顶象iOS加固保护

一、产品简介

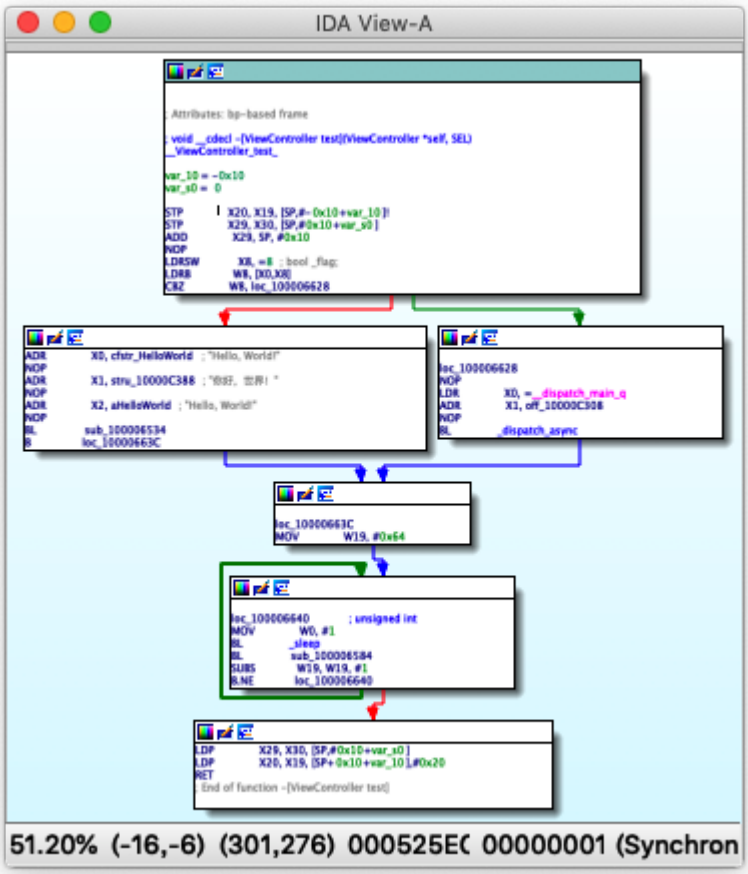
顶象iOS加固保护是顶象基于虚机源码保护技术，针对iOS平台推出的下一代加固产品。可以对iOS APP中的可执行文件进行深度混淆、加固，并使用顶象独创的虚拟机技术对代码进行加密保护，使用任何工具都无法直接进行逆向、破解。

二、产品功能

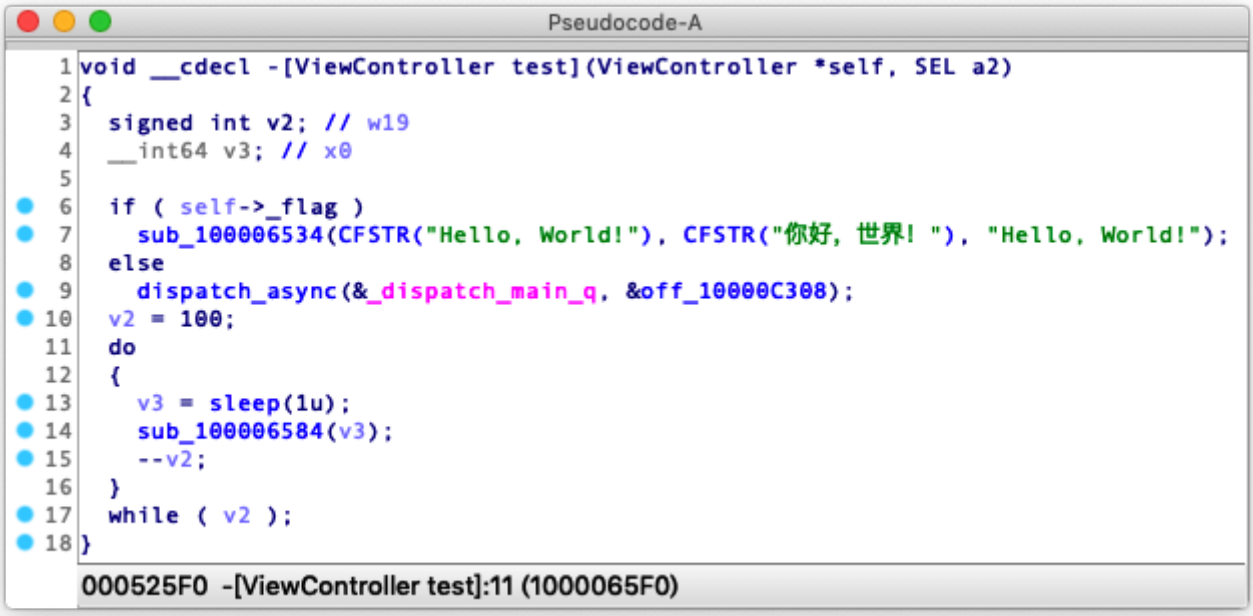
目前iOS加固主要包含逻辑混淆、字符串加密、代码虚拟化保护这三大类功能。通过对下面的代码片段进行保护来展示各个功能的效果：

```
- (void) test {
    if (_flag) {
        test_string(@"Hello, World!", @"你好，世界！", "Hello, World!");
    } else {
        dispatch_async(dispatch_get_main_queue(), ^{
            do_something();
        });
    }
    int i=0;
    while (i++ < 100) {
        sleep(1);
        do_something();
    }
}
```

将代码编译后拖入IDA Pro中进行分析，可以得到这样的控制流图，只有6个代码块，且跳转逻辑简单，可以很容易地判断出if-else以及while的特征：



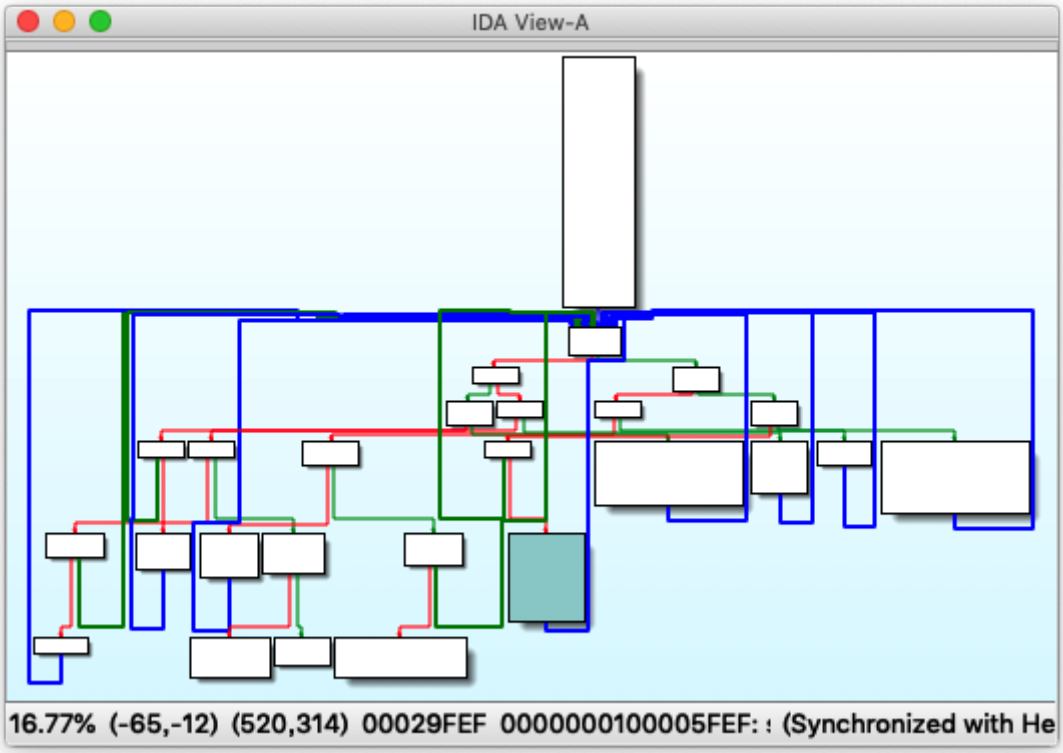
将其反编译为伪代码，代码逻辑及源代码中使用的字符串均清晰可见，与源代码结构基本一致，效果如下



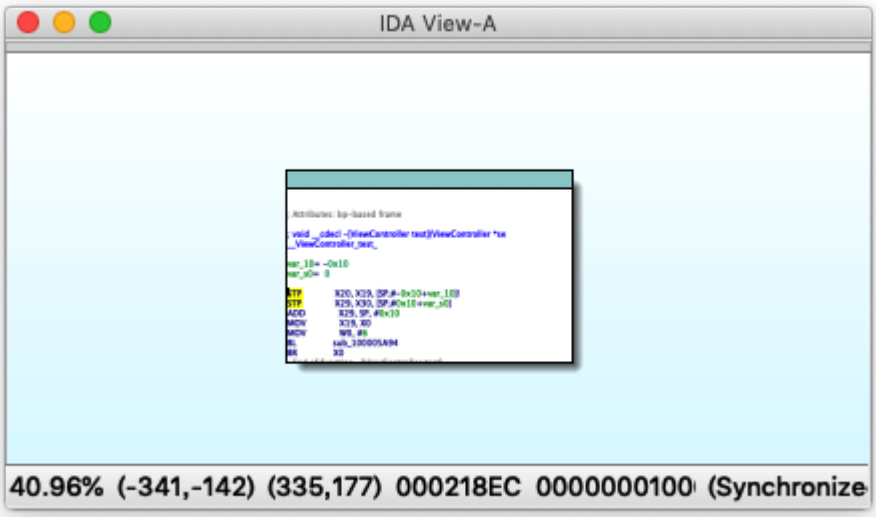
2.1 代码逻辑混淆

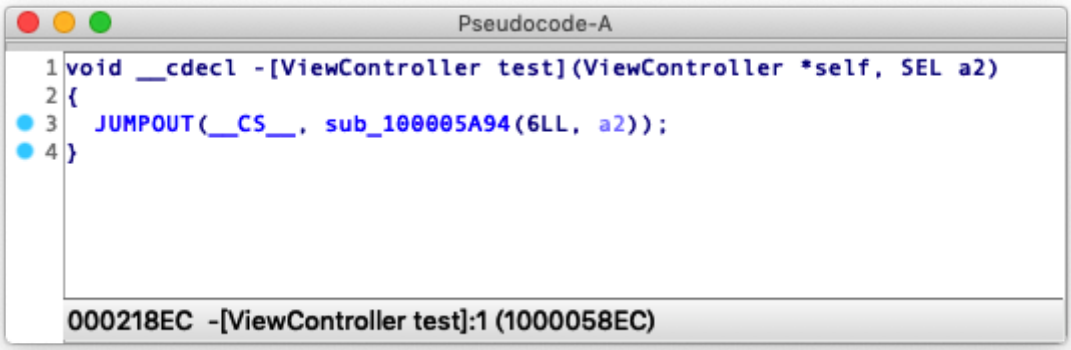
通过将原始代码的控制流进行切分、打乱、隐藏，或在函数中插入花指令来实现对代码的混淆，使代码逻辑复杂化但不影响原始代码逻辑。

对代码进行逻辑混淆保护后，该函数的控制流图会变得十分复杂，且函数中穿插了大量不会被执行到的无用代码块，以及相互间的逻辑跳转，逆向分析的难度大大增强：



若开启防反编译功能，则控制流图会被完全隐藏，只剩下一个代码块，且无法反编译出有效代码(如下图所示)，这对于对抗逆向分析工具来说非常有效，包括但不限于(IDA Pro, Hopper Disassembler, Binary Ninja, GHIDRA等)

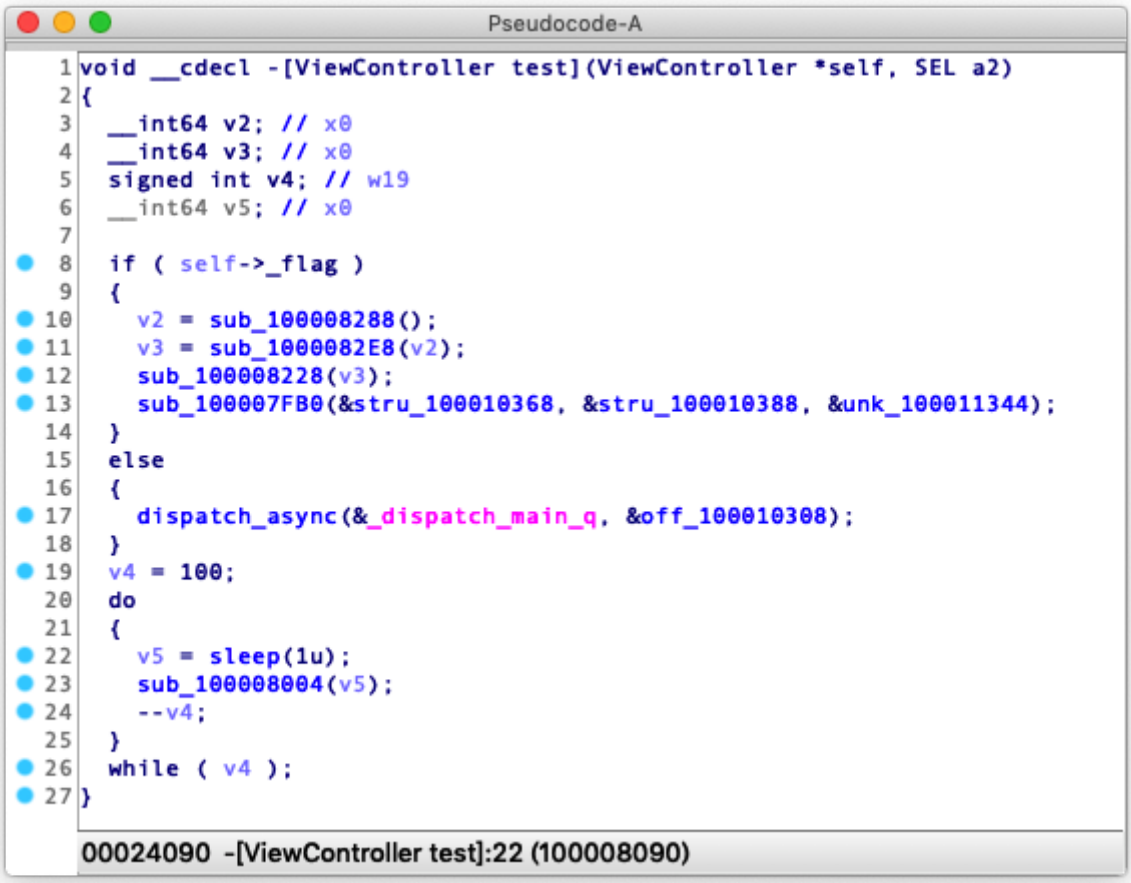




2.2 字符串加密

把所有静态常量字符串(支持C/C++/OC/Swift字符串)进行加密，运行时解密，防止攻击者通过字符串进行静态分析，猜测代码逻辑。

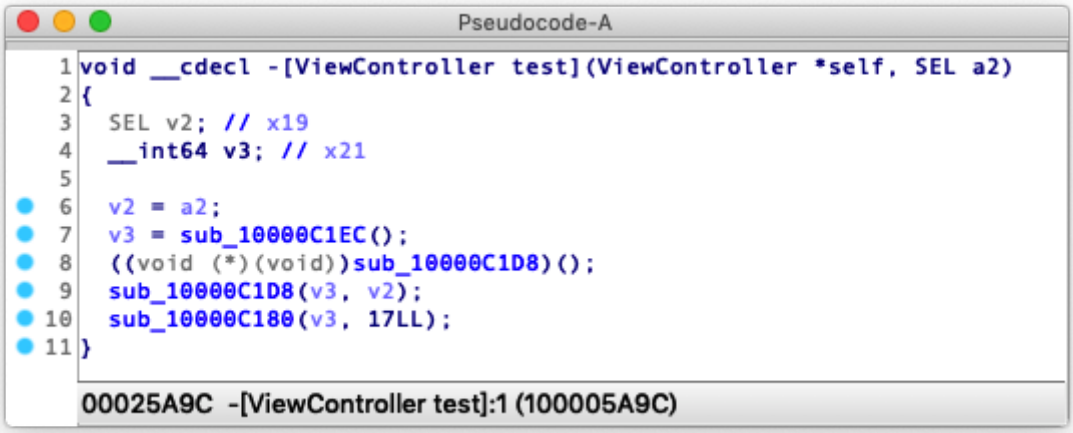
对代码中的字符串进行加密之后，所有的字符串都被替换为加密的引用，任何反编译手段均无法看到明文的字符串。你好，世界!,Hello，World!等字符串原本可以被轻易的反编译出来，但保护之后已经看不到了：



2.3 代码虚拟化

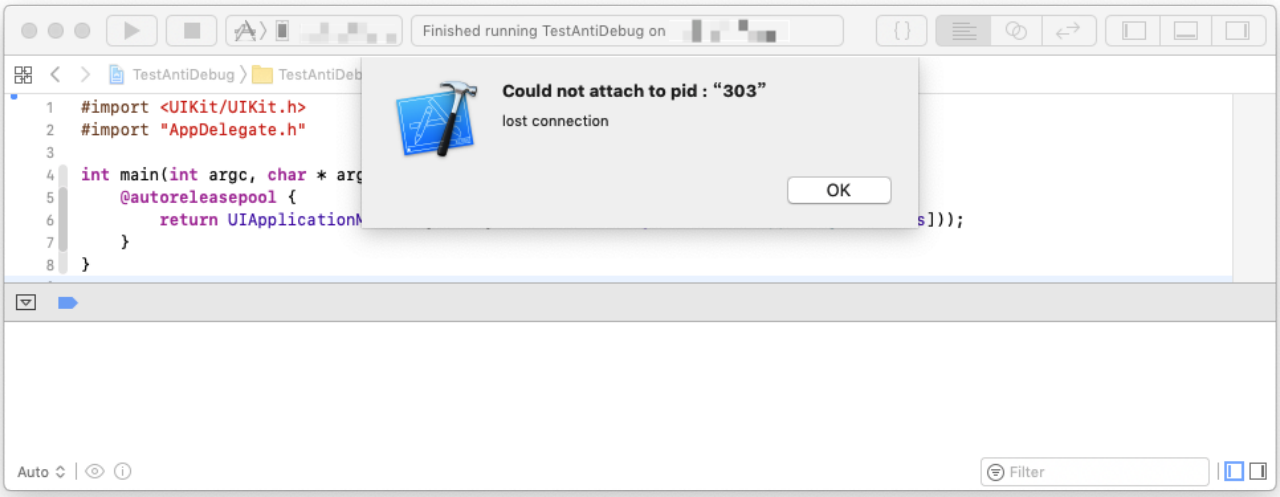
将原始代码编译为动态的DX-VM虚拟机指令，运行在DX虚拟机之上，无法被反编译回可读的源代码，任何工具均无法直接反编译虚拟机指令。

采用代码虚拟化保护后，对函数进行反编译将无法看到任何与原代码相似的内容，函数体中只有对虚拟机子系统的调用：



2.4 防调试

防止通过调试手段分析应用逻辑，开启防调试功能后，App进程可以有效地阻止各类调试器的调试行为：

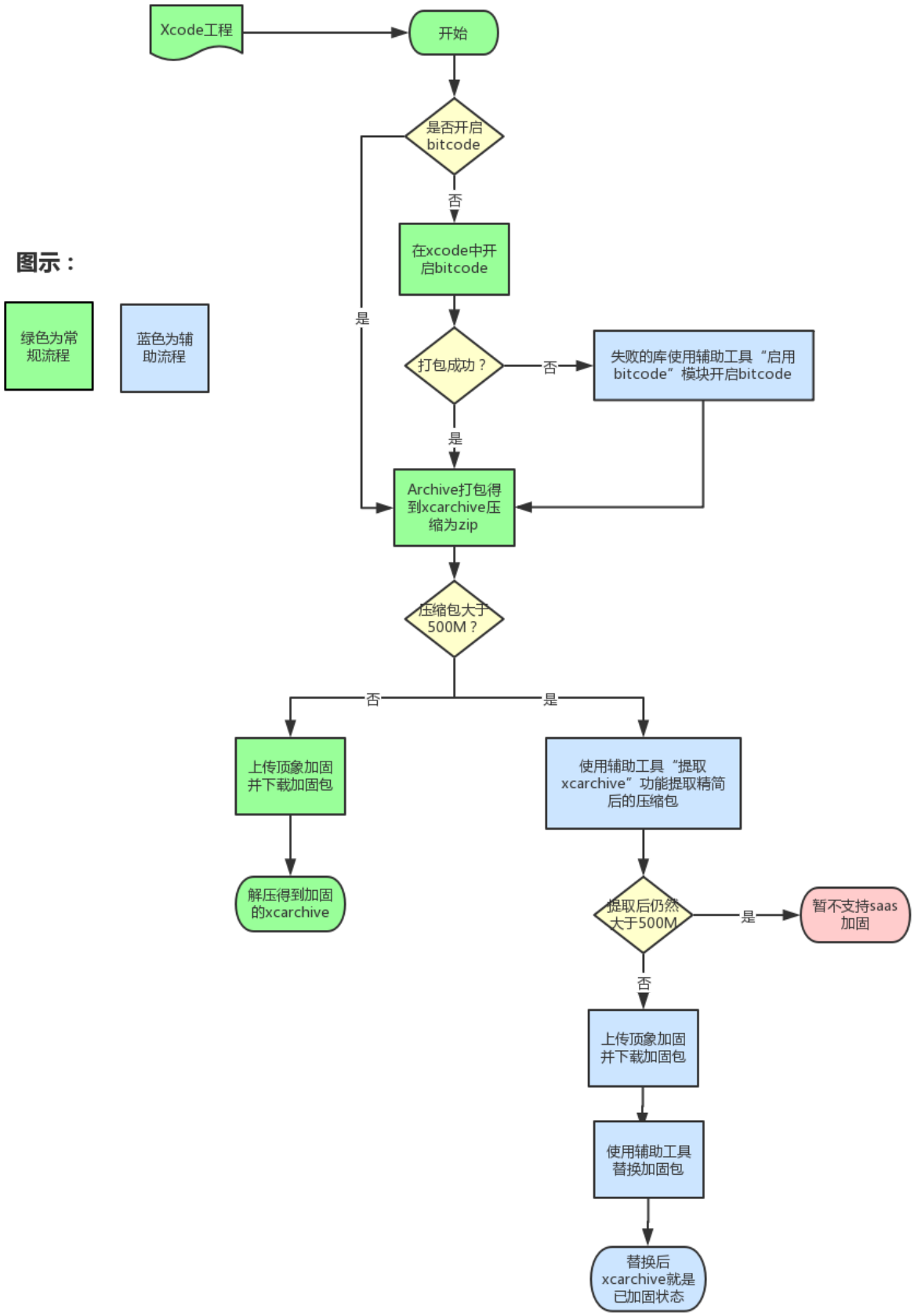


三、适用范围

- **Xcode版本**： 9.0 ~ 11.3
- **iOS版本**：不限
- **支持语言**：Objective-C(C/C++), Swift
- **支持格式**：
  - App类型，仅支持`.xcarchive`格式，不支持`.ipa`格式
  - 动态库类型，仅支持`.framework`格式，不支持`.dylib`格式
  - 静态库类型，支持`.framework`及`.a`格式
- **其他要求**：
  - `Build Setting` 中 `Enable Bitcode` 设置为 `YES`
  - 使用 `Archive` 模式编译以确保Bitcode成功启用，否则编译出的文件将只包含`bitcode-marker`
  - 若无法开启Bitcode，可使用辅助工具进行处理，详见`五、iOS加固辅助工具 -> 5.2 启用bitcode`

- 压缩后体积在 `500M` 以内

四、使用指南

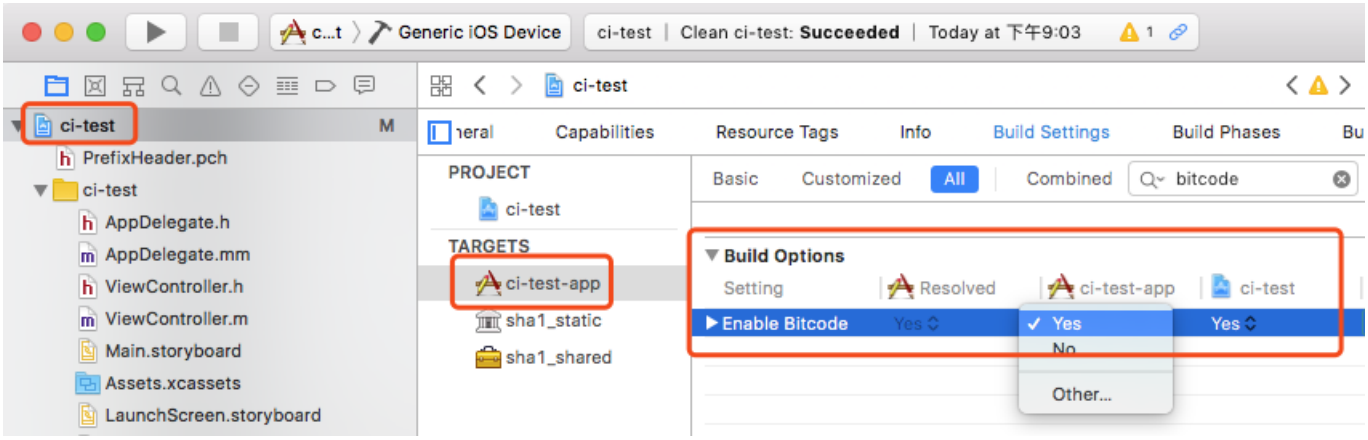


4.1 待加固文件准备

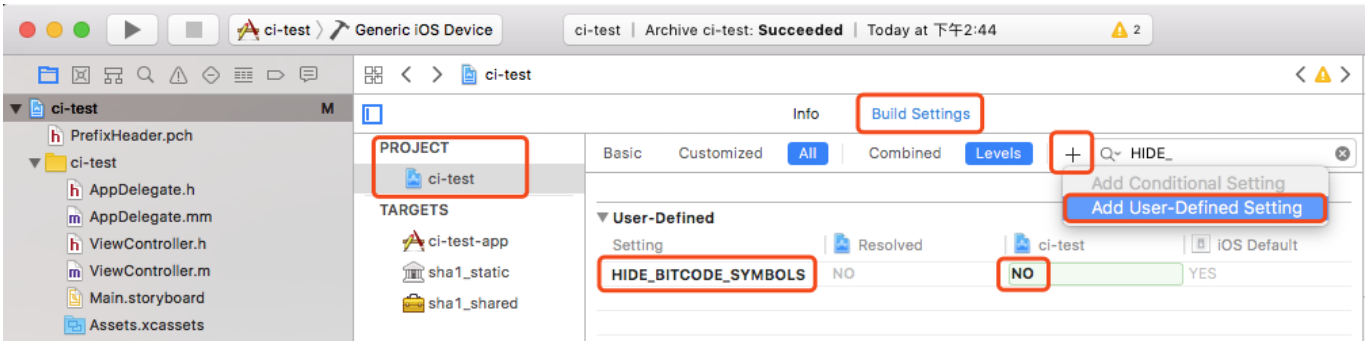
1. 在Xcode左侧导航栏，选择要保护的工程，在Targets列表中，选中要保护的Target
2. 点击进入 `Build Settings` 标签页，在 `Build Options` 一栏找到 `Enable Bitcode` ，设置为 `Yes`

- 注意：需要确保当前要编译的Target，以及所依赖的所有Target及子工程全部开启`Enable Bitcode`

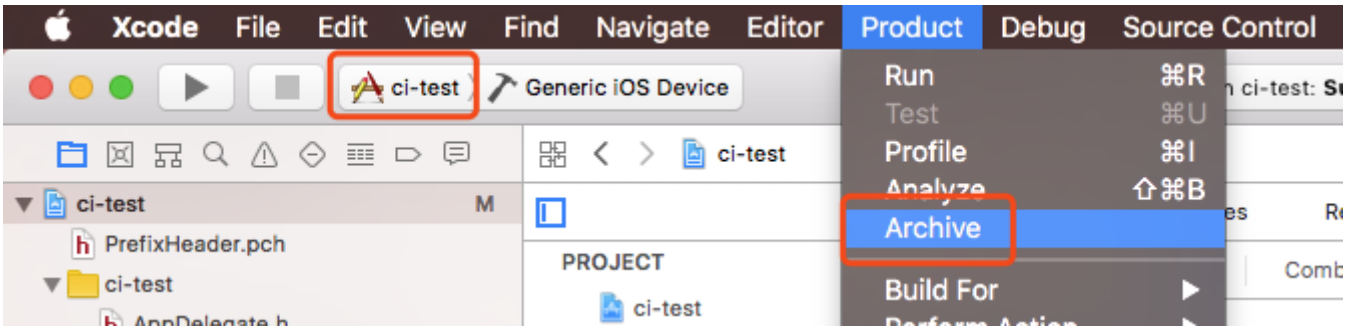
(自Xcode 7.0版本之后，iOS项目的Bitcode开关默认就已开启)



3. 在 Build Settings 中添加自定义配置 `HIDE_BITCODE_SYMBOLS = NO`，此为建议配置，不强制要求



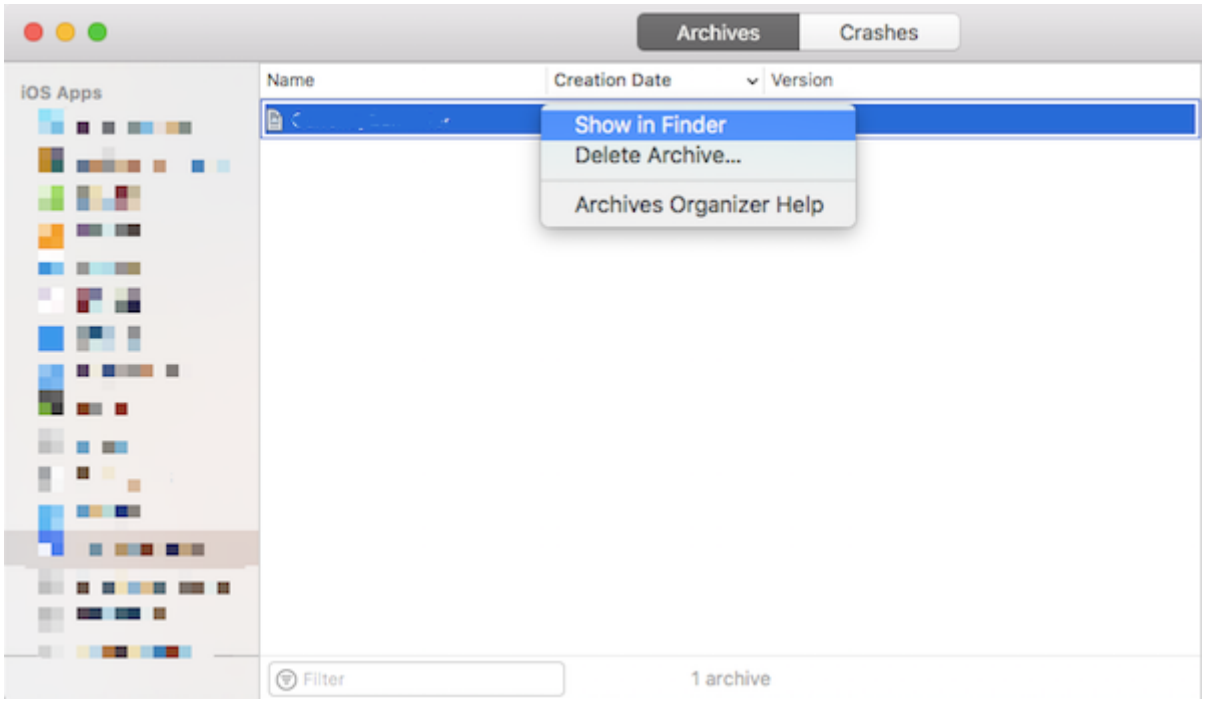
4. 将编译目标(Active Scheme)设置为要保护的Target，Device设置为 `Generic iOS Device` ，并点击点击菜单栏 `Product -> Archive` ，等待Archive完成




```
- 若工程中引入了第三方提供的动、静态库，但没有开启bitcode，则会导致打包失败，此时可以使用顶象iOS加固辅助工具([点此下载](http://appen.dingxiang-inc.com/mobile/saas/ios-util.dmg))强制开启第三方库的bitcode，具体操作详见`5.2 启用 bitcode`
```

5. Archive完成后：

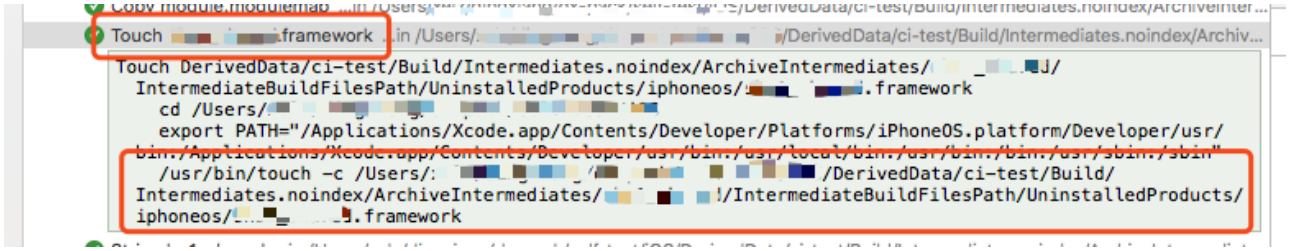
- 若要保护的文件类型是App，则在Archive后自动弹出的 `Organizer` 窗口中，右键点击Archive文件，并选择Show in Finder



然后在Finder中， 右键点击xcarchive文件，点击"压缩...", 得到 .zip 格式的压缩包

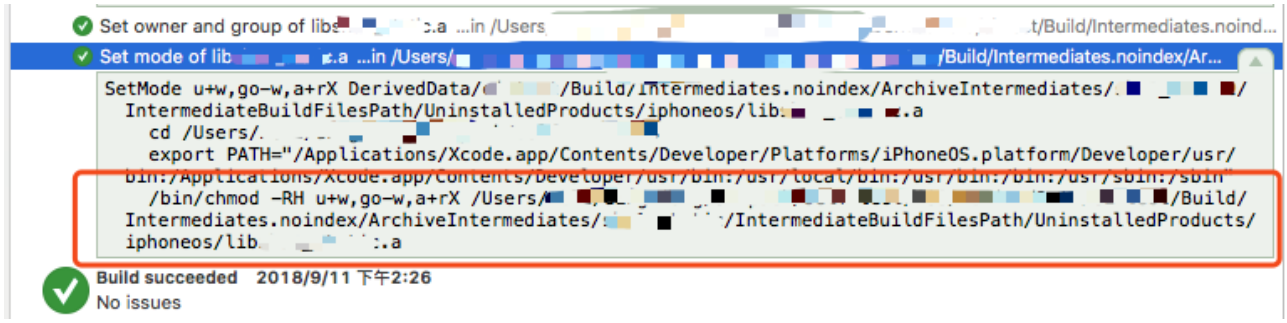
  - 如果压缩包大小超过后台上传限制，可以使用顶象iOS加固辅助工具(点此下载)提取加固必需的文件，具体操作详见`5.3 提取 xcarchive`

- 若要保护的文件类型是 `framework` ，根据编译日志找到Xcode编译输出的路径



然后在Finder中定位到该路径，右键点击.framework文件，点击"压缩...", 得到 .zip 格式的压缩包

- 若要保护的文件类型是 `.a` 格式的静态库，根据编译日志找到Xcode编译输出的路径



该 .a 文件不需要压缩，直接上传即可

4.2 上传加固文件进行加固

1. 登录顶象控制台，并进入iOS加固保护页面

2. 选择加固版本，并点击立即使用，弹出创建任务窗口

3. 根据需要自行调整保护配置

- **代码混淆强度**：针对单个函数的混淆力度，混淆强度越高越难以破解，同时带来的性能影响越大，代码体积增长越明显，一般情况下不建议使用强力混淆。
- **代码混淆比例**：随机选取指定比例的函数/字符串进行混淆/加密，未被选中的函数/字符串则不作处理，此选项可用于平衡安全性和性能、体积的影响。
- **字符串加密**：启用后会对代码中所有的字符串进行加密保护，反编译后无法导出明文字符串。
- **防调试保护**：为App提供运行时防调试能力，防止攻击者通过调试来动态分析App的逻辑。
- **虚机保护强度**：指代码虚拟化保护的强度，被保护的代码将进入虚拟机运行，无法被任何工具反编译。

- **加固后开启Bitcode**：指加固后输出的xcarchive包中，是否需要包含Bitcode，如果需要带bitcode上架Appstore，请选择“是”。
    - 开启后加固后的包中会同时包含二进制代码及Bitcode，会增大加固包的体积，但是不管开启与否最终用户下载到的App体积是相同的。
    - 如果原本工程并未开启Bitcode，因为需要进行加固才开启Bitcode，或者不需要带Bitcode上架，此处建议直接选择“否”。
- **Xcode版本**，请选择编译待加固文件所使用的Xcode版本，选择过低的版本将无法加固，选择过高的版本则可能导致加固后无法使用。
- **上传待加固文件**：将第一步中准备的文件(.zip/.a)上传。

4. 点击确定，创建加固任务

4.3 下载加固包

1. 任务提交成功后，会进入加固等待队列，后台加固处理完成后可至任务列表进行下载

2. 下载后得到加固包，格式与源包格式一致，为.zip压缩包或者.a静态库

- 若上传的是完整的xcarchive压缩包，可直接解压缩，双击即可导入到Xcode Organizer中，可在Organizer中进行ipa导出、提交AppStore等后续处理。
- 如果上传的压缩包是通过顶象iOS加固辅助工具提取出来的，则需要使用工具将加固包替换进原xcarchive中，后续导出或者提交AppStore直接对原xcarchive进行操作即可。

3. 若加固失败，可点击任务列表右侧"查看失败原因"，查看具体原因。

五、iOS加固辅助工具

5.1 功能简介

本工具用于辅助使用顶象iOS在线加固系统，常规情况下，用户无需使用本工具即可进行加固，但在某些特殊情况下需要使用本工具在用户本地进行一些操作以辅助加固的进行，可以点击此处下载。

本工具包含以下功能，各功能模块相互独立，若需要使用其中一个功能，不代表一定需要使用工具中的其他功能。

- 启用 bitcode
- 提取 xcarchive

5.2 启用 bitcode

5.2.1 功能说明

此功能模块用于强制开启第三方库的bitcode，由于顶象iOS在线加固工具是基于bitcode进行，打包时必须确保Enable Bitcode = YES，打出的包才可以用于加固。若此时工程中存在未开启bitcode的第三方库，则会导致打包失败，提示类似如下信息：

> ld: 'xxxxx' does not contain bitcode. You must rebuild it with bitcode enabled (Xcode setting ENABLE_BITCODE), obtain an updated library from the vendor, or disable bitcode for this target.
>
> ld: bitcode bundle could not be generated because 'xxxxx' was built without full bitcode.

此时可以使用本工具将报错提示中的第三方库强制"开启"bitcode。

使用本工具"开启bitcode"后并非真的在库中嵌入了bitcode，而是欺骗xcode令其认为这些库已经开启了bitcode，进而可以正常地进行编译、打包，真实的bitcode必须由源码生成。

5.2.2 适用场景

- 工程中以二进制(非源码)方式引入了第三方库，且该第三方库未开启bitcode
  - 若工程中以源码方式引入第三方库，如第三方提供子工程，或者通过Pods/Carthage等依赖管理工具引入了三方库的源码，则直接在该库对应的 Target 的 Build Settings 中找到 `Enable Bitcode` 设置为 `YES` 即可，不需要使用本工具进行处理

- 支持以下格式的第三方库
  - .framework 格式的静态库(Static Library)
  - .framework 格式的动态库(Dynamic Library)
  - .framework 格式的Object文件(Relocatable Object File)
  - .a 格式的静态库(Static Library)
- 支持处理单个库文件， 或者指定文件夹，由工具自动搜索并处理文件夹内所有未开启bitcode的第三方库

5.2.3 使用说明

1. 开启顶象iOS加固辅助工具，并调至 `启用bitcode` 选项卡
2. 在上方输入框处选择待处理的文件，可以直接将文件/文件夹拖拽至输入框中，也可以点击输入框在弹出的窗口中进行选择
3. 工具会自动判断输入的文件是否包含bitcode，若输入为文件夹，则会自动搜索文件件下的所有动静态库，并判断是否包含bitcode
4. 点击"执行"按钮，将会把所有未开启bitcode的文件进行处理，"强制"开启bitcode

5.2.4 注意事项

1. 被处理的原始原件将会在同一目录下被备份为.bak文件
2. 对于动态库，经过此工具处理后，只是用于欺骗本地Xcode，让其可以在开启Bitcode的情况下顺利完成打包，但打出的包不能直接用于提交AppStore，若直接提交会收到 `Invalid Bundle` 的提示，必须先经过顶象加固处理后方可提交。(静态库以及Object文件不受此影响)
3. 使用此工具处理后，在创建加固任务时，请将 `加固后开启bitcode` 设置为 `否`

5.3 提取 xcarchive

5.3.1 功能说明

此功能模块用于完成xcarchive的压缩以及加固文件的替换。顶象iOS在线加固对上传的文件大小有一定限制，若app体积过大，则xcarchive压缩后可能会超出此限制，此时可以使用此工具提取加固必须的文件进行压缩，减小压缩包体积。待完成加固后，再将下载下来的加固包替换回原始包中。

5.3.2 适用场景

- 待加固的包为xcarchive格式，且压缩后体积超出了上传限制(当前为500M)，若未超过限制，则不需要使用本工具。

5.3.3 使用说明

1. 开启顶象iOS加固辅助工具，并调至"提取xcarchive"选项卡，按照界面提示依次进行操作

2. 在 xcarchive文件 输入框处选择待加固的xcarchive文件，可以直接xcarchive文件拖拽至输入框中，或者点击输入框并在弹出的窗口中选择

3. 输出路径 默认在选择的xcarchive同一目录下，点击 输出路径 输入框，可在弹出的窗口中选择其他路径以进行修改
4. 点击 提取 按钮，工具将提取加固必须的文件进行压缩，保存为上一步中指定的文件
5. 访问顶象控制台，使用上一步提取出的文件进行加固
6. 完成加固后，将加固包下载至本地，拖拽至 加固文件路径 输入框，并点击 替换 按钮，加固后的二进制文件将自动替换进原始xcarchive中

5.3.4 注意事项

1. 替换加固包时，原始xcarchive包若需要保留，需自行备份，或者使用提取出的zip文件再次进行替换即可恢复。
2. 提取完毕后工具可以先关闭，等加固完成后重新开启，替换时重新在 xcarchive文件 处选择之前提取的xcarchive即可
3. 如果提取后的压缩包(只包含加固必须的二进制文件)体积仍然超过了最大限制，将无法使用在线加固系统进行加固。

六、常见问题

6.1 什么是Bitcode?

为什么要开启Bitcode？不开启Bitcode是否可以加固?

A: Bitcode是一种源代码被编译为二进制机器码过程中的中间表示，它既不是源代码，也不是机器码。开启Bitcode之后，Xcode编译出的二进制文件中会嵌入Bitcode。Apple在Xcode7中引入了Bitcode机制，并默认将其开启，通过提交包含Bitcode的ipa，Apple可以在新产品或者新技术发布后，无需开发者参与，即可通过Bitcode对程序进行优化、或者编译出适用于新产品的应用程序。

顶象的加固技术正是基于Bitcode进行处理，因此需要开启Bitcode进行打包，不开启Bitcode无法进行加固。

6.2 开启Bitcode之后打包失败

若提示`xxx does not contain bitcode` 或者 `xxx was built without full bitcode`

A: 原因是项目中依赖的库文件没有开启bitcode

- 若报错的库文件有对应的源代码(自己开发的或者是通过Pod等工具引入的第三方库，在Xcode中存在对应的源代码及Target)，则按照上述方式找到该Target，并开启bitcode即可
- 若报错的库文件为第三方提供的已经编译好的二进制文件(没有源代码)，则需要联系该第三方提供包含Bitcode的版本。也可以使用顶象iOS加固辅助工具强制开启，详见`五、iOS加固辅助工具 -> 5.2 启用bitcode`。
- 若报错为`xxx was built without full bitcode`，说明该第三方库配置了 `Enable Bitcode = YES`，但并不是以Archive方式打包的，而是本地编译的Development版本，由于本地编译、测试过程中并不需要bitcode，因此Xcode在此种场景下只会在二进制文件中添加一个字节的bitcode marker以加快编译速度。解决方案：联系该第三方库的提供方使用Archive方式编译。也可以使用顶象iOS加固辅助工具强制开启，详见`五、iOS加固辅助工具 -> 5.2 启用 bitcode`。

若提示 `-weak_library and -bitcode_bundle (Xcode setting ENABLE_BITCODE=YES) cannot be used together`

A: 在 `Other Linker Flags` 中找到类似 `-weak_library /usr/lib/libxxx.dylib` 的配置，改为 `-weak-lxxx` 的形式，如将 `-weak_library /usr/lib/libstdc++.dylib` 改为 `-weak-lstdc++`

6.3 加固对体积的影响

A: 不同的加固参数会带来不同的体积变化，并且相同的加固配置对于不同的代码所产生的影响也是不确定的，这取决于代码自身的复杂程度。以下为基于历史样本的统计值，仅供参考。

计算方法为：加固后二进制中`__TEXT`的大小 / 加固前`__TEXT`的大小，并非整个二进制文件或者ipa包的体积变化。

类型	体积变化
基础版(限制30%)	约2倍
标准版(简单混淆)	约1.5倍
标准版(普通混淆)	约2.2倍
标准版(强力混淆)	约3.1倍
旗舰版(开启全部功能)	不超过4倍

加固后，`__TEXT`的大小有可能会超出AppStore限制，具体限制请参阅App Store Connect官方文档，请根据实际情况合理设置加固参数。

注：从二进制文件整体的角度来看，加固后体积可能会变小，因为加固前二进制文件中同时包含未加固的机器码和比机器码大数倍的bitcode，若未勾选加固后开启`bitcode`，加固后的二进制文件中将只包含机器码，bitcode会从二进制中被移除。

6.4 加固辅助工具无法打开

A: 这是由于苹果的安全机制导致，Mac OSX默认只允许AppStore中下载软件

- 若提示“来自身份不明的开发者”，请参考Apple官方指导 打开来自身份不明开发者的应用
- 若提示“文件已损坏”，请依次打开“系统偏好设置”->“安全性与隐私”->“通用”->“允许从以下位置下载的应用”，选择“任何来源”，然后再尝试打开
    - 若没有“任何来源”，请先在终端中运行`sudo spctl --master-disable`后，重试

6.5 Bitcode格式无法识别

提示请确保使用官方Xcode编译器进行编译

A: 此提示说明项目中包含非官方Xcode编译的代码，第三方编译器生成的Bitcode无法被Apple识别，如果在开启bitcode的情况下提交AppStore会直接被拒，提示 `Invalid Bundle`，但若不开启bitcode提交并不会有任何影响。顶象加固是基于bitcode进行，仅对Xcode内置的Apple LLVM编译器进行了适配，同样无法识别第三方编译器产出的未知格式的bitcode。这里存在两种情况

- 第一种是在工程中使用了ollvm等第三方编译器工具链去编译整个项目，这种情况，换回Xcode内置的默认工具链即可。
- 第二种是因为项目中集成了使用了ollvm等非官方编译器(Apple LLVM)进行编译的第三方SDK(静态库)，如遇到此种问题，可采取以下方案跳过加固时对第三方SDK的处理：

假如有两个第三方SDK存在此种问题，名字分别为 `libSDK.a` 和 `SDK.framework`，均存放在桌面下，路径为 `~/Desktop`

1. 自行对原SDK进行备份
2. 在终端执行以下命令，将SDK中的bitcode去除

```
$ xcrun bitcode_strip -r ~/Desktop/libSDK.a -o ~/Desktop/libSDK.a
$ xcrun bitcode_strip -r ~/Desktop/SDK.framework/SDK -o
~/Desktop/SDK.framework/SDK
```

3. 使用顶象iOS加固辅助工具强制为上述SDK开启bitcode
4. 将工程内使用的第三方SDK，替换为上进一步处理后的SDK
5. 正常打包并进行加固

> 若存在其他不希望保护的静态库类型的SDK，同样可以参照此流程进行处理。