

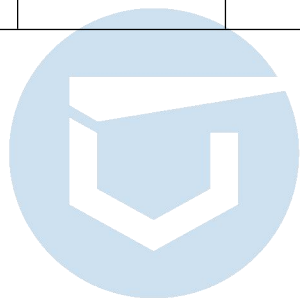
鸿蒙 NEXT 应用加固

用户使用手册



北京梆梆安全科技有限公司

版本号	日期	修订页/修订描述	作者	审批人
V2.0.3	2024/05/10	Init	赵猛	王志锋
V2.1.0	2024/08/09	新增功能： 1、C/C++ bitcode； 2、运行环境安全：防 OpenHarmony 运行和防模拟器运行； 3、ArkTS：表达式混淆； 4、so 加固；	岳永露	王志锋
V2.2.0	2022/10/22	新增功能： 1、ArkTS：防伪造控制流； 2、登录界面增加使用鸿蒙原生混淆器的功能；	岳永露	王志锋



梆梆安全
BANGCLE



目录

1. 引言	5
1.1. 目的	5
1.2. 范围	5
1.3. 运行环境	5
1.4. 术语和定义	6
2. 鸿蒙 NEXT 应用加固	6
2.1. 简要使用流程介绍	6
2.2. 应用加固关键配置	7
3. 加固插件化软件包 (PLUGIN)	8
3.1. 前置条件	8
3.2. 安装与卸载	9
3.2.1. 安装插件客户端	9
3.2.2. 卸载插件客户端	11
3.3. 登录或离线使用	13
3.3.1. 登录梆梆安全 SaaS 服务平台	13
3.3.2. 登录本地部署的加固系统	15
3.3.3. 离线登录	16



3.4. 加固策略配置	17
3.5. 鸿蒙 NEXT 应用加固处理	19
3.5.1. 基于 HarmonyObfuscator 界面加固处理	19
3.5.2. 基于 Terminal 命令行加固处理	21
3.5.3. 基于 Build 菜单加固处理	22
3.5.4. 查看加固效果	23
3.5.5. 工程 SO 加固（可选）	24
3.6. 关于	25
3.6.1. 查看客户端版本	25
3.6.2. 查看授权信息	25
3.6.3. 退出登录	25
3.7. 使用鸿蒙原生混淆器	25
4. 加固命令行软件 COMMAND_LINE	27
4.1. 前置条件	27
4.2. LINUX/MAC/WINDOWS 构建流水线自动化加固	28
附件：常见问题和解决方案	33
● 关于清除项目编译中间信息	33
● 如何通过 HAP 文件查看加固策略	33



1. 引言

1.1. 目的

本文档主要用于使用户熟悉梆梆安全鸿蒙 NEXT 安全加固工具应用加固产品（简称为：梆梆安全鸿蒙 APP 加固）功能，并指导用户掌握操作使用流程。

1.2. 范围

本文档为公开文档，适用于使用梆梆安全鸿蒙 APP 加固产品的相关产品人员、技术人员、测试人员、售后实施人员等。

1.3. 运行环境

梆梆安全鸿蒙 APP 加固客户端有以下两种使用形式：

以插件的形式安装和运行在华为鸿蒙开发工具 DevEco Studio 上，支持 MacOS/Windows 操作系统使用。

以命令行的形式安装和运行在华为鸿蒙命令行 commandline-tools 上，支持 MacOS/Linux/Windows 操作系统使用。

操作系统、鸿蒙开发环境与梆梆安全鸿蒙 APP 加固产品的对应关系如图所示（**推荐使用 mac 开发环境**）：

操作系统	鸿蒙开发软件	梆梆安全鸿蒙 NEXT 应用加固
Mac	devecostudio-mac-*.zip	idea_plugin.zip
	commandline-tools-mac-*.zip	command_line.zip
Windows	devecostudio-windows-*.zip	idea_plugin.zip
	commandline-tools-windows-*.zip	command_line.zip



Linux	commandline-tools-linux-*.zip	command_line.zip
-------	-------------------------------	------------------

1.4. 术语和定义

术语/定义	说明
鸿蒙 NEXT 应用	指 HarmonyOS NEXT 系统上运行的应用软件。本文提到的梆梆鸿蒙 APP 加固,是指对 HarmonyOS NEXT 应用的加固。
usc.an110.com	梆梆安全 SaaS 安全云服务平台。
Plugin	鸿蒙 DevEco Studio 开发工具插件软件包 idea_plugin.zip。安装后可以随同 DevEco Studio 使用。

2. 鸿蒙 NEXT 应用加固

2.1. 简要使用流程介绍

- 获取梆梆安全鸿蒙 NEXT 加固软件包；
- 获取梆梆安全分配的系统账号、密码，或者相关 license 授权文件；
- 部署梆梆安全鸿蒙 NEXT 加固服务器；
- 在 DevEco Studio 内安装梆梆安全鸿蒙 NEXT 加固插件 idea_plugin.zip,更新 license 授权、登录、使用 APP 加固服务（参见[加固插件化软件包（Plugin）](#)）。
- 或者在鸿蒙开发命令行工具基础上使用梆梆安全鸿蒙 NEXT 加固命令行工具 command_line 完成加固引擎部署后，使用 APP 加固服务（参见[加固命令行软件 command_line](#)）。

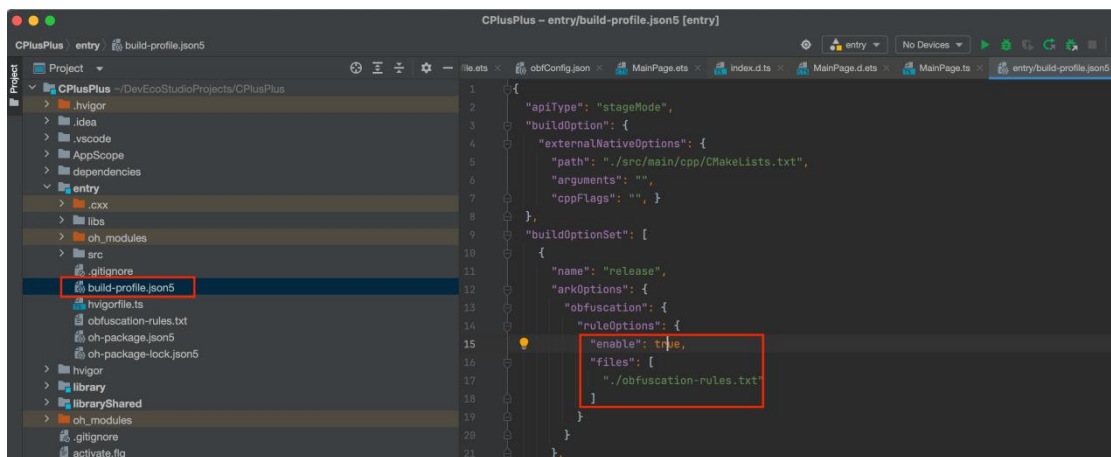


- 加固后的应用生成位置和运行方式无变化，具体请参考华为官方说明。
- 如果未获取 SDK 授权，无法加固编译 HAR。需要联系项目经理获取授权或还原鸿蒙原生混淆。

2.2. 应用加固关键配置

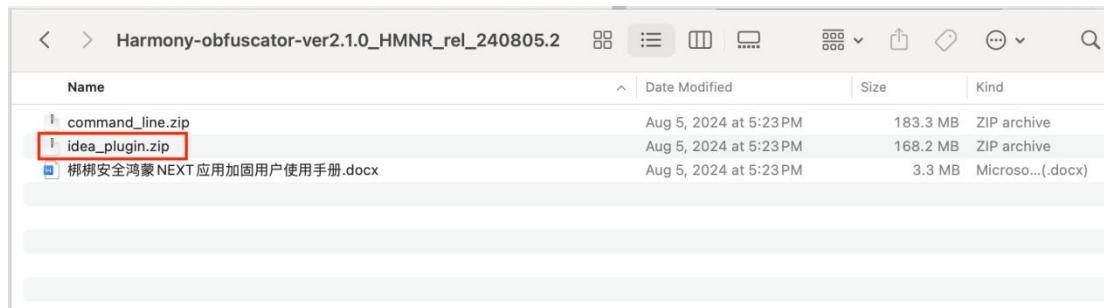
对于 hap 和 hsp 进行 app 加固的关键配置：

参与加固的 module 需在 build-profile.json5 中配置并 enable obfuscation-rules.txt 规则文件。



3. 加固插件化软件包（Plugin）

示例：[Harmony-obfuscator-ver2.2.0_HMNR_rel_YYMMDD.x.zip/idea_plugin.zip](#)



3.1. 前置条件

梆梆安全鸿蒙 NEXT 应用加固客户端软件需依赖华为鸿蒙 DevEco Studio 的开发编译环境使用，请确保：

1. 已成功安装并使用 DevEco Studio。
2. 待加固的鸿蒙 NEXT SDK 可以通过 DevEco Studio 正常编译成功。
3. 当 Windows 提示“以管理员身份运行” DevEco Studio 时，如下图所示运行。



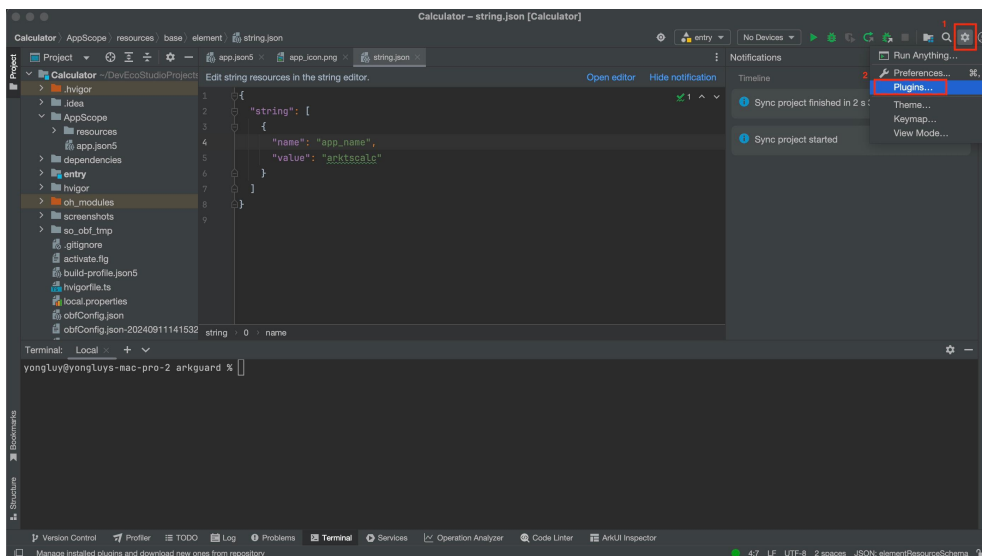
3.2. 安装与卸载

3.2.1. 安装插件客户端

梆梆安全鸿蒙 NEXT 加固客户端为鸿蒙 DevEco Studio 开发工具标准插件软件包，请按照 DevEco Studio 插件软件包安装方法执行即可。

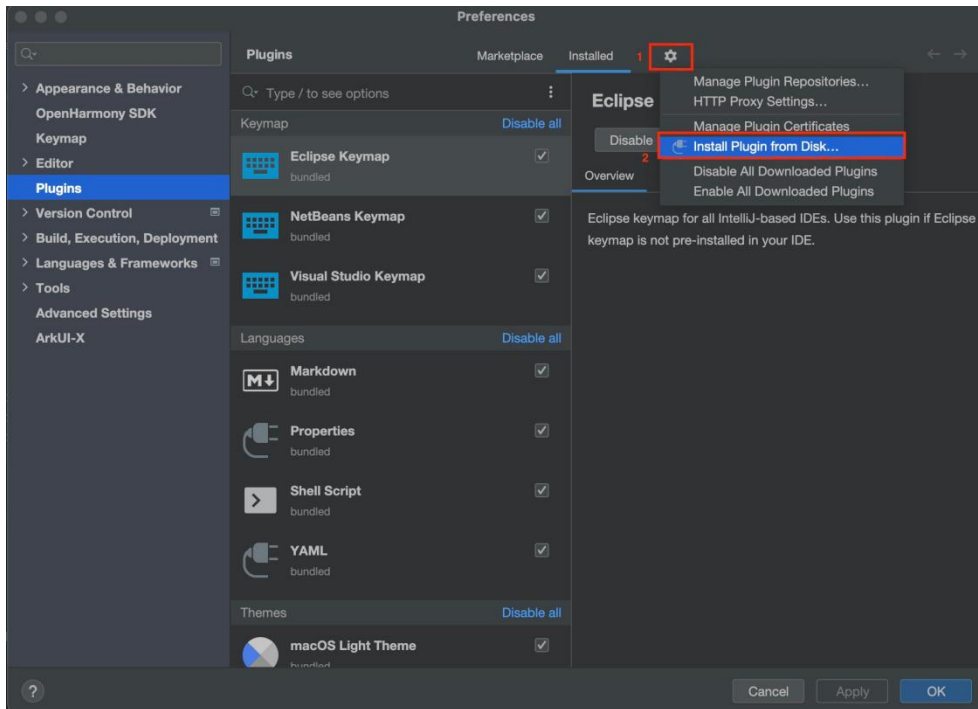
具体步骤如下：

1. 运行鸿蒙 DevEco Studio 开发工具，点击“IDE and Project Settings”按钮，然后选择“Plugins...”菜单项：

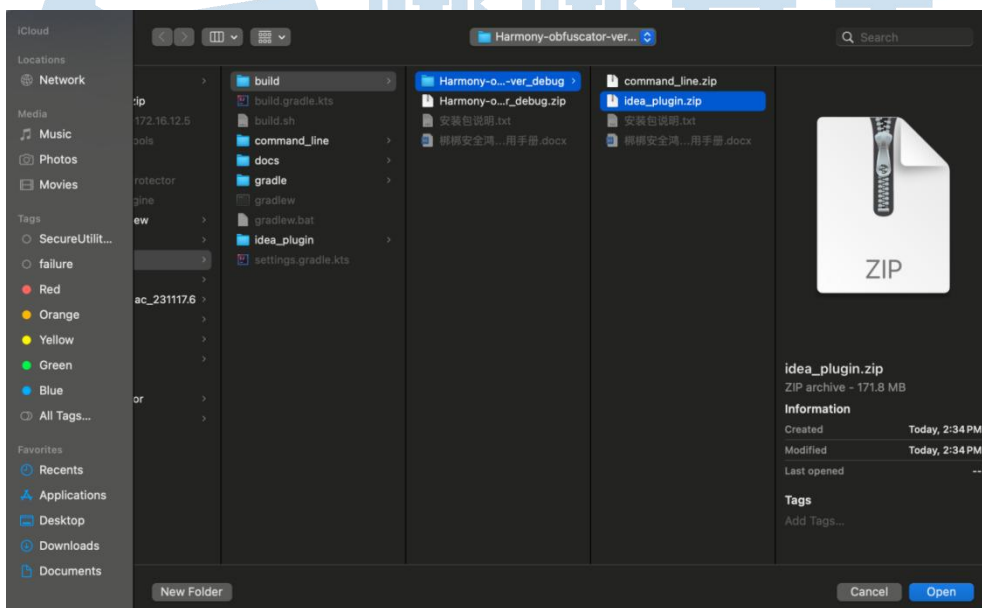


2. 在新窗口“Preferences”界面，点击“Manage Repositories...”设置按钮，在弹窗的菜单列表内选择“Install Plugin from Disk...”菜单项：



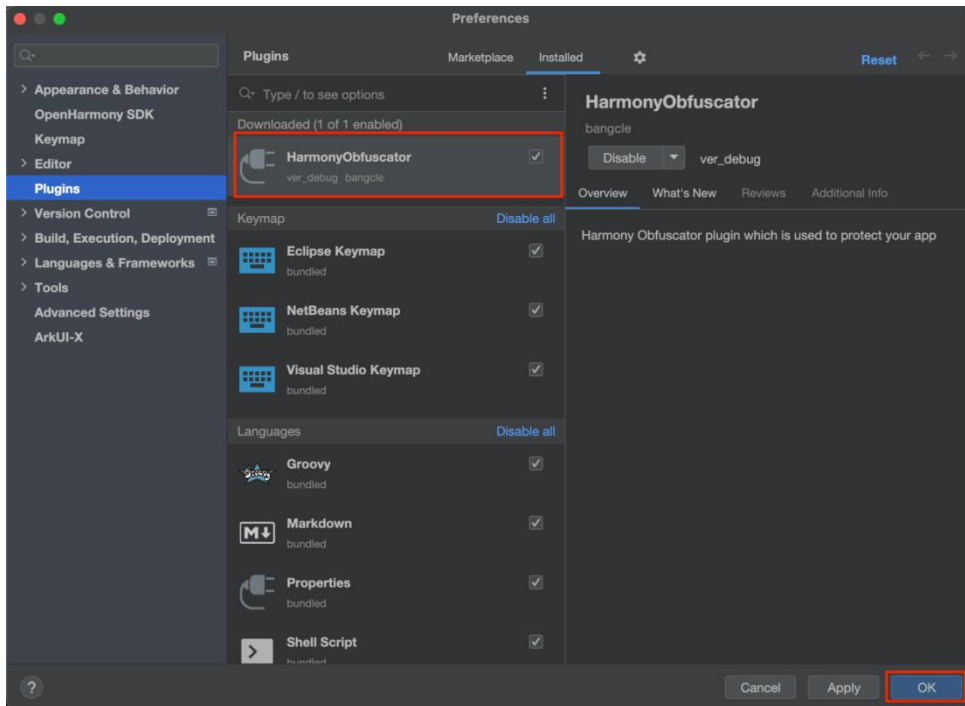


3. 找到梆梆安全鸿蒙 NEXT 加固客户端软件包 idea_plugin.zip，选择并点击” OK”按钮：

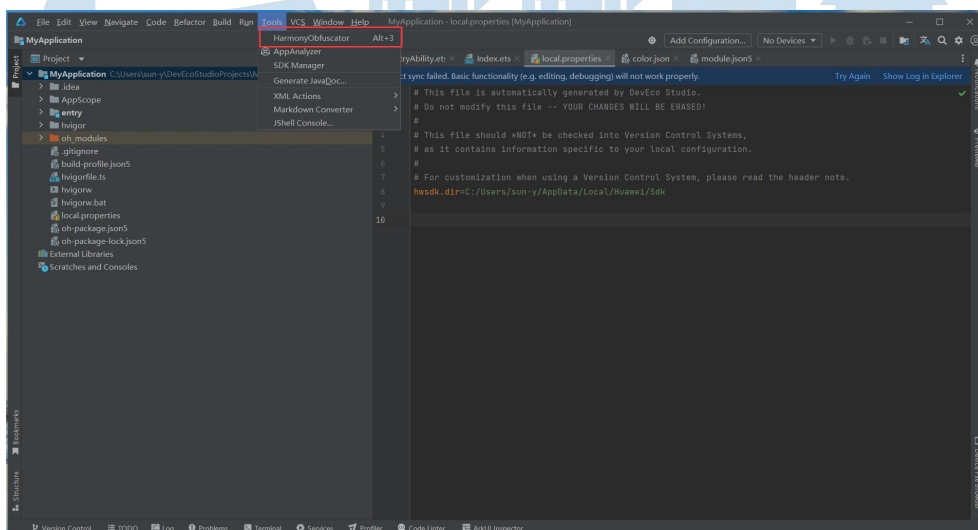


4. 在“Installed”已安全插件列表页面会显示出灰色“HarmonyObfuscator”插件，点击” OK”按钮，完成插件安装：





5. 安装完成验证：通过查看“Tools”菜单，可以查看到新增“HarmonyObfuscator”鸿蒙加固工具菜单选项：



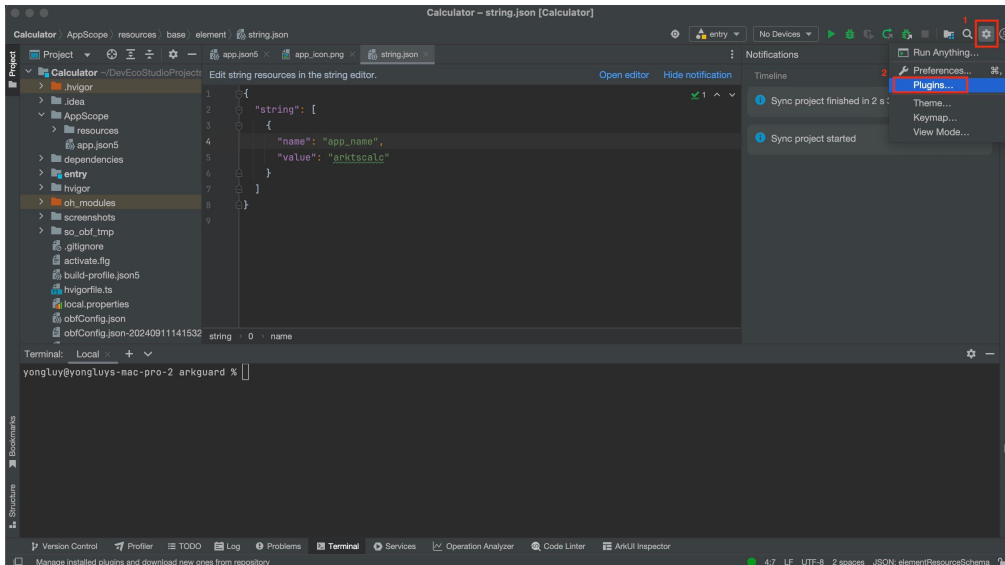
3.2.2. 卸载插件客户端

请在安装梆梆安全鸿蒙 NEXT 加固客户端新版本前，卸载掉已安装的插件软件包。

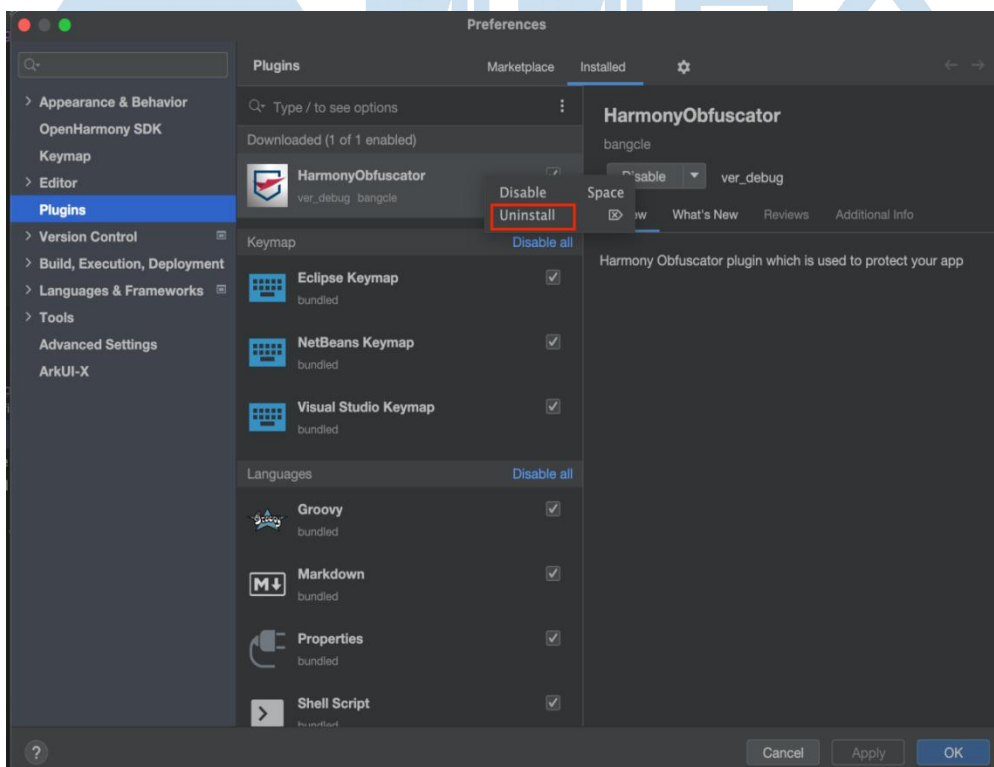
具体步骤如下：



1. 运行鸿蒙 DevEco Studio 开发工具，点击“IDE and Project Settings”功能设置菜单按钮，在弹出菜单内选择“Plugins...”菜单项：

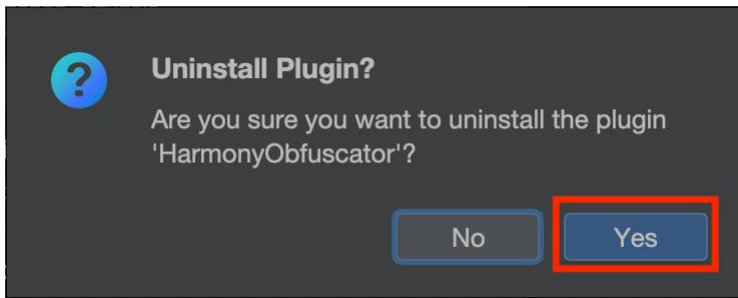


2. 在“Preferences”界面内点击“Installed”导航菜单，找到已安装的“HarmonyObfuscator”插件，并右键点击，选择“Uninstall”菜单项：

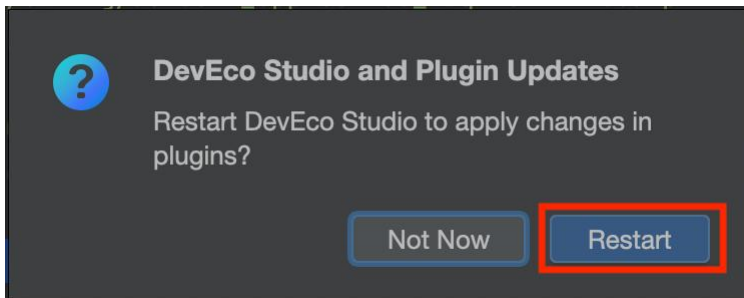


3. 点击“YES”按钮，确认卸载“HarmonyObfuscator”插件：





4. 按照提示重启 DevEco Studio，完成卸载。



3.3. 登录或离线使用

梆梆安全鸿蒙 NEXT 加固系统支持登录梆梆安全 SaaS 服务平台、客户本地部署的服务器以及离线使用。

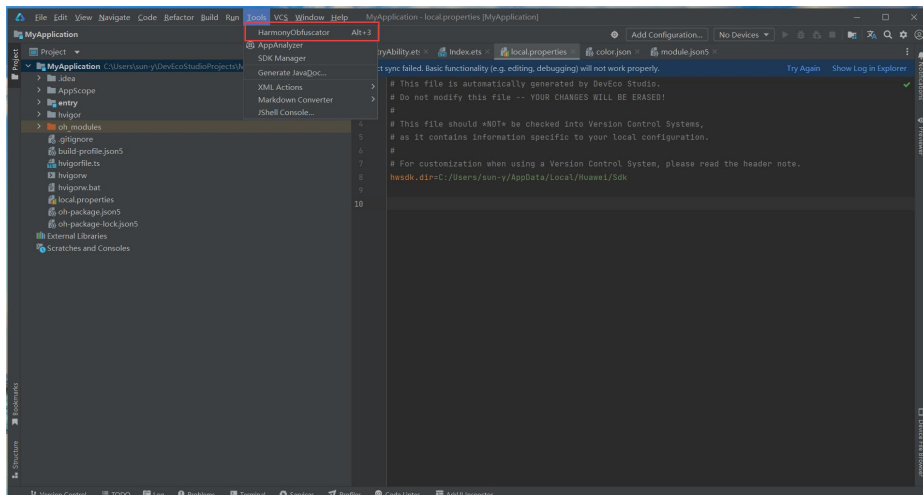
3.3.1. 登录梆梆安全 SaaS 服务平台

此登录方式仅适用于登录访问梆梆安全鸿蒙 NEXT 加固 SAAS 服务平台。

具体步骤：

1. 运行鸿蒙 DevEco Studio 开发工具；打开“Tools”菜单，点击“HarmonyObfuscator”鸿蒙加固工具菜单选项





2. 点击“设置”，配置梆梆安全 SaaS 服务器，默认地址：<https://usc.an110.com>



3. 输入用户名和密码，点击登录使用：





温馨提醒：梆梆安全鸿蒙 NEXT 应用加固的处理操作完全在客户本地电脑内完成，登录梆梆安全 SaaS 平台，不会提交源代码到梆梆安全的 SaaS 平台，无需担心代码泄露风险。

3.3.2. 登录本地部署的加固系统

此登录方式仅适用于登录访问部署在客户本地环境内的梆梆安全鸿蒙 NEXT 应用加固系统。配置和使用方式同 [3.2.1. 登录梆梆安全 SaaS 服务平台](#)，不再重复描述。只是设置页面中，需设置为本地服务器 URL 地址。服务器 URL 示例：<http://192.168.139.47:8000>

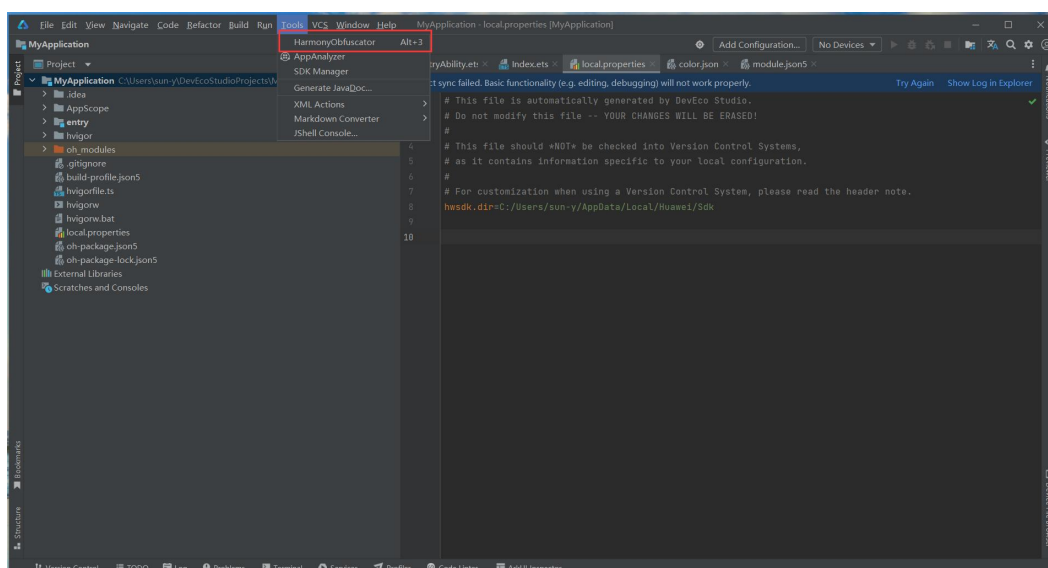


3.3.3. 离线登录

此登录方式主要适用于仅安装梆梆安全鸿蒙 NEXT 加固客户端情况下使用。

具体步骤：

1. 运行鸿蒙 DevEco Studio 开发工具，打开“Tools”菜单，点击“HarmonyObfuscator”鸿蒙加固工具菜单选项：



2. 点击“离线使用”按钮直接进入离线使用模式，无需设置、无需输入用户名和密码



3. 离线模式下，第一次使用会提示未导入 license 授权文件，是否导入。安装引导，



成功导入梆梆安全鸿蒙 APP 加固 license 后才能使用 APP 加固服务：



4. 选择合法的 license 文件后，将提示成功导入授权文件：

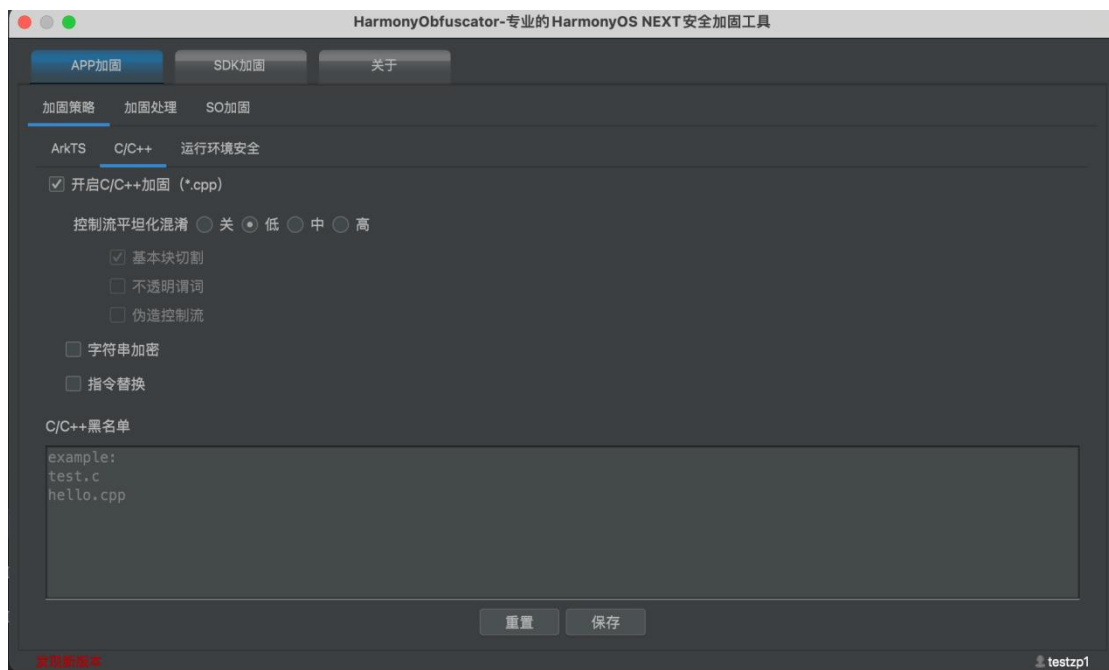


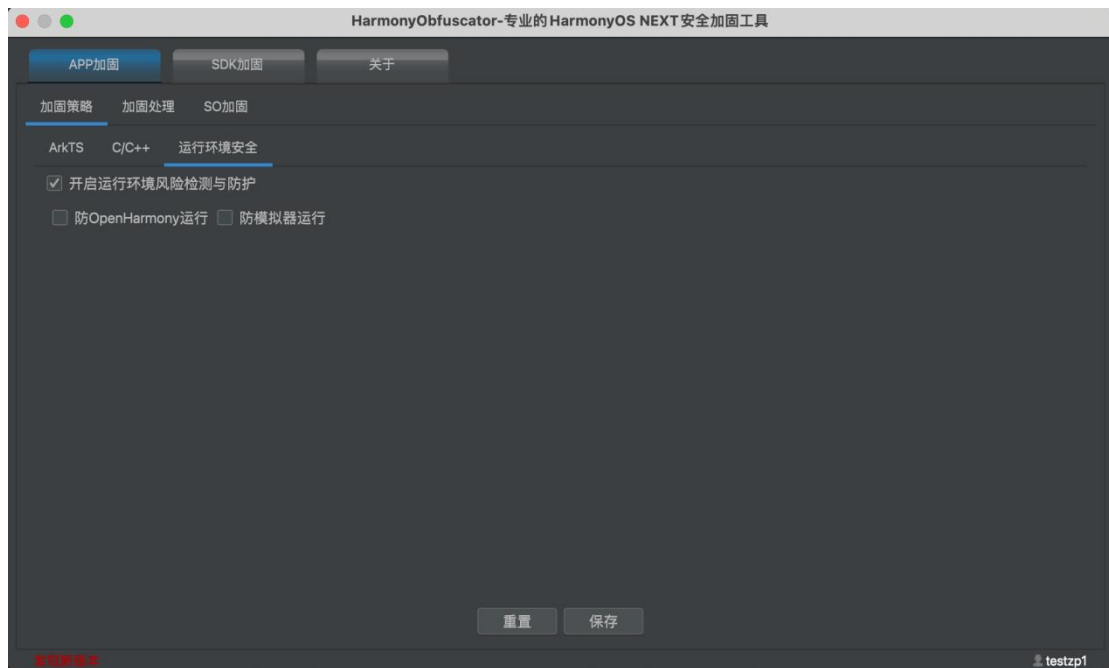
3.4. 加固策略配置

重要说明：无论是否更改加固策略，首次使用时一定登录加固工具以确保插件部署成功。

梆梆安全鸿蒙 APP 加固策略配置项说明如图，点击 APP 加固后，根据需要勾选加固策略中的各选项，最后点击保存：







3.5. 鸿蒙 NEXT 应用加固处理

梆梆安全鸿蒙 NEXT 应用加固技术，加固处理说明：

- 1、使用加固处理前，需要先保存加固策略（参见[加固策略配置](#)）；
- 2、只在编译 release 包时调用加固处理引擎，编译 debug 包不会进行加固处理（参见[基于 Build 菜单加固处理](#)）；
- 3、推荐使用在线登录梆梆安全“HarmonyObfuscator”加固系统并通过“加固处理”界面进行加固的。此时才会向鸿蒙 NEXT 安全加固服务器生成加固任务记录、上传加固策略信息（参见[鸿蒙 NEXT 应用加固服务平台使用说明](#)）。通过命令行，离线使用或 build 菜单编译并加固，不会生成加固任务记录信息。

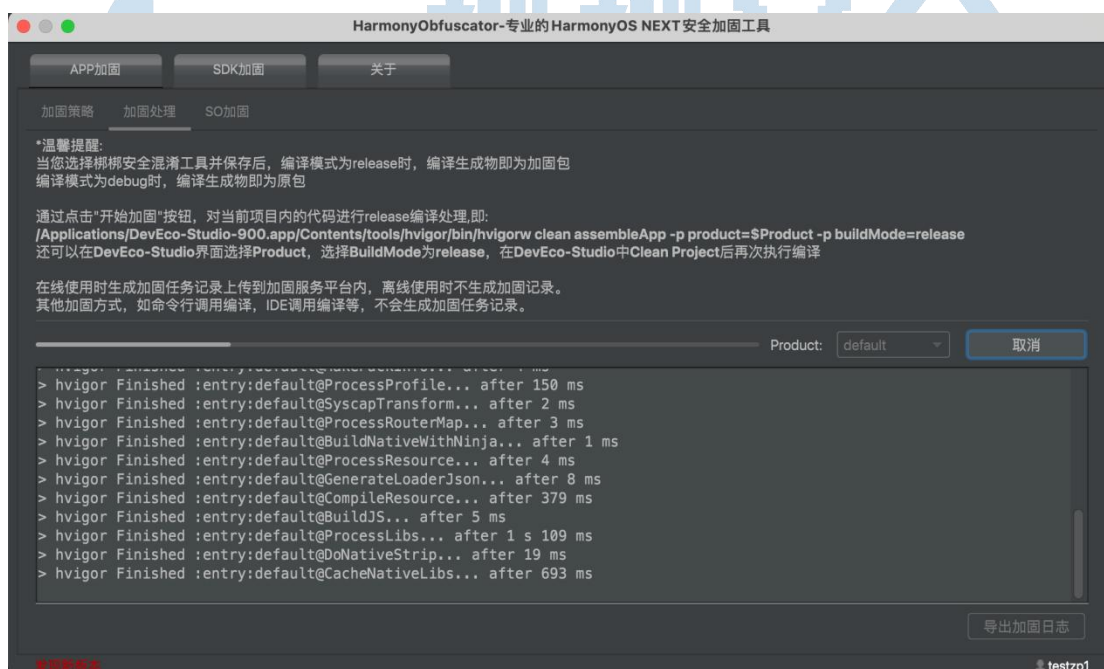
3.5.1. 基于 HarmonyObfuscator 界面加固处理

- 1、选中 APP 加固标签后，再点击“加固处理”标签，选择对应产品后，点击“开始加固”，开始执行加固处理：



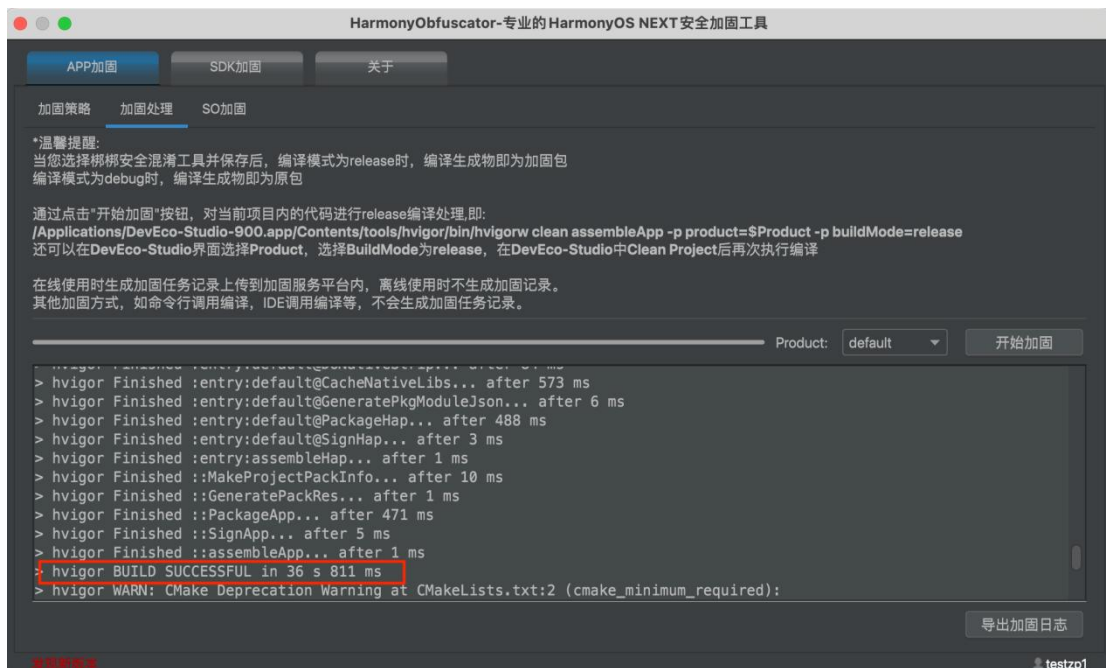


2、处理过程会显示加固处理进行和信息，可以点击“取消”按钮，终止当前加固处理：



3、加固处理完成，显示如下，点击“导出加固日志”可以保存当前加固的输出信息。

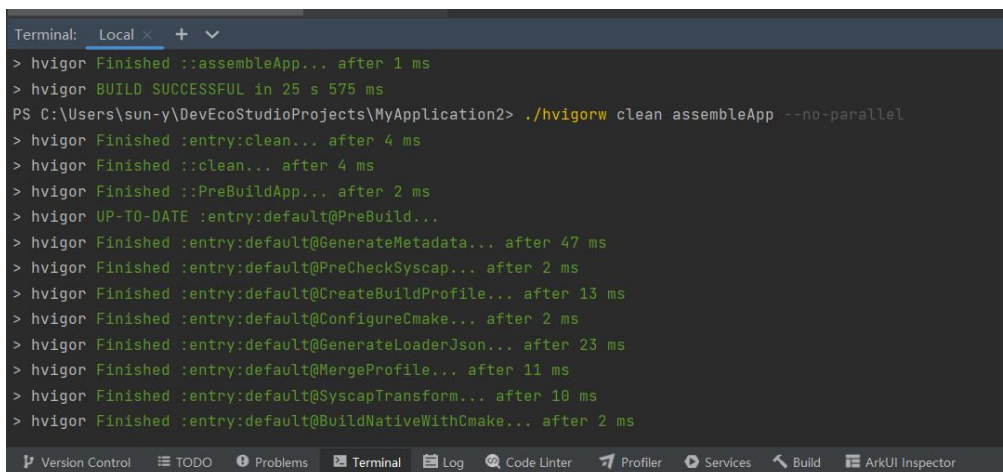




3.5.2. 基于 Terminal 命令行加固处理

- 1、配置并保存加固策略。
- 2、清除项目信息：删除工程目录下的.cxx 文件夹（参见[关于清除项目编译中间信息](#)）
- 3、在 DevEco-Studio Terminal 执行编译 release 包命令，编译过程就会知道调用梆梆安全鸿蒙 NEXT 加固引擎进行加固处理。

例如：`./hvvigorw clean assembleApp --no-parallel`



```

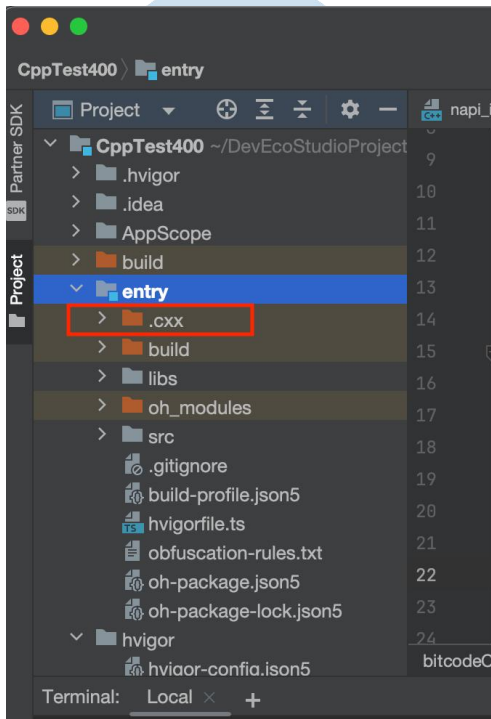
}
all function = 6; obf function = 0; mini function = 6; noreturn function = 0; obf block = 0
all function = 8; obf function = 0; mini function = 6; noreturn function = 1; obf block = 1
all function = 10; obf function = 0; mini function = 6; noreturn function = 2; obf block = 2
all function = 12; obf function = 0; mini function = 6; noreturn function = 3; obf block = 3
obf:C:\Users\sun-y\DevEcoStudioProjects\MyApplication2\entry\build\default\cache\default\default@CompileArkTS\esmodule\releas
all function = 14; obf function = 2; mini function = 10; noreturn function = 0; obf block = 4
all function = 16; obf function = 0; mini function = 10; noreturn function = 1; obf block = 4
all function = 18; obf function = 0; mini function = 10; noreturn function = 2; obf block = 6
all function = 20; obf function = 0; mini function = 10; noreturn function = 3; obf block = 8
> hvmgr Finished :entry:default@CompileArkTS... after 16 s 108 ms
> hvmgr Finished :entry:default@BuildJS... after 9 ms
> hvmgr Finished :entry:default@CacheNativeLibs... after 15 ms
> hvmgr Finished :entry:default@PackageHap... after 1 s 300 ms
> hvmgr WARN: Will skip sign 'hos_hap'. No signingConfigs profile is configured in current project.
  
```

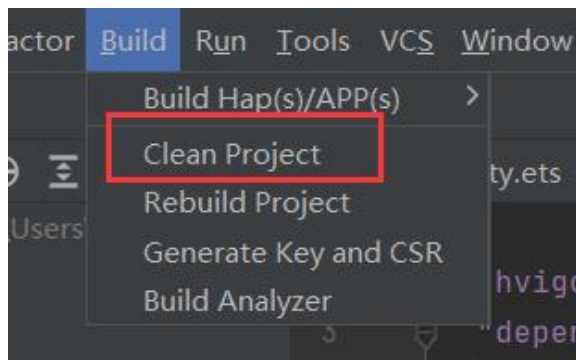
3.5.3. 基于 Build 菜单加固处理

1、配置并保存加固策略。

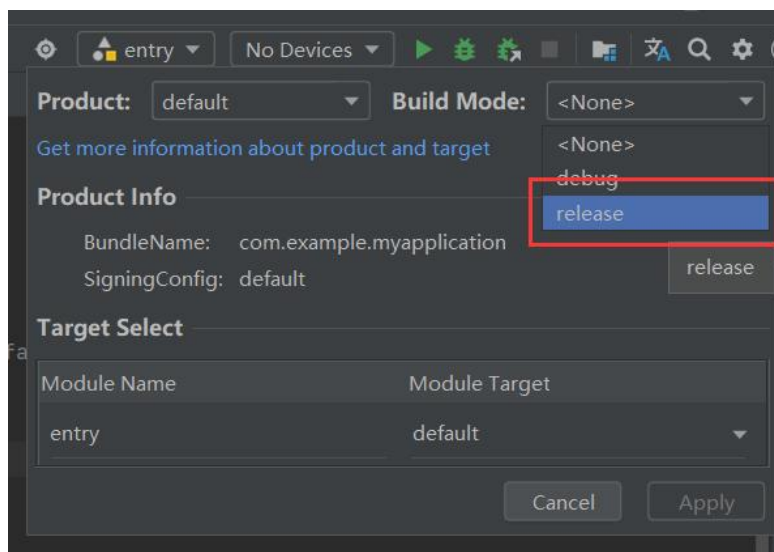
2、清除项目信息：删除工程目录中的.cxx 文件夹（参见[关于清除项目编译中间信息](#)）

，并 Clean Project。

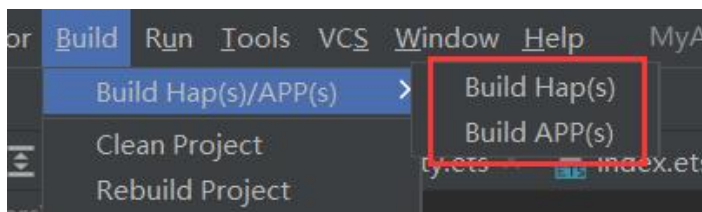




3、指定编译 release 包：



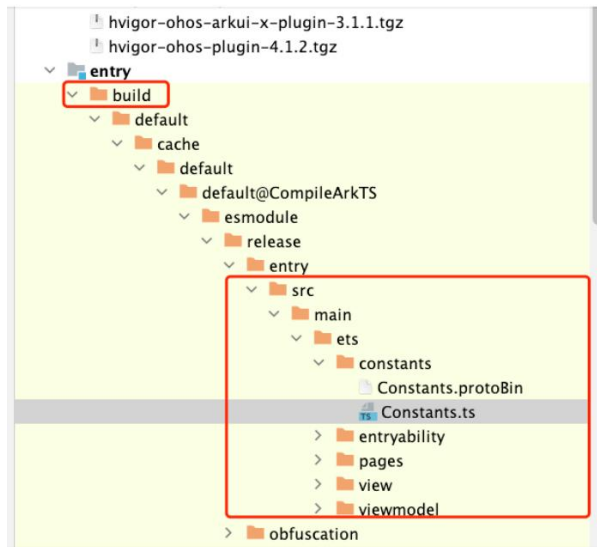
4、通过 Build 菜单编译，因编译的是 release 包，编译过程会自动调用梆梆安全鸿蒙 NEXT 加固引擎进行加固处理：



3.5.4. 查看加固效果

ArkTS 加固后，可以在新生成的下列路径里查看加固效果：

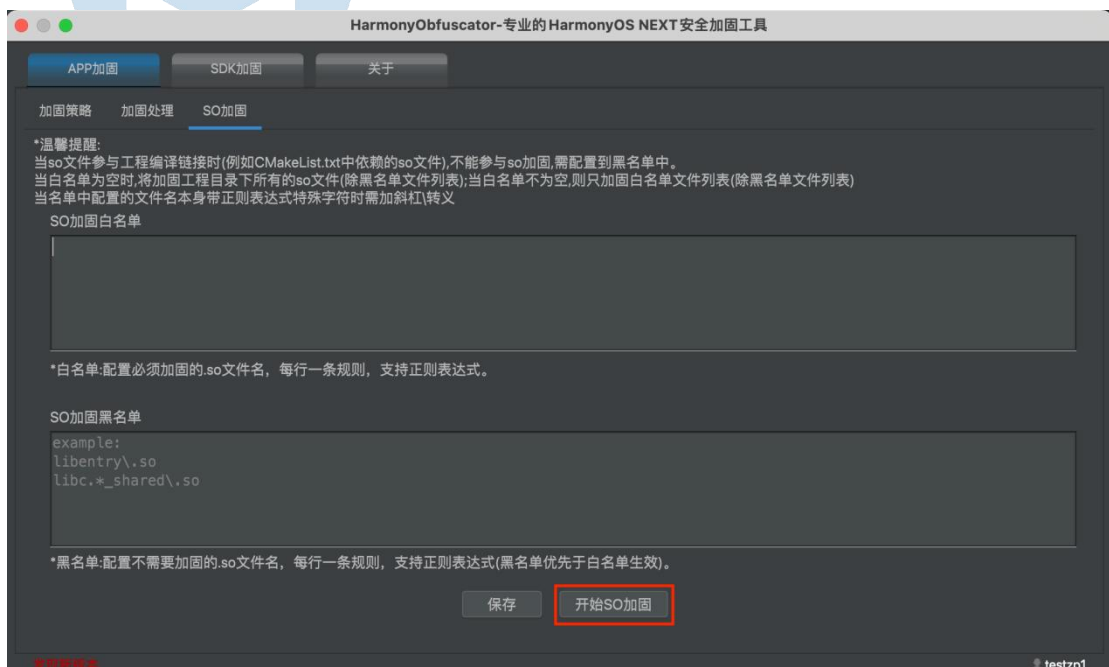




C/C++代码加固效果，可以通过反编译软件 ida 查看编译输出的文件。

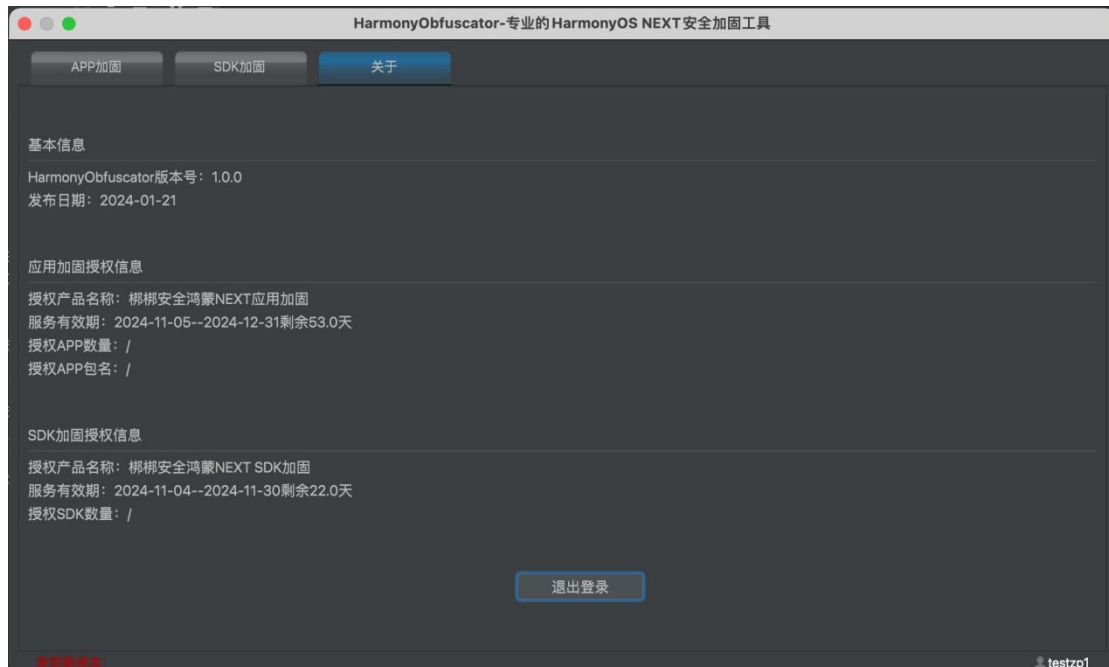
3.5.5. 工程 SO 加固（可选）

选中 APP 加固标签后，在 SO 加固 tab 中，点击“开始 SO 加固”，将对工程目录中的 *.so 文件进行加固。注意事项参见软件上的界面说明。



3.6. 关于

点击“关于”标签，可以查看如下界面：



3.6.1. 查看客户端版本

可以在“关于”页面查看梆梆安全鸿蒙 NEXT 加固客户端版本和发布时间。

3.6.2. 查看授权信息

可以在“关于”页面查看当前 APP 加固授权范围，包括可用时间、授权 APP 数量、授权包名等信息。

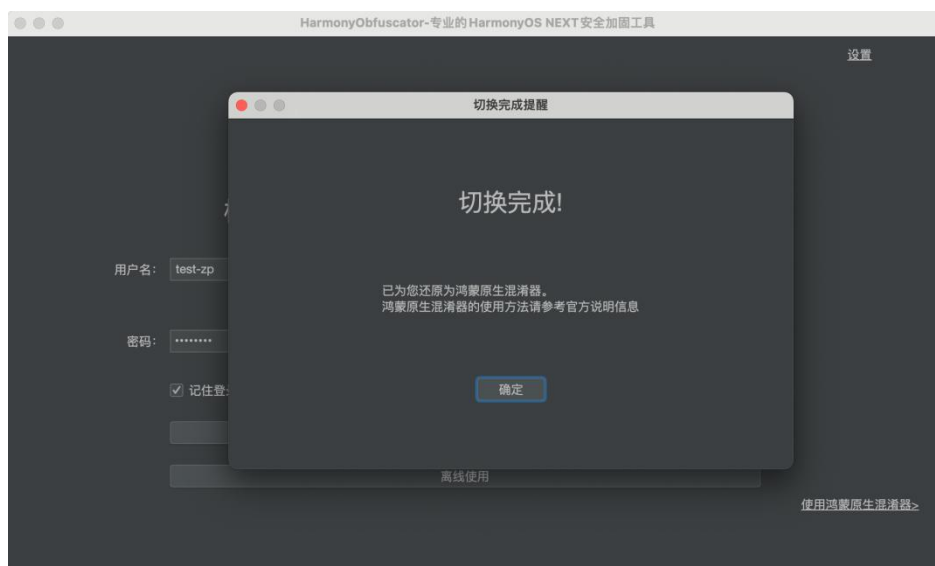
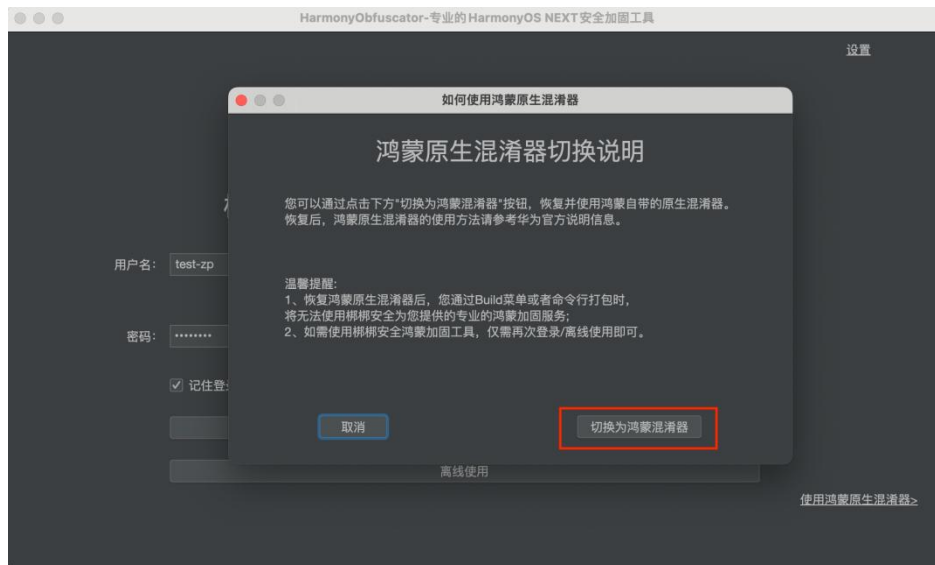
3.6.3. 退出登录

可以通过“关于”页面的退出登录按钮，退出当前登录的用户。

3.7. 使用鸿蒙原生混淆器

在登录界面点击**使用鸿蒙原生混淆器**，根据指引操作，可以恢复 DevEco-Studio 编译环境（一般在排查混淆引擎问题时使用）：





如果想再次使用 Harmony-obfuscator 插件提供的梆梆安全鸿蒙混淆器，只需**登录**或者**离线使用**，此时会再次部署梆梆安全加固工具相关文件，如常使用即可。

4. 加固命令行软件 command_line

本节应用场景为基于华为鸿蒙 commandline-tools 在 Linux/Mac/Windows 构建流水线自动化加固打包。用户通过梆梆应用加固命令行软件 command_line 完成在华为鸿蒙开发工具 commandline-tools 上的混淆部署，根据用户配置的加固策略，就会在每次编译时自动加固处理。

4.1. 前置条件

梆梆安全鸿蒙 NEX 应用加固命令行软件需依赖华为鸿蒙的命令行开发编译环境使用，请确保：

1、在鸿蒙官网下载以下鸿蒙命令行工具 zip。准备 JDK17 及以上版本（鸿蒙命令行的使用前提）

linux 环境: commandline-tools-linux-x64-XXX.zip

mac x86_64 环境: commandline-tools-mac-x64-XXX.zip

mac arm 环境: commandline-tools-mac-arm64-XXX.zip

Windows 环境: commandline-tools-windows-x64-XXX.zip

2、解压鸿蒙命令行工具 zip，定义其所在目录为<dev-commandline-tools>，在工程目录下，执行以下命令可成功构建工程:

mac/linux 环境: <dev-commandline-tools>/bin/hvigorw clean assembleApp



windows 环境: <dev-commandline-tools>\bin\hvigorw.bat clean assembleApp

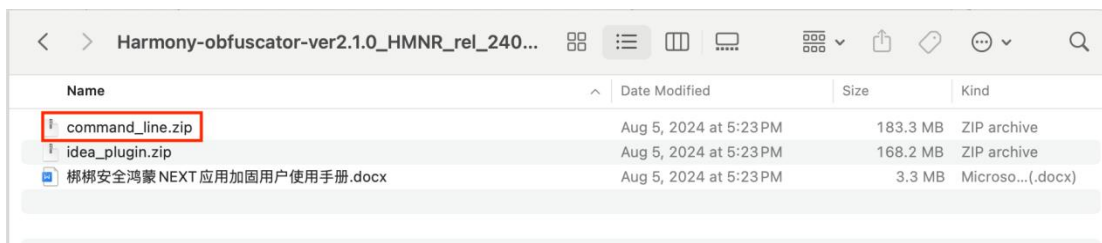
3、需使用 JDK17 及以上版本（鸿蒙命令行的使用前提）。

4.2. Linux/Mac/Windows 构建流水线自动化加固

准备梆梆安全鸿蒙 NEXT 加固命令行工具 command_line.zip，进行

HarmonyObfuscator 的安装和卸载。命令行工具命名示例：

[Harmony-obfuscator-ver2.2.0_HMNR_rel_YYMMDD.x.zip/command_line.zip](#)



先解压梆梆 command_line.zip，执行 bin 目录下可执行文件 command_line 或 command_line.bat 部署或者移除应用加固引擎；然后在工程中配置加固策略文件后；最后执行应用打包编译命令即可完成 APP 加固。离线方式安装的简要操作演示如下：



```
yongliuy@yongliuys-mac-pro-2 Downloads % export JAVA_HOME=/Applications/jdk-17.jdk/Contents/Home 第一步
yongliuy@yongliuys-mac-pro-2 Downloads % unzip Harmony-obfuscator-ver2.2.0_HMNR_rel_241120.1.zip 第二步
Archive:  Harmony-obfuscator-ver2.2.0_HMNR_rel_241120.1.zip
  inflating: updatePlugin.xml
  inflating: ??????????NEXT SDK?????????V2.0.0-?????????使?????????.docx
  inflating: ??????????NEXT SDK?????????V2.2.0-?????????使?????????.docx
  extracting: ideas_plugin.zip
  extracting: command_line.zip
yongliuy@yongliuys-mac-pro-2 Downloads % unzip command_line.zip 第三步
Archive:  command_line.zip
  creating: command_line/
  creating: command_line/lib/
  inflating: command_line/lib/command_line-all.jar
  creating: command_line/bin/
  inflating: command_line/bin/command_line.bat
  inflating: command_line/bin/command_line
yongliuy@yongliuys-mac-pro-2 Downloads % ./command_line/bin/command_line -i ~/tmp/license.lic -d /Applications/DevEco-Studio-900.app 第四步
[-i, /Users/yongliuy/tmp/license.lic, -d, /Applications/DevEco-Studio-900.app]
mac os x
2024-12-02 11:13:05.210 [INFO] sysDefaultLanguage:en_CN
2024-12-02 11:13:05.243 [INFO] invoke install/uninstall in command mode!
2024-12-02 11:13:05.678 [INFO] error:hmsdk_protect:No license or invalid license.
2024-12-02 11:13:05.712 [INFO] START ==>>> find arkguard path in sdk <== START
2024-12-02 11:13:05.724 [INFO] obfToolsPath = /Applications/DevEco-Studio-900.app/Contents/sdk/default/openharmony/ets/build-tools/ets-loader/n
ode_modules/arkguard
2024-12-02 11:13:05.725 [INFO] END ==>>> find arkguard path in sdk <== END
2024-12-02 11:13:05.772 [INFO] copy new directory /Applications/DevEco-Studio-900.app/Contents/sdk/default/openharmony/ets/build-tools/ets-load
er/node_modules/arkguard
2024-12-02 11:13:05.831 [INFO] START ==>>> find clang path in sdk <== START
2024-12-02 11:13:05.837 [INFO] END ==>>> find clang path in sdk <== END
2024-12-02 11:13:07.347 [INFO] xattr -d com.apple.quarantine /Applications/DevEco-Studio-900.app/Contents/sdk/default/openharmony/native/llvm/b
in/clang
2024-12-02 11:13:07.348 [INFO] copy new /Applications/DevEco-Studio-900.app/Contents/sdk/default/openharmony/native/llvm/bin/clang
2024-12-02 11:13:08.898 [INFO] xattr -d com.apple.quarantine /Applications/DevEco-Studio-900.app/Contents/sdk/default/openharmony/native/llvm/b
in/clang++
2024-12-02 11:13:08.900 [INFO] copy new /Applications/DevEco-Studio-900.app/Contents/sdk/default/openharmony/native/llvm/bin/clang++
2024-12-02 11:13:10.462 [INFO] xattr -d com.apple.quarantine /Applications/DevEco-Studio-900.app/Contents/sdk/default/openharmony/native/llvm/b
in/clang-cl
2024-12-02 11:13:10.464 [INFO] copy new /Applications/DevEco-Studio-900.app/Contents/sdk/default/openharmony/native/llvm/bin/clang-cl
2024-12-02 11:13:11.997 [INFO] xattr -d com.apple.quarantine /Applications/DevEco-Studio-900.app/Contents/sdk/default/openharmony/native/llvm/b
in/clang-ls
2024-12-02 11:13:11.999 [INFO] copy new /Applications/DevEco-Studio-900.app/Contents/sdk/default/openharmony/native/llvm/bin/clang-ls
2024-12-02 11:13:13.474 [INFO] xattr -d com.apple.quarantine /Applications/DevEco-Studio-900.app/Contents/sdk/default/openharmony/native/llvm/b
in/clang-cpp
2024-12-02 11:13:13.475 [INFO] copy new /Applications/DevEco-Studio-900.app/Contents/sdk/default/openharmony/native/llvm/bin/clang-cpp
2024-12-02 11:13:13.536 [INFO] copy new /universal_harmony/mac/universal_harmony
2024-12-02 11:13:13.537 [INFO] Activation.invoke: Activation.needDeploy = false
Deploy successfully! OK
```

详细说明如下：

1、部署梆梆安全鸿蒙 NEXT 应用加固引擎：

可以选择在线和离线两者其一方式来进行部署引擎。在线部署需提供用户名密码和登录 url，离线安装则需提供 license 文件。

a) 在线安装命令

linux/mac 环境：bin/command_line -d <dev-commandline-tools> -u <usrName> -p <passWord> -l <urlLink>

windows 环境：bin\command_line.bat -d <dev-commandline-tools> -u <usrName> -p <passWord> -l <urlLink>

例如：./bin/command_line -d ~/dev-command-line-tools -u test-zp -p 12345 -l http://172.16.38.19:8000

b) 离线安装命令

linux/mac 环境：bin/command_line -d <dev-commandline-tools> -i <licenseFile>

windows 环境：bin\command_line.bat -d <dev-commandline-tools> -i <licenseFile>



例如：`./bin/command_line -d ~/dev-command-line-tools -i license-梆梆安全-鸿蒙 NEXT 应用加固-无限制.lic`

2、移除梆梆安全鸿蒙 NEXT 应用加固引擎：

当不再使用梆梆安全鸿蒙 NEXT 应用加固引擎时，可以执行以下命令去除引擎文件：

linux/mac 环境：`bin/command_line -d <dev-commandline-tools> -r`

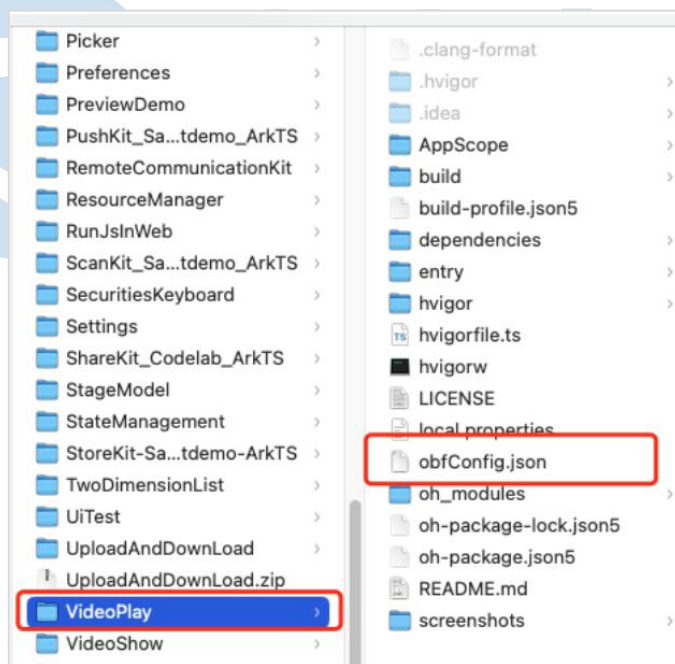
windows 环境：`bin\command_line.bat -d <dev-commandline-tools> -r`

例如：`./bin/command_line -d ~/dev-command-line-tools -r`

3、配置鸿蒙 NEXT 应用加固策略并完成加固：

1) 将策略配置文件 `obfConfig.json` 手动拷贝到项目源工程目录下：

示例：工程目录名称为“VideoPlay”，拷贝 `obfConfig.json` 文件后如下图所示：



2) 配置/修改 `obfConfig.json` 内的加固策略：

`obfConfig.json` 文件附件如下（当前附件内已经是配置好的加固策略，内容如下，无需修改，可直接使用。）：





obfConfig.json

obfConfig.json 内容解释如下：

```
{
  "mEnable": true, //总开关
  "mNameObfuscation": {
    "mEnable": true, //变量名混淆
  },
  "mStringArray": true, //字符串加密
  "mReservedStrings": [ ], //字符串过滤: [过滤内容, .*过滤.*]
  "mReservedFiles": [ ], //文件名过滤: [文件名过滤, .*过滤.*]
  "mControlFlow": true, //控制流混淆
  "mCompact": true, //代码压缩
  "mEnableSourceMap": false, //可忽略
  "mNumberExpression": true, //常量混淆
  "mLogicalExpression": true, //表达式混淆
  "mBogusControlFlow": 0.6, //伪造控制流: 0 表示关闭; 0.6 表示 60%的比率; 1 表示全做
  "bitcodeObf": {
    "enable": true, //C/C++ bitcode 混淆总开关
    "strength": 3, //强度: 0: 关, 1: 低, 2: 中, 3: 高
    "splitBB": 3, //切割基本块数量: 1: 不切割, 2~10: 所有函数都做
    "bcf": 30, //虚假控制流比率
    "fla": 30, //控制流平坦化比率
    "z_opaque": 30, //不透明谓词比率
    "stringCry": 30, //字符串加固比率
    "sub": 30, //指令替换分比率
  }
}
```




```
"mReservedFiles": [ ] //文件名过滤: [文件名匹配]
},
"soObf": {
    "enable": true, //可忽略的开关, 当用户进行 SO 加固时才触发处理
    "whiteFileList": [ ], //文件名过滤: [文件名过滤, .*过滤.*]
    "blackFileList": [ ] //文件名过滤: [文件名过滤, .*过滤.*]
},
"mSecSdk": {
    "mEnable": true, //开启运行环境风险检测与防护总开关
    "mAntiOpenHarmony": true, //防 OpenHarmony 运行
    "mAntiSimulator": true //防模拟器运行
}
}
```

3) 按照原有的鸿蒙 NEXT 应用自动化命令执行即可自动进行加固处理。

例如, 在工程目录下执行如下操作:

删除工程目录下的.cxx 文件夹 (参见[关于清除项目编译中间信息](#))

linux/mac 环境: <dev-commandline-tools>/bin/hvigorw clean assembleApp

Windows 环境: <dev-commandline-tools>\bin\hvigorw.bat clean assembleApp

4) 对于 so 加固命令行使用

对项目源工程中所有 so 文件进行 so 加固

linux/mac 环境: ./bin/command_line -s <projectDir>

windows 环境: .\bin\command_line.bat -s <projectDir>

例如: ./bin/command_line -s ~/CppTest400



附件：常见问题和解决方案

● 关于清除项目编译中间信息

因为经过编译的代码如果没有代码变化，并不会主动再次执行编译，而应用加固是在编译过程中完成加固，因此需要执行以下操作在确保执行完整的加固流程：

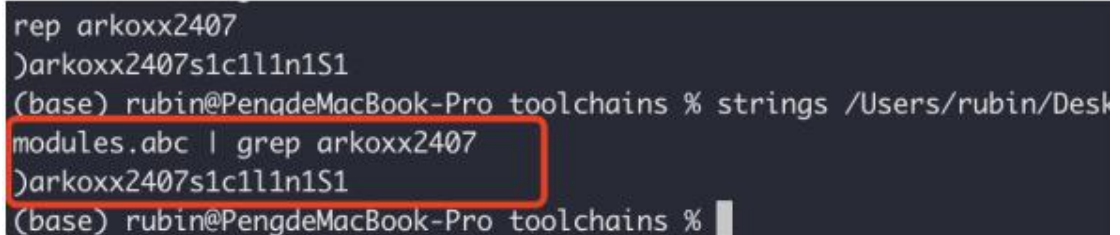
- 1、删除工程目录中的.cxx 文件夹，以确保能触发 C/C++代码重现编译并同时完成加固
- 2、通过菜单 Clean project 或者执行 `hvigorw clean` 命令可以删除其他类型代码编译结果，以重新触发编译并同时完成加固。

● 如何通过 hap文件查看加固策略

1) 查看 Arkts 加固特征。

在解压后的 hap 包中找到 abc 文件,在 Mac 终端中,执行命令 `strings XXX.abc | grep arkox2407` (Windows 可以下载 Strings 命令,在 PowerShell 中执行 `.\strings.exe XXX.abc | Select-String arkox2407`)。

如图所示：执行命令后输出 `arkox2407s1c111n1S1`，表示开启了字符串加密，控制流混淆，表达式混淆，常量混淆和运行环境保护。



```
rep arkox2407
)arkox2407s1c111n1S1
(base) rubin@PenadeMacBook-Pro toolchains % strings /Users/rubin/Desktop/modules.abc | grep arkox2407
)arkox2407s1c111n1S1
(base) rubin@PengdeMacBook-Pro toolchains %
```

输出字段解释：

arkox2407：表示 Arkts 加固特征的字符标志

s：字符串加密，1 为开启，0 为关闭



c: 控制流混淆, 1 为开启, 0 为关闭

l: 表达式混淆, 1 为开启, 0 为关闭

n: 常量混淆, 1 为开启, 0 为关闭

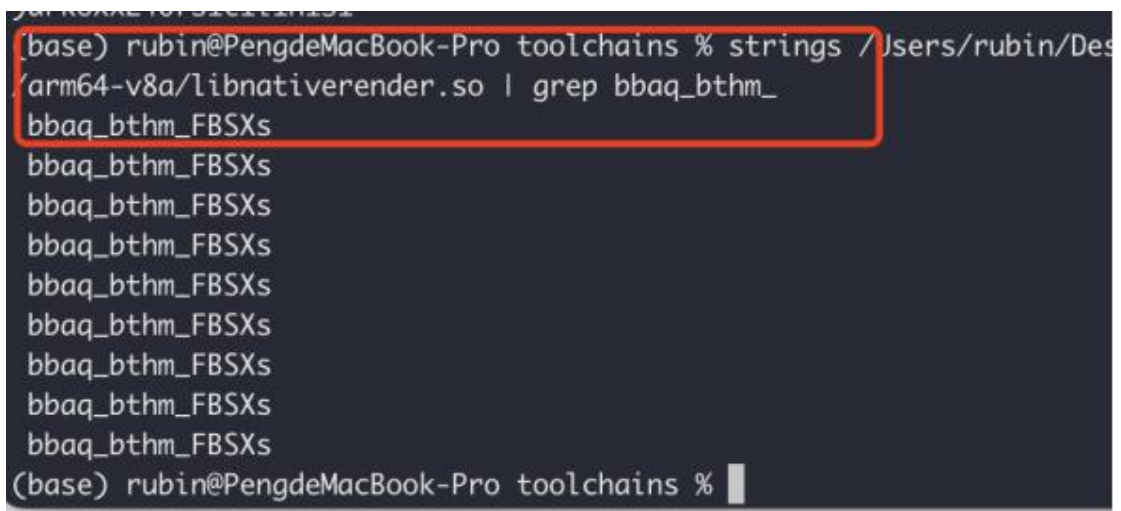
S: 运行环境保护, 1 为开启, 0 为关闭

2) 查看 c/c++加固特征。

在解压后的 hap 包中找到 C/C++编译生成的 so 文件, 在 Mac 终端中执行命令:

`strings XXX.so | grep bbaq_bthm` (Windows 可以下载 Strings 命令, 在 PowerShell 中执行 `.\strings.exe XXX.so | Select-String bbaq_bthm`) 。

如图所示: 执行命令后输出 `bbaq_bthm_FBSXs`, 表示开启了控制流平坦化、伪造控制流、字符串加密、基本块切割和指令替换。



```
(base) rubin@PengdeMacBook-Pro toolchains % strings /Users/rubin/Desktop/arm64-v8a/libnativender.so | grep bbaq_bthm_
bbaq_bthm_FBSXs
bbaq_bthm_FBSXs
bbaq_bthm_FBSXs
bbaq_bthm_FBSXs
bbaq_bthm_FBSXs
bbaq_bthm_FBSXs
bbaq_bthm_FBSXs
bbaq_bthm_FBSXs
bbaq_bthm_FBSXs
bbaq_bthm_FBSXs
(base) rubin@PengdeMacBook-Pro toolchains %
```

输出字段解释:

`bbaq_bthm_`: 表示 c/c++加固特征的字符标志

B: 伪造控制流, 出现为开启, 不出现为关闭

S: 字符串加密, 出现为开启, 不出现为关闭

X: 基本块切割, 出现为开启, 不出现为关闭

s: 指令替换, 出现为开启, 不出现为关闭

