黑客反向工程技术

iOS6 的越狱技术手段

主讲人:

任政驰*(PB10007110)*

整体思路

- Step1.在沙盒(Sandbox)外运行未签名的代码
- Step2.绕开ASLR(Address Space Layout Randomization)
- Step3.夺取内核控制权

Step1.在沙盒外运行未签名的代码

- iOS6.1的加强和不足:
  - 加强:
    - LaunchDaemons 现在从签过名的dyld cache 中加载
    - LaunchDaemons 在文件系统中不可见

Step1.在沙盒外运行未签名的代码

- iOS6.1的加强和不足:
  - 不足:
    - /etc/launchd.conf 仍然可见(曾被Corona Untether用做越狱)
    - /etc/launchd.conf 可以执行任何 launchd 命令

相关概念

- l.Symbolic Link
  - Unix 文件系统提供的一种将不同文件链接至同一个文件的机制，称为链接.相当于 Win 下的"快捷方式"
  - 使单个程序对同一文件使用不同的名字
  - 有自己的 inode,在磁盘上有一小片空间存放路径名,故可跨文件系统

相关概念

- II.Launch Daemon 和 Launch Agents
  - Mac OS X(iOS) 的启动管理,前者在开机时载入,后者在用户登录时载入
- III.launchctl 和 launchd
  - launchd 是系统初始化进程配置,具有 root 权限
  - launchctl 可用于使 launchd 运行命令行

Step1.在沙盒外运行未签名的代码

- We need to:
  - 执行 launchctl 命令
  - 改变 launchd 的 socket 权限(因为我们还没有 root)

Step1.在沙盒外运行未签名的代码

- 构建一枚没有没有代码的应用“Jailbreak”
  - 添加一个 APP 需要/var/mobile/Library/Caches/com.apple.mobile.installation.plist 文件的验证
  - 修改.../com.apple.mobile.installation.plist 文件
  - 写回改好的文件

/var/mobile/Library/Caches/com.apple.mobile.installation.plist

```
<plist version="1.0">
<dict>
        ...
        <key>System</key>
        <dict>
                <key>com.apple.DemoApp</key>
                <dict>
                        <key>ApplicationType</key>
                        <key>System</key>
                        ...
                        <key>SBAppTags</key>
                        <array>
                                <string>hidden</string>
                        </array>
                        ...
                        <key>Path</key>
                        <string>/var/mobile/DemoApp.app</string>
                        ...
                        <key>EnvironmentVariables</key>
                        <dict>
                                <key>LAUNCHD_SOCKET</key>
                                <string>/private/var/tmp/launchd/sock</string>
                        </dict>
                </dict>
                ...
        </dict>
        ...
</dict>
```

Step1.在沙盒外运行未签名的代码

- 构建一枚没有没有代码的应用“Jailbreak”
  - 手段:MobileBackup2(MB2)
    - MobileBackup2有自己的备份空间
    - 在其空间内添加任意文件基本上是不可行的,但是还是有一些可用的路径.例如:
    - MediaDomain:Media/Recordings (/var/mobile/Media/Recordings)

Step1.在沙盒外运行未签名的代码

- 构建一枚没有没有代码的应用“Jailbreak”
  - 手段:MobileBackup2(MB2)
    - 为backup构造一个 Symbolic Link(symlink):Media/Recordings/haxx 其中 haxx 指向/var/mobile
    - 当从备份恢复时,MB2 写入Media/Recordings/haxx/DemoApp.app 但实际上写入了/var/mobile/DemoApp.app

Step1.在沙盒外运行未签名的代码

- 构建一枚没有没有代码的应用 “Jailbreak”
  - 于是最终我们使用 MB2写入了想要的文件:
    - /var/mobile/Library/Caches/com.apple.mobile.installation.plist
    - /var/mobile/DemoApp.app
    - /var/mobile/DemoApp.app/Info.plist
    - /var/mobile/DemoApp.app/Icon.png
    - ...

Step1.在沙盒外运行未签名的代码

- 构建一枚没有没有代码的应用“Jailbreak”
  - ✓使用 MB2写入想要的文件
  - 重启设备以使改动生效

Step1.在沙盒外运行未签名的代码

✓构建一枚没有没有代码的应用“Jailbreak”

- 该图标只含有一条 Shebang 命令:
  - #!/bin/launchctl submit -l remount -- /sbin/mount -v -t hfs -o rw /dev/disk0s1s1
- 该命令将调用一个已经签名的应用代码来通过 Code-Signing 的审查,以执行 root 级的命令

Step1.在沙盒外运行未签名的代码

- ✓构建一枚没有没有代码的应用“Jailbreak”
  - 这就是为什么，我们在越狱时，要运行一次右面这个LOGO的应用
- ✓ROOT 文件系统(RFS)可写!

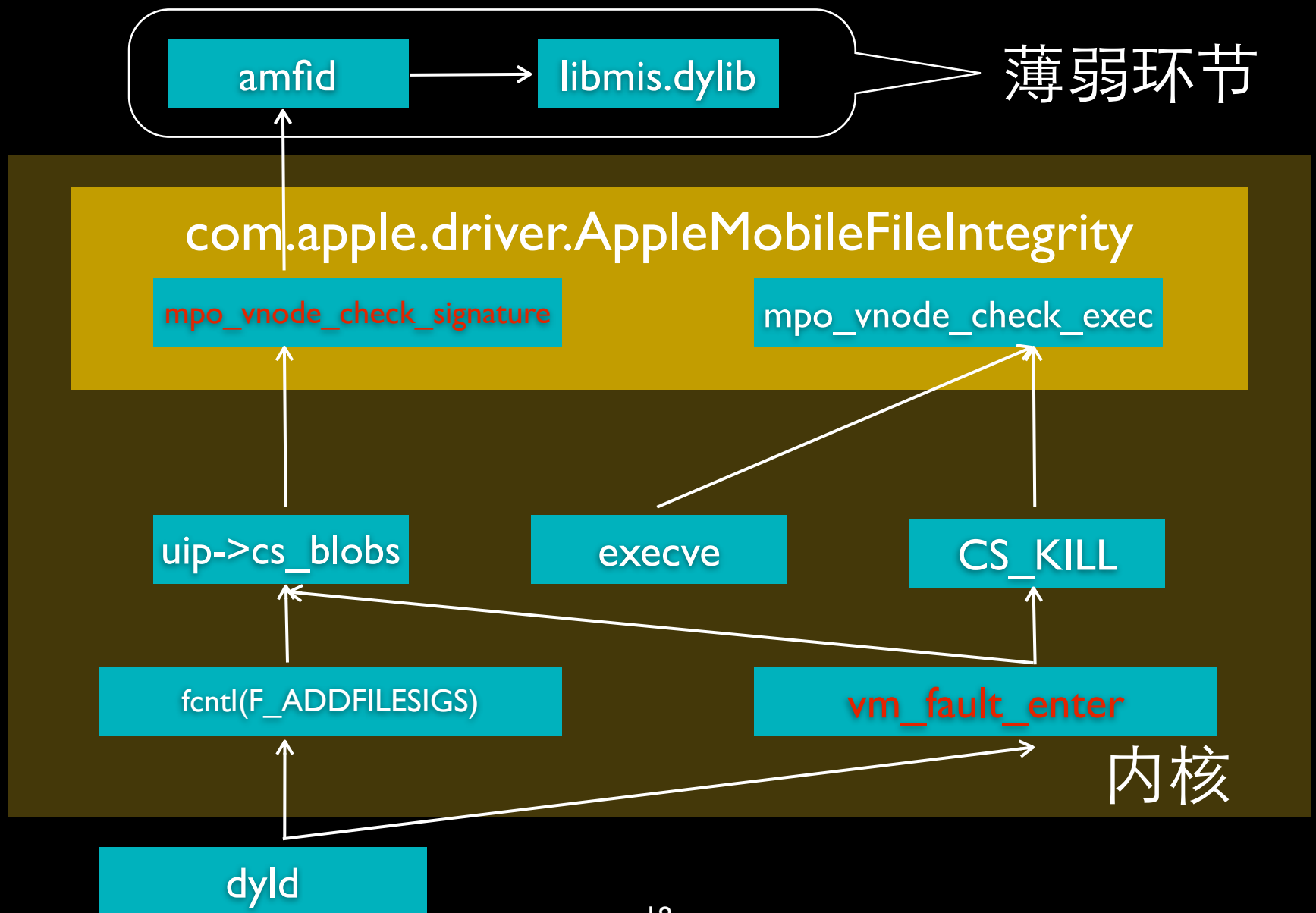
Step1.在沙盒外运行未签名的代码

- ✓拿到 root 级文件 launchd.conf 的权限
- ✓修改 launchd.conf 文件即可保存前面的工作,这样每次 iOS 设备重启,就不必再连上 USB 进行越狱操作了(完美越狱)
- 不完美越狱:没拿到 launchd.conf 权限,导致每次开机都要进行越狱操作

相关概念

- IV.AMFID(Apple Mobile File Integrity Daemon)
  - 监测移动文件完整性的守护进程
  - 校验代码签名,防止流氓程序运行
  - 但是...

AMFID工作原理图


amfid
libmis.dylib
薄弱环节
com.apple.driver.AppleMobileFileIntegrity
mpo_vnode_check_signature
mpo_vnode_check_exec
uip->cs_blobs
execve
CS_KILL
fcntl(F_ADDFILESIGS)
vm_fault_enter
内核
dyld

Step1.在沙盒外运行未签名的代码

- 破解 AMFID 防护机制
  - 不解释...

整体思路

- ✓Step1.在沙盒(Sandbox)外运行未签名的代码
- Step2.绕开ASLR(Address Space Layout Randomization)
- Step3.夺取内核控制权

相关概念

- V.ASLR
  - 随机地址空间布局(Address Space Layout Randomization)
  - 系统加载时,ASLR 都会将加载项随机分配到内存的不同位置
  - 防止某些内容总是存在内存的特定部分,让黑客有规律可寻

Step2.绕开ASLR

- 手段:异常处理向量(ARM conception vector)
  - 处理器异常、出错时(不完全是出错),即会抓这个 vector 找相应的处理函数
    - 例如,遇到看不懂的指令,它就会抓0x4地址指令,也就是“ldr pc,_undefined_instruction”,接着就跳到 undefined instruction 的函数去处理

Step2.绕开ASLR

- 手段:异常处理向量(ARM conception vector)
  - 异常向量地址不变,总是指向0xFFFF0000
  - 当程序出现异常时,ARM 异常向量就会给出崩溃的内存地址
  - 拿到足够多的信息,就可以找到内核在内存的分布范围,这样整个内核的内容也就被抓到了

整体思路

- ✓ Step1.在沙盒(Sandbox)外运行未签名的代码
- ✓ Step2.绕开ASLR(Address Space Layout Randomization)
- Step3.夺取内核控制权

Step3.夺取内核控制权

- 利用之前的成果,把内核内存的某个地址传给一个应用
- 应用返回的内容可以用来改写这部分内存地址
- 由此即可修改内核任何部分的内容

整体思路

- ✓ Step1.在沙盒(Sandbox)外运行未签名的代码
- ✓ Step2.绕开ASLR(Address Space Layout Randomization)
- ✓ Step3.夺取内核控制权

黑客反向工程技术

iOS6 的越狱技术手段

主讲人:

任政驰*(PB10007110)*

谢谢!

Thank You!