

吾爱破解
www.52pojie.cn

设置消息提醒退出

春节红包

网站

新帖

搜索

帮助

快捷导航

请输入搜索内容

帖子

Q

领取今日签到奖励

» 网站 » 【软件安全】 » 『软件调试区』

« 返回列表5681 页»

[调试逆向] Mac下 XX 及任意程序的内置浏览器调试的实现复制链接

楼主

精华

电梯直达

本帖最后由 楼主 于 2020-2-6 14:14 编辑

° 前言

在 iOS 以及 Android 上，Google 提供了功能强大的基于 Xcode 的调试器功能，方便开发者调试 iOS 的内置浏览器。在某聊天软件盛行的今天，很多网页都必须在该软件的内置浏览器里才能运行，作为开发者的我们，如果想一窥这类网页的实现细节该怎么办呢？在浩瀚的网络里搜了一下，安卓可以通过 Burp Suite 来调试，iOS 系统越狱后，通过 Cydia 也能做到，唯独 Android 系统甚少提及。这篇文章里，我尝试通过逆向的手段，达到对任意 iOS 的内置浏览器实现调试。

° 原理分析

从现有的资料，可以查到，Google 判断是否可以进行调试依赖于守护进程 Xcode，而实际的逻辑判断实现在动态库 Xcode 中，其路径为

“ /usr/lib/system/libsystem.dylib 中的 _system_xcode 函数 ”

而其中的关键函数为签名如下的 _system_xcode 方法：

1. 系统 Xcode 纯文本查看 复制代码

```
1 | -[RWIRelayDelegateMac _allowApplication:bundleIdentifier:];
```

放进 IDA 分析这个方法的实现过程，整个流程，可以注释如下：

分析可得，要使这个方法返回真，有以下几个条件

- 为内部程序，即 $\mathbf{A} \in \mathbf{X}_{int}$ 为真，且 $\mathbf{X}_{int} \vdash \mathbf{X}_{int}$ 为真
- 是代过过滤应用（包名包含 $\mathbf{h} \hat{\mathbf{n}} \mathbf{H} \hat{\mathbf{B}} \mathbf{X} \mathbf{X} \mathbf{b} \mathbf{b} \mathbf{e} \mathbf{n} \mathbf{X}$ ），即 $\mathbf{A} \in \mathbf{H} \hat{\mathbf{U}} \mathbf{2} \mathbf{H} \hat{\mathbf{B}} \mathbf{r} \mathbf{a} \mathbf{q}$ 函数返回 $\mathbf{h} \hat{\mathbf{n}}$
- 应用签名包含 $\mathbf{h} \hat{\mathbf{n}} \mathbf{H} \hat{\mathbf{B}} \mathbf{X} \mathbf{X} \mathbf{b} \mathbf{b} \mathbf{e} \mathbf{n} \mathbf{X}$ 且 \mathbf{T} 权限
- 应用签名包含 $\mathbf{h} \hat{\mathbf{n}} \mathbf{H} \hat{\mathbf{B}} \mathbf{X} \mathbf{X} \mathbf{b} \mathbf{b} \mathbf{e} \mathbf{n} \mathbf{X}$ 且 \mathbf{T} 权限

以上条件实现任一即可。对于普通开发者来说，最可操作的是第7点，只要用1\开发证书重签名应用即可达成。但是重签也是有局限性的，会造成某些重要Xenith Xenith丢失从而导致运行不正常。根据以上的分析，我们来尝试一下如何用技术的手段实现。

在开始之前，需要做的第一件事情是关闭系统的 `gadget` 以及 `mkaaj nxl`。因为我们需要修改系统的实现，就需要获得目标进程的 `mkae` 获得了 `mkae` 也就得到了该进程的控制权，系统的 `gadget` 正是保护这一机制不被滥用，拦截不信任的应用调用 `mkaaj nxl` 来获得 `mkae` 另一个安全机制 `mkaaj nxl` 对调用敏感内核函数的程序进行签名校验，也要关闭。关于关闭的方法网上有大量资料，不再累述。

0 \tilde{U}^6 尝试

由于我们要修改的为一个 `6.4` 方法，首先想到的就是最常用最方便的 `h5py` 技术，我们参考 `6.4` 的版本，直接用以下的 `h5py` 脚本验证一下：

[÷ ! g l h m](#)
[纯文本查看](#)
[复制代码](#)

```
01 Interceptor.attach(ObjC.classes.RWIRelayDelegateMac['_allowApplication:bundleIdentifier:'].implementati
02 onEnter: function(args) {
03     this.bundleId = new ObjC.Object(args[3]);
04 },
05 onLeave: function(retval) {
06     const allow = !retval.equals(NULL)
07     console.log(this.bundleId + (allow ? ' allows' : ' does not allow') + ' WebInspect')
08     if (!allow) {
09         console.log('now patch it');
10         retval.replace(ptr(1));
11     }
12 }
13 });
```

打开g N后，在命令行输入N T X6 进入N 界面提示，输入以上的÷ t g脚本。然后在某聊天软件打开任意文章以拉起T X6 t ãT，这时可以看到N 给出进程终止的提示。

N 不起作用，而我对N 的注入机制也不了解，只能靠自己来实现对T X6 的注入了。对于在í 上可以实现进程注入的技术手段，据我所知有以下三种：

- 线程注入
- 线程劫持
- sEb助手服务

其中线程劫持这种远程注入的难度太大，因为要从进程中取出某个线程，保存当前线程的状态，运行劫持代码，再恢复原来的状态，很容易造成线程出错。我在尝试了另外两种方法后，都以T X6 的Hk而失败告终。打开系统的日志，可以看到如下的输出：

类型	时间	进程	信息
	18:25:34.026593	trustd	could not enable test hierarchy: no UAT pinning preferences set
	18:25:34.028171	trustd	cert[2]: AnchorTrusted =(leaf)[force]> 0
	18:25:34.038202	amfid	MacOS error: -67050
	18:25:34.038281	amfid	MacOS error: -67050
	18:25:34.043467	Safari	Device (CURRENTMACHINE) Application (PID:4976) Removed Debuggable
	18:25:34.043545	Safari	Device (CURRENTMACHINE) Application (PID:4970) Removed Debuggable
	18:25:34.043628	Safari	Target (CURRENTMACHINE) Removed Application (PID:4970)
	18:25:34.043722	Safari	Target (CURRENTMACHINE) Removed Application (PID:4976)
	18:25:34.043796	Safari	Manager Removed Target (CURRENTMACHINE)
	18:25:34.043890	Safari	Removed CURRENTMACHINE target: <RWIMachine: 0x6040000cf5e0>

amfid (Security)

子系统： com.apple.securityd 类别： security_exception [详细信息](#) 2019-11-28 18:25:34.03820

MacOS error: -67050

吾爱破解论坛
www.52pojie.cn

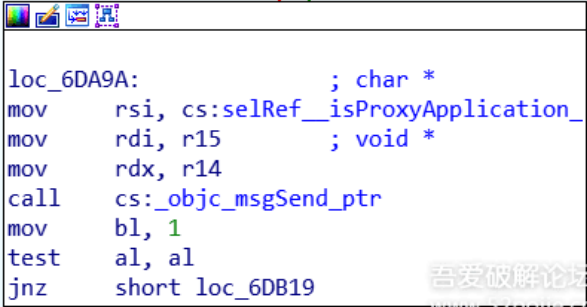
通过命令行输入kXl 可以查看这个出错的详细说明，英文就不写出来了，用人话来说，就是不符合某些要求，但是没指名是什么要求。这个错误是 h N进程发出的，查阅相关资料，有点头绪了，我们来分析一下。

我是首先生成一个动态库，内容很简单，在初始化函数中利用h Xr I kT 冷äÉY ÆEÜk Xñj nÍ ¼ E T2H rã eBóel ÆE rãM 始终返回真。而注入程序首先是通过mkæ N H 获取T X6 的mkæ 并在T X6 的进程空间里通过h ¼ h ¼ E 函数分配代码空间和栈空间。然后将E H加载动态库所在路径的相关汇编代码通过h ¼ h ¼ E 函数写入到刚才开辟的代码空间中。最后通过结构体Ü · üN X I Ümr X ° ü设置线程的栈指针Eg E和栈基指针E^E指向栈空间底部，通过rÜ X I Ü X nÜö qæ j创建并运行线程。一切看起来如此正常，然而即使I Ü和注入程序都用开发者证书签名，依然过不了 h N的拦截。

h N H XÍ "6æ öæ ænj ä I Xñ "e 苹果可移动文件完整性守护进程，对系统的保护简单而高效。在í 系统里，当一个程序被签名的时候，签名工具I Xæ 会对该程序的二进制文件的每个页H n计算一个哈希值，以及所有哈希值一起计算出一个总的哈希值。对于í 系统以及tæ Ó系统，内核通过H nXNö 的机制来分配内存，当我们访问一个内存地址时，如果该地址非法，或者我们对其没有访问权限，或者该地址对应的物理内存还未分配，内核都会生成一个H nXNö 进而执行操作系统的H nXNö ü eI ÆH。在í 上，还有一种情况会产生H nXNö 当某个文件（例如I Ü）被内核以h h H函数映射进进程的虚拟内存中。这个时候 h N会校验新进来的内存页是否有签名，签名是否与当前执行的二进制程序一致，如果没有签名或者签名的哈希值不一致，那么加载过程将中断，对于某些苹果签名的二进制，系统为了安全起见会直接æ 掉该二进制，T X6 就是个例子。故事到了这里，如果要继续发展下去，就需要H nÜ h N了，了解过ä g越狱的朋友会知道，越狱其中关键的一步就是H nÜ h N，可见这条路一定是荆棘丛丛，先暂不考虑。我想到的另一个办法是内存补丁，既然系统不允许我们加载我们的代码，那就直接改内部实现好了，也就是我接下来要说的内存补丁。

° Ü 内存补丁

回顾图 的流程图，已知其中一个可以实现调试的条件是函数ä E H Ü2H rã eB返回°，观察了一下判断过程，惊喜的发现，非常容易扭转状态，见下图：



```
loc_6DA9A:                ; char *
mov     rsi, cs:selRef__isProxyApplication_
mov     rdi, r15            ; void *
mov     rdx, r14
call    cs:_objc_msgSend_ptr
mov     bl, 1
test    al, al
jnz     short loc_6DB19
```

吾爱破解论坛
www.52pojie.cn

首先是给寄存器赋值，然后判断存放 `loc_6DB19` 返回值的寄存器是否不为零，如果不为零则跳转到结束并把 `loc_6DB19` 的值返回。很容易想到把 `loc_6DB19` 改为 `loc_6DB18` 即可。用机器码来表示就是 `b0` 改为 `k^`。

```
text:000000000006DAA1 4C 89 FF      mov     rdi, r15      ; void
text:000000000006DAA4 4C 89 F2      mov     rdx, r14
text:000000000006DAA7 FF 15 0B 16+   call    cs:_objc_msgSend_ptr
text:000000000006DAAD B3 01         mov     bl, 1
text:000000000006DAAF 84 C0         test    al, al
text:000000000006DAB1 75 66         jnz     short loc_6DB19
text:000000000006DAB3 4C 8D 7D 10    lea     r15, [rbp+arg_0]
text:000000000006DAB7 49 8B 47 18    mov     rax, [r15+18h]
```

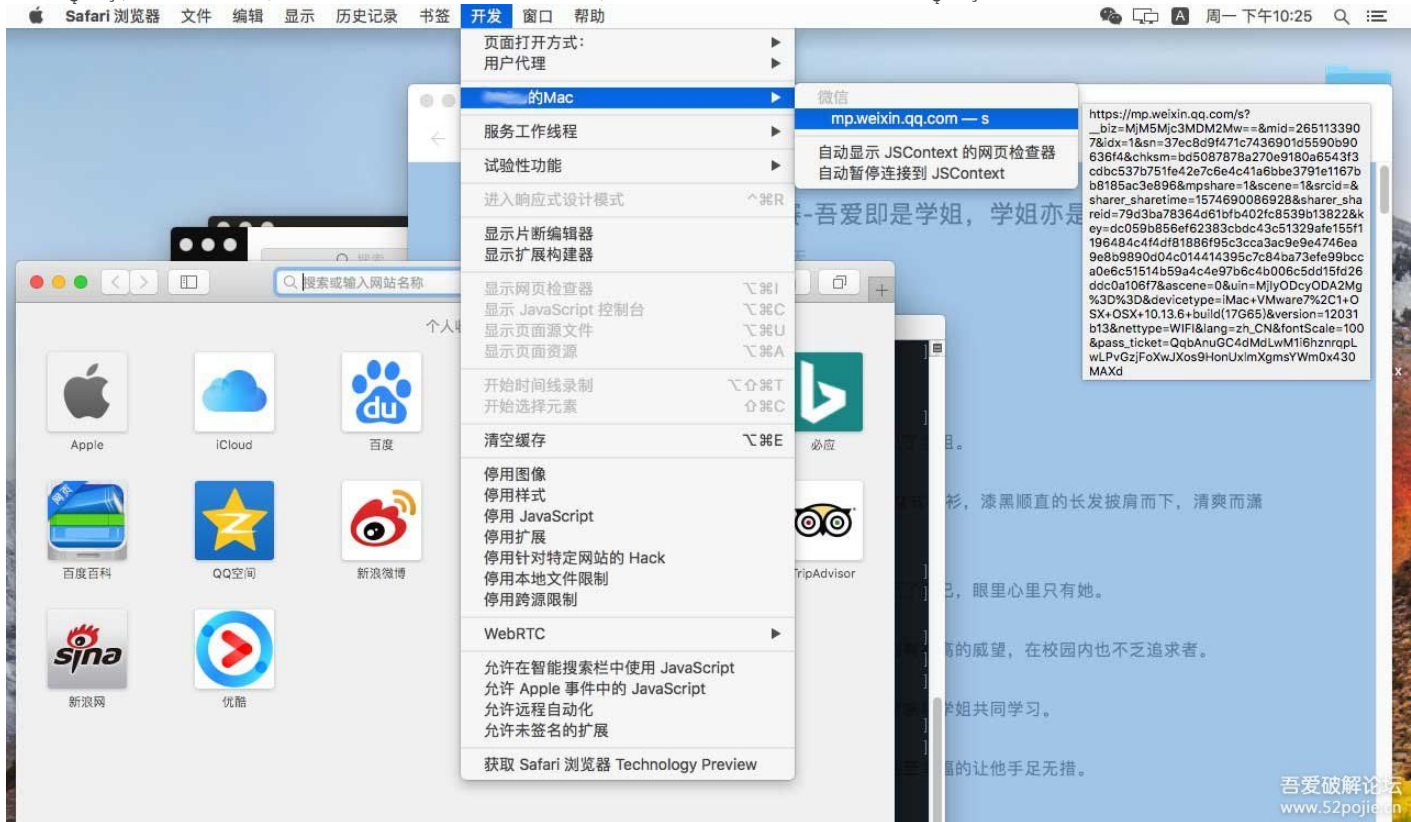
现在面临的最大问题也是整个实现过程最大的挑战，已知这段代码在虚拟内存中的位置是 `000000000006DAB1`，如何确定在进程的虚拟内存中的位置？这段代码是以系统共享动态库的形式加载进 `mach_vm_region` 的内存空间。我首先想到的方法是通过 `mach_vm_region` 函数枚举出所有进程中活跃的内存块（`mach_vm_region`）并搜索 `mach_vm_region` 的文件头以找到动态库的开始位置。然而实际的输出内存区间却没有我想象中那样可以对齐到动态库的起始头位置，对比一下 `mach_vm_region` 的输出如下：

```
0x7fff90000000 2097152
0x7fff90200000 2097152
0x7fff90400000 2097152
0x7fff90600000 2097152
0x7fff90800000 2097152
0x7fff90a00000 2097152
0x7fff90c00000 2097152
0x7fff90e00000 155648
00007fff9008e000-00007fff90090000 [ 8K 8K 0K 0K] rw-/rwx SM=COW /System/Library/PrivateFrameworks/TCC.framework/Versions/A/TCC
00007fff90129000-00007fff90130000 [ 28K 28K 0K 0K] rw-/rwx SM=COW /System/Library/PrivateFrameworks/TextureIO.framework/Versions/A/TextureIO
00007fff90170000-00007fff90171000 [ 4K 4K 0K 0K] rw-/rwx SM=COW /System/Library/PrivateFrameworks/TrustEvaluationAgent.framework/Versions/A/TrustEvaluationAgent
00007fff90173000-00007fff901bd000 [ 296K 296K 4K 0K] rw-/rwx SM=COW /System/Library/PrivateFrameworks/UIFoundation.framework/Versions/A/UIFoundation
00007fff90432000-00007fff9044f000 [ 116K 112K 8K 0K] rw-/rwx SM=COW /System/Library/PrivateFrameworks/WebInspector.framework/Versions/A/WebInspector
00007fff9061b000-00007fff9061c000 [ 4K 4K 0K 0K] rw-/rwx SM=COW ...m/Library/PrivateFrameworks/Login.framework/Versions/A/Frameworks/Loginsupport.
00007fff90694000-00007fff90696000 [ 8K 4K 0K 0K] rw-/rwx SM=COW /usr/lib/closure/libclosure.dylib
00007fff906a0000-00007fff906a1000 [ 4K 4K 4K 0K] rw-/rwx SM=COW /usr/lib/libCRFSuite.dylib
00007fff906a1000-00007fff906a2000 [ 4K 4K 4K 0K] rw-/rwx SM=COW /usr/lib/libChineseTokenizer.dylib
00007fff906a2000-00007fff906a5000 [ 12K 8K 0K 0K] rw-/rwx SM=COW /usr/lib/libCoreStorage.dylib
```

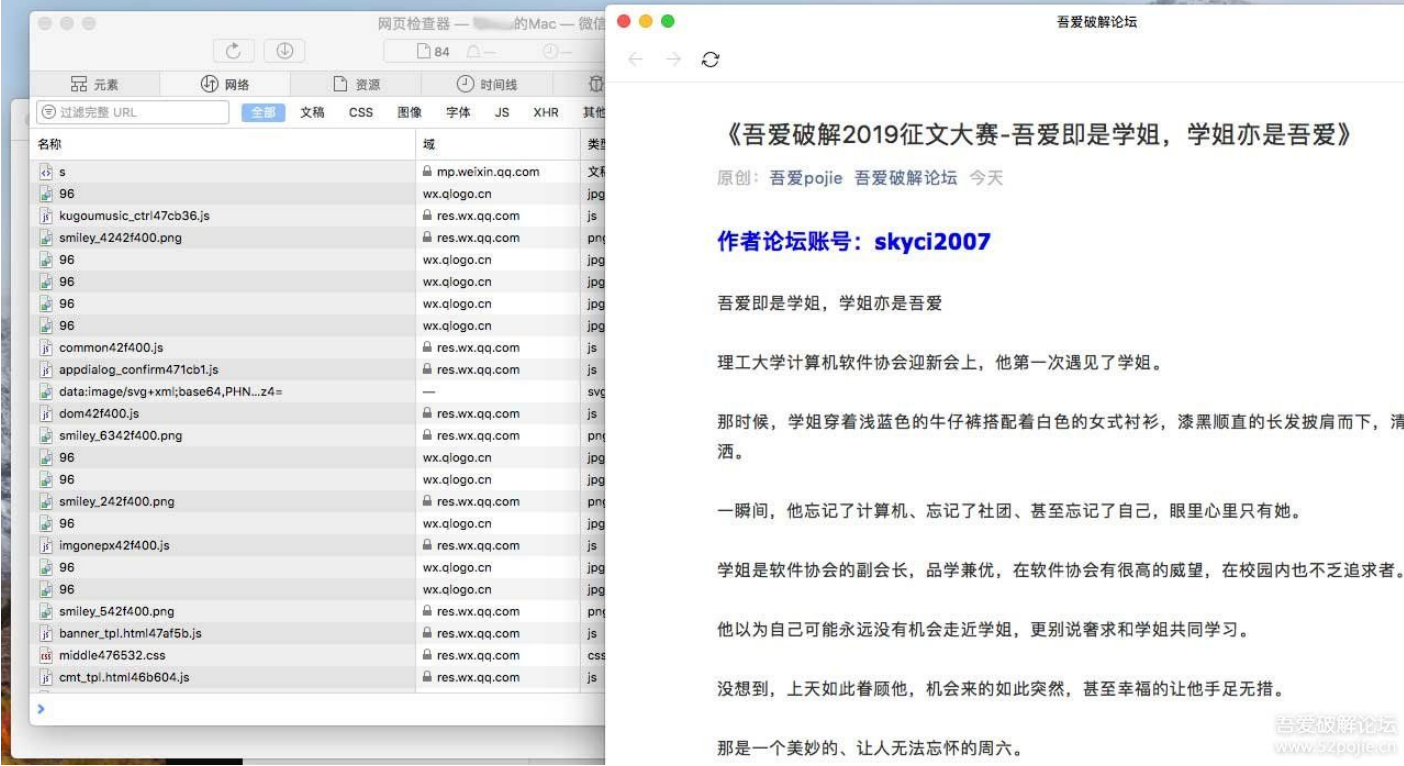
尝试了若干方法后，始终没办法获得准确的偏移量，百般无奈下，我只好采取最没技术含量的方法，直接调用命令 `mach_vm_region` 获取偏移量。在这里我也问一下论坛里的大牛，如何通过代码计算某个动态库加载后的偏移量？

获得 `mach_vm_region` 的输出后，字符串查找代码所在的动态库的名称 `mach_vm_region` 从而定位到所在行，系统从 `mach_vm_region` 中映射共享动态库到进程的虚拟内存空间中，起始位置的 8 位数字一定是以 `00000000` 开头（根据 `mach_vm_region` 中的定义，起始偏移 `00000000` 到 `00000000`），通过正则表达式匹配出开始和结束位置，从而获得偏移值。以上图为例，需要打补丁的位置就在 `000000000006DAB1`。

先打开 `gdb`，然后运行程序，在聊天软件中打开一篇公众号文章，可以看到该文章的地址已经出现在 `gdb` 的开发调试列表中。



点击文章的 `url` 即可进入调试界面，开始愉快的调试了。



代码在i”ÁtX、\$āŋŪgāqH两个版本的系统上测试过。由于是内存补丁，重启g N后就会失效。
完整代码在这里 ŪmK HnāŋŪ66ŋ”h 1VŪ”iH nŋ ŪŪX6āqkHŪŋm

免费评分

	吾爱币	热心值	理由	收起 ▲
I m 7 (0	15	15	厉害祝福哈哈	
清炒藕片、	15	15	用心讨论，共获提升！	
穷光蛋	15	15	我很赞同！	
七个涨停一倍	15	15	欢迎分析讨论交流，吾爱破解论坛有你更精彩！	
Ēgó°	15	15	用心讨论，共获提升！	
劣酒先生	15	15	已经处理，感谢您对吾爱破解论坛的支持！	
小鱼儿飞呀飞	15	15	谢谢? KŪ ēāk!	
H' ā" ē6\ m	15	15	谢谢? KŪ ēāk!	
K āH" k	15	15	用心讨论，共获提升！	
Vmŋh 50 0 0	15	15	热心回复！	
ūŪxqŋ 5 5 n	15	15	谢谢? KŪ ēāk!	
Ák" ē" 8	15	15	谢谢? KŪ ēāk!	
6ŋāŪxŋŋ 1 1)	15	15	鼓励转贴优秀软件安全工具和文档！	
NkH ēāē	15	15	谢谢? KŪ ēāk!	
+ 6 破解 5 5)	15	15	我很赞同！	
nj " kĒ	15	15	用心讨论，共获提升！	
WŪ ēnjāēŋ 50 - 6 +	15	15	厉害，赞	
kŪŪ" \X ē		15	用心讨论，共获提升！	

查看全部评分