotInfoArray;
    uint64_t                aotInfoArrayChangeTimestamp;
    uint64_t                aotSharedCacheBaseAddress;
    std::array<uint8_t, 16> aotSharedCacheUUID[16];
};
```

然后：

【未解决】dyld-932.4中gProcessInfo编译后symbol却是__ZN5dyld412gProcessInfoE

自己编译arm64e版的Frida

安装依赖库：

```
pip install colorama prompt-toolkit pygments
```

设置Python用新版 `3.10.6`

```
local再去设置为3.10.6的版本：
 crifan@licrifandeMacBook-Pro「 ~/dev/dev_src/ios_reverse/frida」 pyenv versions
  system
  3.5.2
  3.6.6
```

```
  3.7.3
* 3.9.4 (set by /Users/crifan/.pyenv/version)
  3.10.6
 crifan@licrifandeMacBook-Pro~/dev/dev_src/ios_reverse/frida pyenv local 3.10.6
 crifan@licrifandeMacBook-Pro~/dev/dev_src/ios_reverse/frida python --version
Python 3.10.6
```

clone frida的代码：

```
git clone --recurse-submodules https://github.com/frida/frida.git
```

先make看看有哪些编译选项：

```
crifan@licrifandeMacBook-Pro~/dev/dev_src/ios_reverse/frida/frida\main make
make[1]: Entering directory '/Users/crifan/dev/dev_src/ios_reverse/frida/frida'

Usage: make TARGET [VARIABLE value]

Where TARGET specifies one or more of:

  /* gum */
  gum-macos                 Build for macOS
  gum-ios                   Build for iOS
  gum-watchos               Build for watchOS
  gum-tvos                  Build for tvOS
  gum-android-x86           Build for Android/x86
  gum-android-x86_64        Build for Android/x86-64
  gum-android-arm           Build for Android/arm
  gum-android-arm64         Build for Android/arm64
  check-gum-macos           Run tests for macOS

  /* core */
  core-macos                Build for macOS
  core-ios                  Build for iOS
  core-watchos              Build for watchOS
  core-tvos                 Build for tvOS
  core-android-x86          Build for Android/x86
  core-android-x86_64       Build for Android/x86-64
  core-android-arm          Build for Android/arm
  core-android-arm64        Build for Android/arm64
  check-core-macos          Run tests for macOS

  /* python */
  python-macos              Build Python bindings for macOS
  check-python-macos        Test Python bindings for macOS

  /* node */
  node-macos                Build Node.js bindings for macOS
  check-node-macos          Test Node.js bindings for macOS

  /* tools */
  tools-macos               Build CLI tools for macOS
  check-tools-macos         Test CLI tools for macOS
```

```
And optionally also VARIABLE values:
  PYTHON                    Absolute path of Python interpreter including version suff
ix
  NODE                      Absolute path of Node.js binary

For example:
  $ make python-macos PYTHON=/usr/local/bin/python3.6
  $ make node-macos NODE=/usr/local/bin/node

make[1]: Leaving directory '/Users/crifan/dev/dev_src/ios_reverse/frida/frida'
```

此处要去编译：`ios` 的，所以看起来是：

`core-ios Build for iOS`

所以最后去：

```
make core-ios
```

期间解决了证书问题：

- 【已解决】Mac中编译frida-core报错：FAILED /usr/bin/codesign IOS_CERTID not set

继续：

```
✘ crifan@licrifandeMacBook-Pro ~/dev/dev_src/ios_reverse/frida/frida ＼ main export
MACOS_CERTID=frida-cert
 crifan@licrifandeMacBook-Pro ~/dev/dev_src/ios_reverse/frida/frida ＼ main export I
OS_CERTID=frida-cert
 crifan@licrifandeMacBook-Pro ~/dev/dev_src/ios_reverse/frida/frida ＼ main export W
ATCHOS_CERTID=frida-cert
 crifan@licrifandeMacBook-Pro ~/dev/dev_src/ios_reverse/frida/frida ＼ main export T
VOS_CERTID=frida-cert
 crifan@licrifandeMacBook-Pro ~/dev/dev_src/ios_reverse/frida/frida ＼ main make cor
e-ios
make[1]: Entering directory '/Users/crifan/dev/dev_src/ios_reverse/frida/frida'
. build/frida-env-ios-arm64.rc; \
builddir=build/tmp-ios-arm64/frida-core; \
if [ ! -f $builddir/build.ninja ]; then \
            meson_args="--native-file build/frida-macos-x86_64.txt"; if [ ios-arm64 !=
macos-x86_64 ]; then meson_args="$meson_args --cross-file build/frida-ios-arm64.txt"; fi
; python3 /Users/crifan/dev/dev_src/ios_reverse/frida/frida/releng/meson/meson.py setup
$meson_args \
        --prefix /usr \
        --default-library static -Doptimization=s -Db_ndebug=true --strip -Dconnectivity
=enabled -Dmapper=auto \
        -Dassets=installed \
        frida-core $builddir || exit 1; \
fi \
    && python3 /Users/crifan/dev/dev_src/ios_reverse/frida/frida/releng/meson/meson.py
compile -C $builddir \
    && DESTDIR="/Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64"
 python3 /Users/crifan/dev/dev_src/ios_reverse/frida/frida/releng/meson/meson.py install
 -C $builddir
INFO: autodetecting backend as ninja
```

```
INFO: calculating backend command to run: /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/toolchain-macos-x86_64/bin/ninja -C /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/tmp-ios-arm64/frida-core
ninja: Entering directory `/Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/tmp-ios-arm64/frida-core'
[85/85] Generating lib/gadget/frida-gadget with a custom command
ninja: Entering directory `/Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/tmp-ios-arm64/frida-core'
ninja: no work to do.
Installing lib/base/libfrida-base-1.0.a to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/lib
Installing lib/base/frida-base.h to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/include/frida-1.0
Installing lib/base/frida-base-1.0.vapi to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/share/vala/vapi
Installing lib/payload/libfrida-payload-1.0.a to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/lib
Installing lib/payload/frida-payload.h to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/include/frida-1.0
Installing lib/payload/frida-payload-1.0.vapi to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/share/vala/vapi
Installing lib/agent/frida-agent.dylib to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/lib/frida
Installing lib/gadget/frida-gadget.dylib to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/lib/frida
Installing src/frida-helper to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/lib/frida
Installing src/api/frida-core.h to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/include/frida-1.0
Installing src/api/frida-core-1.0.vapi to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/share/vala/vapi
Installing src/api/frida-core-1.0.deps to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/share/vala/vapi
Installing src/api/libfrida-core-1.0.a to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/lib
Installing server/frida-server to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/bin
Installing portal/frida-portal to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/bin
Installing inject/frida-inject to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/bin
Installing /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/tmp-ios-arm64/frida-core/meson-private/frida-base-1.0.pc to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/lib/pkgconfig
Installing /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/tmp-ios-arm64/frida-core/meson-private/frida-payload-1.0.pc to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/lib/pkgconfig
Installing /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/tmp-ios-arm64/frida-core/meson-private/frida-core-1.0.pc to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/lib/pkgconfig
make[1]: Leaving directory '/Users/crifan/dev/dev_src/ios_reverse/frida/frida'
```

即可：编译完成。

我们要找到的：frida-server，貌似是：

- Installing server/frida-server to /Users/crifan/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-arm64/usr/bin

去看看build目录

```
crifan@licrifandeMacBook-Pro㎡ ~/dev/dev_src/ios_reverse/frida/frida㎡ ＼ main㎡ cd build
 crifan@licrifandeMacBook-Pro㎡ ~/dev/dev_src/ios_reverse/frida/frida/build㎡ ＼ main㎡ ll
total 56
-rw-r--r--  1 crifan  staff   190B  1 16 11:45 frida-env-ios-arm64.rc
-rw-r--r--  1 crifan  staff   193B  1 16 11:45 frida-env-macos-x86_64.rc
drwxr-xr-x  3 crifan  staff    96B  1 16 11:47 frida-ios-arm64
-rwxr-xr-x  1 crifan  staff   430B  1 16 11:45 frida-ios-arm64-pkg-config
-rw-r--r--  1 crifan  staff   2.9K  1 16 11:45 frida-ios-arm64.txt
-rwxr-xr-x  1 crifan  staff   436B  1 16 11:45 frida-macos-x86_64-pkg-config
-rw-r--r--  1 crifan  staff   2.9K  1 16 11:45 frida-macos-x86_64.txt
-rw-r--r--  1 crifan  staff   217B  1 16 11:44 frida-version.h
drwxr-xr-x  8 crifan  staff   256B  1 16 11:45 sdk-ios-arm64
drwxr-xr-x  8 crifan  staff   256B  1 16 11:44 sdk-macos-x86_64
drwxr-xr-x  4 crifan  staff   128B  1 16 11:47 tmp-ios-arm64
drwxr-xr-x  8 crifan  staff   256B  1 16 11:44 toolchain-macos-x86_64
 crifan@licrifandeMacBook-Pro㎡ ~/dev/dev_src/ios_reverse/frida/frida/build㎡ ＼ main㎡ cd
 frida-ios-arm64
 crifan@licrifandeMacBook-Pro㎡ ~/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-ar
m64㎡ ＼ main㎡ ll
total 0
drwxr-xr-x  6 crifan  staff   192B  1 16 11:47 usr
 crifan@licrifandeMacBook-Pro㎡ ~/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-ar
m64㎡ ＼ main㎡ cd usr
 crifan@licrifandeMacBook-Pro㎡ ~/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-ar
m64/usr㎡ ＼ main㎡ ll
total 0
drwxr-xr-x   6 crifan  staff   192B  1 16 15:14 bin
drwxr-xr-x   3 crifan  staff    96B  1 16 11:47 include
drwxr-xr-x  13 crifan  staff   416B  1 16 15:14 lib
drwxr-xr-x   3 crifan  staff    96B  1 16 11:47 share
 crifan@licrifandeMacBook-Pro㎡ ~/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-ar
m64/usr㎡ ＼ main㎡ cd bin
 crifan@licrifandeMacBook-Pro㎡ ~/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-ar
m64/usr/bin㎡ ＼ main㎡ ll
total 40752
-rwxr-xr-x  1 crifan  staff   6.2M  1 16 15:13 frida-inject
-rwxr-xr-x  1 crifan  staff   5.2M  1 16 15:13 frida-portal
-rwxr-xr-x  1 crifan  staff   6.4M  1 16 15:13 frida-server
-rwxr-xr-x  1 crifan  staff   2.1M  1 16 11:47 gum-graft
 crifan@licrifandeMacBook-Pro㎡ ~/dev/dev_src/ios_reverse/frida/frida/build/frida-ios-ar
m64/usr/bin㎡ ＼ main㎡ file ./frida-server
./frida-server: Mach-O 64-bit executable arm64
```

的确是编译成功了

回去看看之前frida-server的大小和file输出信息

```
crifan@licrifandeMacBook-Pro㎡ ~/dev/dev_root/iosReverse/AppleStore/debug/gProcessInfo/i
```

```
Phone11_151ᵐ ll
total 55248
-rwxr-xr-x@ 1 crifan  staff   629K  1 13 10:06 dyld
-rw-r--r--  1 crifan  staff   2.5M  1 13 22:14 dyld.id0
-rw-r--r--  1 crifan  staff   1.6M  1 13 22:13 dyld.id1
-rw-r--r--  1 crifan  staff    40K  1 13 22:13 dyld.nam
-rw-r--r--  1 crifan  staff   1.3K  1 13 22:13 dyld.til
-rw-r--r--  1 crifan  staff   173K  1 13 10:44 dyld_jtool2_S_symbol.txt
-rw-r--r--  1 crifan  staff   277B  1 13 10:07 dyld_rabin2_E_exports.txt
-rw-r--r--  1 crifan  staff   125B  1 13 10:07 dyld_rabin2_i_imports.txt
-rw-r--r--  1 crifan  staff   249K  1 13 21:33 dyld_rabin2_s_r_symbols.txt
-rw-r--r--  1 crifan  staff   381K  1 13 10:13 dyld_rabin2_s_symbols.txt
-rwxr-xr-x  1 crifan  staff    20M  1 12 17:55 frida-server
-rw-r--r--  1 crifan  staff   169B  1 12 17:58 fridaServer_rabin2_E_exports.txt
-rw-r--r--  1 crifan  staff    17K  1 12 17:58 fridaServer_rabin2_i_imports.txt
-rw-r--r--  1 crifan  staff   1.9M  1 13 10:18 fridaSever_rabin2_s_symbols.txt
 crifan@licrifandeMacBook-Proᵐ ~/dev/dev_root/iosReverse/AppleStore/debug/gProcessInfo/
iPhone11_151ᵐ file ./frida-server
./frida-server: Mach-O universal binary with 3 architectures: [arm64:Mach-O 64-bit exec
utable arm64] [arm64e] [arm64e]
./frida-server (for architecture arm64):    Mach-O 64-bit executable arm64
./frida-server (for architecture arm64e):    Mach-O 64-bit executable arm64e
./frida-server (for architecture arm64e):    Mach-O 64-bit executable arm64e
```

-》此处frida-server很大：20M

不过明显是：FAT格式，包含多个架构：

- `arm64`
- `arm64e`

那看起来，貌似有个问题：

此处，从iPhone导出的真实在用的 `frida-server` ，支持： `arm64e`

而此处自己编译出来的，只支持 `arm64` ，不支持 `arm64e`

而记得：此处的 `iPhone` 中的架构都是： `arm64e` 的?

感觉需要：

- 确认 `iOS 15.1` 的 `iPhone11` 中，此处 `arm` 的架构是： `arm64e` 还是 `arm64`
  - 确定其中的 `frida-server` 是否需要支持 `arm64e`
- 如果需要支持 `arm64e` ，再去看： `frida` 编译 `core-ios` 时，如何指定或加上 `arm64e` 的支持

先去：

【基本解决】iOS 15.1的iPhone11中frida-server所用架构是arm64e还是arm64

-》 `arm64` 的二进制，是能放到 `arm64e` 的 `A13` 的 `iPhone11` 中运行的。

那先继续看看：

【未解决】自己编译出的arm64的frida-server能否在iPhone11正常运行

其他过程详见：

- 【未解决】用frida源码自己编译出frida的iOS的包含frida-server的deb安装包
- 【未解决】自己编译出包含arm64和arm64e的FAT格式的frida-server二进制
- 【未解决】自己修改编译frida-core源码以尝试解决Frida的Missing gProcessInfo问题
- 【未解决】Frida中如何编译出iOS的arm64e的frida-server二进制
- 【未解决】自己编译Frida的frida-core代码生成可用二进制frida-server

找arm64e版的Frida

从Frida源码和build中找

- 【未解决】找Frida中iOS的arm6e4：从Frida源码和build中找

从Frida的github中找

- 【未解决】找Frida中iOS的arm6e4：从Frida的github中找

自己给make加echo打印日志调试

- 【未解决】找Frida中iOS的arm6e4：自己给make加echo打印日志调试

从github的ci的workflow中找arm64e

- 【未解决】找Frida中iOS的arm6e4：从github的ci的workflow中找arm64e

从编译日志中的Downloading ios-arm64入手

- 【未解决】找Frida中iOS的arm6e4：从编译日志中的Downloading ios-arm64入手

make时如何传入arm64e的arch参数

- 【未解决】找Frida中iOS的arm6e4：make时如何传入arm64e的arch参数

从make编译时的log日志入手

- 【未解决】找Frida中iOS的arm6e4：从make编译时的log日志入手

字符串反混淆

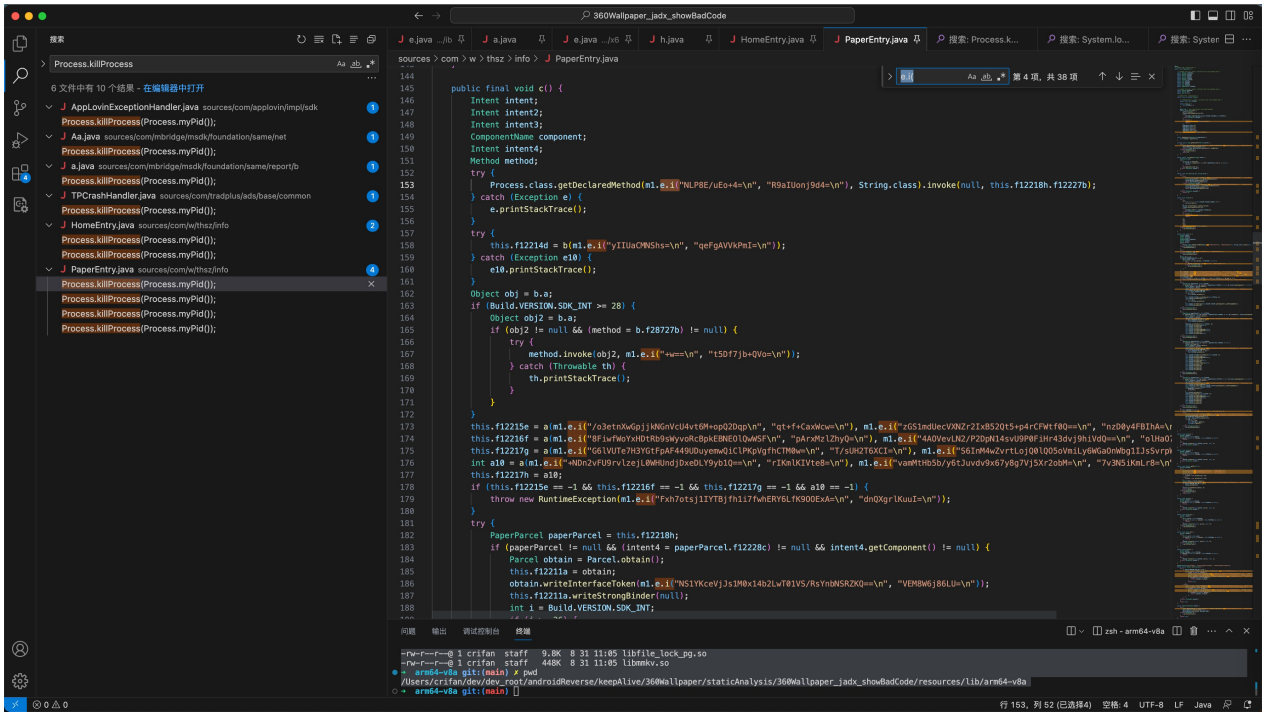
找到并调用字符串解密函数m1.e.i

需求

之前用Frida调试某安卓app逆向期间，遇到：

`/Users/crifan/dev/dev_root/androidReverse/keepAlive/360Wallpaper/staticAnalysis/360Wallpaper_jadx_showBadCode/sources/com/w/thsz/info/PaperEntry.java`

```
            Process.class.getDeclaredMethod(m1.e.i("NLP8E/uEo+4=\n", "R9aIUonj9d4=\n"),
String.class).invoke(null, this.f12218h.f12227b);

                    method.invoke(obj2, m1.e.i("+w==\n", "t5Df7jb+QVo=\n"));

        this.f12215e = a(m1.e.i("/o3etnXwGpjjkNGnVcU4vt6M+opQ2Dqp\n", "qt+f+CaxWcw=\n"),
m1.e.i("zGS1mdUecVXNZr2IxB52Qt5+p4rCFWtf0Q==\n", "nzD0y4FBIhA=\n"));

            throw new RuntimeException(m1.e.i("Fxh7otsj1IYTBjfh1i7fwhERY6LfK9OOExA=\n",
"dnQXgrlKuuI=\n"));

                obtain.writeInterfaceToken(m1.e.i("NS1YKceVjJs1M0x14b2LwT01VS/RsYnbNSRZ
KQ==\n", "VEM8W6j86LU=\n"));
```



发现 `m1.e.i` ，是通用的字符串加密解密函数。

而此处，想要搞清楚原始字符串是什么

所以就想要去：给上述加了密的字符串，比如：

- `m1.e.i("/o3etnXwGpjjkNGnVcU4vt6M+opQ2Dqp\n", "qt+f+CaxWcw=\n")`

- `m1.e.i("zGS1mdUecVXNZr2IxB52Qt5+p4rCFWtf0Q==\n", "nzD0y4FBIhA=\n")`
- `m1.e.i("8FiwfWoYxHDtRb9sWyvoRcBpkEBNEOlQwWSF\n", "pArxMzlZhyQ=\n")`
- `m1.e.i("4AOVevLN2/P2DpN14svU9P0FiHr43dvj9hiVdQ==\n", "olHaO7aOmqA=\n")`
- `m1.e.i("G6lVUTe7H3YGtFpAF449UDuyemwQiClPKpVgfhCTM0w=\n", "T/sUH2T6XCI=\n")`
- `m1.e.i("S6InM4wZvrtLojQ0lQO5oVmiLy6WGaOnWbg1IJsSvrpW\n", "GPZmYdhG9/U=\n")`
- `m1.e.i("+NDn2vFU9rvlzejL0WHUndjDxeDLY9yb1Q==\n", "rIKmlKIVte8=\n")`
- `m1.e.i("vamMtHb5b/y6tJuvdv9x67y8g7Vj5Xr2obM=\n", "7v3N5iKmLr8=\n")`

去解密字符串

== 字符串反混淆

具体实现：解密字符串

`/Users/crifan/dev/dev_root/androidReverse/keepAlive/360Wallpaper/dynamicDebug/Frida/frida/hook_360Wallpaper.js`

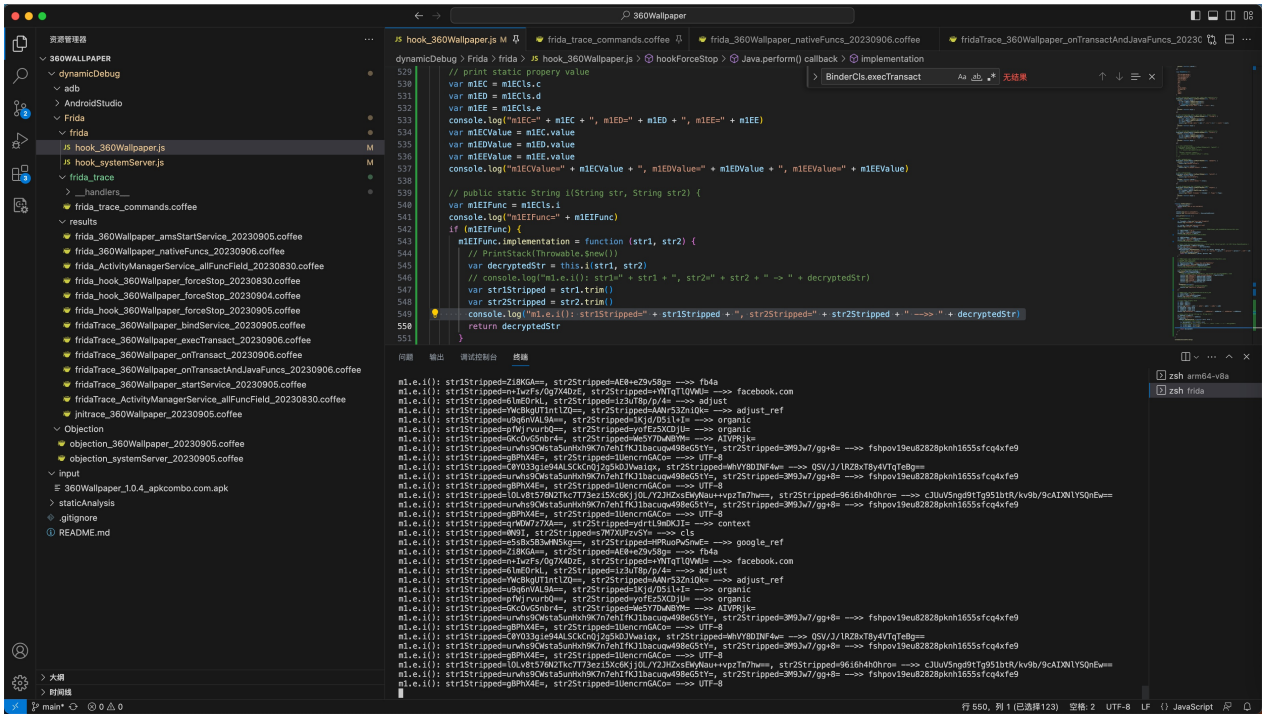
```
// convert Object(dict/list/...) to JSON string
function toJsonStr(curObj, singleLine=false, space=2){
  // console.log("toJsonStr: singleLine=" + singleLine)
  // var jsonStr = JSON.stringify(curObj, null, 2)
  var jsonStr = JSON.stringify(curObj, null, space)
  if(singleLine) {
    // jsonStr = jsonStr.replace(/\\n/g, '')
    jsonStr = jsonStr.replace(/\n/g, '')
  }
  return jsonStr
  // return curObj.toString()
}


var decryptedStrDict = {}

// example:
//   str1=n+IwzFs/Og7X4DzE\n, str2=+YNTqTlQVWU=, decryptedStr=facebook.com
//   str1=C0YO33gie94ALSCkCnQj2g5kDJVwaiqx\n, str2=WhVY8DINF4w= -->> decryptedStr=QSV/J/lRZ8xT8y4VTqTeBg==
function saveDecrptedString(str1, str2, decryptedStr){
  var delimChar = ","
  var curToDecryptStrGroup = str1 + delimChar + str2
  // console.log("curToDecryptStrGroup=" + curToDecryptStrGroup)
  // if (!(curToDecryptStrGroup in decryptedStrDict)){
  if (!decryptedStrDict.hasOwnProperty(curToDecryptStrGroup)){
    decryptedStrDict[curToDecryptStrGroup] = decryptedStr
    console.log("Updated -> decryptedStrDict=" + toJsonStr(decryptedStrDict))
  }
}

// manual decrypt obfuscated string
function manualDecryptString(m1ECls, m1EIFunc){
  // /Users/crifan/dev/dev_root/androidReverse/keepAlive/360Wallpaper/staticAnalysis/360Wallpaper_jadx_showBadCode/sources/com/w/thsz/info/HomeEntry.java
  // /Users/crifan/dev/dev_root/androidReverse/keepAlive/360Wallpaper/staticAnalysis/36
```

```
0Wallpaper_jadx_showBadCode/sources/com/w/thsz/info/PaperEntry.java
  let toDecryptStrArr = [
    // ["/o3etnXwGpjjkNGnVcU4vt6M+opQ2Dqp\n", "qt+f+CaxWcw=\n"],
    // ["zGS1mdUecVXNZr2IxB52Qt5+p4rCFWtf0Q==\n", "nzD0y4FBIhA=\n"],
    // ["8FiwfWoYxHDtRb9sWyvoRcBpkEBNEOlQwWSF\n", "pArxMzlZhyQ=\n"],
    // ["4AOVevLN2/P2DpN14svU9P0FiHr43dvj9hiVdQ==\n", "olHaO7aOmqA=\n"],
    // ["G6lVUTe7H3YGtFpAF449UDuyemwQiClPKpVgfhCTM0w=\n", "T/sUH2T6XCI=\n"],
    // ["S6InM4wZvrtLojQ0lQO5oVmiLy6WGaOnWbg1IJsSvrpW\n", "GPZmYdhG9/U=\n"],
    // ["+NDn2vFU9rvlzejL0WHUndjDxeDLY9yb1Q==\n", "rIKmlKIVte8=\n"],
    // ["vamMtHb5b/y6tJuvdv9x67y8g7Vj5Xr2obM=\n", "7v3N5iKmLr8=\n"],

    ["dKEMTz+a/hF0vxgTGbL5S3y5AUkpvvtRdKgNT3Sg7kp3\n", "Fc9oPVDzmj8=\n"],
    ["G96QImCjHHQbwIR+RosbLhPGnSR2hxk0G9eRIg==\n", "erD0UA/KeFo=\n"],
    ["QA==\n", "DHIuWtcRHPM=\n"],
    ["97HxJBfO7nI=\n", "hNSFZWWpuEI=\n"],
    ["hQG5PXN9gEE=\n", "5GLNVAUU9Dg=\n"],
    ["6ibSaFhtSRDrJNp5SW1OB/g8wHtPZlMa9w==\n", "uXKTOgwyGlU=\n"],
    ["qbfO4YGsTom/usjukapBjrSx0+GLvE6Zv6zO7g==\n", "6+WBoMXvD9o=\n"],
    ["RGOVY5WGeT9EY4ZkjJx+JVZjnX6PhmQjVnmHcIKNeT5Z\n", "FzfUMcHZMHE=\n"],
    ["a/ZMVX/pp3Zv6AAWcuSsMm3/VFV74aB+b/4=\n", "CpogdR2AyRI=\n"],
    ["L1IU+CJsgCkvTACkBESHcydKGf40SIVpL1sV+A==\n", "Tjxwik0F5Ac=\n"],
    ["23Kbgc7H69HbbI/d6O/si9NqlofY4+6R23uagQ==\n", "uhz/86Guj/8=\n"],
    ["pMYKuJux1TCk2B7kvZnSaqzeB76NldBwpM8LuA==\n", "xahuyvTYsR4=\n"],
    ["a8JgfYSH7R9y/g==\n", "BowBCe3xiE8=\n"],
    ["jn6X/WJ9JeWOYIOhRFUiv4Zmmvt0WSCljneW/SlHNb6N\n", "7xDzjw0UQcs=\n"],
    ["dx2TBGOGukN3A4dYRa69GX8FngJ1or8DdxSSBA==\n", "FnP3dgzv3m0=\n"],
    ["NLP8E/uEo+4=\n", "R9aIUonj9d4=\n"],
    ["yIIUaCMNShs=\n", "qeFgAVVkPmI=\n"],
    ["+w==\n", "t5Df7jb+QVo=\n"],
    ["zGS1mdUecVXNZr2IxB52Qt5+p4rCFWtf0Q==\n", "nzD0y4FBIhA=\n"],
    ["4AOVevLN2/P2DpN14svU9P0FiHr43dvj9hiVdQ==\n", "olHaO7aOmqA=\n"],
    ["S6InM4wZvrtLojQ0lQO5oVmiLy6WGaOnWbg1IJsSvrpW\n", "GPZmYdhG9/U=\n"],
    ["vamMtHb5b/y6tJuvdv9x67y8g7Vj5Xr2obM=\n", "7v3N5iKmLr8=\n"],
    ["Fxh7otsj1IYTBjfh1i7fwhERY6LfK9OOExA=\n", "dnQXgrlKuuI=\n"],
    ["NS1YKceVjJs1M0x14b2LwT01VS/RsYnbNSRZKQ==\n", "VEM8W6j86LU=\n"],
    ["Kb+u3htr/+MpobqCPUP4uSGno9gNT/qjKbav3g==\n", "SNHKrHQCm80=\n"],
    ["ZIkFqddEwPhklxH18WzHomyRCK/BYMW4ZIAEqQ==\n", "Bedh27gtpNY=\n"],
    ["m7XwCB9NrZabq+RUOWWqzJOt/Q4JaajWm7zxCA==\n", "+tuUenAkybg=\n"],
    ["DdaB2+RT5g==\n", "bLXitJE9kn4=\n"],
    ["Mea6/fsOqEUx673g4Qm4GH7Bn+z3CLkFJMW/4fUAqRl026r69g==\n", "UIjej5RnzGs=\n"],
    ["wcIRcGqP4zrc3x5hS6vNAeP1EV1aodUA4dUoTlWnwwfh/Ck=\n", "lZBQPjnOoG4=\n"],
    ["Pzv0yGSqC/UUP+fPZJk=\n", "S1OGrQH1apY=\n"],
    ["5NI4MmJGDgjP1is1YnU=\n", "kLpKVwcZb2s=\n"],
    ["qZTwT62fXkCCiPtarQ==\n", "3fyCKsjAPyM=\n"],
    ["u+oaes1wGJO75x1n13cIzvTNP2vBdgnTrskfZsN+Gc8=\n", "2oR+CKIZfL0=\n"],
    ["7EI3G+06AmTzUww66DEVdQ==\n", "iydDWIJUdgE=\n"],
    ["j8OjMFDZCyyB3ukRWsIZa43IiiNR0QhnnOOmNlbGCg==\n", "7q3HQj+wbwI=\n"],
    ["whQiaETnSy/CBA4=\n", "o2drBjCCOUk=\n"],
    ["ee/hgMqVxoN97w==\n", "HoqV06/nsOo=\n"],
    ["vX0MZNDuohKzYEZF2vWwVb92JXfR5qFZrl0JYtbxow==\n", "3BNoFr+Hxjw=\n"],
    ["TMkZbWldkwpT2CJMbFaEGw==\n", "K6xtLgYz528=\n"],
    ["fkp5AgYdMph+WlU=\n", "HzkwbHJ4QP4=\n"],
    ["4gCYURyyJQDmAA==\n", "hWXsAnnAU2k=\n"],
    ["7Mysgi2Jd1Xo4bixMA==\n", "gYXf1ET/GAI=\n"],
  ]
  if (m1EIFunc) {
```

```
    toDecryptStrArr.forEach((curDecryptRow) => {
      // console.log("curDecryptRow=" + curDecryptRow)
      var str1 = curDecryptRow[0]
      var str2 = curDecryptRow[1]
      // console.log("str1=" + str1 + ", str2=" + str2)
      var str1Stripped = str1.trim()
      var str2Stripped = str2.trim()
      // var decryptdStr = m1EIFunc(str1, str2)
      var decryptdStr = m1ECls.i(str1, str2)
      // console.log("str1=" + str1 + ", str2=" + str2 + " -->> " + decryptdStr)
      console.log(decryptdStr + " <<<--- str1Stripped=" + str1Stripped + ", str2Strippe
d=" + str2Stripped)
    })

    // toDecryptStrArr.forEach((item, index, arr) => {
    // console.log("item=" + item + ", index=" + index + ", arr=" + arr)


    // toDecryptStrArr.forEach((curDecryptRow, curIdx) => {
    //   console.log("curDecryptRow=" + curDecryptRow + ", curIdx=" + curIdx)
    //   var str1 = curDecryptRow[0]
    //   var str2 = curDecryptRow[1]
    //   console.log("str1=" + str1 + ", str2=" + str2)
    //   // var decryptdStr = m1EIFunc(str1, str2)
    //   var decryptdStr = m1ECls.i(str1, str2)
    //   console.log("str1=" + str1 + ", str2=" + str2 + " -->> " + decryptdStr)
    // })
  }

}



// 360Wallpaper_jadx_showBadCode/sources/m1/e.java
var m1EClassName = "m1.e"
var m1ECls = Java.use(m1EClassName)
console.log("m1ECls=" + m1ECls)
// printClassAllMethodsFields(m1EClassName)

// print static propery value
var m1EC = m1ECls.c
var m1ED = m1ECls.d
var m1EE = m1ECls.e
console.log("m1EC=" + m1EC + ", m1ED=" + m1ED + ", m1EE=" + m1EE)
var m1ECValue = m1EC.value
var m1EDValue = m1ED.value
var m1EEValue = m1EE.value
console.log("m1ECValue=" + m1ECValue + ", m1EDValue=" + m1EDValue + ", m1EEValue=" + m1
EEValue)

// public static String i(String str, String str2) {
var m1EIFunc = m1ECls.i
console.log("m1EIFunc=" + m1EIFunc)

manualDecryptString(m1EIFunc)
```

```
// hook function
if (m1EIFunc) {
  m1EIFunc.implementation = function (str1, str2) {
    // PrintStack(Throwable.$new())
    var decryptedStr = this.i(str1, str2)
    // console.log("m1.e.i(): str1=" + str1 + ", str2=" + str2 + " -> " + decryptedStr)
    var str1Stripped = str1.trim()
    var str2Stripped = str2.trim()
    console.log("m1.e.i(): str1Stripped=" + str1Stripped + ", str2Stripped=" + str2Stri
pped + " -->> " + decryptedStr)
    // saveDecrptedString(str1, str2, decryptedStr)
    return decryptedStr
  }
}
```

效果



```
...
Stack: m1.e.i Native Method)
    at a1.c.e Unknown Source 4)
    at a1.c.d onReceive OutHelper.kt 4)
    at android.app.LoadedApk ReceiverDispatcher$Args.lambda getRunnable 0 android.app.L
oadedApk ReceiverDispatcher Args LoadedApk.java 1790)
    at android.app.LoadedApk ReceiverDispatcher$Args$ExternalSyntheticLambda0.run Unkn
own Source 2)
    at android.os.Handler.handleCallback Handler.java 942)
    at android.os.Handler.dispatchMessage Handler.java 99)
    at android.os.Looper.loopOnce Looper.java 201)
    at android.os.Looper.loop Looper.java 288)
    at android.app.ActivityThread.main ActivityThread.java 7918)
    at java.lang.reflect.Method.invoke Native Method)
    at com.android.internal.os.RuntimeInit MethodAndArgsCaller.run RuntimeInit.java 548)
```

```
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:936)
m1.e.i(): str1: e5sBx5B3wHN5kg==
, str2: HPRuoPwSnwE==

...

m1.e.i(): str1Stripped: n IwzFs Og7X4DzE, str2Stripped: YNTqTlQVWU ---> facebook.com
curToDecryptStrGroup: n IwzFs Og7X4DzE
, YNTqTlQVWU==


m1.e.i(): str1Stripped: 6lmEOrkL, str2Stripped: iz3uT8p/p/4== ---> adjust
curToDecryptStrGroup: 6lmEOrkL
,iz3uT8p/p/4==


m1.e.i(): str1Stripped: YWcBkgUT1ntlZQ==, str2Stripped: AANr53ZniQk== ---> adjust_ref
curToDecryptStrGroup: YWcBkgUT1ntlZQ==
,AANr53ZniQk==


m1.e.i(): str1Stripped: u9q6nVAL9A==, str2Stripped: 1Kjd/D5il/I== ---> organic
curToDecryptStrGroup: u9q6nVAL9A==
,1Kjd/D5il/I==


m1.e.i(): str1Stripped: pfWjrvurbQ==, str2Stripped: yofEz5XCDjU== ---> organic
curToDecryptStrGroup: pfWjrvurbQ==
,yofEz5XCDjU==



m1.e.i(): str1Stripped: Zi8KGA==, str2Stripped: AE0/eZ9v58g== ---> fb4a
m1.e.i(): str1Stripped: n IwzFs Og7X4DzE, str2Stripped: YNTqTlQVWU== ---> facebook.com
m1.e.i(): str1Stripped: 6lmEOrkL, str2Stripped: iz3uT8p/p/4== ---> adjust
m1.e.i(): str1Stripped: YWcBkgUT1ntlZQ==, str2Stripped: AANr53ZniQk== ---> adjust_ref
m1.e.i(): str1Stripped: u9q6nVAL9A==, str2Stripped: 1Kjd/D5il/I== ---> organic
m1.e.i(): str1Stripped: pfWjrvurbQ==, str2Stripped: yofEz5XCDjU== ---> organic
m1.e.i(): str1Stripped: GKcOvG5nbr4=, str2Stripped: We5Y7DwNBYM== ---> AIVPRjk=
m1.e.i(): str1Stripped: urwhs9CWsta5unHxh9K7n7ehIfKJ1bacuqw498eG5tY=, str2Stripped: 3M9Jw7
/gg/8== ---> fshpov19eu82828pknh1655sfcq4xfe9
m1.e.i(): str1Stripped: gBPhX4E=, str2Stripped: 1UencrnGACo== ---> UTF-8
m1.e.i(): str1Stripped: C0YO33gie94ALSCkCnQj2g5kDJVwaiqx, str2Stripped: WhVY8DINF4w== --->
 QSV/J/lRZ8xT8y4VTqTeBg==
m1.e.i(): str1Stripped: urwhs9CWsta5unHxh9K7n7ehIfKJ1bacuqw498eG5tY=, str2Stripped: 3M9Jw7
/gg/8== ---> fshpov19eu82828pknh1655sfcq4xfe9
m1.e.i(): str1Stripped: gBPhX4E=, str2Stripped: 1UencrnGACo== ---> UTF-8
m1.e.i(): str1Stripped: lOLv8t576N2Tkc7T73ezi5Xc6KjjOL/Y2JHZxsEWyNau/vpzTm7hw==, str2St
ripped: 96i6h4hOhro== ---> cJUuV5ngd9tTg951btR/kv9b/9cAIXNlYSQnEw==
m1.e.i(): str1Stripped: urwhs9CWsta5unHxh9K7n7ehIfKJ1bacuqw498eG5tY=, str2Stripped: 3M9Jw7
/gg/8== ---> fshpov19eu82828pknh1655sfcq4xfe9
m1.e.i(): str1Stripped: gBPhX4E=, str2Stripped: 1UencrnGACo== ---> UTF-8
m1.e.i(): str1Stripped: qrWDW7z7XA==, str2Stripped: ydrtL9mDKJI== ---> context
m1.e.i(): str1Stripped: 0N9I, str2Stripped: s7M7XUPzvSY== ---> cls
m1.e.i(): str1Stripped: e5sBx5B3wHN5kg==, str2Stripped: HPRuoPwSnwE== ---> google_ref
m1.e.i(): str1Stripped: Zi8KGA==, str2Stripped: AE0/eZ9v58g== ---> fb4a
```

```
m1.e.i() str1Stripped n IwzFs Og7X4DzE, str2Stripped YNTqTlQVWU ->> facebook.com
m1.e.i() str1Stripped 6lmEOrkL, str2Stripped iz3uT8p p 4= ->> adjust
m1.e.i() str1Stripped YWcBkgUT1ntlZQ==, str2Stripped AANr53ZniQk ->> adjust_ref
m1.e.i() str1Stripped u9q6nVAL9A==, str2Stripped 1Kjd D5il I ->> organic
m1.e.i() str1Stripped pfWjrvurbQ==, str2Stripped yofEz5XCDjU ->> organic
m1.e.i() str1Stripped GKcOvG5nbr4==, str2Stripped We5Y7DwNBYM ->> AIVPRjk=
m1.e.i() str1Stripped urwhs9CWsta5unHxh9K7n7ehIfKJ1bacuqw498eG5tY, str2Stripped 3M9Jw7
gg 8 ->> fshpov19eu82828pknh1655sfcq4xfe9
m1.e.i() str1Stripped gBPhX4E, str2Stripped 1UencrnGACo ->> UTF 8
m1.e.i() str1Stripped C0YO33gie94ALSCkCnQj2g5kDJVwaiqx, str2Stripped WhVY8DINF4w ->>
QSV J lRZ8xT8y4VTqTeBg==
m1.e.i() str1Stripped urwhs9CWsta5unHxh9K7n7ehIfKJ1bacuqw498eG5tY, str2Stripped 3M9Jw7
gg 8 ->> fshpov19eu82828pknh1655sfcq4xfe9
m1.e.i() str1Stripped gBPhX4E, str2Stripped 1UencrnGACo ->> UTF 8
m1.e.i() str1Stripped lOLv8t576N2Tkc7T73ezi5Xc6KjjOL Y2JHZxsEWyNau vpzTm7hw==, str2St
ripped 96i6h4hOhro ->> cJUuV5ngd9tTg951btR kv9b 9cAIXNlYSQnEw==
m1.e.i() str1Stripped urwhs9CWsta5unHxh9K7n7ehIfKJ1bacuqw498eG5tY, str2Stripped 3M9Jw7
gg 8 ->> fshpov19eu82828pknh1655sfcq4xfe9
m1.e.i() str1Stripped gBPhX4E, str2Stripped 1UencrnGACo ->> UTF 8



Updated -> decryptedStrDict {"0mIyyw==\n,pQNbvzxsUjA=\n":"wait","qrWDW7z7XA==\n,ydrtL9m
DKJI=\n":"context"}

Updated -> decryptedStrDict {"0mIyyw==\n,pQNbvzxsUjA=\n":"wait","qrWDW7z7XA==\n,ydrtL9m
DKJI=\n":"context","0N9I\n,s7M7XUPzvSY=\n":"cls"}

...

Updated -> decryptedStrDict {
  "RHk=\n,NEk26SxLQpY=\n": "p0",
  "Ctg=\n,euj9oN1O89Q=\n": "p0",
  "WNoEx1td\n,ZK5srihjjHA=\n": "<this>",
  "6tol5TJn\n,xK1VtlcTvZ4=\n": ".wpSet",
  "ybAkZWjlVXzbuzRjbuJWIYaNBUNTxX8V+w==\n,qN5AFweMMVI=\n": "android.settings.SETTINGS",
  "iznC7BGVypmSDsLzOg==\n,/1Gxlk7zuPY=\n": "thsz_from_set",
  "MzhD1Fyu+7sqD0PLdw==\n,R1AwrgPIidQ=\n": "thsz_from_set",
  "TD0nYsjkP49UOAxS1vQ8\n,O014DbiBUdA=\n": "wp_open_out_num",
  "NIU=\n,RLXuwslcso4=\n": "p0",
  "v68=\n,z59QGAfzgVI=\n": "p0",
  "10g=\n,p3jUwVR1TlE=\n": "p0",
  "bTY=\n,HQYlg2SZsI8=\n": "p0",
  "+pM=\n,iqNAgnhpDJw=\n": "p0",
  "+mc=\n,ilcKLu3y9Hc=\n": "p0",
  "qfk=\n,2ckbQ5GN/s8=\n": "p0",
  "aog=\n,GrhJtEmflVA=\n": "p0",
  "qjI=\n,2gK64i8EPUY=\n": "p0",
  "ZNg=\n,FOgMtYm5CFA=\n": "p0",
  "+5/dnwJs0ULhmcau\n,iPqpwGMcoR0=\n": "set_app_icon",
  "b4w=\n,H7wXwoPm++I=\n": "p0",
  "Z3U=\n,F0R7J/92o8c=\n": "p1",
  "0mIyyw==\n,pQNbvzxsUjA=\n": "wait",
  "qrWDW7z7XA==\n,ydrtL9mDKJI=\n": "context",
  "0N9I\n,s7M7XUPzvSY=\n": "cls",
  "e5sBx5B3wHN5kg==\n,HPRuoPwSnwE=\n": "google_ref",
```

```
  "Zi8KGA==\n,AE0+eZ9v58g=\n": "fb4a",
  "n+IwzFs/Og7X4DzE\n,+YNTqTlQVWU=\n": "facebook.com",
  "6lmEOrkL\n,iz3uT8p/p/4=\n": "adjust",
  "YWcBkgUT1ntlZQ==\n,AANr53ZniQk=\n": "adjust_ref",
  "u9q6nVAL9A==\n,1Kjd/D5il+I=\n": "organic",
  "pfWjrvurbQ==\n,yofEz5XCDjU=\n": "organic",
  "GKcOvG5nbr4=\n,We5Y7DwNBYM=\n": "AIVPRjk=",
  "urwhs9CWsta5unHxh9K7n7ehIfKJ1bacuqw498eG5tY=\n,3M9Jw7/gg+8=\n": "fshpov19eu82828pknh
1655sfcq4xfe9",
  "gBPhX4E=\n,1UencrnGACo=\n": "UTF-8",
  "C0YO33gie94ALSCkCnQj2g5kDJVwaiqx\n,WhVY8DINF4w=\n": "QSV/J/lRZ8xT8y4VTqTeBg==",
  "lOLv8t576N2Tkc7T73ezi5Xc6KjjOL/Y2JHZxsEWyNau++vpzTm7hw==\n,96i6h4hOhro=\n": "cJUuV5n
gd9tTg951btR/kv9b/9cAIXNlYSQnEw==",
  "lGsWeqSRpxScawZtpYztW5ZxG2el1oB2ulY3V5ihkG6wSC1MgrmPdbJW\n,9QVyCMv4wzo=\n": "android
.intent.action.CLOSE_SYSTEM_DIALOGS"
}
```

》手动解密：

```
m1EIFunc function e() {
    [native code]
}
TRANSACTION_startService <---> str1Stripped: o3etnXwGpjjkNGnVcU4vt6M opQ2Dqp, str2Stri
pped: qt f CaxWcw
START_SERVICE_TRANSACTION <---> str1Stripped: zGS1mdUecVXNZr2IxB52Qt5 p4rCFWtf0Q==, str
2Stripped: nzD0y4FBIhA
TRANSACTION_broadcastIntent <---> str1Stripped: 8FiwfWoYxHDtRb9sWyvoRcBpkEBNEOlQwWSF, s
tr2Stripped: pArxMzlZhyQ
BROADCAST_INTENT_TRANSACTION <---> str1Stripped: 4AOVevLN2 P2DpN14svU9P0FiHr43dvj9hiVdQ
=, str2Stripped: olHaO7aOmqA
TRANSACTION_startInstrumentation <---> str1Stripped: G6lVUTe7H3YGtFpAF449UDuyemwQiClPKp
VgfhCTM0w, str2Stripped: T sUH2T6XCI
START_INSTRUMENTATION_TRANSACTION <---> str1Stripped: S6InM4wZvrtLojQ0lQO5oVmiLy6WGaOnW
bg1IJsSvrpW, str2Stripped: GPZmYdhG9 U
TRANSACTION_startActivity <---> str1Stripped: NDn2vFU9rvlzejL0WHUndjDxeDLY9yb1Q==, str
2Stripped: rIKmlKIVte8
START_ACTIVITY_TRANSACTION <---> str1Stripped: vamMtHb5b y6tJuvdv9x67y8g7Vj5Xr2obM, st
r2Stripped: 7v3N5iKmLr8
```

```
/Users/crifan/dev/dev_root/androidReverse/keepAlive/360Wallpaper/staticAnalysis/360Wall
paper_jadx_showBadCode/sources/com/w/thsz/info/
  PaperEntry.java
  HomeEntry.java
```

》字符串解密解码结果：

```
m1EIFunc function e() {
    [native code]
}
android.app.IActivityManager$Stub <---> str1Stripped: dKEMTz a hF0vxgTGbL5S3y5AUkpvvtRd
KgNT3Sg7kp3, str2Stripped: Fc9oPVDzmj8
android.app.IActivityManager <---> str1Stripped: G96QImCjHHQbwIR RosbLhPGnSR2hxk0G9eRIg
=, str2Stripped: erD0UA KeFo
L <---> str1Stripped: QA==, str2Stripped: DHIuWtcRHPM
```

```
setArgV0 <<< --- str1Stripped 97HxJBfO7nI , str2Stripped hNSFZWWpuEI
activity <<< --- str1Stripped hQG5PXN9gEE , str2Stripped 5GLNVAUU9Dg
START_SERVICE_TRANSACTION <<< --- str1Stripped 6ibSaFhtSRDrJNp5SW1OB g8wHtPZlMa9w =, str
2Stripped uXKTOgwyGlU
BROADCAST_INTENT_TRANSACTION <<< --- str1Stripped qbfO4YGsTom usjukapBjrSx0 GLvE6Zv6zO7g
=, str2Stripped 6 WBoMXvD9o
START_INSTRUMENTATION_TRANSACTION <<< --- str1Stripped RGOVY5WGeT9EY4ZkjJx JVZjnX6PhmQjV
nmHcIKNeT5Z, str2Stripped FzfUMcHZMHE
all binder code get failed <<< --- str1Stripped a ZMVX pp3Zv6AAWcuSsMm3 VFV74aB b 4 , st
r2Stripped CpogdR2AyRI
android app IActivityManager <<< --- str1Stripped L1IU CJsgCkvTACkBESHcydKGf40SIVpL1sV A
=, str2Stripped Tjxwik0F5Ac
android app IActivityManager <<< --- str1Stripped 23Kbgc7H69HbbI d6O si9NqlofY4 6R23uagQ
=, str2Stripped uhz 86Guj 8
android app IActivityManager <<< --- str1Stripped pMYKuJux1TCk2B7kvZnSaqzeB76NldBwpM8LuA
=, str2Stripped xahuyvTYsR4
mNativePtr <<< --- str1Stripped a8JgfYSH7R9y g =, str2Stripped BowBCe3xiE8
android app IActivityManager Stub <<< --- str1Stripped jn6X WJ9JeWOYIOhRFUiv4Zmmvt0WSClj
neW SlHNb6N, str2Stripped 7xDzjw0UQcs
android app IActivityManager <<< --- str1Stripped dx2TBGOGukN3A4dYRa69GX8FngJ1or8DdxSSBA
=, str2Stripped FnP3dgzv3m0
setArgV0 <<< --- str1Stripped NLP8E uEo 4 , str2Stripped R9aIUonj9d4
activity <<< --- str1Stripped yIIUaCMNShs , str2Stripped qeFgAVVkPmI
L <<< --- str1Stripped w =, str2Stripped t5Df7jb QVo
START_SERVICE_TRANSACTION <<< --- str1Stripped zGS1mdUecVXNZr2IxB52Qt5 p4rCFWtf0Q =, str
2Stripped nzD0y4FBIhA
BROADCAST_INTENT_TRANSACTION <<< --- str1Stripped 4AOVevLN2 P2DpN14svU9P0FiHr43dvj9hiVdQ
=, str2Stripped olHaO7aOmqA
START_INSTRUMENTATION_TRANSACTION <<< --- str1Stripped S6InM4wZvrtLojQ0lQO5oVmiLy6WGaOnW
bg1IJsSvrpW, str2Stripped GPZmYdhG9 U
START_ACTIVITY_TRANSACTION <<< --- str1Stripped vamMtHb5b y6tJuvdv9x67y8g7Vj5Xr2obM , st
r2Stripped 7v3N5iKmLr8
all binder code get failed <<< --- str1Stripped Fxh7otsj1IYTBjfh1i7fwhERY6LfK9OOExA , st
r2Stripped dnQXgrlKuuI
android app IActivityManager <<< --- str1Stripped NS1YKceVjJs1M0x14b2LwT01VS RsYnbNSRZKQ
=, str2Stripped VEM8W6j86LU
android app IActivityManager <<< --- str1Stripped Kb u3htr MpobqCPUP4uSGno9gNT qjKbav3g
=, str2Stripped SNHKrHQCm80
android app IActivityManager <<< --- str1Stripped ZIkFqddEwPhklxH18WzHomyRCK BYMW4ZIAEqQ
=, str2Stripped Bedh27gtpNY
android app IActivityManager <<< --- str1Stripped m7XwCB9NrZabq RUOWWqzJOt Q4JaajWm7zxCA
=, str2Stripped tuUenAkybg
account <<< --- str1Stripped DdaB2 RT5g =, str2Stripped bLXitJE9kn4
android accounts IAccountManager Stub <<< --- str1Stripped Mea6 fsOqEUx673g4Qm4GH7Bn z3C
LkFJMW 4fUAqRl026r69g =, str2Stripped UIjej5RnzGs
TRANSACTION_removeAccountExplicitly <<< --- str1Stripped wcIRcGqP4zrc3x5hS6vNAeP1EV1aodU
A4dUoTlWnwwfh Ck , str2Stripped lZBQPjnOoG4
three_ac_label <<< --- str1Stripped Pzv0yGSqC UUP fPZJk , str2Stripped S1OGrQH1apY
three_ac_label <<< --- str1Stripped 5NI4MmJGDgjP1is1YnU , str2Stripped kLpKVwcZb2s
three_ac_type <<< --- str1Stripped qZTwT62fXkCCiPtarQ =, str2Stripped 3fyCKsjAPyM
android accounts IAccountManager <<< --- str1Stripped u oaes1wGJO75x1n13cIzvTNP2vBdgnTrs
kfZsN Gc8 , str2Stripped 2oR CKIZfL0
getContextObject <<< --- str1Stripped 7EI3G 06AmTzUww66DEVdQ =, str2Stripped iydDWIJUdgE

android os ServiceManagerNative <<< --- str1Stripped j8OjMFDZCyyB3ukRWsIZa43IiiNR0QhnnOO
mNlbGCg =, str2Stripped 7q3HQj wbwI
```

```
asInterface <<< --- str1Stripped whQiaETnSy CBA4, str2Stripped o2drBjCCOUk
getService <<< --- str1Stripped ee hgMqVxoN97w==, str2Stripped HoqV06 nsOo
android.os.ServiceManagerNative <<< --- str1Stripped vX0MZNDuohKzYEZF2vWwVb92JXfR5qFZrl0
JYtbxow==, str2Stripped 3BNoFr Hxjw
getContextObject <<< --- str1Stripped TMkZbWldkwpT2CJMbFaEGw==, str2Stripped K6xtLgYz528

asInterface <<< --- str1Stripped fkp5AgYdMph WlU, str2Stripped HzkwbHJ4QP4
getService <<< --- str1Stripped 4gCYURyyJQDmAA==, str2Stripped hWXsAnnAU2k
mIsVivoWidget <<< --- str1Stripped 7Mysgi2Jd1Xo4bixMA==, str2Stripped gYXf1ET GAI
m1.e: Found instance object m1.e b55a893
m1.e: Found instance object m1.e 187c0d0
m1.e: Found instance object m1.e 46ca8c9
m1.e: onComplete
```

即可自动解密，得到我们要的原始的字符串的值了。

用JEB自动反编译字符串

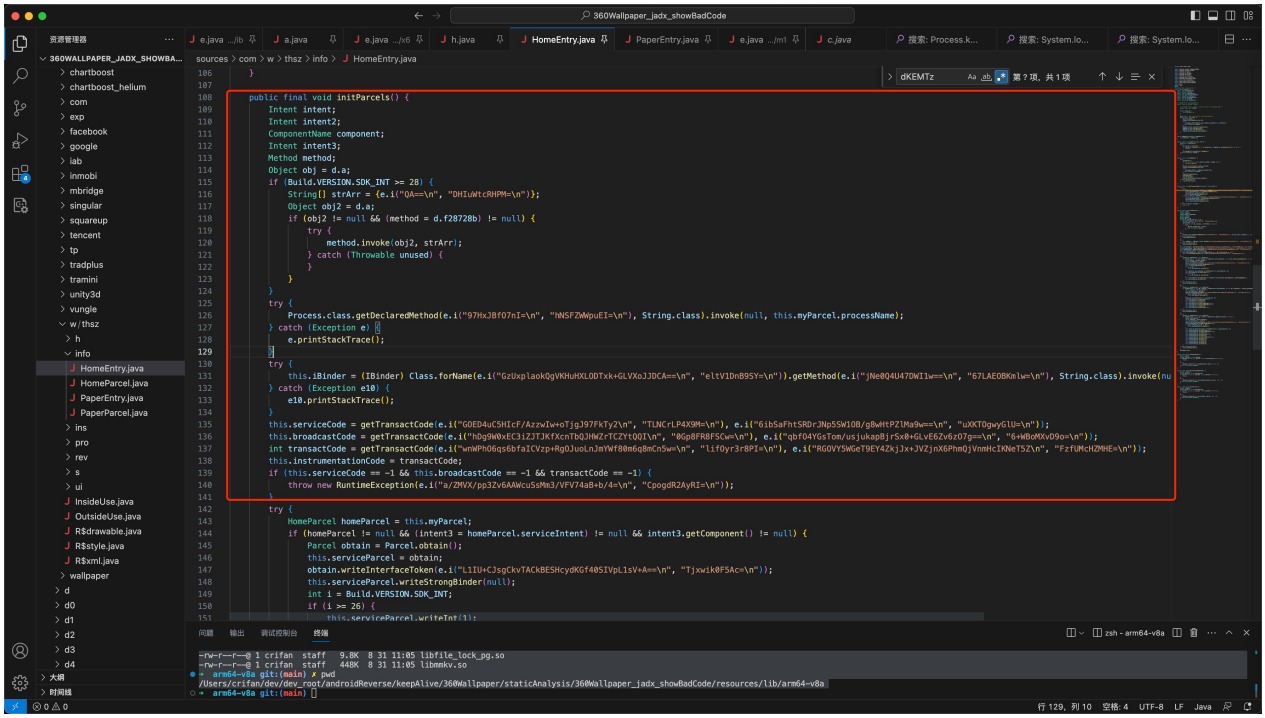
用JEB反编译代码时，默认自动已开启：字符串反混淆=字符串反编译

举例：

效果对比

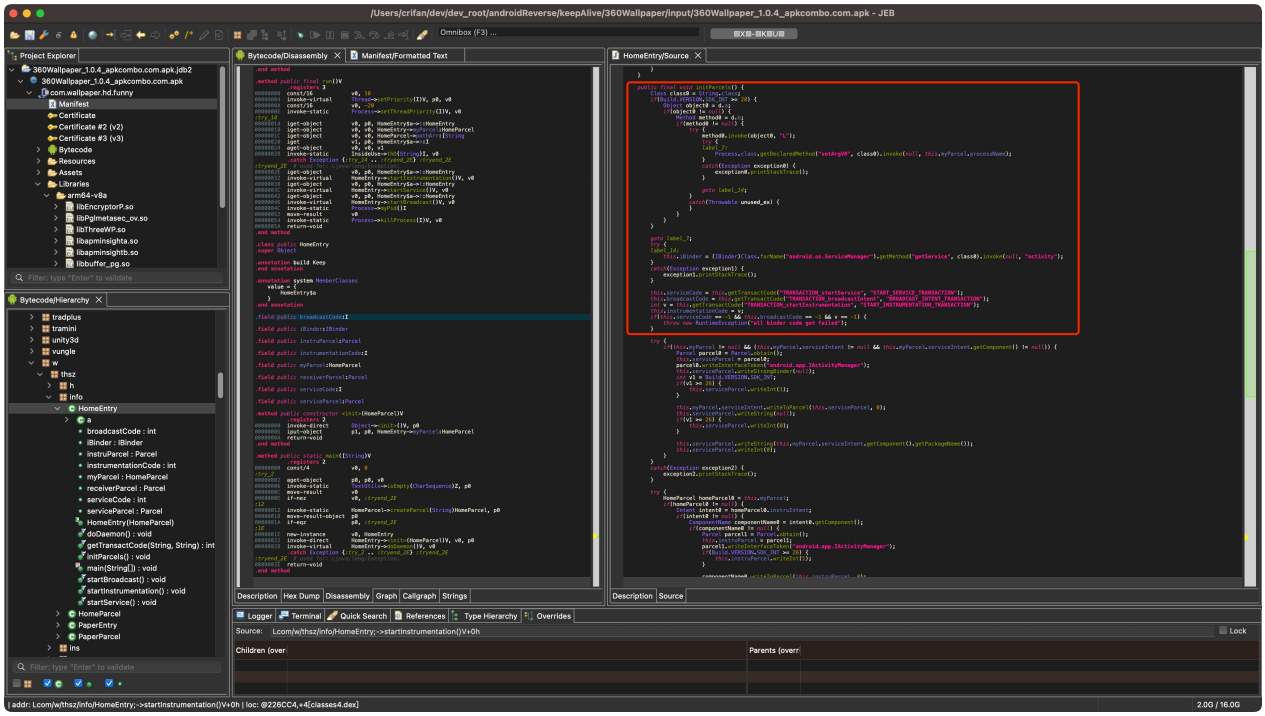
jadx：无法自动反混淆字符串

```
    public final void initParcels() {
...
            Process.class.getDeclaredMethod(e.i("97HxJBfO7nI=\n", "hNSFZWWpuEI=\n"), St
ring.class).invoke(null, this.myParcel.processName);
        } catch (Exception e) {
            e.printStackTrace();
        }
        try {
            this.iBinder = (IBinder) Class.forName(e.i("GzUxplaokQgVKHuHXLODTxk+GLVXoJJ
DCA==\n", "eltV1DnB9SY=\n")).getMethod(e.i("jNe0Q4U47DWI1w==\n", "67LAEOBKmlw=\n"), Str
ing.class).invoke(null, e.i("hQG5PXN9gEE=\n", "5GLNVAUU9Dg=\n"));
        } catch (Exception e10) {
            e10.printStackTrace();
        }
        this.serviceCode = getTransactCode(e.i("GOED4uC5HIcF/AzzwIw+oTjgJ97FkTy2\n", "T
LNCrLP4X9M=\n"), e.i("6ibSaFhtSRDrJNp5SW1OB/g8wHtPZlMa9w==\n", "uXKTOgwyGlU=\n"));
        this.broadcastCode = getTransactCode(e.i("hDg9W0xEC3iZJTJKfXcnTbQJHWZrTCZYtQQI\
n", "0Gp8FR8FSCw=\n"), e.i("qbfO4YGsTom/usjukapBjrSx0+GLvE6Zv6zO7g==\n", "6+WBoMXvD9o=\
n"));
        int transactCode = getTransactCode(e.i("wnWPhO6qs6bfaICVzp+RgOJuoLnJmYWf80m6q8m
Cn5w=\n", "lifOyr3r8PI=\n"), e.i("RGOVY5WGeT9EY4ZkjJx+JVZjnX6PhmQjVnmHcIKNeT5Z\n", "FzfU
McHZMHE=\n"));
```



JEB：可以自动反混淆字符串 = 还原出原始字符串

```
    public final void initParcels() {
...
                        Process.class.getDeclaredMethod("setArgV0", class0).invoke(
null, this.myParcel.processName);
...
        }
...
            this.iBinder = (IBinder)Class.forName("android.os.ServiceManager").getMethod
("getService", class0).invoke(null, "activity");
...
        this.serviceCode = this.getTransactCode("TRANSACTION_startService", "START_SERV
ICE_TRANSACTION");
        this.broadcastCode = this.getTransactCode("TRANSACTION_broadcastIntent", "BROAD
CAST_INTENT_TRANSACTION");
        int v = this.getTransactCode("TRANSACTION_startInstrumentation", "START_INSTRUM
ENTATION_TRANSACTION");
```



常见问题

通用

Process terminated

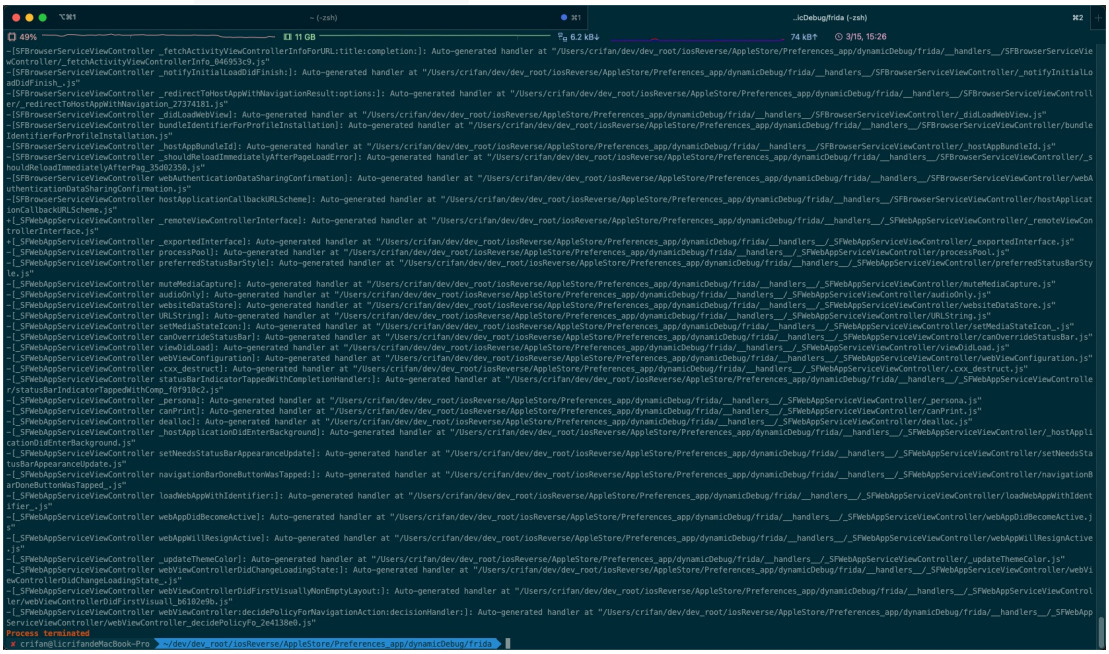
关于 `Process terminated` = `进程结束` = `崩溃退出` ，目前遇到多种现象和可能原因：

由于hook函数太多导致崩溃

- 现象一：用frida-trace去hook太多的类Obj的函数

```
frida-trace -U -F com.apple.Preferences -m "*[AA* *]" -m "*[AK* *]" -m "*[AS* *]" -m "*[NS* *]" -M "-[ASDBundle copyWithZone:]" -M "-[ASDInstallationEvent copyWithZone:]"
```

  - 导致崩溃退出报错： `Process terminated`



- 原因： `frida-trace` 去hook的函数太多了，估计是，加了 `-m "*[NS* *]"` 后导致崩溃
  - 注：iOS的ObjC的内部的多数，甚至是大多数，都是 `NS` 开头的，导致匹配到太多的类和函数，系统处理不过来了，导致frida崩溃，同时导致被调试的app崩溃。
    - 注： `NS` = `NextStep` ，是iOS系统前身的苹果收购的NextStep公司名字
  - 解决办法：减少hook的范围=缩小匹配范围，比如此处改为： `-m "*[NSXPC* *]"` ，暂时只关注我们要调试的 `NSXPCConnection` 的相关内容，基本上可以：避免崩溃

frida的 `new ObjC.Object` 方面的bug

- 现象：
  - 概述 `frida` 调试时，由于加了 `new ObjC.Object(someArg)` 的 `ptr` 转换成 `ObjC的对象` ，结果就会时不时的遇到 `Process terminated` ，而崩溃停止退出调试
  - 详解=具体现象

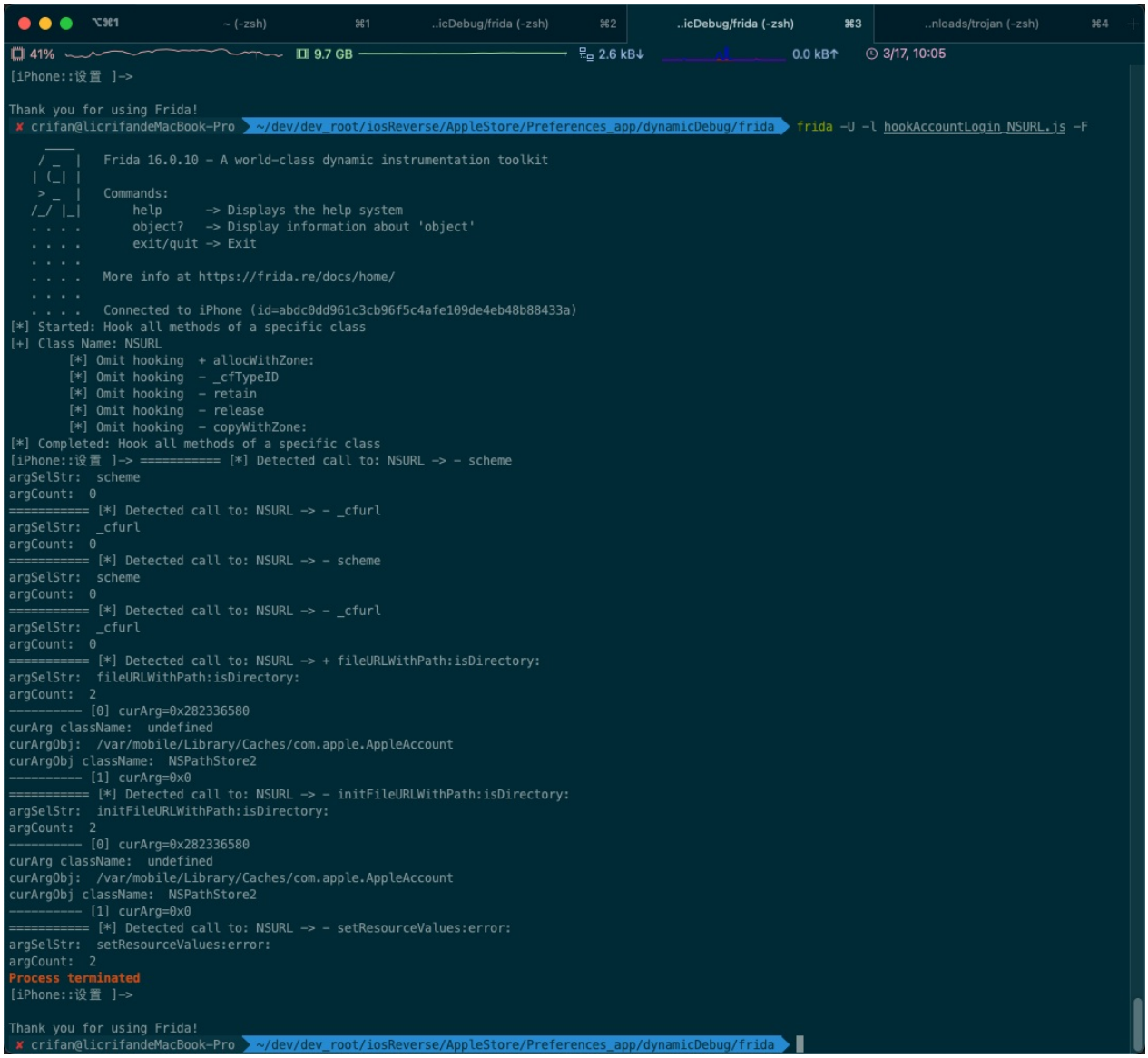
代码：

```
...
function hook_class_method(class_name, method_name)
{
    var hook = ObjC.classes[class_name][method_name];
    Interceptor.attach(hook.implementation, {
        onEnter: function(args) {
            console.log("=========== [*] Detected call to: " + class_name + " -> " + me
thod_name);
            //objc的函数，第0个参数是id，第1个参数是SEL，真正的参数从args[2]开始
            const argId = args[0];
            // console.log("argId: ", argId);

            const argSel = args[1];
            // console.log("argSel: ", argSel);

            const argSelStr = ObjC.selectorAsString(argSel);
            console.log("argSelStr: ", argSelStr);


            const argCount = occurrences(argSelStr, ":");
            console.log("argCount: ", argCount);


            for (let curArgIdx = 0; curArgIdx < argCount; curArgIdx++) {
                const curArg = args[curArgIdx + 2];
                // console.log("[%d] curArg: ", curArgIdx, curArg);
                // console.log("[%d]=", curArgIdx, " ,curArg=", curArg);
                // console.log("[%d]=" + curArgIdx + " ,curArg=" + curArg);
                console.log("---------- [" + curArgIdx + "] curArg=" + curArg);
                if (curArg && (curArg != 0x0)) {
                    console.log("curArg className: ", curArg.$className);
                    const curArgObj = new ObjC.Object(curArg);
                    console.log("curArgObj: ", curArgObj);
                    console.log("curArgObj className: ", curArgObj.$className);
                }
            }
        }
    });
}
...
```

可以hook输出部分log日志，但是很快，时不时的，就崩溃退出了：

```
✘ crifan@licrifandeMacBook-Pro ~/dev/dev_root/iosReverse/AppleStore/Preferences_app/dy
namicDebug/frida frida -U -l hookAccountLogin_NSURL.js -F

     ____
    / _  |   Frida 16.0.10 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
```

```
   . . . .  Connected to iPhone (id abdc0dd961c3cb96f5c4afe109de4eb48b88433a)
[*] Started: Hook all methods of a specific class
[+] Class Name: NSURL
    [*] Omit hooking  + allocWithZone:
    [*] Omit hooking  - _cfTypeID
    [*] Omit hooking  - retain
    [*] Omit hooking  - release
    [*] Omit hooking  - copyWithZone:
[*] Completed: Hook all methods of a specific class
[iPhone::设置 ]-> ========== [*] Detected call to: NSURL -> - scheme
argSelStr:  scheme
argCount:  0
========== [*] Detected call to: NSURL -> - _cfurl
argSelStr:  _cfurl
argCount:  0
========== [*] Detected call to: NSURL -> - scheme
argSelStr:  scheme
argCount:  0
========== [*] Detected call to: NSURL -> - _cfurl
argSelStr:  _cfurl
argCount:  0
========== [*] Detected call to: NSURL -> + fileURLWithPath:isDirectory:
argSelStr:  fileURLWithPath:isDirectory:
argCount:  2
---------- [0] curArg 0x282336580
curArg className:  undefined
curArgObj:  /var/mobile/Library/Caches/com.apple.AppleAccount
curArgObj className:  NSPathStore2
---------- [1] curArg 0x0
========== [*] Detected call to: NSURL -> - initFileURLWithPath:isDirectory:
argSelStr:  initFileURLWithPath:isDirectory:
argCount:  2
---------- [0] curArg 0x282336580
curArg className:  undefined
curArgObj:  /var/mobile/Library/Caches/com.apple.AppleAccount
curArgObj className:  NSPathStore2
---------- [1] curArg 0x0
========== [*] Detected call to: NSURL -> - setResourceValues:error:
argSelStr:  setResourceValues:error:
argCount:  2
Process terminated
[iPhone::设置 ]->


Thank you for using Frida
```



- 原因：暂不完全清楚
  - 可能原因：`frida` 的 `ObjC` 的 `Objcect` 转换方面的bug，暂时无法解决
    - 详见：
      - 【未解决】frida中hook函数打印参数值时最后app崩溃frida输出Process terminated
      - 【未解决】frida中hook调试iOS的ObjC的函数参数时始终出现崩溃Process terminated

Failed to spawn: unable to find process with name

- 报错：`Failed to spawn: unable to find process with name 'Preferences'`
  -
- 原因：frida命令用的是

```
frida -U -l ./hookAccountLogin.js -n Preferences
```

  - 其中`-n`是加二进制名称，此处`Preferences`是app，所以属于参数使用错误，调试目标语法搞错了
- 解决办法：
  - 搞懂frida的调试目标方式，改为别的方式即可
    - 方式1：用`-N app_package_id`

```
frida -U -l ./hookAccountLogin.js -N com.apple.Preferences
```

    - 方式2：换`-p PID`

```
frida -U -l ./hookAccountLogin.js -p 18031
```

      - 其中是用iPhone中ssh中通过ps查看到

```
~ ps -A | grep Preferences
...
18031 ??         0:02.43 /Applications/Preferences.app/Preferences
```

        - 得知`Preferences`的`PID`是`18031`
    - 方式3：用`-F`

```
frida -U -l ./hookAccountLogin.js -F
```

      - 注：确保`Preferences`=系统的设置app，处于最前台在运行才能用`-F`

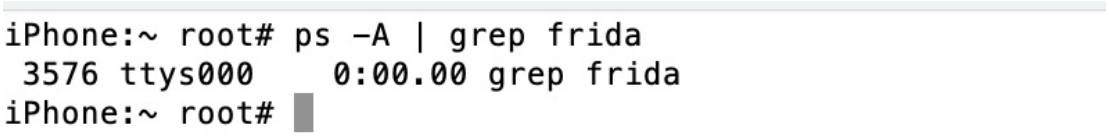
Failed to attach unable to attach to the specified proces

- 背景：用 `palera1nC` 越狱后
  - 注：之前只有 `palera1nC` 能越狱 `15.6` 的系统
- 现象： `frida-trace` 去hook调试报错

```
~% frida-trace -U -i CCCrypt -p 12959
Failed to attach: unable to attach to the specified proces
```

  -
- 原因：iPhone端的frida（ `frida-server` ）没有正常运行
  - 如何确认=如何找到的问题的原因?

```
iPhone:~ root# ps -A l grep frida
3576 ttys000
0:00.00 grep frida
iPhone:~root#
```



    -
- 解决办法
  - iPhone端，重新安装frida，确保安装后 `frida-server` 已正常启动在运行

--no-pause和--pause

关于frida启动后，被调试的目标，是否暂停运行的问题：

- 背景
  - frida启动调试后，被调试的目标（app或进程），是否已经暂停运行
    - 旧版frida：自动暂停运行
      - 支持参数：`--no-pause`
    - 新版frida：（逻辑已经变成了）不暂停运行 = 已经继续运行了
      - 支持参数：`--pause`
- 所以
  - 旧版frida
    - 常会遇到一个问题：每次frida（以Attach后Spawn去）启动调试后，程序自动暂停运行
      - 解决办法：手动输入 `%resume`
      - 

    - 所以就希望：frida调试开始后，自动继续运行，不要每次都输入 `%resume` 才继续运行
      - 解决办法：加 `--no-pause` 参数
        - 参数含义：`--no-pause automatically start main thread after startup`
        - 举例

```
frida -U --no-pause -f com.ss.iphone.ugc.Aweme -l frida/dyldImage.js
```

  - 新版frida
    - 用新版frida，加了参数 `--no-pause` ，报错不支持此参数：`frida: error: unrecognized arguments: --no-pause`
      - 解决办法：不要加任何参数
        - 举例

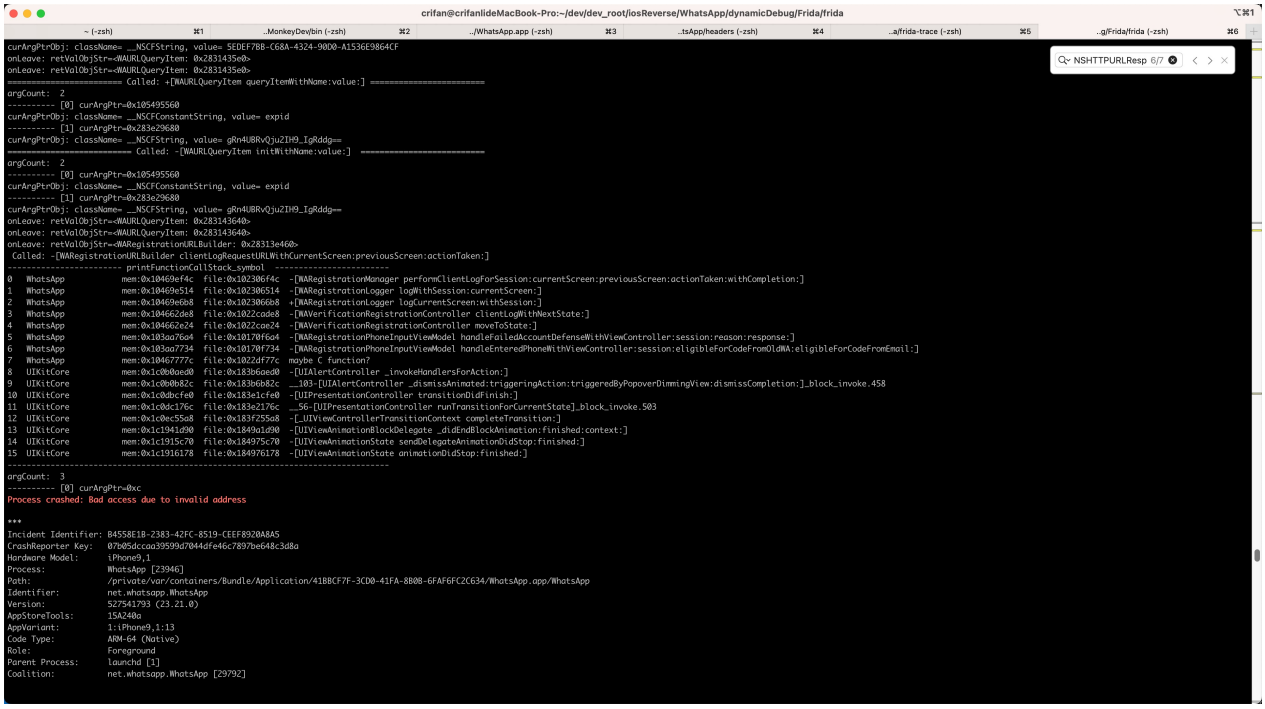
```
frida -U -N com.apple.Preferences -l hookAccountLogin_singleClassAllMethod.js
```

    - -》
      - 如果需要启动后
        - 自动继续运行
          - 则：无需加任何参数
            - 因为新版frida已经变成这个逻辑了
        - 自动暂停运行
          - 再去加新版才支持的参数：`--pause`
            - 参数含义：`--pause leave main thread paused after spawning program`

Process crashed: Bad access due to invalid address

- 问题：

```
Called: -[WARegistrationURLBuilder clientLogRequestURLWithCurrentScreen:previousScreen:
actionTaken:]
'''
argCount:  3
---------- [0] curArgPtr=0xc
Process crashed: Bad access due to invalid address
'''
```



- 原因
  - 表面原因：访问了非法地址：`0xc`
  - 深层次原因：对于值明显异常的地址，没有做过滤，没有排除掉
- 解决办法：加上过滤，排除掉，地址值明显异常的地址
- 具体步骤：

代码改为：

```
// check pointer is valid or not
// example
//          0x103e79560 => true
//          0xc => false
function isValidPointer(curPtr){
    let MinValidPointer = 0x10000
    var isValid = curPtr > MinValidPointer
    return isValid
}

'''
```

```
if (isValidPointer(curArg)) {
  ...
}
```

即可避免访问非法地址指针，避免崩溃。

ValueError: file descriptor cannot be a negative integer

- 问题：`frida-ps`、`frida-ls-devices`等frida-tools工具运行时报错：`ValueError: file descriptor cannot be a negative integer (-42)`
- 原因：当前`12.0.3`的`frida-tools`有bug
- 解决办法：升级到最新版`frida-tools`
- 具体步骤：

```
pip install --upgrade frida_tools
```

  - 注：查看当前frida-tools的版本：

```
pip show frida_tools
```

卡死在Spawning

- 现象：frida去hook时，一直显示=卡死在 `Spawning`

```
➜  frida frida -U -f com.ss.android.ugc.aweme -l hook_douyin.js
     ____
    / _  |   Frida 16.4.6 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
  /_/ |_|       help      -> Displays the help system
  . . . .       object?   -> Display information about 'object'
  . . . .       exit/quit -> Exit
  . . . .
  . . . .   More info at https://frida.re/docs/home/
  . . . .
  . . . .   Connected to Pixel 5 (id=9C181A8D3C3F3B)
Spawning `com.ss.android.ugc.aweme`...
```

  -
  - -> 当时的错误操作：以为真的卡死了，所以去找别的可能的原因，所以就去直接用 `Ctrl+C` 中断运行了
- 原因：其实并没有卡死，而是多等会，会有最终结果
  - 是超时报错，还是能正常运行
- 解决办法：多等一会，看看最终的结果

结果1：正常

结果1：正常继续hook和输出log

```
➜  frida frida -U -f com.ss.android.ugc.aweme -l hook_douyin.js
...
Spawning `com.ss.android.ugc.aweme`...
Java is available
Java.androidVersion=13
```

```
Spawned `com.ss.android.ugc.aweme`. Resuming main thread
[Pixel 5::com.ss.android.ugc.aweme ]->
```

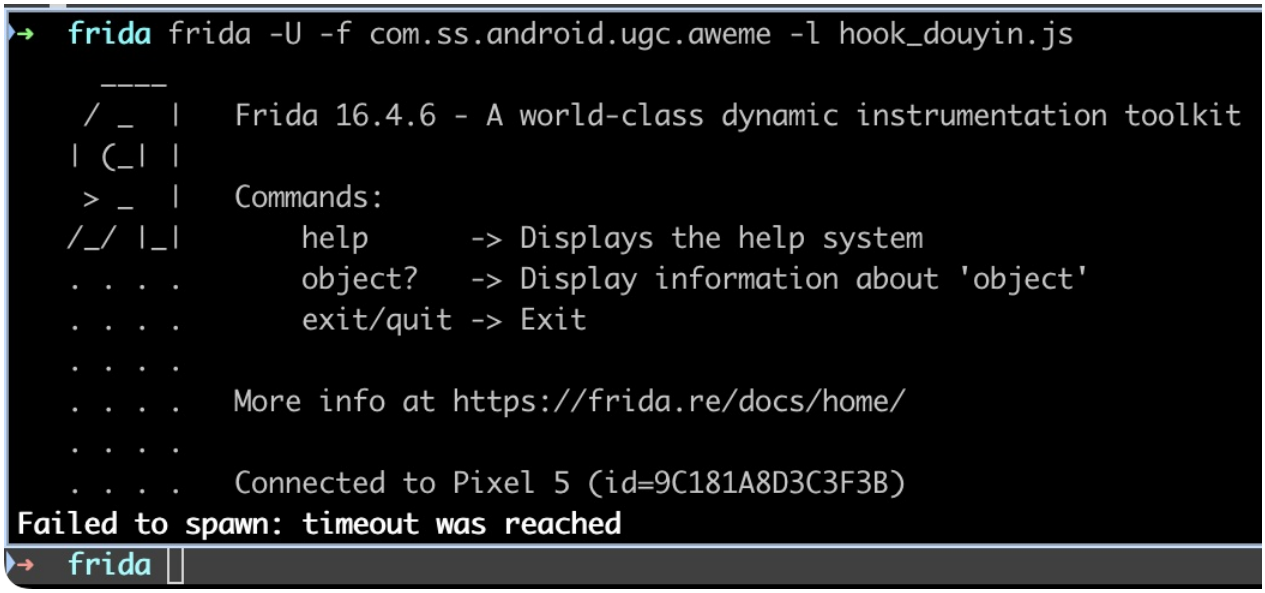
- 解决办法：无需任何操作，只是稍微多等会

结果2：异常

结果1：报错超时等异常

情况1

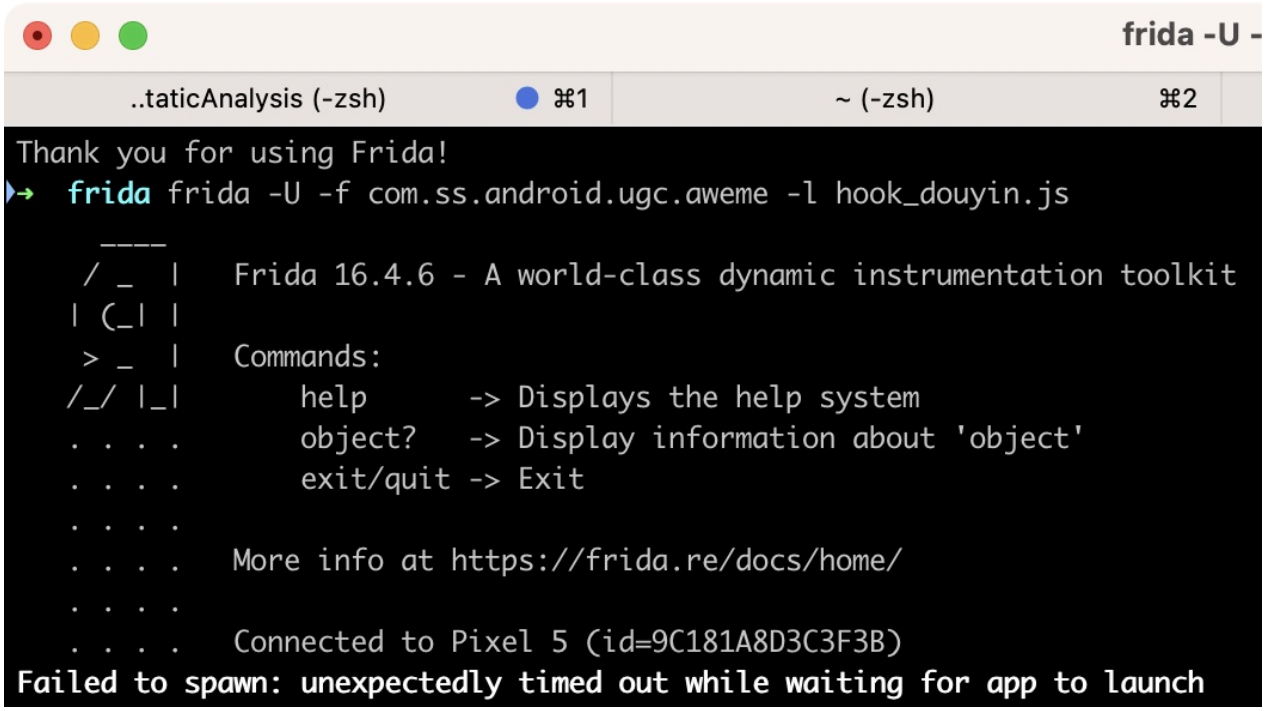
```
➜  frida frida -U -f com.ss.android.ugc.aweme -l hook_douyin.js
...
   . . . .   Connected to Pixel 5 (id 9C181A8D3C3F3B)
Failed to spawn: timeout was reached
```



- 此处的背景
  - 此处之前Frida的hook，触发输出太多log，自己强制输入（多次输入，因为输出log太多，都无法顺利输入）exit退出Frida后，然后不知道什么情况，总之是：导致了安卓端的Frida异常
- 解决办法：重启Android手机

情况2

```
➜  frida frida -U -f com.ss.android.ugc.aweme -l hook_douyin.js
...
   . . . .   Connected to Pixel 5 (id 9C181A8D3C3F3B)
Failed to spawn: unexpectedly timed out while waiting for app to launch
```



- 解决办法：多试试几次（有时候就正常了）

js常见问题

RangeError: invalid array index

- 错误：Frida的hook脚本js中

```
Interceptor.attach(functionRealAddress, {
  onEnter: function(args) {
    console.log(args);
```

  - 会报错：`RangeError: invalid array index`
- 原因：无法知道准确的`args`的数组的大小，无法直接打印，所以报错
- 解决办法：不去直接打印，而改为去获取对应的前几个参数（前提：已知参数个数），再去打印：

```
Interceptor.attach(functionRealAddress, {
  onEnter: function(args) {
    var arg0 = args[0]
    var arg1 = args[1]
    var arg2 = args[2]
    console.log("arg0=" + arg0 + ", arg1=" + arg1 + ", arg2=" + arg2);
```

  - 进一步优化：甚至是，计算出此处的ObjC的参数的个数，循环批量打印所有参数
    - 具体详见：ObjC的参数 · 逆向调试利器：Frida中的`计算ObjC的函数的真正参数的个数 + 打印全部参数`

iOS

最后更新：2024-07-22 15:25:54

用ApiResolver去hook已触发函数但没日志

- 问题：之前用了 `ApiResolver` 去写了hook代码，后续调试的函数确保被触发了，但是却没输出我们希望的日志
  - 注：`ApiResolver` 的完整代码详见
    - ApiResolver · Frida逆向实例和工具函数

```
var resolver = new ApiResolver('objc');
resolver.enumerateMatches('*[AAUISignInViewController *]', {
  onMatch: function(match) {
    console.log(match['name'] + ":" + match['address']);
  },

  onComplete: function() {}
});
```

- 原因：此处 `ApiResolver` 只是search查找，函数是否存在，只运行一次，后续的确，本身就不会再次触发，所以后续代码被触发时，就不会打印日志
- 解决办法：真正的要实现，hook函数，待函数被运行后触发日志打印，应该改用：`Interceptor`
- 具体写法
  - 注：
    - `Interceptor` 的完整示例代码详见：Interceptor=hook函数 · Frida逆向实例和工具函数

```
// https://github.com/noobpk/frida-ios-hook/blob/master/frida-ios-hook/frida-scripts/hook-specific-method-of-class.js
function hook_specific_method_of_class(className, funcName)
{
  // console.log("className=" + className + ", funcName=" + funcName)
  var curClass = ObjC.classes[className];
  if (typeof(curClass) !== 'undefined') {
    var curMethod = curClass[funcName];
    if (typeof(curMethod) !== 'undefined') {
      Interceptor.attach(curMethod.implementation, {
        onEnter: function(args) {
          console.log("argSelf=" + args[0] + ", argSel=" + args[1] + ", realArg1=" + args[2]);
        }
      });
    }else{
      console.log("Can't find method", funcName);
    }
  }else{
    console.log("Can't find class: ", className);
  }
}

//Your class name  and function name here
hook_specific_method_of_class("NSXPCConnection", "- setExportedObject:")
```

XinaA15中的frida

- XinaA15
  - 截至 `XinaA15 v1.1.8` + `Sileo Nightly v2.4` + `Frida v16.0.11`：在rootless越狱的 `XinaA15` 中，无法通过 `Sileo Nightly` 正常安装和使用 `Frida`
    - 只要一使用frida工具（比如 `frida-ps -U` 等）就会导致iPhone重启
    - 且安装和卸载都会出现一些异常报错
      - 安装Frida
        - `/var/jb/var/ib/Library/LaunchDaemons/re.frida.server.plist service is disabled`
        - 且此处 `Sileo Nightly v2.4` 中看到的最新版 `Frida v16.0.13`，竟然还会出现无法安装：404错误

- 卸载Frida
  - `/var/jb/var/jb/Library/LaunchDaemons/re.frida.server.plist: Could not find specified service`

Failed to attach: need Gadget to attach on jailed iOS

- 问题：Mac中用frida启动iOS版抖音

```
frida -U -f com.ss.iphone.ugc.Aweme
```

  - 报错

```
Failed to attach: need Gadget to attach on jailed iOS, its default location is:
 /Users/crifan/.cache/frida/gadget-ios.dylib
```

- 原因：缺少对应的Frida的 `gadget` 库文件
- 解决办法
  - 概述：下载对应版本的 `gadget` 库文件，放到对应位置（此处提示的 `/Users/crifan/.cache/frida/gadget-ios.dylib` ）即可
  - 详见：Mac · 逆向调试利器：Frida中的 `安装Frida的gadget`

导致iPhone重启

之前遇到过：

XinaA15的rootless越狱的iPhone11中，用最新版 `v16.0.10` 的 `Frida` ，但是却会导致iPhone重启（从而丢失XinaA15的越狱，需要再去恢复越狱）

但是：没有解决方案。

目前的结论是：

（XinaA15等）rootless越狱后， `Frida` 的使用，基本上是个大问题，有时候（某个旧版本）能用，有时候（此处新版 `v16.0.10` ）却又无法正常使用。

Waiting for USB device to appear

- 问题：Mac中运行frida去调试app

```
frida -U -f com.apple.store.Jolly
```

  - 但是报错

```
Waiting for USB device to appear...
```

- 原因：（连接iPhone到Mac的）USB数据线没插好
- 解决办法：重新拔插USB数据线，确保USB连接正常
  - 注：
    - 如何确认iPhone是否已插好
      - 方式1：通过**爱思助手**可以确认
        - 已插好：能看到iPhone详情
        - 没插好：看不到iPhone设备
      - 方式2：用frida的工具：frida-ls-devices

Error unable to intercept function at please file a bug

- 现象：Frida的hook的js代码中，函数地址写的是：

```
var akdSymbol2575_functionAddress = 0x1000a0460;
```

  - 导致报错：

```
moduleName= akd moduleBaseAddress= 0x102b40000
functionRealAddress)= 0x202be0460
Error: unable to intercept function at 0x202be0460, please file a bug
    at value (frida/runtime/core.js:367)
```

- 原因：函数地址写错了 -》 找不到函数地址 -》 所以报错
- 解决办法：确保函数地址是正确的
- 具体做法：代码改为：

```
var akdSymbol2575_functionAddress = 0xa0460;
```

  - 说明
    - `0x1000a0460` 是 加了 `VM` 虚拟地址后的 akd二进制内函数的地址
      - `0xa0460`：是不带VM的，真正的函数内的偏移地址

TypeError: cannot read property 'implementation' of undefined

- 问题：

写frida的hook代码调试ObjC的函数

```
function hook_specific_method_of_class(className, funcName)
{
    var hook = ObjC.classes[className][funcName];
    Interceptor.attach(hook.implementation, {
      onEnter: function(args) {
        console.log("[*] Detected call to: " + className + " -> " + funcName);
      }
    });
}

//Your class name  and function name here
hook_specific_method_of_class("AAUISignInViewController", "_nextButtonSelected:")
```

始终报错：

```
TypeError: cannot read property 'implementation' of undefined
```

- 原因：此处iOS的ObjC的函数名 `_nextButtonSelected:` 写法有误
- 解决办法：改为正确的写法： `- _nextButtonSelected:`
  - 注：此处的ObjC的类的函数名的写法：
    - 加上类型是 `class` 的还是 `instance` ，所对应的 `加号` = `+` 还是 `减号` = `-` ，以及中间带上 `空格` = ` `
  - 相关代码改动：

    ```
    hook_specific_method_of_class("AAUISignInViewController", "- _nextButtonSelecte
    d:")
    ```

  - 完整代码详见
    - 单个类的单个函数 · Frida逆向实例和工具函数

Android

Frida中的Java和js变量类型的映射关系

此处Java中的变量类型，转换成Frida中的类型的写法是：

Frida（的js）中的变量类型，其底层来自于**Dex**

所以其实就是：

- Frida的类型 == Dex中的类型 的语法

即：

- Java变量类型 -》 Frida=Dex变量类型
  - 非数组列表
    - 简单类型
      - 完整写法
        - `int` `int`
        - `byte` `byte`
        - `short` `short`
        - `long` `long`
        - `float` `float`
        - `double` `double`
        - `char` `char`
      - -》 缩写 ShortyDescriptor
        - 缩写 说明
        - `V` void；仅对返回类型有效
        - `Z` boolean
        - `B` byte
        - `S` short
        - `C` char
        - `I` int
        - `J` long
        - `F` float
        - `D` double
    - 特殊的对象Object == 带父类，带路径的Java的类：前缀L + 点. 变成 斜杠/ + 后缀分号;
      - 语法：`Lfully/qualified/Name;`
      - 说明：类 `fully.qualified.Name`
      - 举例
        - `Ljava/lang/String;`
        - `Ljava/lang/Object;`
        - `Ljava/lang/System;`
        - `Ljava/io/PrintStream;`
        - `Ljava/net/InetAddress;`
        - `Ljava/time/LocalDateTime;`
        - `Ljava/lang/ClassLoader;`
  - 列表=数组：前面再加个：左中括号 `[`

        - 语法
            - `[descriptor`
        - 说明
            - descriptor 的数组，可递归地用于“数组的数组”，但维数不能超过 255
        - 举例
            - 普通类型：字母缩写
                - `int[]` -> `[I`
                - `byte[]` -> `[B`
                - `short[]` -> `[S`
                - `long[]` -> `[J`
                - `float[]` -> `[F`
                - `double[]` -> `[D`
                - `char[]` -> `[C`
            - 特殊类型：用（Frida=Dex中的）完整类名写法
                - `[Ljava/lang/String;`
                - `[Ljava/lang/Object;`
                - `[Ldalvik/system/DexPathList$Element;`

含义 = 使用场景举例：

（1）Java相关：函数的签名=定义

举例1：

runtime/native/dalvik_system_DexFile.cc - platform/art - Git at Google (googlesource.com)

```
  NATIVE_METHOD(DexFile, createCookieWithArray, "([BII)Ljava/lang/Object;"),
```

-》

- `([BII)Ljava/lang/Object;`
    - `([BII)`
        - 括号内是函数参数
            - `[BII`
                - `[B` = `byte[]` = 字节码数组列表
                - `I` = `int`
                - `I` = `int`
    - `Ljava/lang/Object;`
        - 类型：`Object`

-》所以函数的完整定义 类型 就是这样的：

```
Object createCookieWithArray(byte[], int, int)
```

举例2：

```
  NATIVE_METHOD(DexFile, isProfileGuidedCompilerFilter, "(Ljava/lang/String;)Z"),
```

-》

- `(Ljava/lang/String;)Z`
  - `(Ljava/lang/String;)`
    - `Ljava/lang/String;`
  - `Z`

-》

```
void isProfileGuidedCompilerFilter(String)
```

（2）Frida中

Frida中，定义Object数组的写法：

```
    var JavaObjArr = Java.use("[Ljava.lang.Object;")
    console.log("JavaObjArr=" + JavaObjArr)
```

输出：

```
JavaObjArr=class: [Ljava.lang.Object;
```

把变量转换为Object的数组：

```
    var objArr = Java.cast(obj, JavaObjArr)
    console.log("objArr=" + objArr)
```

输出：

```
objArr=[Ljava.lang.Object;@fc9e9af
```

Error java.lang.ClassNotFoundException Didn't find class on path DexPathList zip file

情况1：类名错了

- 现象

hook报错：

```
Error: java.lang.ClassNotFoundException: Didn't find class zzz.xxx on path: DexPathList[
[zip file
```

即：找不到类名

- 原因：类名写错了
- 根本原因：
  - 之前Jadx反编译出的源码中的文件名 `sources/zzz/xxx.java` 中的 `xxx` 不是真正的类名
  - 而注释中 `renamed from: yyy` 才是真正的类名
- 解决办法：把名字改为真正的Java的类名： `yyy`

情况2：类不在当前hook的二进制中

- 背景

安卓逆向期间，去根据之前Logcat日志：

```
2023-08-31 10:26:23.911  6481-6481  AppButtonsPrefCtl       com.android.settings
          D  Stopping package com.wallpaper.hd.funny
```

找到了安卓内部的类： `com.android.server.pm.Settings` 中的函数： `createNewSetting` ，输出了上述的log

- 现象

去Frida中用js代码：

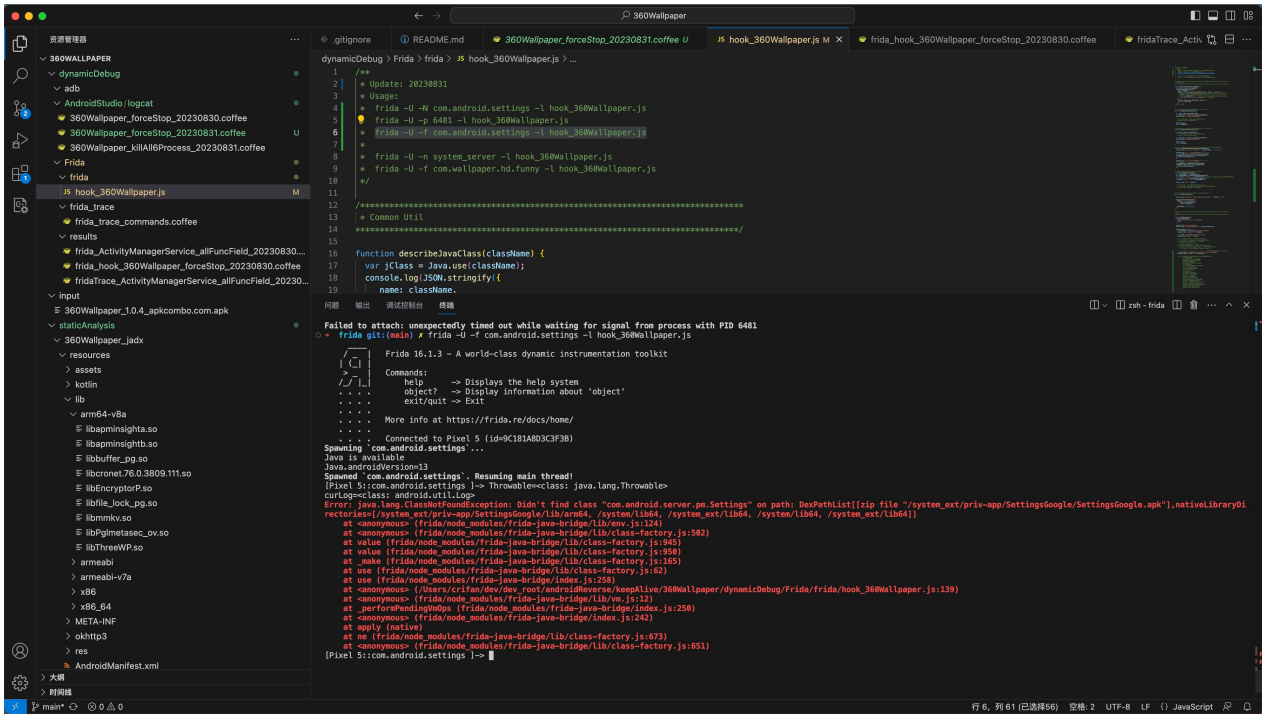
```
    var SettingsCls = Java.use("com.android.server.pm.Settings")
```

去hook安卓内部的类： `com.android.server.pm.Settings`

```
frida -U -f com.android.settings -l hook_360Wallpaper.js
```

报错：

```
Error: java.lang.ClassNotFoundException: Didn't find class "com.android.server.pm.Settings" on path: DexPathList[[zip file "/system_ext/priv-app/SettingsGoogle/SettingsGoogle.apk"],nativeLibraryDirectories [/system_ext/priv-app/SettingsGoogle/lib/arm64, /system/lib64, /system_ext/lib64, /system/lib64, /system_ext/lib64]]
    at anonymous (frida/node_modules/frida-java-bridge/lib/env.js:124)
    at anonymous (frida/node_modules/frida-java-bridge/lib/class-factory.js:502)
    at value (frida/node_modules/frida-java-bridge/lib/class-factory.js:945)
    at value (frida/node_modules/frida-java-bridge/lib/class-factory.js:950)
    at _make (frida/node_modules/frida-java-bridge/lib/class-factory.js:165)
    at use (frida/node_modules/frida-java-bridge/lib/class-factory.js:62)
    at use (frida/node_modules/frida-java-bridge/index.js:258)
    at anonymous (/Users/crifan/dev/dev_root/androidReverse/keepAlive/360Wallpaper/dynamicDebug/Frida/frida/hook_360Wallpaper.js:139)
    at anonymous (frida/node_modules/frida-java-bridge/lib/vm.js:12)
    at _performPendingVmOps (frida/node_modules/frida-java-bridge/index.js:250)
    at anonymous (frida/node_modules/frida-java-bridge/index.js:242)
    at apply (native)
    at ne (frida/node_modules/frida-java-bridge/lib/class-factory.js:673)
    at anonymous (frida/node_modules/frida-java-bridge/lib/class-factory.js:651)
```



- 原因
  - 此处安卓系统的app：**设置**，包名：`com.android.settings`
  - 好像没包含，此处输出的Stopping package的日志的代码：
  - 类 `com.android.server.pm.Settings` 的 函数 `createNewSetting`
  - 因为是另外的二进制程序 `system_server` 才包含此函数
- 解决办法

经过实测，换 `system_server` 去hook，即可找到。

- 具体步骤

```
frida -U -n system_server -l hook_360Wallpaper.js
```

即可找到该类，正常输出：

```
SettingsCls=class: com.android.server.pm.Settings
```

Error cannot call instance method without an instance

现象

对于Java的类 `java.util.HashMap` 的 `put` 函数：

```
public          Object          HashMap.put(          Object,          Object)
```

去hook：

```
  var func_HashMap_put = cls_HashMap.put('java.lang.Object', 'java.lang.Object')
```

但报错：

```
Error: put: cannot call instance method without an instance
```

原因

此处语法错误 == 笔误，写成：重载overload函数的写法（加上函数参数类型）

`cls_HashMap.put('java.lang.Object', 'java.lang.Object')`

但该写法其实是函数调用的写法

而如果去调用函数的话，的确需要：Instance实例

但是此处是hook，不是函数调用

此处只是需要去hook该函数，所以只需要

解决办法

只写出函数名（而无需额外的，类似overload的参数类型）

具体写法

```
var func_HashMap_put = cls_HashMap.put
```

附录

下面列出相关参考资料。

参考资料

-
- 【已解决】frida-trace去hook导致app崩溃最后输出Process terminated
- 【未解决】Mac中frida-trace报错：Failed to spawn unable to find process with name
- 【已解决】frida-ios-dump砸壳ipa报错：need Gadget to attach on jailed iOS
- 【已解决】用frida去hook报错：Failed to spawn unable to find process with name
- 【已解决】Frida启动js脚本报错：Error could not parse line 1 expecting ,
- 【规避解决】frida中console.log打印时参数格式化无效
- 【未解决】Frida中js的console.log日志打印格式化参数
- 【已解决】frida的脚本中console.log打印args报错：RangeError invalid array index
- 【已解决】给Frida的Stalker中输出log日志到文件
- 【已解决】Frida的Stalker中优化hook到指令时的log日志输出
- 【已解决】js中console.log如何打印对象[object Object]
- 【已解决】Frida启动js脚本报错：Error could not parse line 1 expecting ,
- 【已解决】Frida的Stalker中调试报错：Fatal Python error _enter_buffered_busy: could not acquire lock
- 【未解决】frida报错：Failed to spawn: the connection is closed
- 【已解决】frida启动抖音app报错：Failed to attach need Gadget to attach on jailed iOS
- 【已解决】frida运行报错：Waiting for USB device to appear
- 【未解决】Mac中用Frida但报错Failed to enumerate processes the connection is closed且导致iPhone重启
- 【未解决】frida-server运行报错：Failed to spawn this feature requires an iOS Developer Disk Image to be mounted
- 【已解决】frida去hook函数报错：TypeError cannot read property implementation of undefined
- 【未解决】frida中hook函数打印参数值时最后app崩溃frida输出Process terminated
- 【未解决】frida去hook监控iOS的ObjC函数时经常会崩溃Process terminated
- 【未解决】研究frida崩溃Process terminated：通过Preferences的崩溃日志找原因
- 【记录】Frida调试导致iPhone8重启而丢失palera1n越狱
- 【未解决】Mac中Frida报错且导致iPhone重启：重新修复XinaA15越狱环境
- 【未解决】frida导致iPhone重启：从崩溃日志ips文件分析去找可能原因
- 【未解决】Mac中用Frida但报错Failed to enumerate processes the connection is closed且导致iPhone重启
- 【未解决】Frida调试Apple Store报错：Failed to attach missing gProcessInfo
- 【已解决】frida-ios-dump给iOS 15.1的iPhone中app砸壳报错：missing gProcessInfo
- 【已解决】研究frida中是否存在导入外部变量gProcessInfo
- 【未解决】研究Frida中Missing gProcessInfo出错的逻辑和原因
- 【已解决】研究二进制/usr/lib/dyld中是否包含或导出变量_gProcessInfo
- 【记录】dyld源码中的gProcessInfo
- 【已解决】Frida源码中找到了：gProcessInfo
- 【未解决】找Frida中iOS的arm6e4：从Frida源码和build中找
- 【未解决】找Frida中iOS的arm6e4：从Frida的github中找
- 【未解决】找Frida中iOS的arm6e4：自己给make加echo打印日志调试
- 【未解决】找Frida中iOS的arm6e4：从github的ci的workflow中找arm64e

- 【未解决】找Frida中iOS的arm6e4：从编译日志中的Downloading ios-arm64入手
- 【未解决】找Frida中iOS的arm6e4：make时如何传入arm64e的arch参数
- 【未解决】找Frida中iOS的arm6e4：从make编译时的log日志入手
- 【基本解决】iOS 15.1的iPhone11中frida-server所用架构是arm64e还是arm64
- 【未解决】自己编译出的arm64的frida-server能否在iPhone11正常运行
- 【未解决】用frida源码自己编译出frida的iOS的包含frida-server的deb安装包
- 【未解决】自己编译出包含arm64和arm64e的FAT格式的frida-server二进制
- 【未解决】自己修改编译frida-core源码以尝试解决Frida的Missing gProcessInfo问题
- 【未解决】Frida中如何编译出iOS的arm64e的frida-server二进制
- 【未解决】自己编译Frida的frida-core代码生成可用二进制frida-server
- 【已解决】iOS逆向：如何写Frida的Stalker代码去监控函数___lldb_unnamed_symbol2575$$akd的指令运行
- 【未解决】iOS逆向：如何反代码混淆反混淆去混淆
- 【未解决】Mac中Frida启动抖音app进程并调试和hook函数
- 【未解决】用Frida的frida-trace去hook函数iOS版抖音
- 【未解决】frida调试抖音app去hook函数：_dyld_get_image_name
- 【未解决】frida去hook函数_dyld_get_image_name时打印参数为字符串
- 【未解决】用Frida动态调试iOS版抖音app
- 【未解决】Mac中用Frida调试iOS版抖音
- 【已解决】用frida启动hook调试iOS抖音app
- 【未解决】尝试Frida的stalker能否修复抖音AwemeCore中函数名常量字符串
- 【记录】iOS逆向Apple账号：用frida和frida-trace去hook打印更多账号相关函数调用
- 【已解决】Frida的hook类的初始化构造函数中返回值为空undefined
- 【已解决】Frida去hook调试java的map报错：Error cannot call instance method without an instance
- 【记录】Frida调试参数ak_sh逻辑：com.bytedance.retrofit2.client.Request的构造函数
- 【已解决】安卓保活逆向360Wallpaper：字符串解密函数e.i相关代码
- 【已解决】安卓保活逆向360Wallpaper：jadx反编译代码中e.i的通用字符串加密解密函数
- 【已解决】安卓保活逆向360Wallpaper：去使用m1.e.i去解密已混淆加密字符串得到原始字符串
- 【已解决】安卓保活逆向360Wallpaper：用JEB反编译
- 【已解决】安卓保活逆向360Wallpaper：用JEB反编译得到java代码
- 【记录】安卓apk反编译java代码效果对比：JEB对比jadx
-