ios平台clang-ollvm编译器安装及使用说明

1. clang-ollvm安装：

1）下载并编译项目: https://github.com/obfuscator-llvm/obfuscator

$ git clone -b llvm-4.0 <https://github.com/obfuscator-llvm/obfuscator.git>

$ mkdir build

$ cd build

$ cmake -DCMAKE\_BUILD\_TYPE=Release ../obfuscator/

$ make -j7

2）将build目录重命名并放到你自己指定的路径下，例如：

$ mv build /User/sensetime/hexu/obuscator

3）创建C++标准库的头文件链接：主要解决用ollvm-clang编译C++程序时找不到标准库头文件的问题

$ cd /path/to/obuscator\_include

#（例如/User/sensetime/hexu/obuscator/include）

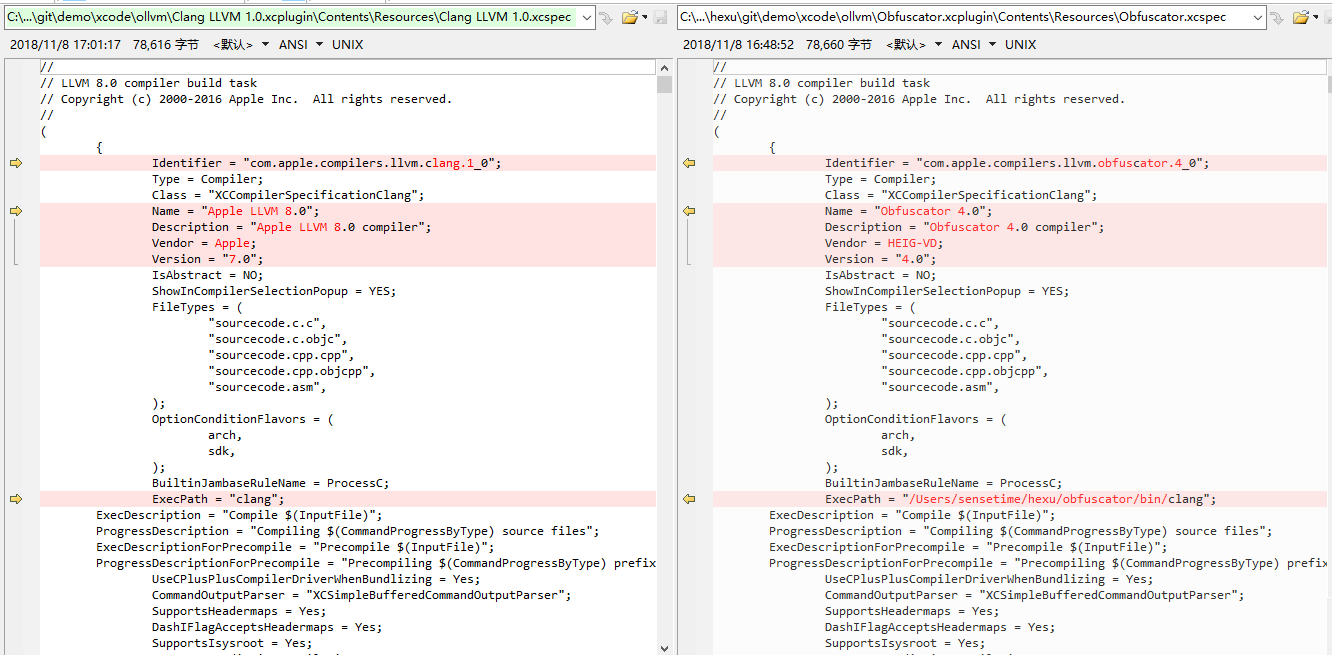
$ ln -s /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/include/c++/ c++

1. 以xcode插件的方式使用clang-ollvm：

1）增加clang-ollvm插件：

参考<https://github.com/obfuscator-llvm/obfuscator/wiki/Installation>

在Xcode 8上增加插件时，Resources/下的.xcspec文件已经是json格式了，可以参考下图：



2）在Xcode中指定使用clang-ollvm编译器：

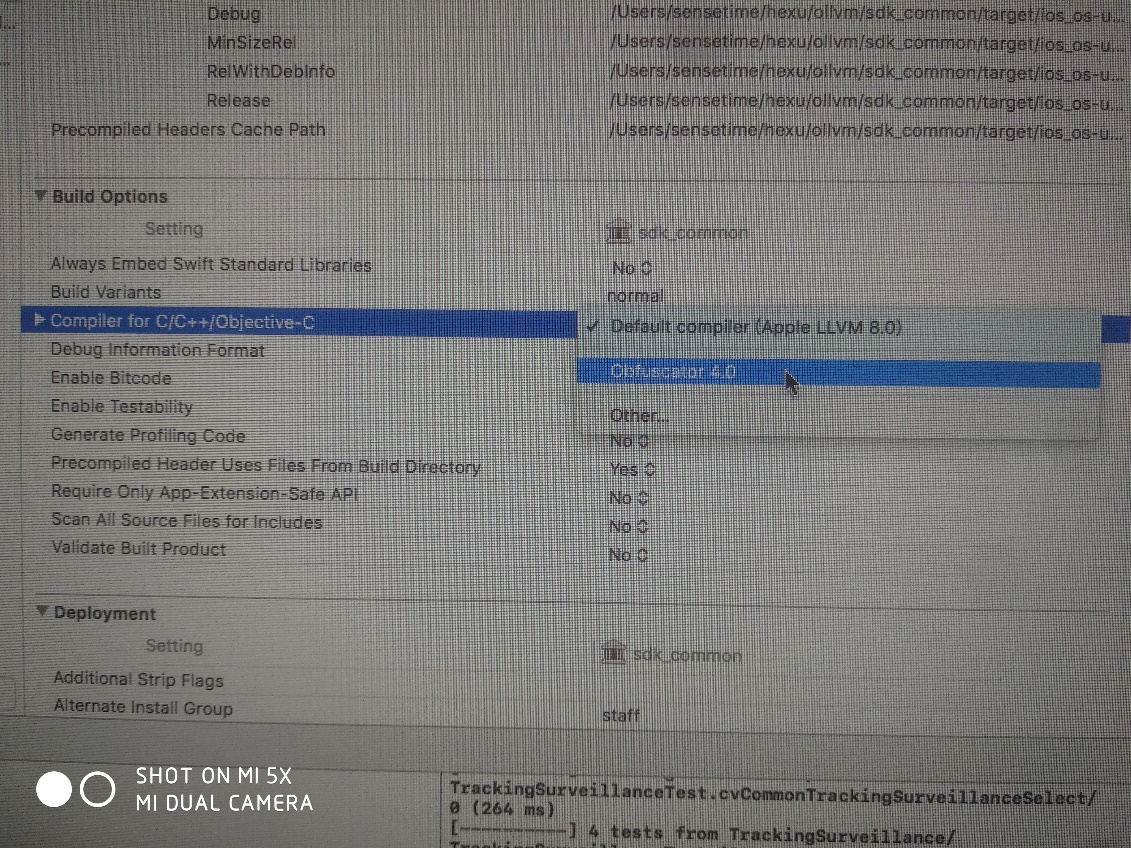
点击项目图标:

-> Build Settings: 选择All和Combined,

-> Build Options

-> Compiler for C/C++/Objective-C: 选择Obfuscator4.0

参考下图：



1. 在xcode中手动设置ollvm-clang的路径

如果通过在xcode手动设置ollvm-clang的路径，就不需要自己添加ollvm-clang插件那些步骤了。具体操作如下：

点击项目图标:

菜单: Editor

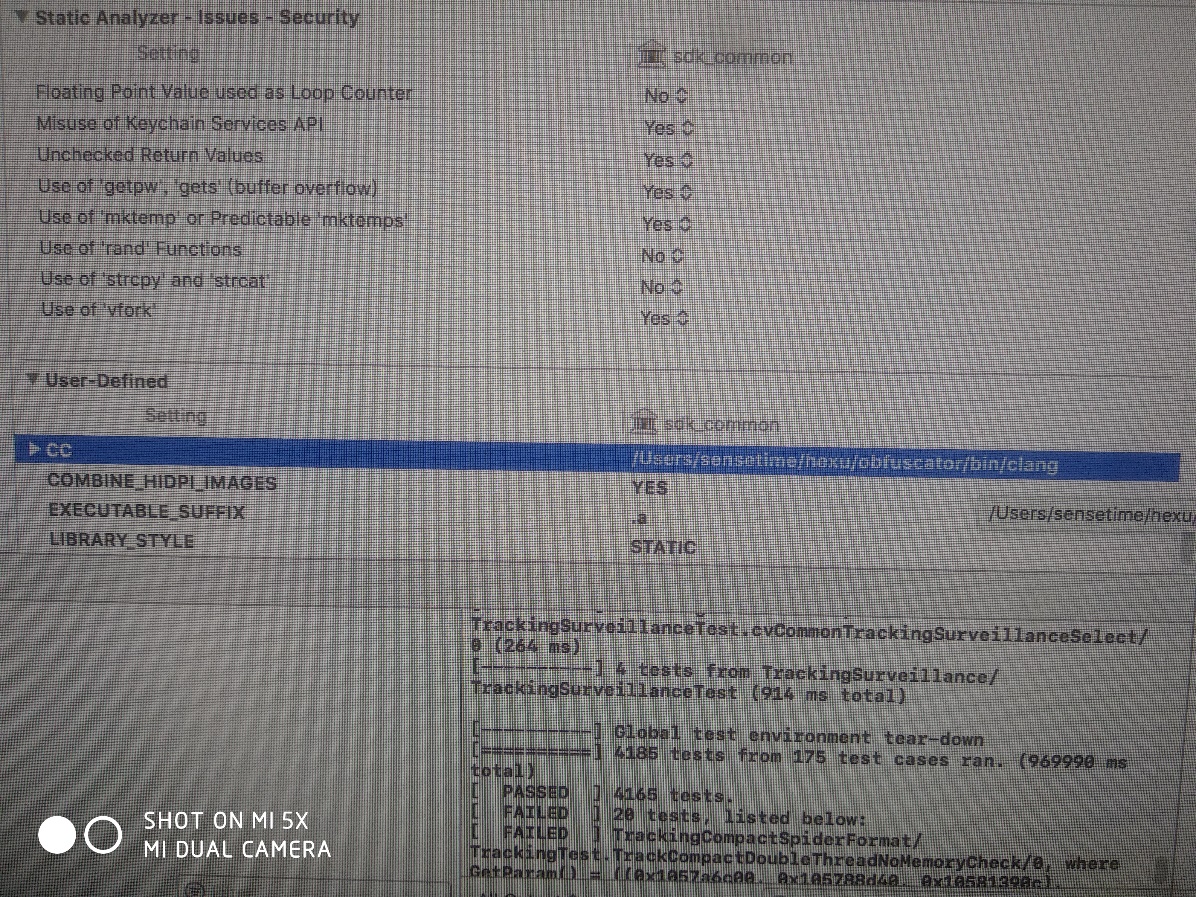
-> Add Build Setting

-> Add User-Defined Setting

将NEW\_SETTING改成CC, value里填/path/to/obfuscator\_bin/clang

可以同样方式增加CXX来指定c++编译器。

参考下图：



1. 通过CMake生成Xcode项目工程并设置ollvm-clang的路径

如果是通过CMake脚本生成Xcode项目工程的话，可以通过设置Xcode属性达到自动设置ollvm-clang路径的效果，具体操作是在CMake脚本中增加如下代码：

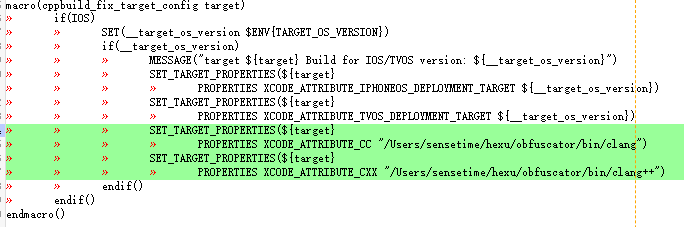
set\_target\_properties(your-target PROPERTIES XCODE\_ATTRIBUTE\_CC "${YOUR\_CC}")

set\_target\_properties(your-target PROPERTIES XCODE\_ATTRIBUTE\_CXX "${YOUR\_CXX}")

参考:

<https://stackoverflow.com/questions/39327952/how-can-i-force-xcode-to-use-a-custom-compiler>

sdk\_common中的sample如下图：



1. 用ollvm-clang直接替换Xcode默认的clang编译器：

这种方法最简单粗暴，但是在使用中不再需要修改任何编译器选项。具体操作如下：

$ cd /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin

$ mv clang clang.backup

$ ln -s /path/to/obfuscator\_bin/clang clang

#（例如/User/sensetime/hexu/obuscator/bin/clang）

1. 如何开启clang-ollvm的代码混淆功能：

1）Obfuscator-llvm支持三种混淆模式分别是：

Instructions Substitution： -mllvm -sub

Bogus Control Flow： -mllvm -bcf

Control Flow Flattening： -mllvm -fla

具体原理参考<https://github.com/obfuscator-llvm/obfuscator/wiki/Features>，

2）在Xcode中指定相应混淆模式的编译器选项：

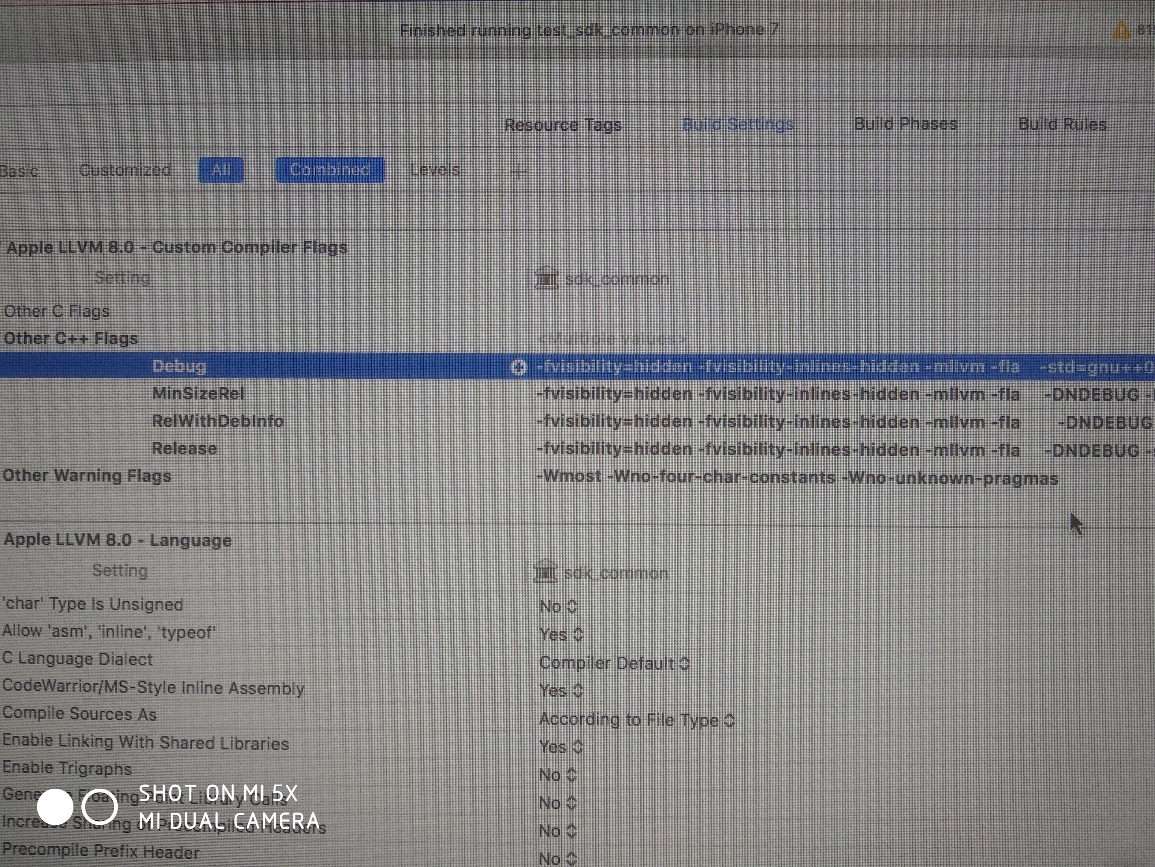
点击项目图标:

-> Build Settings: 选择All和Combined,

-> Obfuscator 4.0 - Custom Compiler Flags

-> Other C++ Flags: 增加 -mllvm -fla（根据自己需要修改）

参考下图：



3）在CMake脚本中指定相应混淆模式的编译器选项

通过设置CMAKE\_CXX\_FLAGS变量，可以达到效果，例如：

SET(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} -mllvm -fla")

4）只混淆特定函数：

有的时候，由于效率或其它原因的考虑，我们只想给指定的函数混淆，OLLVM也提供了对这一特性的支持。比如，想对函数func()使用bcf混淆，只需要给函数func()增加bcf属性即可。

int func() \_\_attribute((\_\_annotate\_\_(("bcf"))));

fla，sub和bcf三个属性可以搭配使用。如果不想对func()函数使用bcf属性，那标记为“nobcf”即可。

参考下图：

