Frida-学习使用

环境搭建

Mac安装frida

```
brew install python3

pip3 install frida-tools  //安装frida
```

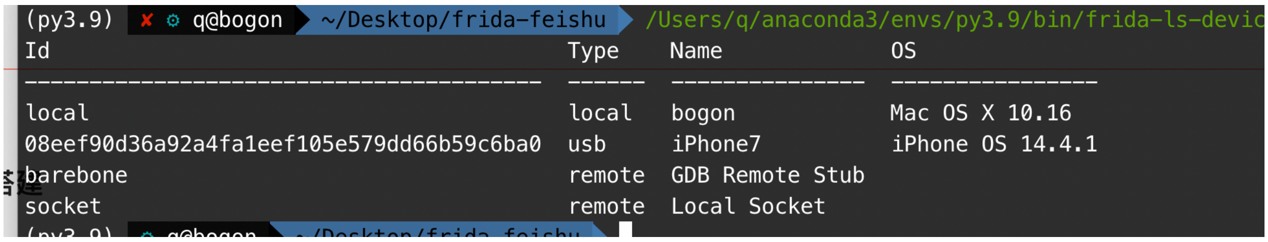
iphone-手机安装frida-server

```
在cydia中添加frida源 https://build.frida.re, 点击 添加源 进行添加,如下所示
cydia添加frida源后, 搜索frida,根据iOS设备版本安装对应的frida服务端---注意版本对应
```

Frida-常用命令

frida-ls-devices

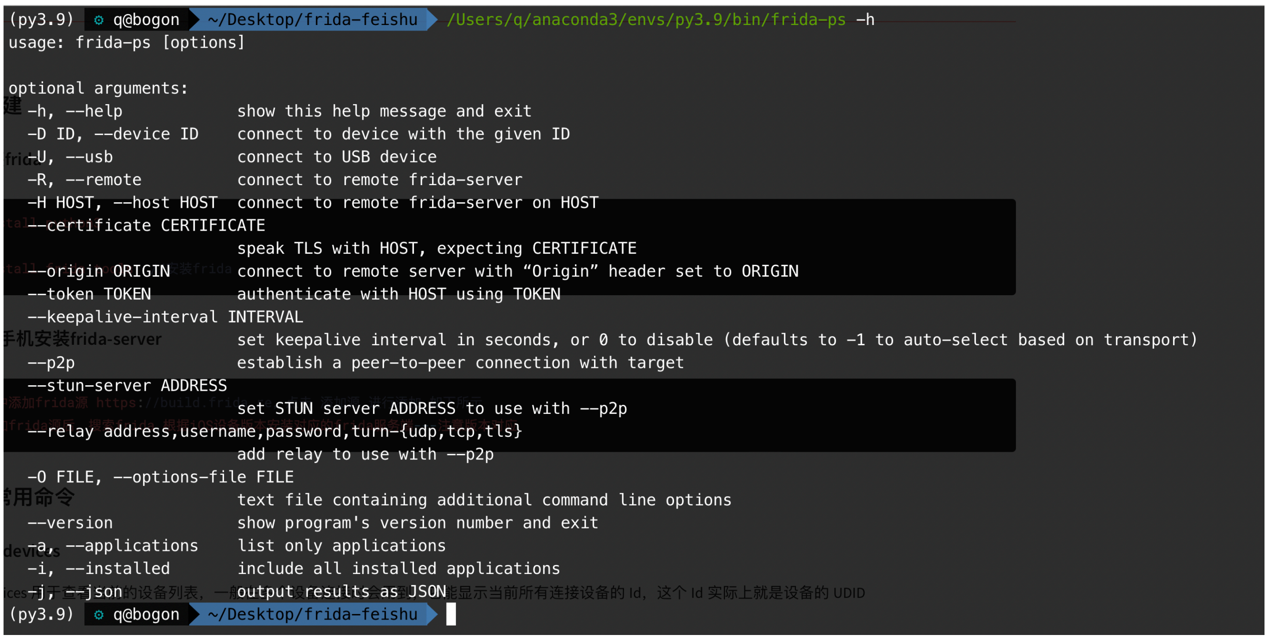
frida-ls-devices 用于查看当前的设备列表，一般在多个设备连接时会用到，它能显示当前所有连接设备的 Id，这个 Id 实际上就是设备的 UDID



```
(py3.9)  ✘ ⚙ q@bogon  ~/Desktop/frida-feishu  /Users/q/anaconda3/envs/py3.9/bin/frida-ls-devic
Id                                        Type    Name              OS
----------------------------------------  ------  ----------------  ----------------
local                                     local   bogon             Mac OS X 10.16
08eef90d36a92a4fa1eef105e579dd66b59c6ba0  usb     iPhone7           iPhone OS 14.4.1
barebone                                  remote  GDB Remote Stub
socket                                    remote  Local Socket
(py3.9)  ⚙ q@bogon  ~/Desktop/frida-feishu
```

Frida-ps

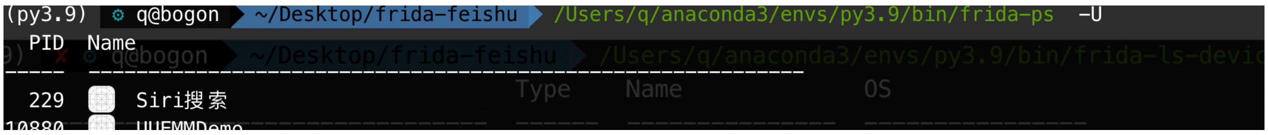
frida-ps 用于查看设备上当前所有运行的进程



```
(py3.9)  q@bogon  ~/Desktop/frida-feishu  /Users/q/anaconda3/envs/py3.9/bin/frida-ps -h
usage: frida-ps [options]

optional arguments:
  -h, --help            show this help message and exit
  -D ID, --device ID    connect to device with the given ID
  -U, --usb             connect to USB device
  -R, --remote          connect to remote frida-server
  -H HOST, --host HOST  connect to remote frida-server on HOST
  --certificate CERTIFICATE
                        speak TLS with HOST, expecting CERTIFICATE
  --origin ORIGIN       connect to remote server with “Origin” header set to ORIGIN
  --token TOKEN         authenticate with HOST using TOKEN
  --keepalive-interval INTERVAL
                        set keepalive interval in seconds, or 0 to disable (defaults to -1 to auto-select based on transport)
  --p2p                 establish a peer-to-peer connection with target
  --stun-server ADDRESS
                        set STUN server ADDRESS to use with --p2p
  --relay address,username,password,turn-{udp,tcp,tls}
                        add relay to use with --p2p
  -O FILE, --options-file FILE
                        text file containing additional command line options
  --version             show program's version number and exit
  -a, --applications    list only applications
  -i, --installed       include all installed applications
  -j, --json            output results as JSON
(py3.9)  q@bogon  ~/Desktop/frida-feishu 
```

查看手机-正在运行的进程：

-U 连接的USB手机



```
(py3.9)  q@bogon  ~/Desktop/frida-feishu  /Users/q/anaconda3/envs/py3.9/bin/frida-ps  -U
  PID  Name
-----  ------------------------------------------------------------
  229     Siri搜索
10880     UUEMMDemo
```

查看正在运行的应用

```
$ frida-ps -U -a
 PID  Name         Identifier
----  -----------  -------------------------
1161  App Store    com.apple.AppStore
1087  Cydia        com.saurik.Cydia
1167  Safari       com.apple.mobilesafari
......
```

查看所有安装的应用

```
frida-ps -U -a -i
PID  Name             Identifier
----  ---------------  -------------------------------
1161  App Store        com.apple.AppStore
1087  Cydia            com.saurik.Cydia
1167  Safari           com.apple.mobilesafari
1163  电话              com.apple.mobilephone
   -  QQ               com.tencent.mqq
   -  邮件              com.apple.mobilemail
   -  音乐              com.apple.Music
......
```

指定查看某个设备的进程

```
frida-ps -D cca1b9055ac2684999cd81e525ac03fe6028b9f9 -a -i
```

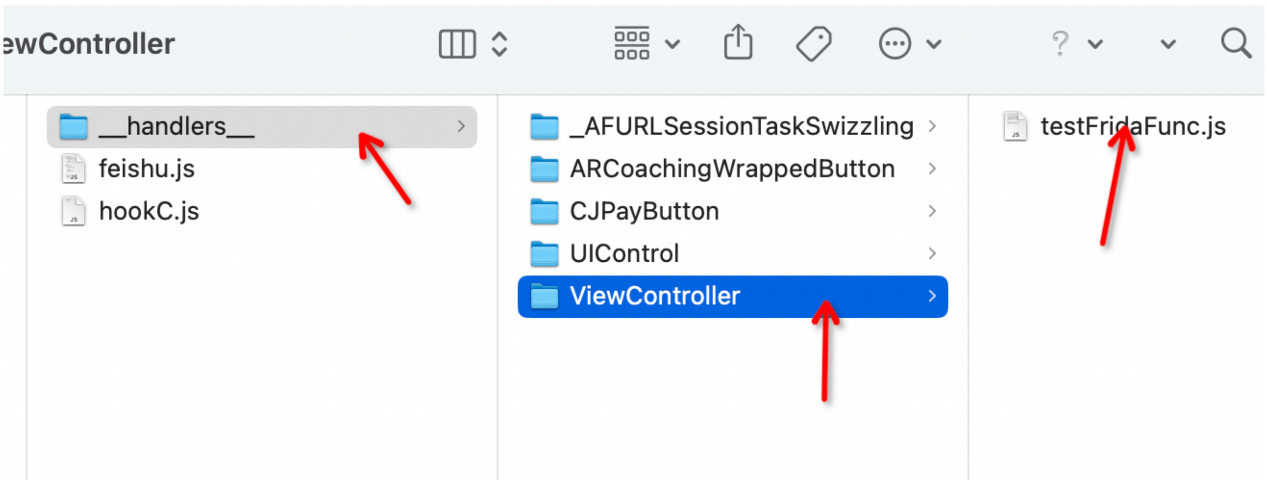
```
# 将Frida通过USB连接到iPad，并列出正在运行的进程
frida-ps -U

# 运行中的应用程序列表
frida-ps -Ua

# 已安装的应用程序列表
frida-ps -Uai

# 把Frida连接到特定的设备上
frida-ps -D 0216027d1d6d3a03
```

结束某个进程

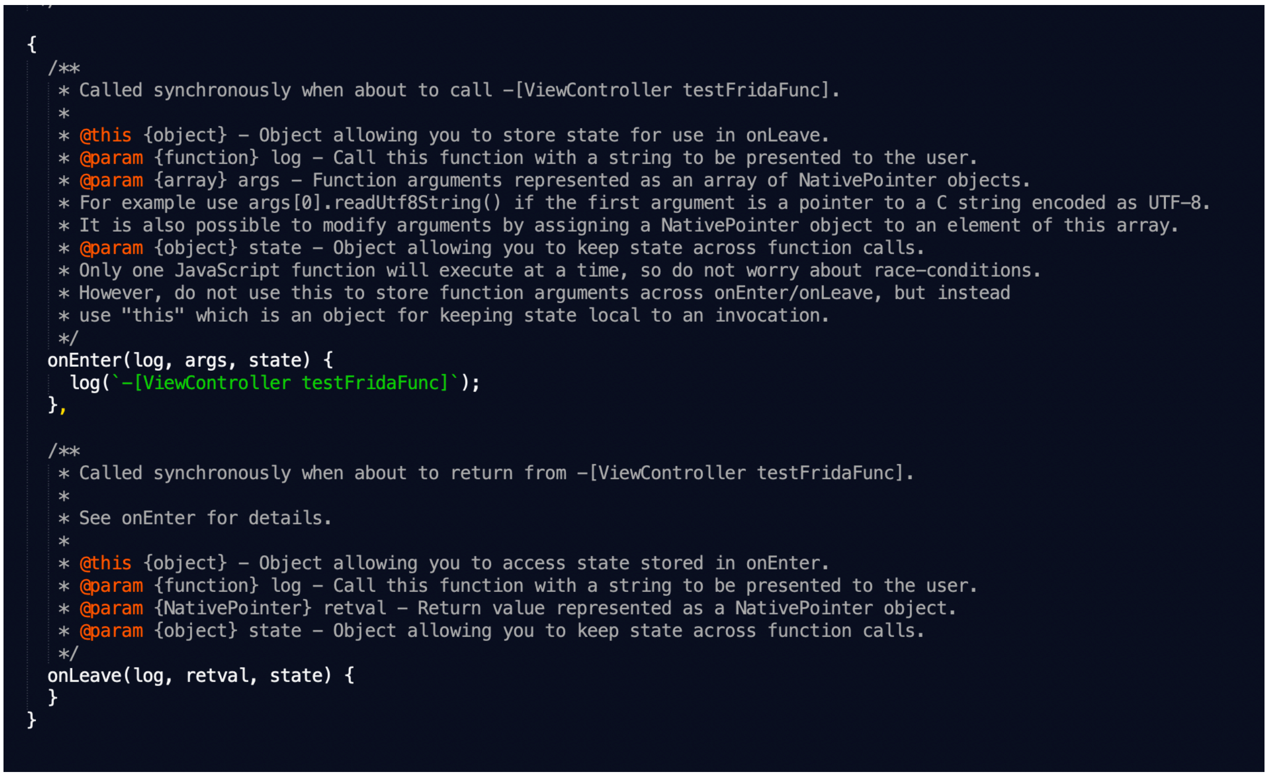
```
frida-kill -U 10880
```

frida-trace

> 执行frida-trace 会在当前目录下生成一个-hook方法的JS文件：
>
> frida-trace -U -m "-[ViewController testFridaFunc]" -f "com.uusafe.sdk.TestFrida"



Frida-trace-js说明



```
{
  /**
   * Called synchronously when about to call -[ViewController testFridaFunc].
   *
   * @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 NativePointer 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, but instead
   * use "this" which is an object for keeping state local to an invocation.
   */
  onEnter(log, args, state) {
    log(`-[ViewController testFridaFunc]`);
  },

  /**
   * Called synchronously when about to return from -[ViewController testFridaFunc].
   *
   * 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 object.
   * @param {object} state - Object allowing you to keep state across function calls.
   */
  onLeave(log, retval, state) {
  }
}
```

```
onEnter:hook方法的入口--可以对传参进行处理
args是要hook方法的阐述列表（arg0:对象自身，arg1:方法名，arg2:参数-如果有）

onLeave:方法执行完毕返回时候--可以对返回值进行处理
```

指令参数列表

```
frida-trace -h
positional arguments:
  args                  extra arguments and/or target

optional arguments:
  -h, --help            show this help message and exit
  -D ID, --device ID    connect to device with the given ID
  -U, --usb             connect to USB device
  -R, --remote          connect to remote frida-server
  -H HOST, --host HOST  connect to remote frida-server on HOST
  --certificate CERTIFICATE
                        speak TLS with HOST, expecting CERTIFICATE
  --origin ORIGIN       connect to remote server with “Origin” header set to
ORIGIN
  --token TOKEN         authenticate with HOST using TOKEN
  --keepalive-interval INTERVAL
                        set keepalive interval in seconds, or 0 to disable
(defaults to -1 to auto-select based on transport)
  --p2p                 establish a peer-to-peer connection with target
  --stun-server ADDRESS
                        set STUN server ADDRESS to use with --p2p
  --relay address,username,password,turn-{udp,tcp,tls}
                        add relay to use with --p2p
  -f TARGET, --file TARGET
                        spawn FILE
  -F, --attach-frontmost
                        attach to frontmost application
  -n NAME, --attach-name NAME
                        attach to NAME
  -N IDENTIFIER, --attach-identifier IDENTIFIER
                        attach to IDENTIFIER
  -p PID, --attach-pid PID
                        attach to PID
  -W PATTERN, --await PATTERN
                        await spawn matching PATTERN
  --stdio {inherit,pipe}
                        stdio behavior when spawning (defaults to “inherit”)
  --aux option          set aux option when spawning, such as “uid=(int)42”
(supported types are: string, bool, int)
  --realm {native,emulated}
                        realm to attach in
  --runtime {qjs,v8}    script runtime to use
  --debug               enable the Node.js compatible script debugger
  --squelch-crash       if enabled, will not dump crash report to console
  -O FILE, --options-file FILE
                        text file containing additional command line options
  --version             show program's version number and exit
  -I MODULE, --include-module MODULE
                        include MODULE
  -X MODULE, --exclude-module MODULE
                        exclude MODULE
  -i FUNCTION, --include FUNCTION
                        include [MODULE!]FUNCTION
  -x FUNCTION, --exclude FUNCTION
                        exclude [MODULE!]FUNCTION
  -a MODULE!OFFSET, --add MODULE!OFFSET
                        add MODULE!OFFSET
  -T INCLUDE_IMPORTS, --include-imports INCLUDE_IMPORTS
                        include program's imports
  -t MODULE, --include-module-imports MODULE
                        include MODULE imports
  -m OBJC_METHOD, --include-objc-method OBJC_METHOD
                        include OBJC_METHOD
  -M OBJC_METHOD, --exclude-objc-method OBJC_METHOD
                        exclude OBJC_METHOD
  -j JAVA_METHOD, --include-java-method JAVA_METHOD
                        include JAVA_METHOD
  -J JAVA_METHOD, --exclude-java-method JAVA_METHOD
                        exclude JAVA_METHOD
  -s DEBUG_SYMBOL, --include-debug-symbol DEBUG_SYMBOL
                        include DEBUG_SYMBOL
  -q, --quiet           do not format output messages
  -d, --decorate        add module name to generated onEnter log statement
  -S PATH, --init-session PATH
                        path to JavaScript file used to initialize the session
  -P PARAMETERS_JSON, --parameters PARAMETERS_JSON
                        parameters as JSON, exposed as a global named
'parameters'
  -o OUTPUT, --output OUTPUT
                        dump messages to file
```

重要指令说明：

```
-i 跟踪某个函数，-x 排除某个函数。
-m 跟踪某个 Objective-C 方法，-M 排除某个 Objective-C 方法。
-a 跟踪某一个地址，需要指名模块的名称。

// 设备相关
-D  连接到指定的设备，多个设备时使用。示例:frida-trace -D
555315d66cac2d5849408f53da9eea514a90547e -F
-U  连接到USB设备，只有一个设备时使用。示例fria-trace -U -F

// 应用程序相关
-f  目标应用包名。spawn模式。示例:frida-trace -U -f com.apple.www
```

```
-F  当前正在运行的程序。attach模式示例。示例:frida-trace -U -F或frida-trae -UF
-n  正在运行的程序的名字。attach模式。示例:frida-trace -U -n QQ
-N  正在运行的程序的包名。attach模式。示例:frida-trace -U -N com.apple.www
-p  正在运行的程序的pid。attach模式。示例:frida-trace -U -p 2302

// 方法相关，以下参数在一条跟踪命令中可重复使用
-I  包含模块。示例:frida-trace -UF -I "libcommonCrypto*"
-X  不包含模块。示例:frida-trace -UF -X "libcommonCrypto*"
-i  包含c函数。示例:frida-trace -UF -i "CC_MD5"
-x  不包名c函数。示例:frida-trace -UF -i "*MD5" -x "CC_MD5"
-a  包含模块+偏移跟踪。示例:frida-trace -UF -a 模块名\!0x7B7D48
-m  包含某个oc方法。示例:frida-trace -UF -m "+[NSURL URLWithString:]"
-M  不包含某个oc方法。示例:frida-trace -UF -M "+[NSURL URLWithString:]"

// 日志相关
-o  日志输出到文件。示例:frida-trace -UF -m "*[* URL*]" -o run.log
```

模糊匹配-命令使用

```
-m "-[NSURL *]" // 匹配NSURL类的所有实例方法
-m "+[NSURL *]" // 匹配NSURL类的所有类方法
-m "*[NSURL *]" // 匹配NSURL类的所有方法
-m "*[*URL *]"  // 匹配以URL结尾类的所有方法
-m "*[URL* *]"  // 匹配以URL开头类的所有方法
-m "*[*URL* *]" // 匹配包含URL的类的所有方法
-m "*[*URL* *login*]" // 匹配包含URL的类的带login的所有方法
-m "*[????? *]" // 匹配类名只有五个字符的类的所有方法
```

指定某个应用-某个方法启动流程-强制启动hook

-f 强制

com.uusafe.sdk.TestFrida：应用bundleID

```
frida-trace -U  -m "-[ViewController testFridaFunc]" -f
"com.uusafe.sdk.TestFrida"
```

连续跟踪hook-多个函数方法

```
frida-trace -U  -m "-[ViewController testFridaFunc]"  -m "方法2️⃣" 应用名
```

Frida-trace-配合py搭配js使用

Frida-跟踪hook-C方法---自定义

```
function frida_hook_open(){
  var func_ptr = Module.findExportByName(null, "open"); // 找到要hook的函数
  Interceptor.attach(func_ptr, { // hook函数
    onEnter: function(args) {
      if (args[0].isNull()) return;
          var path = args[0].readUtf8String();
      console.log("open " + path);
      console.log("=====current thraed:" + Process.getCurrentThreadId());

    },
    onLeave: function(retVal) {
      console.log("leave function");
    }
  });
}

frida_hook_open();
```

Frida-hook-C方法-系统

frida -U -l open.js 应用名

frida -U -l open.js -f "bundleID"

```
function frida_hook_open(){
  var func_ptr = Module.findExportByName(null, "open"); // 找到要hook的函数
  Interceptor.attach(func_ptr, { // hook函数
    onEnter: function(args) {
      if (args[0].isNull()) return;
          var path = args[0].readUtf8String();
      console.log("open " + path);
      console.log("=====current thraed:" + Process.getCurrentThreadId());
    },
    onLeave: function(retVal) {
      console.log("leave function");
    }
  });
}
frida_hook_open();
```

Frida-hook-C方法-自己定义方法Interceptor.replace

执行命令：

```
frida -D 08eef90d36a92a4fa1eef105e579dd66b59c6ba0 -l /Users/q/Desktop/frida-feishu/hookC.js  -f "com.uusafe.sdk.TestFrida"
```

```
   / _  |   Frida 16.1.1 - 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=08eef90d36a92a4fa1eef105e579dd66b59c6ba0)
Spawned `com.uusafe.sdk.TestFrida`. Resuming main thread!
[iPhone::com.uusafe.sdk.TestFrida ]-> 替换后的函数
```

```
//hook-自定义C接口
//此函数在module模块中寻找地址为offset
function get_func_addr(module, offset) {
   var base_addr = Module.findBaseAddress(module);
   // console.log("base_addr: " + base_addr);
   // console.log(hexdump(ptr(base_addr), {
   //          length: 16,
   //          header: true,
   //          ansi: true
   //      }))
   var func_addr = base_addr.add(offset);
   if (Process.arch == 'arm')
      return func_addr.add(1);  //如果是32位地址+1
   else
      return func_addr;
}
```

```
//替换获取所在模块TestFrida模块中，hopper中的地址为0xeba6c处的testFridaFunC函数
var func_addr_replace_eba6c = get_func_addr('TestFrida', 0x5e88);
var add_replace_eba6c = new NativeFunction(func_addr_replace_eba6c, 'void',
[]);
// 进行替换
Interceptor.replace(add_replace_eba6c, new NativeCallback(function() {
console.log('替换后的函数');

}, 'void', []));
```

Frida-hook-C追踪自定义的C方法Interceptor.attach

执行命令：

```
frida -D 08eef90d36a92a4fa1eef105e579dd66b59c6ba0 -l /Users/q/Desktop/frida-feishu/hookC.js  -f
"com.uusafe.sdk.TestFrida"
```

```
  / _ |   Frida 16.1.1 - 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=08eef90d36a92a4fa1eef105e579dd66b59c6ba0)
Spawned `com.uusafe.sdk.TestFrida`. Resuming main thread!
[iPhone::com.uusafe.sdk.TestFrida ]-> onEnter
num1: 0x102f0a110
onLeave
```

```
//此函数在module模块中寻找地址为offset
function get_func_addr(module, offset) {
   var base_addr = Module.findBaseAddress(module);
   // console.log("base_addr: " + base_addr);
   // console.log(hexdump(ptr(base_addr), {
   //          length: 16,
   //          header: true,
   //          ansi: true
   //      }))
   var func_addr = base_addr.add(offset);
   if (Process.arch == 'arm')
      return func_addr.add(1);  //如果是32位地址+1
```

```
    else
        return func_addr;

}
//替换获取所在模块TestFrida模块中，hopper中的地址为0xeba6c处的testFridaFunC函数
var func_addr_replace_eba6c = get_func_addr('TestFrida', 0x5e88);
Interceptor.attach(ptr(func_addr_replace_eba6c), {
    onEnter: function(args) {
        console.log("onEnter");
        var num1 = args[0];
        console.log("num1: " + num1);
    },
    onLeave: function(retval) {
        console.log("onLeave");
    }
});
```

Frida-hookIMP-OC方法

Hook-方法无参数

```
// 普通参数
function hook01(){
    let class_name = 'TestFrida'
    let method = '+ sharedInstance'
    let sharedInstance = ObjC.classes[class_name][method]
    let oldImpl = sharedInstance.implementation
    sharedInstance.implementation =
ObjC.implement(sharedInstance,function(handle,selector,arg1,arg2){
        // 如果有参数的话，就是第三个参数为该方法的第一个参数: args[n-2]

        // 查看值
        console.log("arg1",new ObjC.Object(arg1))

        let result = oldImpl(handle,selector,arg1,arg2) //此处可以判断是否执行原函数
        return result
    })
}
```

Hook-方法block参数

```
// 参数类型为NSStackBlock
```

```
let method = '+ autoLogin:session:callBack:'
    let impl = ObjC.classes.ALBBSessionRpcService[method]
    const oldImpl = impl.implementation;
    impl.implementation = ObjC.implement(impl,
function(handle,selector,arg1,arg2,arg3) {
        try{
            console.log("arg1",new ObjC.Object(arg1))
            console.log("arg2",arg2,JSON.stringify(new ObjC.Object(arg2)))
            console.log("arg3",arg3,new ObjC.Object(arg3)) //此参数为NSStackBlock

            var block = new ObjC.Block(arg3);
            const appCallback = block.implementation;
            block.implementation = function (resp)  {
                console.log(resp) // ALBBResponse
                const result = appCallback(resp);
                return result;
            };

            var ret = oldImpl(handle,selector,arg1,arg2,arg3);

            // console.log(ret)
        }catch(err){
            console.log(err)
        }
        return ret
    });
```

Frida-JS-OC语法

> JS-objc-官网

Frida-常用模块API

Module-处理库相关的操作

findExportByName(moduleName|null, exportName)

```
moduleName: lib名字
exportName: 函数名字
返回exportName的地址
```

findBaseAddress(moduleName)

```
moduleName: lib名字
返回lib的基地址
```

Process模块:

findModuleByAddress(address)

```
address: lib的指针地址
该函数返回一个Module对象
```

Momery模块

readCString(pointer)

```
pointer:指针地址
把pointer还原成字符串
```

Interceptor模块: 监听

attach(target, callbacks)

```
target:指针地址
callbacks: 回调函数
onEnter
onLeave
```

replace(target, replacement) —— 该函数用以改变原函数逻辑的

参考

> Frida-官网
>
> ios逆向之frida安装与使用基础 - 简书 (jianshu.com)