目录

iOS逆向开发：静态分析

- 最新版本：`v1.7.5`
- 更新时间：`20250101`

简介

介绍iOS逆向开发期间的静态分析。先是静态分析的概览；接着是从ipa中找到要分析的二进制文件；再是静态分析主要涉及的内容，包括用Mach-O工具，用strings、nm、otool等导出字符串资源，用class-dump导出ObjC头文件，用IDA、Hopper等分析代码逻辑等等；以及相关工具，比如radare2以及GUI版Cutter等；然后给出一些相关实例；最后再整理一些相关的经验心得。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

HonKit源码

- crifan/ios_re_static_analysis: iOS逆向开发：静态分析

如何使用此HonKit源码去生成发布为电子书

详见：crifan/honkit_template: demo how to use crifan honkit template and demo

在线浏览

- iOS逆向开发：静态分析 book.crifan.org
- iOS逆向开发：静态分析 crifan.github.io

离线下载阅读

- iOS逆向开发：静态分析 PDF
- iOS逆向开发：静态分析 ePub
- iOS逆向开发：静态分析 Mobi

版权和用途说明

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

其他

作者的其他电子书

本人 `crifan` 还写了其他 `150+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

关于作者

关于作者更多介绍，详见：

关于CrifanLi李茂 – 在路上

静态分析概览

iOS逆向过程中，在砸壳出ipa后，就是：`静态分析`。

什么是静态分析

iOS逆向的目的，往往要搞清楚，app底层的某些功能和逻辑的具体实现机制和原理。

而搞懂底层逻辑，从过程角度来说，主要分：

- 静态分析：不运行程序的前提下，利用工具和手段，搞懂程序逻辑
- 动态调试：运行程序的前提下，动态运行期间，研究和调试程序的逻辑

从ipa中找到二进制

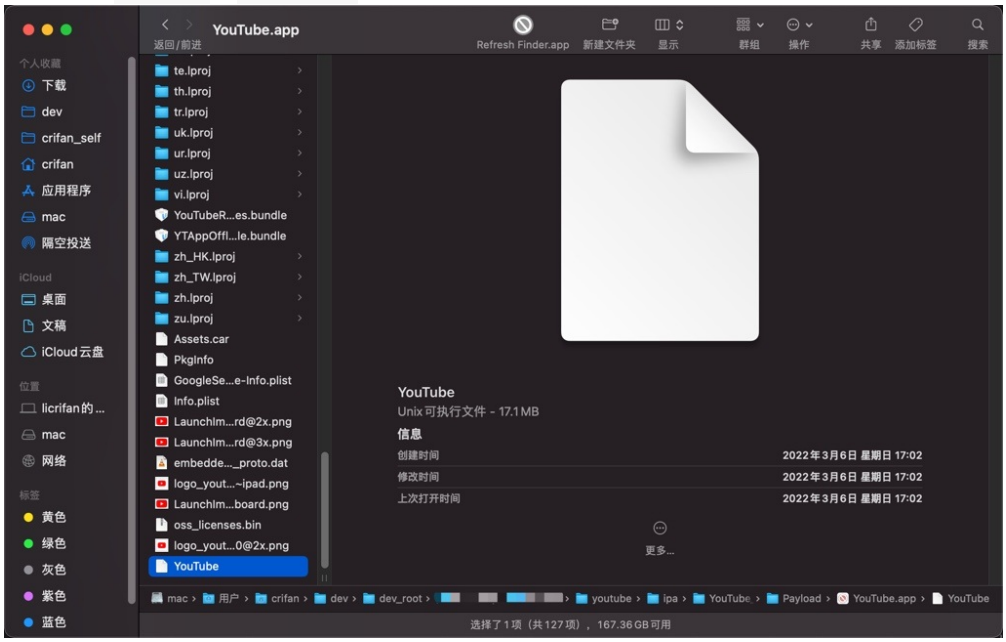
TODO：

- 【未解决】静态分析抖音二进制寻找越狱检测手段
- 【已解决】如何从脱壳后的抖音的IPA文件中得到二进制文件
- 相关
  - 【已解决】越狱iPhone中抖音app的安装目录安装位置
  - 【已解决】研究iOS中app的目录的UUID类的值和app名称如何映射

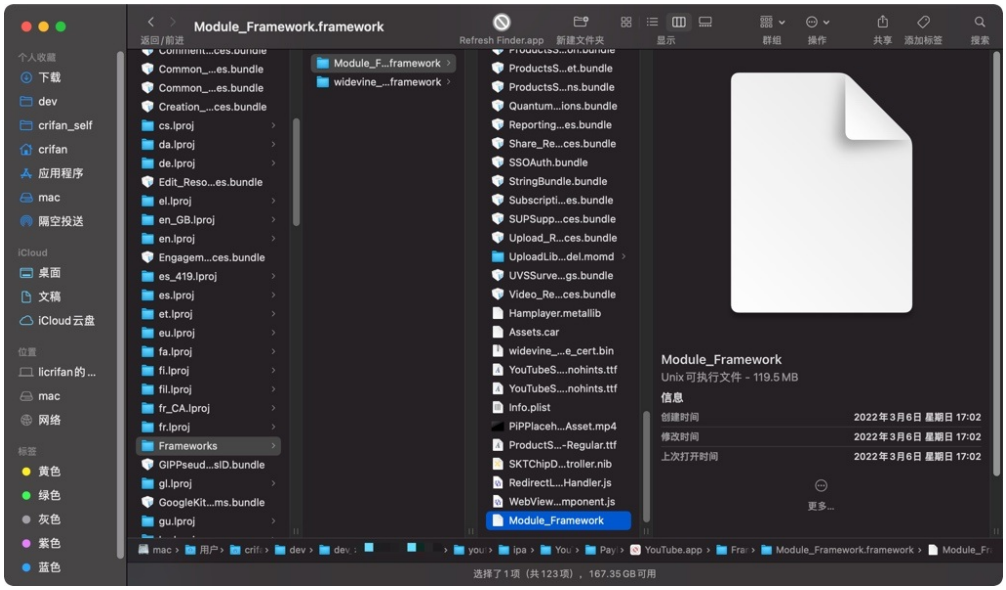
作为iOS逆向的静态分析，其输入文件是iOS的app的二进制文件。

对应的就是，在前一步，从app砸壳得到的ipa文件中，找到对应的二进制文件，用于后续的静态分析。

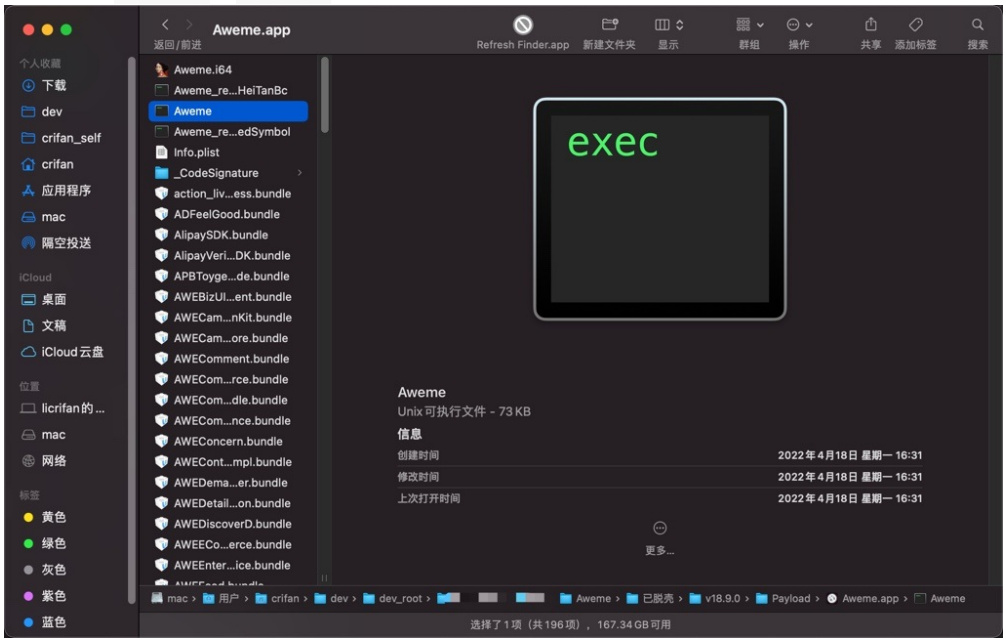
比如：

- YouTube
  - v17.08.2
    - ipa解压后得到：`YouTube.app`
      - 入口二进制：`17MB+ 的 YouTube`
      - 

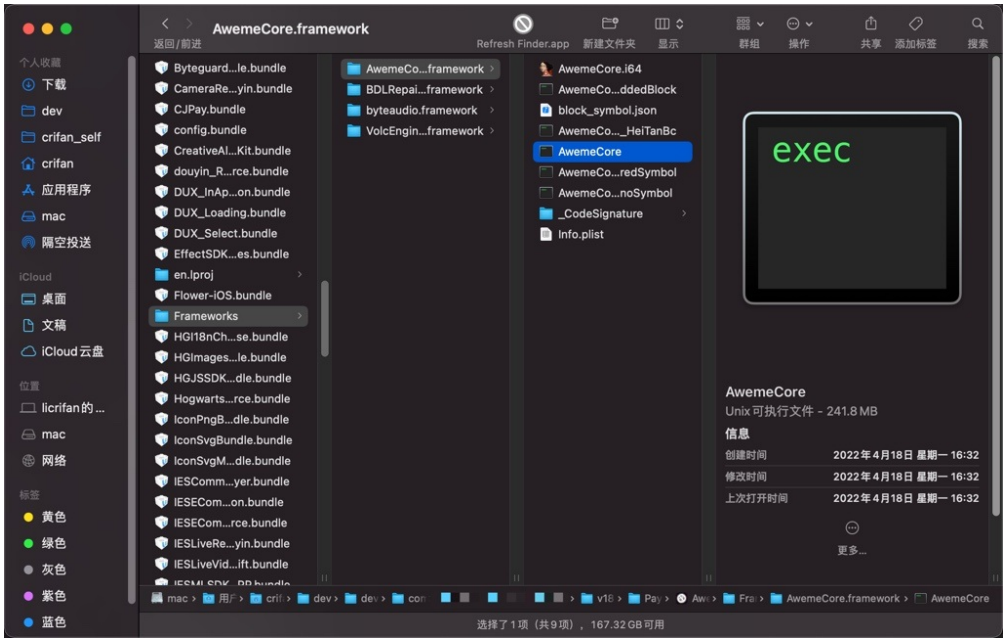
      - 核心二进制：`100MB+ 的 Frameworks/Module_Framework.framework/Module_Framework`



- 抖音
  - v18.9.0
    - ipa解压后得到：`Aweme.app`
      - 入口二进制：`70KB+ 的 Aweme`



      - 核心二进制：`240MB+ 的 Frameworks/AwemeCore.framework/AwemeCore`

- 

静态分析内容

iOS逆向的静态分析，主要针对的是：

- iOS的二进制格式：`Mach-O`
  - `Mach-O` 工具
    - 概述
      - 查看解析Mach-O
        - `MachOView`
        - `rabin2`
        - `jtool2`
        - `otool`
      - FAT瘦身
        - `lipo`
    - 详解
      - 独立子教程
        - 可执行文件格式：Mach-O
  - 其他处理
    - 导出字符串等资源：用于后续研究搜索函数、类等用途
      - `nm`
      - `strings`
    - 处理签名：重新签名避免运行崩溃
      - `ldid`
      - `codesign`
    - 导出头文件：研究类的具体函数和属性
      - `class-dump`
    - 分析代码逻辑：分析研究代码实现逻辑和其他各种细节
      - `IDA`
      - `Hopper`

导出字符串资源

iOS逆向期间，对于iOS的app，即 `Mach-O` 二进制，的静态分析，往往涉及到：从二进制中导出字符串等信息和其他资源，供后续研究和分析。

常用的导出字符串等资源的工具有：

- `strings`
- `nm`
- `Mach-O` 的工具
  - `rabin2`
  - `jtool2`
  - `otool`
- entitlement和codesign的工具
  - `ldid` / `ldid2`
  - `codesign`

下面详细解释如何操作：

通用成套做法

- 典型的**成套的做法** = `nm` + `strings` + `otool` + `jtool2` + `rabin2` + `ldid2` / `codesign`

自动处理

用下面的自动化脚本，一键自动处理：

- `exportMacho_StrResInfo.sh`

```
#!/bin/bash
# Function: Export/Extract single Mach-O file string and resources related info
# Author: Crifan Li
# Update: 20241228
# Refer:
#  /Users/crifan/dev/dev_root/androidReverse/CloudPhone/Douyin/dy297/soLibs/common/exportElf_StrResInfo.sh
#  https://book.crifan.org/books/ios_re_static_analysis/website/analysis_content/export_str_res/common_suite/
# Usage:
#   chmod +x exportMacho_StrResInfo.sh
#   ./exportMacho_StrResInfo.sh input/dyld output

# SEPERATOR="------------------"
SEPERATOR="=================="

# outpuFileSuffix="txt"
outpuFileSuffix="coffee"

function log() {
  echo "${SEPERATOR} $1 ${SEPERATOR}"
}

function extractInputFolder(){
  curInputFile=$1
  # echo "curInputFile=${curInputFile}"
  retInputFolder="$(dirname "${curInputFile}")"
  # echo "retInputFolder=${retInputFolder}"
  # return retInputFolder
  # return $retInputFolder
  # echo ${retInputFolder}
  echo $retInputFolder
}

function extractFilenameNoSuffix(){
  curInputFile=$1
  # echo "curInputFile=${curInputFile}"
  filenameWithSuffix="$(basename "${inputFile}")"
  # echo "filenameWithSuffix=${filenameWithSuffix}"
  filenameNoSuffix=${filenameWithSuffix%.*}
  # echo "filenameNoSuffix=${filenameNoSuffix}"
  echo ${filenameNoSuffix}
```

```
}

function initOutputFolerFromInputFolder(){
  inputFolder=$1
  # echo "inputFolder=${inputFolder}"
  outputFoler=$2
  # echo "outputFoler=${outputFoler}"
  if [ -z "$outputFoler" ]
  then
    if [ -z "$inputFolder" ]
    then
      outputFoler="."
    else
      outputFoler=${inputFolder}
    fi
    # echo "outputFoler=${outputFoler}"
    echo ${outputFoler}
  else
    echo ${outputFoler}
  fi
}

inputFile=$1
echo "inputFile=${inputFile}"
outputFoler=$2
echo "outputFoler=${outputFoler}"

# inputFolder="$(dirname "${inputFile}")"
inputFolder=$(extractInputFolder $inputFile)
echo "inputFolder=${inputFolder}"

inputMachoFile=$(extractFilenameNoSuffix $inputFile)
echo "inputMachoFile=${inputMachoFile}"
origgOutputMachoFile=${inputMachoFile}
echo "origgOutputMachoFile=${origgOutputMachoFile}"
outputMachoFile=${origgOutputMachoFile// /_}
outputMachoFile=${outputMachoFile//-/_}
echo "after remove special char: hyphen, space -> outputMachoFile=${outputMachoFile}"

# if [ -z "$outputFoler" ]
# then
#   if [ -z "$inputFolder" ]
#   then
#     outputFoler="."
#   else
#     outputFoler=${inputFolder}
#   fi
#   echo "outputFoler=${outputFoler}"
# fi

outputFoler=$(initOutputFolerFromInputFolder $inputFolder $outputFoler)
echo "outputFoler=${outputFoler}"

log "Exporting info use otool"
otool -l "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_otool_l.${outpuFileSuffix}
```

```
otool -oV "${inputFile}" > ${outputFoler}/""${outputMachoFile}""_otool_oV.${outpuFileSuffix}

log "Exporting info use nm"
nm "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_nm.${outpuFileSuffix}

log "Exporting info use strings"
strings "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_strings.${outpuFileSuffix}

log "Exporting info use jtool2"
export ARCH=arm64
jtool2 "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_strings.${outpuFileSuffix}

jtool2 -h "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_jtool2_h_header.${outpuFileSuffix}
jtool2 -l "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_jtool2_l_list.${outpuFileSuffix}
jtool2 -L "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_jtool2_L_library.${outpuFileSuffix}
jtool2 -S "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_jtool2_S_symbol.${outpuFileSuffix}

jtool2 --analyze "${inputFile}"
# outpu: dyld.ARM64.C21DBA37-9DF9-3FC7-B286-734030E18BB1

# findNameRegex="${outputMachoFile}".*.*
# findNameRegex="${inputFile}".*.*
findNameRegex="${inputMachoFile}".*.*
# findNameRegex="${origgOutputMachoFile}".*.*
echo "findNameRegex=${findNameRegex}"
# jtool2AnalyzeOutputFile="$(find . -maxdepth 1 -name ${findNameRegex} -printf 1 -quit)"

# jtool2AnalyzeOutputFile=$(find . -maxdepth 1 -name ${findNameRegex} -printf 1 -quit)
# jtool2AnalyzeOutputFile=$(find . -maxdepth 1 -name ${findNameRegex})
# jtool2AnalyzeOutputFile=$(find . -maxdepth 1 -name 'dyld.*')
# jtool2AnalyzeOutputFile=$(find . -maxdepth 1 -name ${findNameRegex} -printf 1)
# jtool2AnalyzeOutputFile=$(find . -name ${findNameRegex})
jtool2AnalyzeOutputFile=$(find . -maxdepth 1 -name ${findNameRegex})
echo "jtool2AnalyzeOutputFile=${jtool2AnalyzeOutputFile}"  # jtool2AnalyzeOutputFile=./dyld.ARM64.C21DBA37-9DF9-3FC7-B286-734030E18BB1
jtool2AnalyzeOutputFilename="$(basename "${jtool2AnalyzeOutputFile}")"
echo "jtool2AnalyzeOutputFilename=${jtool2AnalyzeOutputFilename}" # jtool2AnalyzeOutputFilename=dyld.ARM64.C21DBA37-9DF9-3FC7-B286-734030E18BB1
mv ${jtool2AnalyzeOutputFilename} ${outputFoler}/"${outputMachoFile}"_jtool2_analyze.${outpuFileSuffix}

log "Exporting info use rabin2"
rabin2 -I "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_rabin2_I_identification.${outpuFileSuffix}
rabin2 -i "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_rabin2_i_imports.${outpuFileSuffix}
rabin2 -E "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_rabin2_E_exports.${outpuFileSuffix}
rabin2 -l "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_rabin2_l_libraries.${outpuFileSuffix}
rabin2 -z "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_rabin2_z_strings.${outpu
```

```
FileSuffix}
rabin2 -s "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_rabin2_s_symbols.${outpu
FileSuffix}
rabin2 -S "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_rabin2_S_sections.${outp
uFileSuffix}

log "Exporting entitlement info use ldid and codesign"
# export entitlements
ldid2 -e "${inputFile}" > ${outputFoler}/"${outputMachoFile}"_ldld_entitlement.xml
codesign -d --entitlements - "${inputFile}" &> ${outputFoler}/"${outputMachoFile}"_code
sign_entitlement.xml

log "Exporting code sign info use codesign"
# export code sign info
#   for binary
codesign -vv -d "${inputFile}" &> ${outputFoler}/"${outputMachoFile}"_codesign.${outpuF
ileSuffix}
# #   for app
# codesign -vv -d xxx.App > iOSApp_codesign.txt

log "Exporting info Done"
```

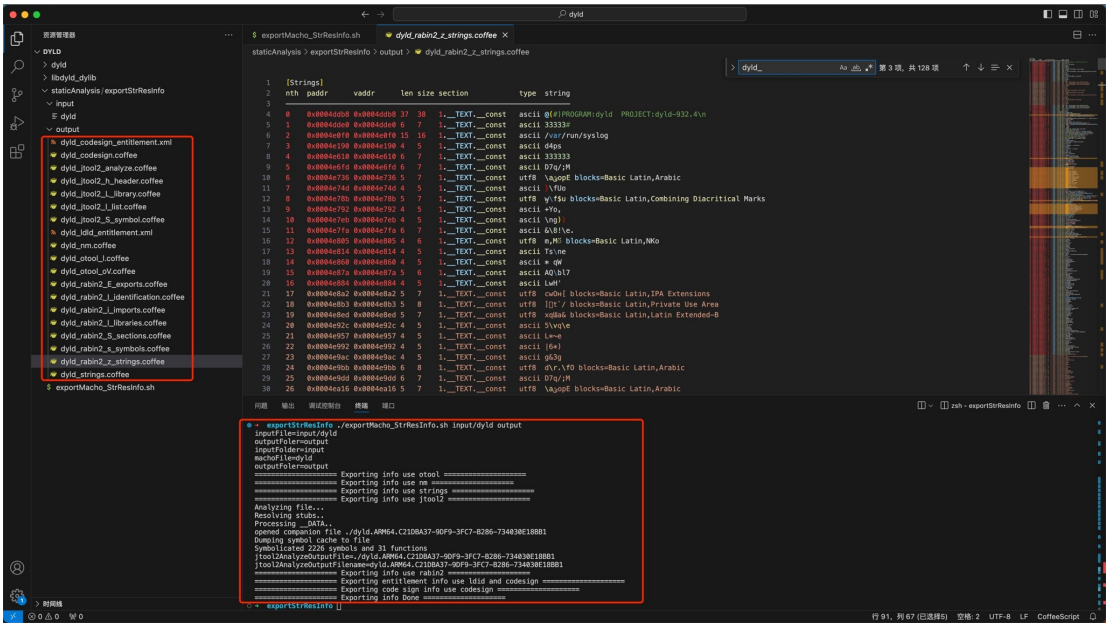
运行方式：

- 添加可执行权限
  - `chmod +x exportMacho_StrResInfo.sh`
- 运行：输入文件的传入Mach-O二进制和（可选）输出目录

```
./exportMacho_StrResInfo.sh <inputMachoFile> <outputFolder>
```

  - 举例

```
./exportMacho_StrResInfo.sh input/dyld output
```



手动处理

手动一行行的，单独分批去运行下面的命令，去导出Mach-O的字符串资源等信息：

```
nm iOSBinaryFile > iOSBinaryFile_nm.txt

strings iOSBinaryFile > iOSBinaryFile_strings.txt

otool -l iOSBinaryFile > iOSBinaryFile_otool_l.txt
otool -oV iOSBinaryFile > iOSBinaryFile_otool_oV.txt

jtool2 -h iOSBinaryFile > iOSBinaryFile_jtool2_h_header.txt
jtool2 -l iOSBinaryFile > iOSBinaryFile_jtool2_l_list.txt
jtool2 -L iOSBinaryFile > iOSBinaryFile_jtool2_L_library.txt
jtool2 -S iOSBinaryFile > iOSBinaryFile_jtool2_S_symbol.txt

jtool2 --analyze iOSBinaryFile
mv iOSBinaryFile.ARM64.xxx-xxx-xxx-xxx-xxx iOSBinaryFile_jtool2_analyze.txt

rabin2 -I iOSBinaryFile > iOSBinaryFile_rabin2_I_identification.txt
rabin2 -i iOSBinaryFile > iOSBinaryFile_rabin2_i_imports.txt
rabin2 -E iOSBinaryFile > iOSBinaryFile_rabin2_E_exports.txt
rabin2 -l iOSBinaryFile > iOSBinaryFile_rabin2_l_libraries.txt
rabin2 -z iOSBinaryFile > iOSBinaryFile_rabin2_z_strings.txt
rabin2 -s iOSBinaryFile > iOSBinaryFile_rabin2_s_symbols.txt
rabin2 -S iOSBinaryFile > iOSBinaryFile_rabin2_S_sections.txt

# export entitlements
codesign -d --entitlements - iOSBinaryFile > iOSBinaryFile_codesign_entitlement.xml
# or
# ldid -e iOSBinaryFile > iOSBinaryFile_ldld_entitlement.xml
# ldid2 -e iOSBinaryFile > iOSBinaryFile_ldld_entitlement.xml

# export code sign info
#   for binary
codesign -vv -d iOSBinaryFile > iOSBinaryFile_codesign.txt
#   for app
codesign -vv -d xxx.App > iOSApp_codesign.txt
```

- 说明
  - 把 `iOSBinaryFile` 换成实际的 `Mach-O` 二进制文件名
- 特殊
  - 如果二进制是 `FAT` 格式= `胖二进制` ，那么对于 `jtool2` ，要指定架构才能继续

    ```
    export ARCH=arm64
    jtool2 -h iOSBinaryFile > iOSBinaryFile_jtool2_h_header.txt
    ```

举例

HeiBao的dylib

对于二级制文件，此处是dylib的动态库：

```
➜  DynamicLibraries pwd
/Users/crifan/dev/DevRoot/Aweme/exportFromiPhone/iPhoneX-137/Library/MobileSubstrate/DynamicLibraries
➜  DynamicLibraries ll
total 152568
-rwxr-xr-x@ 1 crifan  staff   6.2M  3 14 10:34 zzzzHeiBaoLib.dylib
-rw-r--r--  1 crifan  staff    68M  3 17 21:39 zzzzHeiBaoLib.i64
```

去导出字符串等资源：

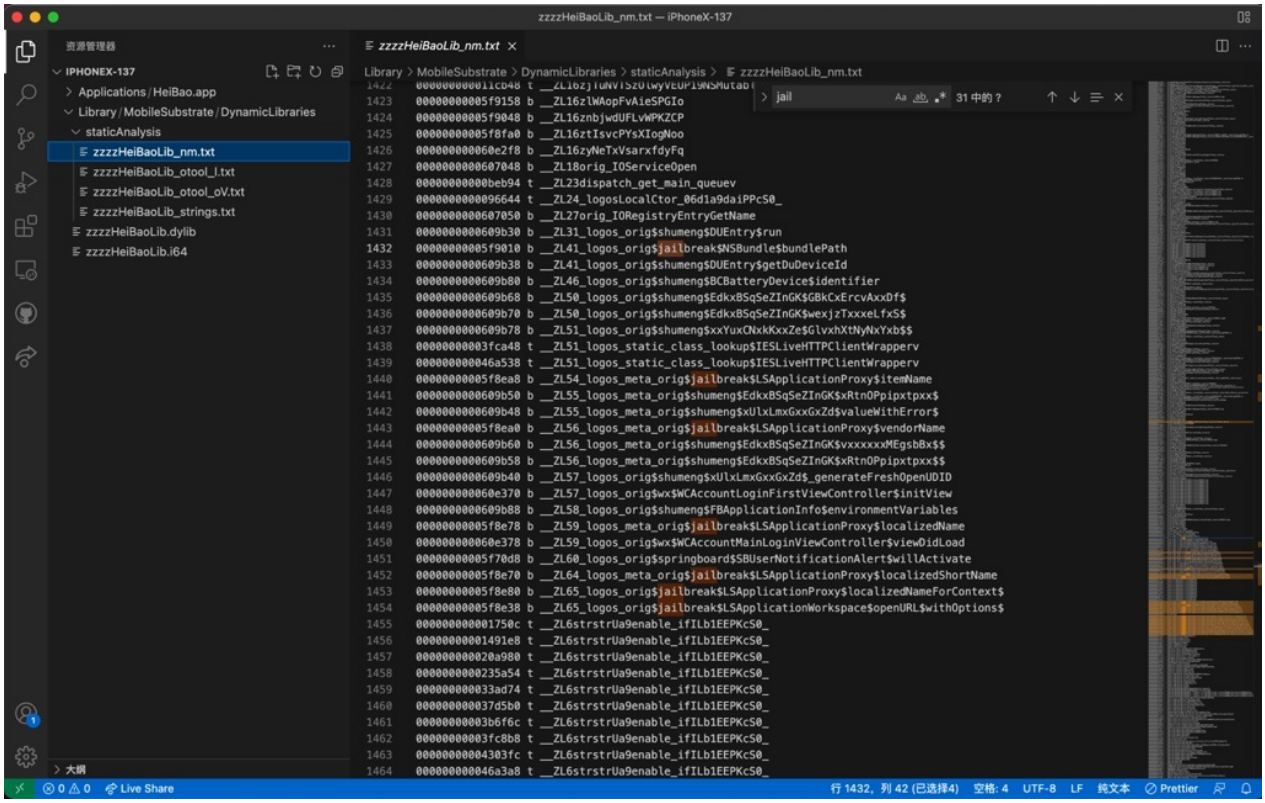
```
➜  DynamicLibraries otool -l zzzzHeiBaoLib.dylib > HeiBaoLib_otool_l.txt
➜  DynamicLibraries otool -oV zzzzHeiBaoLib.dylib > HeiBaoLib_otool_oV.txt
➜  DynamicLibraries nm zzzzHeiBaoLib.dylib > HeiBaoLib_nm.txt
➜  DynamicLibraries strings zzzzHeiBaoLib.dylib > HeiBaoLib_strings.txt

➜  DynamicLibraries ll
total 153112
-rw-r--r--  1 crifan  staff   108K  3 21 10:00 HeiBaoLib_nm.txt
-rw-r--r--  1 crifan  staff    12K  3 21 10:00 HeiBaoLib_otool_l.txt
-rw-r--r--  1 crifan  staff    51K  3 21 10:00 HeiBaoLib_otool_oV.txt
-rw-r--r--  1 crifan  staff    89K  3 21 10:00 HeiBaoLib_strings.txt
-rwxr-xr-x@ 1 crifan  staff   6.2M  3 14 10:34 zzzzHeiBaoLib.dylib
-rw-r--r--  1 crifan  staff    68M  3 17 21:39 zzzzHeiBaoLib.i64
```

后续即可去分析和搜索想要研究的值了。

比如：

搜索越狱 jailbreak 相关内容：



抖音的AwemeCore

```
jtool2 -h ../../../../已脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.framework/Aw
emeCore

jtool2 -l ../../../../已脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.framework/Aw
emeCore > AwemeCore_jtool2_l_list.txt
jtool2 -L ../../../../已脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.framework/Aw
emeCore > AwemeCore_jtool2_L_library.txt

jtool2 -S ../../../../已脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.framework/Aw
emeCore > AwemeCore_jtool2_S.txt

jtool2 --analyze ../../../../已脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.frame
work/AwemeCore > AwemeCore_jtool2_analyze.txt


rabin2 -I ../../../../已脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.framework/Aw
emeCore

rabin2 -i ../../../../已脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.framework/Aw
emeCore > AwemeCore_rabin2_i.txt

rabin2 -E ../../../../已脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.framework/Aw
emeCore > AwemeCore_rabin2_E.txt

rabin2 -l ../../../../已脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.framework/Aw
emeCore > AwemeCore_rabin2_l.txt

rabin2 -z ../../../../已脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.framework/Aw
emeCore > AwemeCore_rabin2_z.txt
```

```
rabin2 -s ../../../已脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.framework/AwemeCore > AwemeCore_rabin2_s.txt

rabin2 -S ../../../已脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.framework/AwemeCore > AwemeCore_rabin2_S_section.txt
```

MaskPro.dylib

```
➜  DynamicLibraries otool -l MaskPro.dylib > MaskProDylib/MaskProDylib_otool_l.txt
➜  DynamicLibraries otool -oV MaskPro.dylib > MaskProDylib/MaskProDylib_otool_oV.txt
➜  DynamicLibraries nm MaskPro.dylib > MaskProDylib/MaskProDylib_nm.txt
➜  DynamicLibraries strings MaskPro.dylib > MaskProDylib/MaskProDylib_strings.txt
```

期间遇到 `FAT = 胖二进制` 的问题：

```
➜  DynamicLibraries jtool2 -h MaskPro.dylib > MaskProDylib/MaskProDylib_jtool2_h_header.txt
Fat binary, little-endian, 2 architectures: armv7, arm64
Select an architecture setting the ARCH= environment variable
```

解决办法：

```
➜  DynamicLibraries export ARCH=arm64
➜  DynamicLibraries jtool2 -h MaskPro.dylib > MaskProDylib/MaskProDylib_jtool2_h_header.txt
```

继续：

```
➜  DynamicLibraries jtool2 -l MaskPro.dylib > MaskProDylib/MaskProDylib_jtool2_l_list.txt
➜  DynamicLibraries jtool2 -L MaskPro.dylib > MaskProDylib/MaskProDylib_jtool2_L_library.txt
➜  DynamicLibraries jtool2 -S MaskPro.dylib > MaskProDylib/MaskProDylib_jtool2_S_symbol.txt
➜  DynamicLibraries jtool2 --analyze MaskPro.dylib > MaskProDylib/MaskProDylib_jtool2_analyze.txt
...
```

继续：

```
➜  DynamicLibraries rabin2 -I MaskPro.dylib > MaskProDylib/MaskProDylib_rabin2_I_identification.txt
➜  DynamicLibraries rabin2 -i MaskPro.dylib > MaskProDylib/MaskProDylib_rabin2_i_imports.txt
➜  DynamicLibraries rabin2 -E MaskPro.dylib > MaskProDylib/MaskProDylib_rabin2_E_exports.txt
➜  DynamicLibraries rabin2 -l MaskPro.dylib > MaskProDylib/MaskProDylib_rabin2_l_libraries.txt
➜  DynamicLibraries rabin2 -z MaskPro.dylib > MaskProDylib/MaskProDylib_rabin2_z_string
```

```
s.txt
➜  DynamicLibraries rabin2 -s MaskPro.dylib > MaskProDylib/MaskProDylib_rabin2_s_symbol
s.txt
➜  DynamicLibraries rabin2 -S MaskPro.dylib > MaskProDylib/MaskProDylib_rabin2_S_sectio
ns.txt
```

Mask的dylib

对于一个二进制，此处是一个动态库文件 `Mask.dylib` ，想要导出字符串等资源，供后续分析。

- 典型的**成套的做法** = `otool` + `nm` + `strings` + `jtool2` + `rabin2` ：

```
otool -l Mask.dylib > MaskDylib_otool_l.txt
otool -oV Mask.dylib > MaskDylib_otool_oV.txt

nm Mask.dylib > MaskDylib_nm.txt

strings Mask.dylib > MaskDylib_strings.txt

jtool2 -h Mask.dylib > MaskDylib_jtool2_h_header.txt
jtool2 -l Mask.dylib > MaskDylib_jtool2_l_list.txt
jtool2 -L Mask.dylib > MaskDylib_jtool2_L_library.txt
jtool2 -S Mask.dylib > MaskDylib_jtool2_S_symbol.txt
jtool2 --analyze Mask.dylib > MaskDylib_jtool2_analyze.txt

rabin2 -I Mask.dylib > MaskDylib_rabin2_I_identification.txt
rabin2 -i Mask.dylib > MaskDylib_rabin2_i_imports.txt
rabin2 -E Mask.dylib > MaskDylib_rabin2_E_exports.txt
rabin2 -l Mask.dylib > MaskDylib_rabin2_l_libraries.txt
rabin2 -z Mask.dylib > MaskDylib_rabin2_z_strings.txt
rabin2 -s Mask.dylib > MaskDylib_rabin2_s_symbols.txt
rabin2 -S Mask.dylib > MaskDylib_rabin2_S_sections.txt
```

Mach-O工具

- `Mach-O` 工具
  - 概述
    - 查看解析Mach-O
      - MachOView
      - rabin2
      - jtool2
      - otool
    - FAT瘦身
      - lipo
  - 详解
    - 独立子教程
      - 可执行文件格式：Mach-O

nm

用法

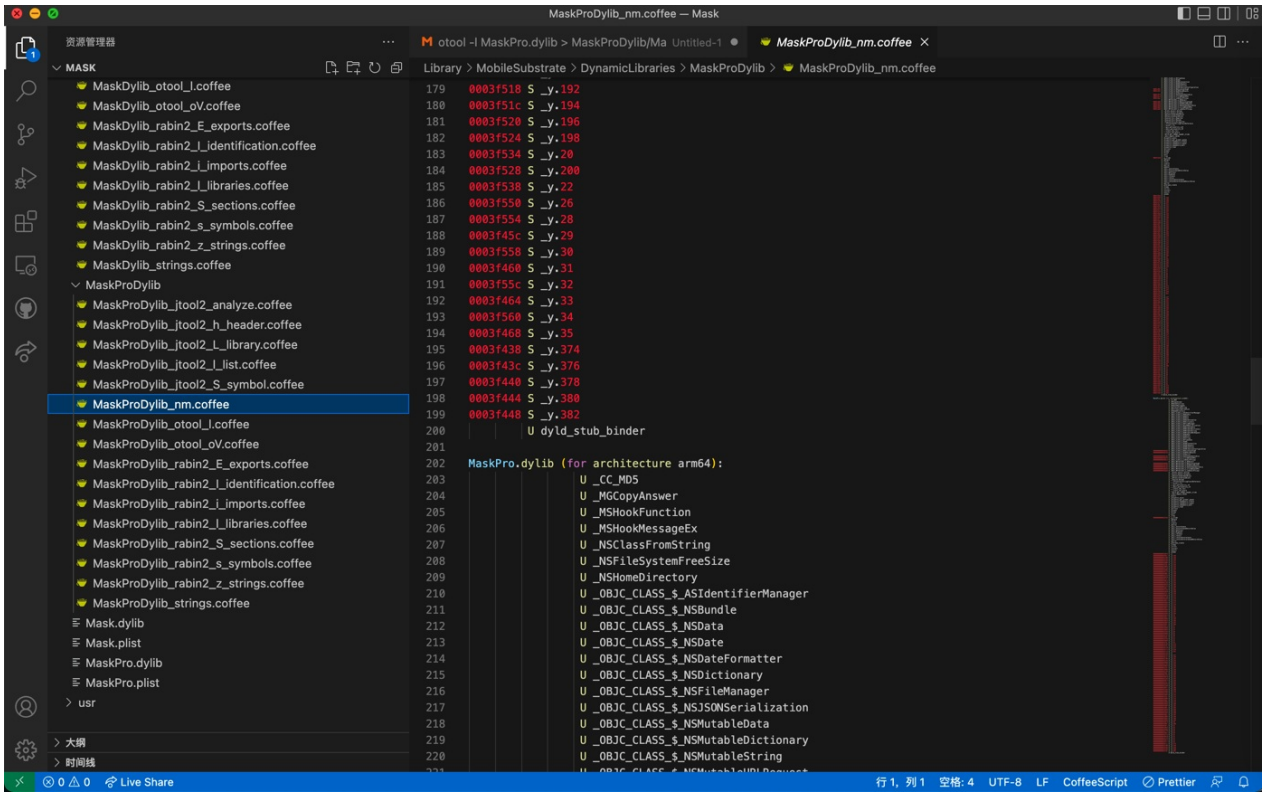
```
nm iOSBinaryFile > iOSBinaryFile_nm.txt
```

举例

MaskPro.dylib

```
➜  DynamicLibraries nm MaskPro.dylib > MaskProDylib/MaskProDylib_nm.txt
```

输出：




```
MaskPro.dylib (for architecture armv7):
         U _CC_MD5
         U _MGCopyAnswer
         U _MSHookFunction
         U _MSHookMessageEx
         U _NSClassFromString
         U _NSFileSystemFreeSize
         U _NSHomeDirectory
         U _OBJC_CLASS_$_ASIdentifierManager
         U _OBJC_CLASS_$_NSBundle
...
0003f440 S _y.378
0003f444 S _y.380
0003f448 S _y.382
         U dyld_stub_binder


MaskPro.dylib (for architecture arm64):
                 U _CC_MD5
                 U _MGCopyAnswer
```

```
                 U _MSHookFunction
                 U _MSHookMessageEx
                 U _NSClassFromString
                 U _NSFileSystemFreeSize
                 U _NSHomeDirectory
                 U _OBJC_CLASS_$_ASIdentifierManager
...
```

strings

用法

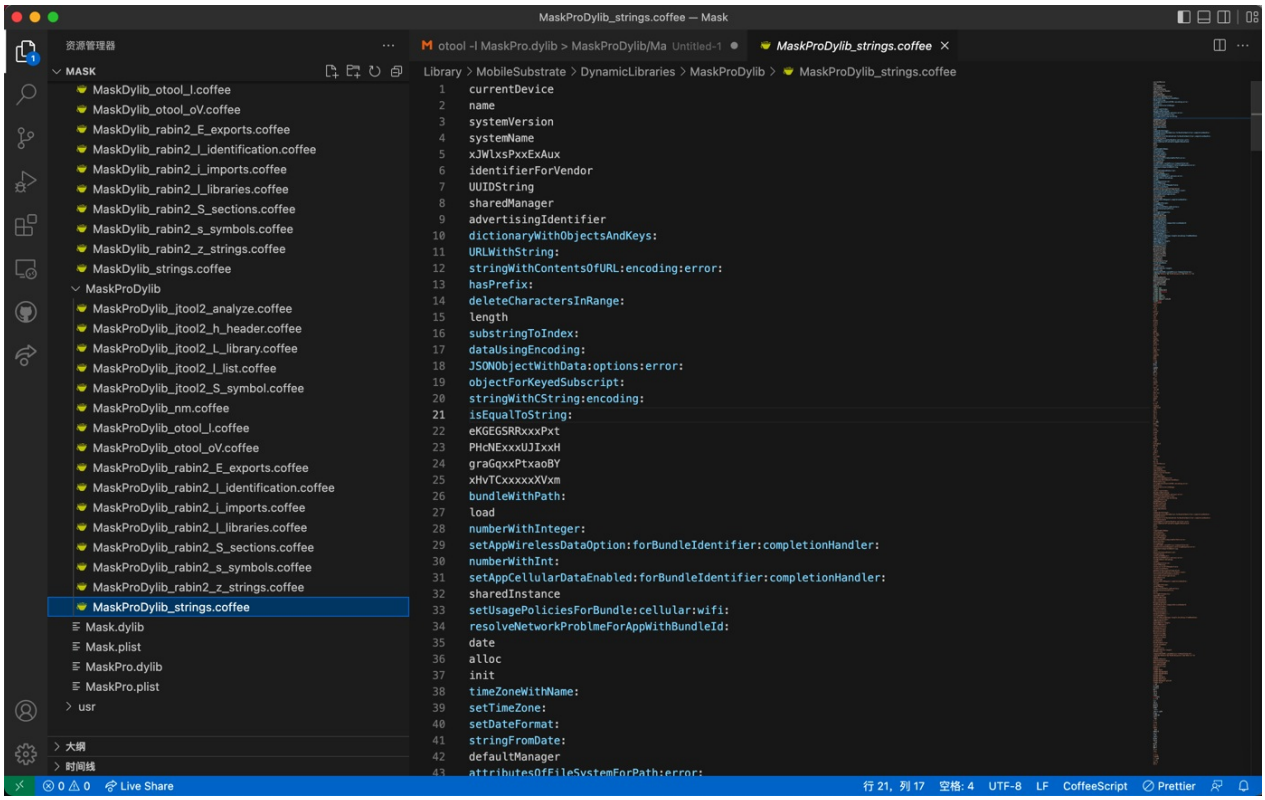
```
strings iOSBinaryFile > iOSBinaryFile_strings.txt
```

strings用法举例

MaskPro.dylib

```
➜  DynamicLibraries strings MaskPro.dylib > MaskProDylib/MaskProDylib_strings.txt
```

输出：



其中的：

```
eKGEGSRRxxxPxt
PHcNExxxUJIxxH
graGqxxPtxaoBY
xHvTCxxxxxXVxm
```

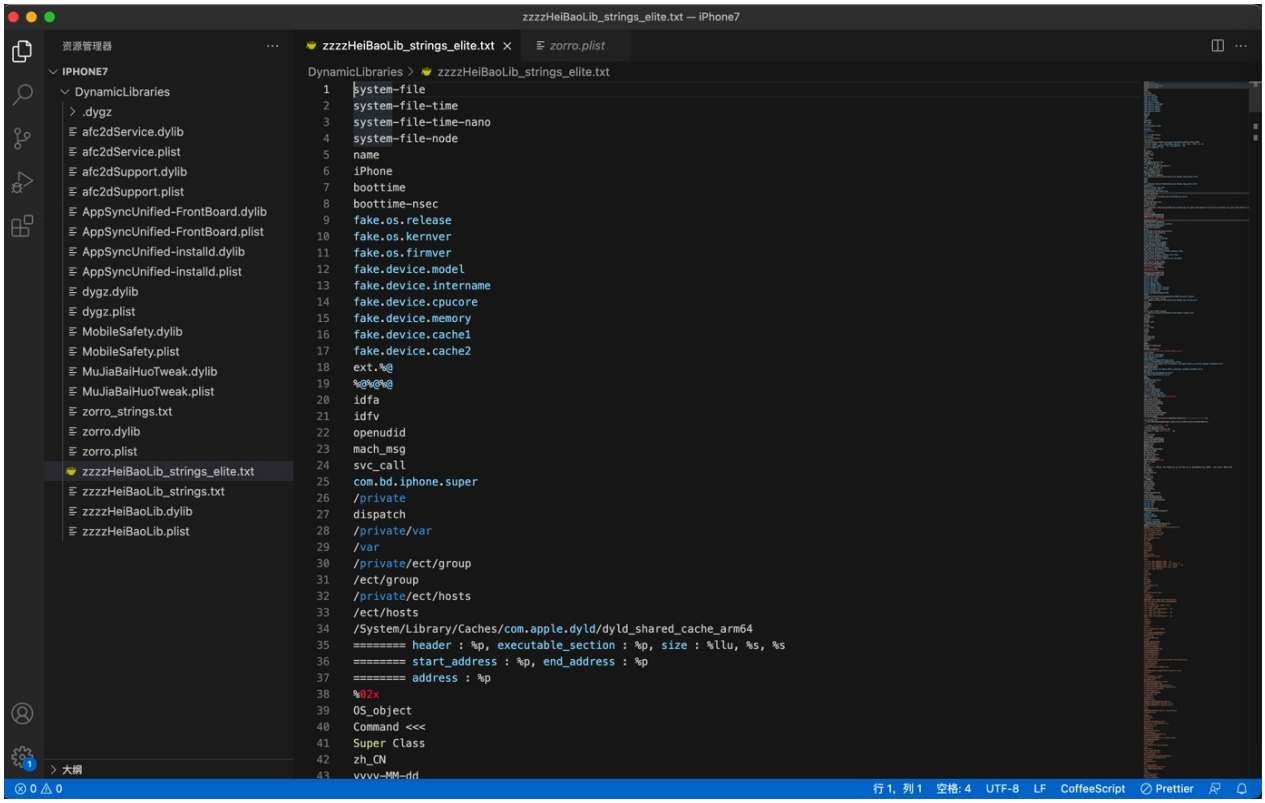
很明显是：字符串加密的一种，手动替换或者hash成固定长度的字符串了。

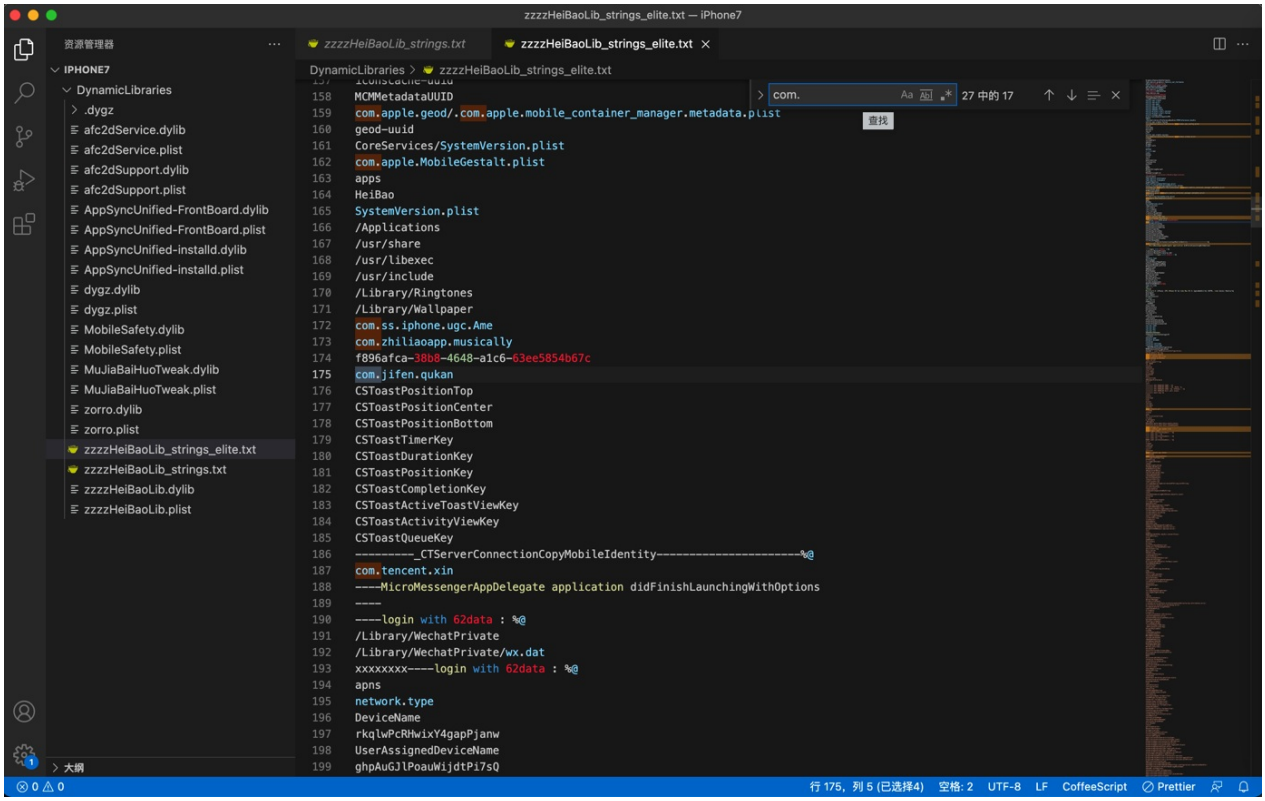
zzzzHeiBaoLib.dylib

```
➜  DynamicLibraries strings zzzzHeiBaoLib.dylib > zzzzHeiBaoLib_strings.txt
```

输出：`zzzzHeiBaoLib_strings.txt`

改名和提取核心字段后为：`zzzzHeiBaoLib_strings_elite.txt`





摘录部分内容：

- 疑似app
  - com.bd.iphone.super
  - com.apple.springboard
  - com.apple.geod/.com.apple.mobile_container_manager.metadata.plist
  - com.apple.MobileGestalt.plist
  - com.zhiliaoapp.musically

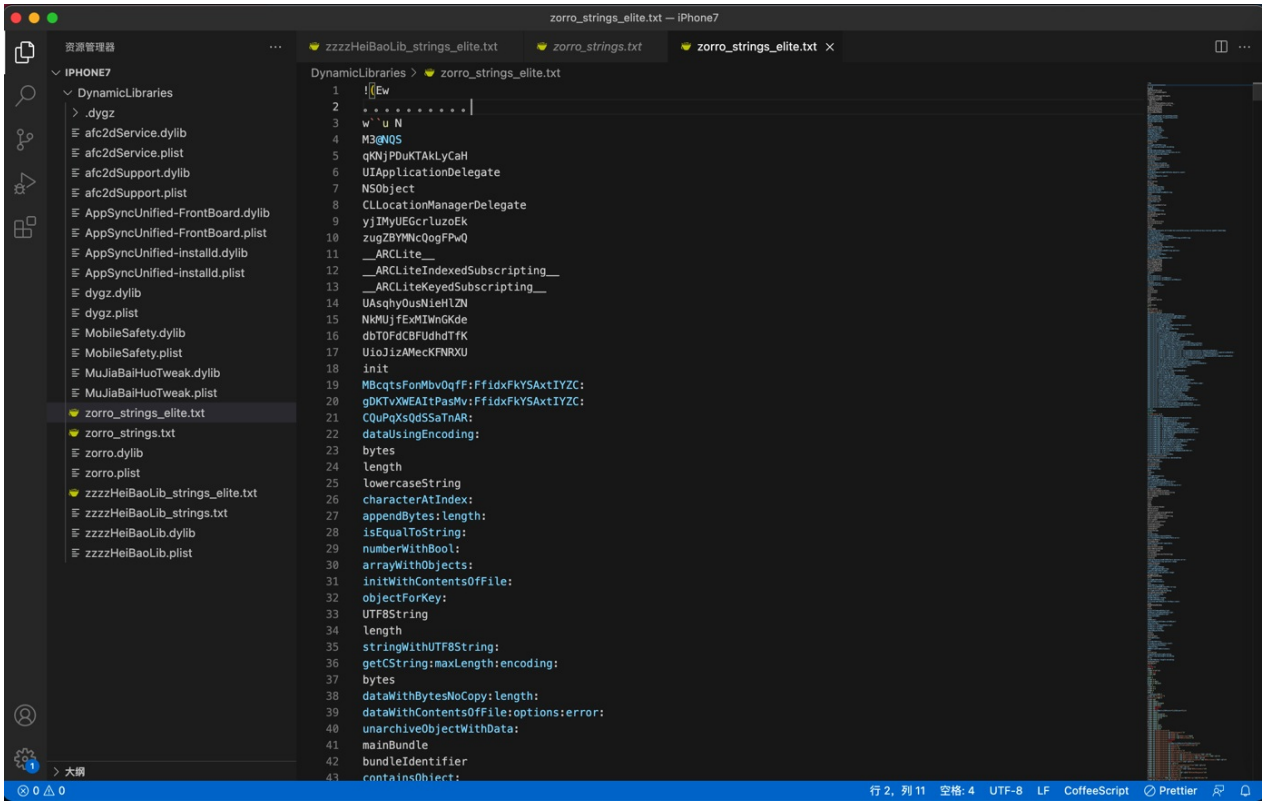
    - com.jifen.qukan
    - 趣头条
    - com.tencent.xin
    - com.kuaishou.nebula
    - 快手极速版
    - com.ss.iphone.ugc.Live
    - com.xunmeng.pinduoduo
    - com.soulapp.cn
    - 抖音
    - iPhone版
        - com.ss.iphone.ugc.Ame
        - com.ss.iphone.ugc.Aweme
        - com.ss.iphone.ugc.aweme.lite
    - 注：安卓版 是
        - com.ss.android.ugc.aweme
        - com.ss.android.ugc.aweme.lite
    - com.dragon.read
    - com.wemomo.momoappdemo1
    - com.jiangjia.gif
- 系统属性
    - carrier.name
    - carrier.mcc
    - carrier.mnc
    - carrier.icc
    - network.addr
    - network.dstaddr
    - location.latitude
    - location.longitude
    - identifierForVendor
    - systemVersion
    - operatingSystemVersionString
    - operatingSystemVersion
    - carrierName
    - mobileCountryCode
    - mobileNetworkCode
    - isoCountryCode
    - allowsVOIP
- 疑似越狱检测路径相关
    - /
        - /Applications
        - /usr
            - /usr/share
            - /usr/libexec
            - /usr/include
        - /ect/hosts
        - /private

        - /private/var
            - /private/var/containers/Bundle/Application/
        - /private/ect/group
    - /Library
        - /Library/Ringtones
        - /Library/Wallpaper
    - /System
        - /System/Library
            - /System/Library/Caches/com.apple.dyld/dyld_shared_cache_arm64
            - /System/Library/CoreServices/SystemVersion.plist
            - /System/Library/PreferenceBundles/VPNPreferences.bundle
    - /var
        - /var/containers/Shared/SystemGroup/systemgroup.com.apple.mobilegestaltcache/Library/Caches/com.apple.MobileGestalt.plist
- 黑豹本身有关
    - /var/mobile/Library/Preferences/com.heibao.fake_params.plist
    - /var/mobile/Library/Preferences/com.heibao.fake_prefs.plist
    - /var/mobile/Library/Preferences/com.heibao.vpn_config.plist
    - /var/mobile/Library/Preferences/com.heibao.scheme.plist
- 微信相关
    - /Library/WechatPrivate
    - /Library/WechatPrivate/wx.dat
- 反越狱检测相关
    - isJailbroken
    - btd_isJailBroken
    - isJailBreak
    - isJailBroken
    - HasInstallJailbreakPluginInvalidIAPPurchase
    - HasInstallJailbreakPlugin:
    - IsJailBreak
    - getJailbreakPath
    - getJailbreakRootDir
    - JailBroken
    - isDeviceJailBreak
    - isDeviceJailBroken
    - cannotPurchaseDuetoJailbreakPlugin:

zorro.dylib

```
➜  DynamicLibraries strings zorro.dylib > zorro_strings.txt
```

输出：



结果分析：

- 反越狱检测的相关路径
  - /var/containers
  - /System/Library/Frameworks/
  - /var/containers/Bundle/Application/
  - /var/stash/
  - /private/var/db/stash/
  - /private/
  - /no/jailbreak
  - /System/Library/CoreServices/SystemVersion.plist

处理签名

处理iOS的app或二进制的签名（和entitlement）等内容的常见工具有：

- ldid
- codesign

ldid

- `ldid` / `ldid2`
  - 用法概述
    - 查看/导出二进制的entitlement权限

      ```
      ldid -e <binary>
      ```

      - 举例

        ```
        ldid -e debugserver
        ldid -e amsaccountsd
        ldid -e Preferences.app/Preferences > Preferences_entitlement.plist
        ```

    - 设置二进制的entitlement权限

      ```
      ldid -Sent.xml <binary>
      ```

      - 注：`-S` 和文件名 `ent.xml` 中间**没有空格**！
      - 举例
        - 使用对应的entitlement文件给 `debugserver` 重新签名

          ```
          ldid -S/Users/your/Desktop/Entitlements.xml ./debugserver
          ```

  - 详见子教程
    - ldid · iOS逆向开发：签名和权限

codesign

- `codesign`
  - 用法概述
    - 二进制文件

      ```
      codesign -d --entitlements - BinaryFile
      ```

      - 举例

        ```
        codesign -d --entitlements - debugserver
        codesign -d --entitlements - amsaccountsd_orig
        ```

    - app

      ```
      codesign -d --entitlements - xxx.app
      ```

      - 举例

        ```
        codesign -d --entitlements - Preferences.app
        ```

  - 详见子教程
    - codesign · iOS逆向开发：签名和权限

导出头文件

- 概述
  - 用 `ObjC` 的MonkeyDev的class-dump 或 Swift的crifan/dsdump去导出头文件
- 详解
  - iOS逆向分析：导出头文件
    - 最新结论 · iOS逆向分析：导出头文件

分析代码逻辑

iOS逆向的静态分析期间，最大的最有用的，还是：分析代码逻辑

iOS逆向方面，常用的分析代码逻辑的工具有：

- `IDA`
- `Hopper`

IDA

TODO：

- 【记录】IDA分析脱壳后抖音ipa的二进制AwemeCore
- 【已解决】iOS底层函数：objc_enumerationMutation
- 【未解决】研究抖音越狱检测逻辑：___lldb_unnamed_symbol1841884即sub_11326A84

后已整理出独立教程，详见：

逆向利器：IDA (crifan.org)

iOS逆向之IDA使用心得

objc_enumerationMutation表示循环的逻辑

iOS逆向期间，IDA的伪代码中，有时候会看到：`objc_enumerationMutation`

其含义是：`只是表示这段代码是循环遍历而已`

代码举例：

```
v4 = (void *)objc_retain(v2->_allTrackRenderers);
v5 = v4;
v6 = objc_msgSend(v4, "countByEnumeratingWithState:objects:count:", &v24, &v28, 16LL);

if ( v6 )
{
  v7 = v6;
  v8 = **(_QWORD **)v25;
  do
  {
    v9 = 0LL;
    do
    {
      if ( *(_QWORD *)v25 != v8 )
        objc_enumerationMutation(v5);
      objc_msgSend(*(void **)(*((_QWORD *)&v24 + 1) + 8 * v9++), "terminate");
    }
    while ( v9 < (unsigned __int64)v7 );
    v7 = objc_msgSend(v5, "countByEnumeratingWithState:objects:count:", &v24, &v28, 1
6LL);
  }
  while ( v7 );
}
```

和：

```
    do
    {
      v34 = 0LL;
...
      v81 = v31;
      do
      {
        if ( *(_QWORD *)v98 != v32 )
          objc_enumerationMutation(v28);
        v86 = v34;
        v35 = *(_QWORD *)(*((_QWORD *)&v97 + 1) + 8 * v34);
        v85 = jmp_objc_msgSend_x19tox2(v26[83], objectForKeyedSubscript__);
...
      }
      while ( v86 + 1 < (unsigned __int64)v31 );
      v31 = objc_msgSend(v28, v73, &v97, &v109, 16LL);
    }
    while ( v31 );
```

Hopper

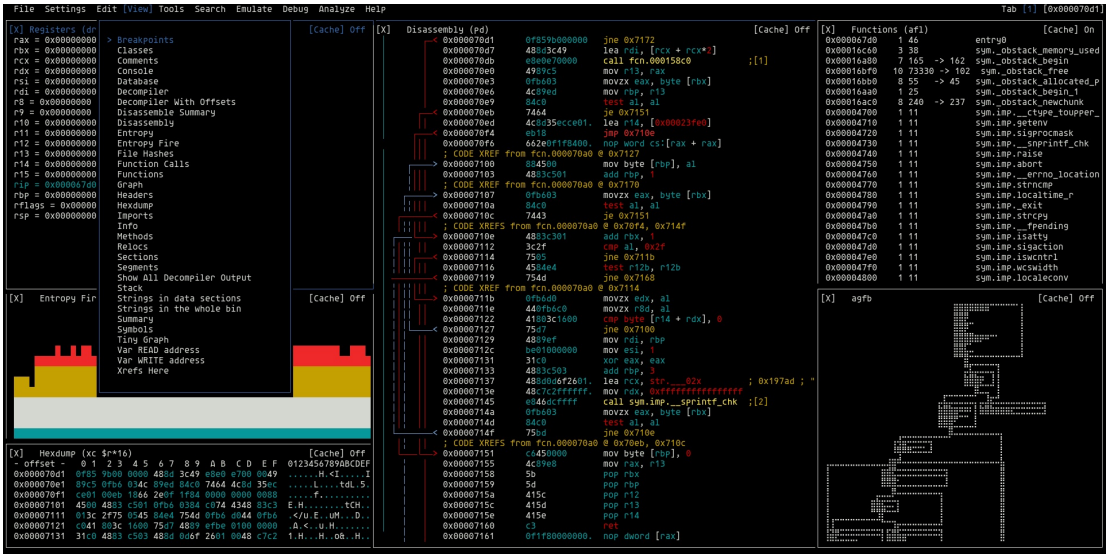
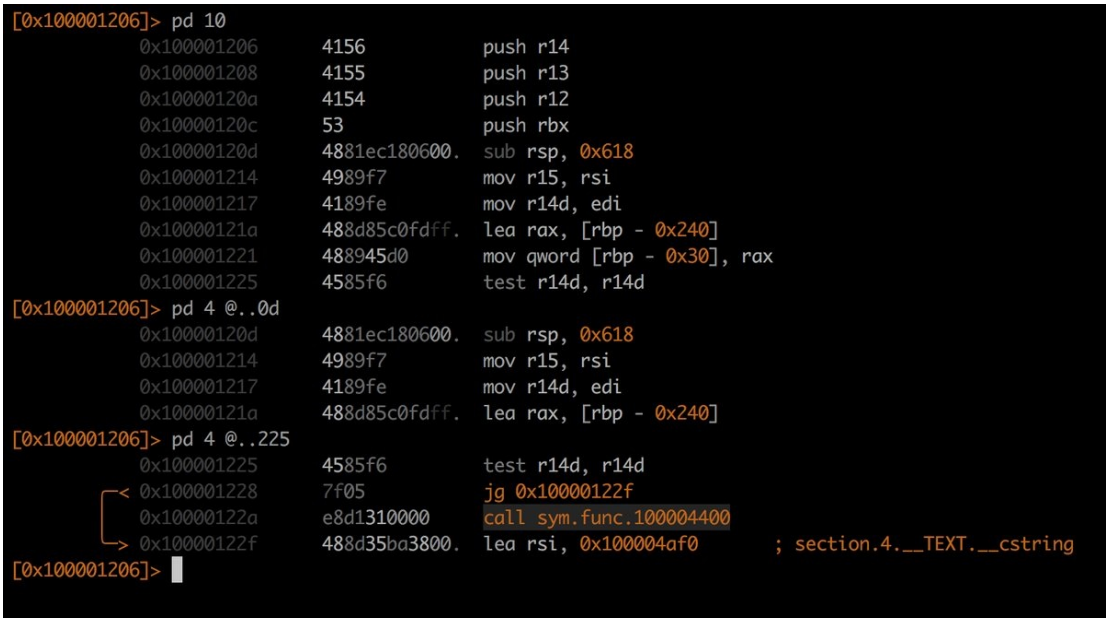
- `Hopper = Hopper Disassembler`
  - 概述
    - Hopper是常用的iOS逆向工具之一，主要用来静态分析iOS二进制的代码逻辑
  - 详解
    - iOS逆向工具：Hopper

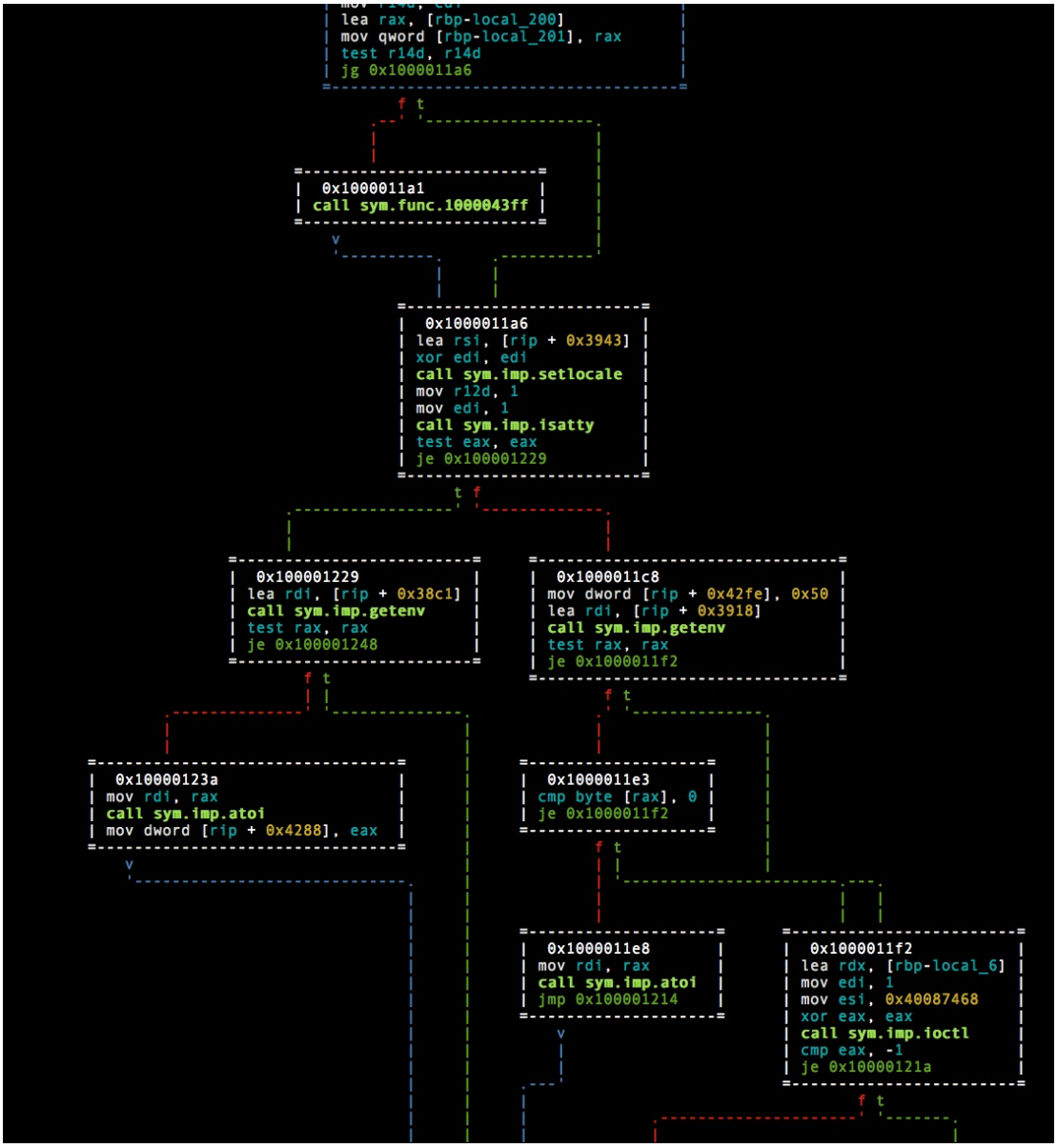
相关工具

对于iOS逆向中的静态分析，其实还有其他一些相关工具，值得介绍。

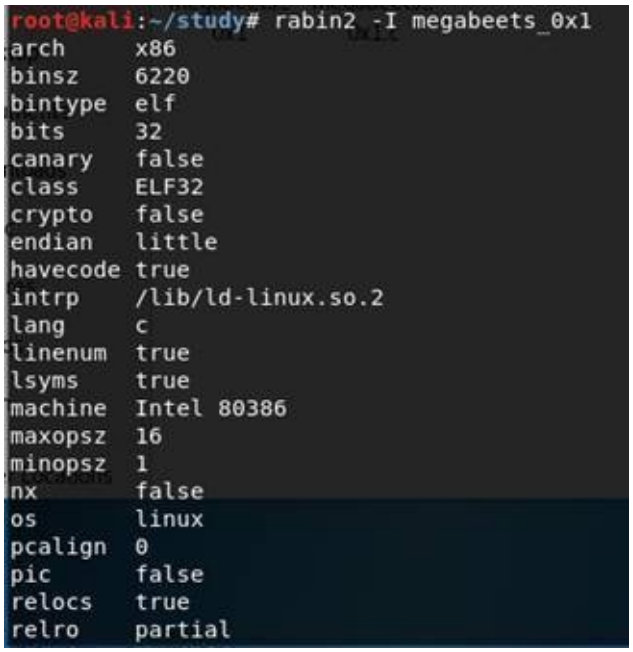
- `radare2`：成套的逆向工具
  - 包含多个命令行的工具，前面提到的rabin2就是其中之一
- `Cutter`：radare2的GUI版本
- Miasm
  - 一句话介绍：用Python实现的逆向工程框架
    - Reverse engineering framework in Python
  - 用途：分析、修改、生成 二进制程序
    - analyze / modify / generate binary programs
  - 特性
    - Opening / modifying / generating PE / ELF 32 / 64 LE / BE
    - Assembling / Disassembling X86 / ARM / MIPS / SH4 / MSP430
    - Representing assembly semantic using intermediate language
    - Emulating using JIT (dynamic code analysis, unpacking, ...)
    - Expression simplification for automatic de-obfuscation
  - 资料
    - GitHub
      - cea-sec/miasm: Reverse engineering framework in Python
        - https://github.com/cea-sec/miasm
    - 官网
      - Home — Miasm's blog
        - https://miasm.re/blog/

radare2

- `radare2`
  - 是什么：一个著名的开源逆向工程平台
    - Unix-like reverse engineering framework and commandline tools
  - 评价
    - 可谓是一大神器
      - 支持包括反汇编、分析数据、打补丁、比较数据、搜索、替换、虚拟化等等，同时具备超强的脚本加载能力，并且可以运行在几乎所有主流的平台
  - 竞品
    - `IDA`
  - 截图
    - 
    - 



- 支持平台
  - Mac
  - Windows
  - Linux
  - Solaris
  - Android
  - iOS
  - Haiku
- 历史
  - Radare project started as a forensics tool, a scriptable commandline hexadecimal editor able to open disk files but later support for analyzing binaries, disassembling code, debugging programs, attaching to remote gdb servers
- 功能：Radare is a portable reversing framework that can
  - Disassemble (and assemble for) many different architectures
  - Debug with local native and remote debuggers (gdb, rap, webui, r2pipe, winedbg, windbg)
  - Run on Linux, *BSD, Windows, OSX, Android, iOS, Solaris and Haiku
  - Perform forensics on filesystems and data carving

    - Be scripted in Python, Javascript, Go and more
    - Support collaborative analysis using the embedded webserver
    - Visualize data structures of several file types
    - Patch programs to uncover new features or fix vulnerabilities
    - Use powerful analysis capabilities to speed up reversing
    - Aid in software exploitation
  - 特性
    - Batch, commandline, visual and panels interactive modes
    - Embedded webserver with js scripting and webui
    - Assemble and disassemble a large list of CPUs
    - Runs on Windows and any other UNIX flavour out there
    - Analyze and emulate code with ESIL
    - Native debugger and GDB, WINDBG, QNX and FRIDA
    - Navigate ascii-art control flow graphs
    - Ability to patch binaries, modify code or data
    - Search for patterns, magic headers, function signatures
    - Easy to extend and modify
    - Commandline, C API, script with r2pipe in any language
  - 包含工具
    - `rabin2`
      - 获取 `ELF` , `PE` , `Mach-O` , `Java CLASS` 文件的区段、头信息、导入导出表、字符串相关、入口点等等
        - 包括打印出二进制文件的系统属性、语言、字节序、框架、以及使用了哪些加固技术
      - 支持多种格式的输出文件
      - 截图
        - 

    - `radiff2`：比较文件不同的
    - `rahash2`：各种密码算法，hash算法集成
    - `rasm2`：汇编和反汇编
    - `ragg2`：开发shellcode工具(radare2自己编写的编译器)
    - `radare2`：整合了所有工具

- 资料
  - 官网
    - radare
      - https://rada.re/n/radare2.html
    - GitHub
      - radareorg/radare2: UNIX-like reverse engineering framework and command-line toolset
        - https://github.com/radareorg/radare2
    - 教程
      - The Official Radare2 Book
        - https://book.rada.re/index.html

help帮助语法

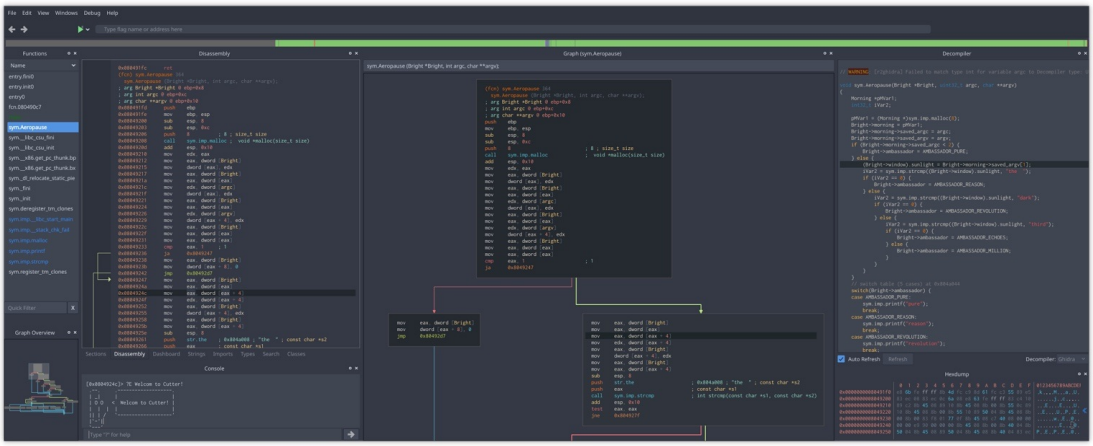
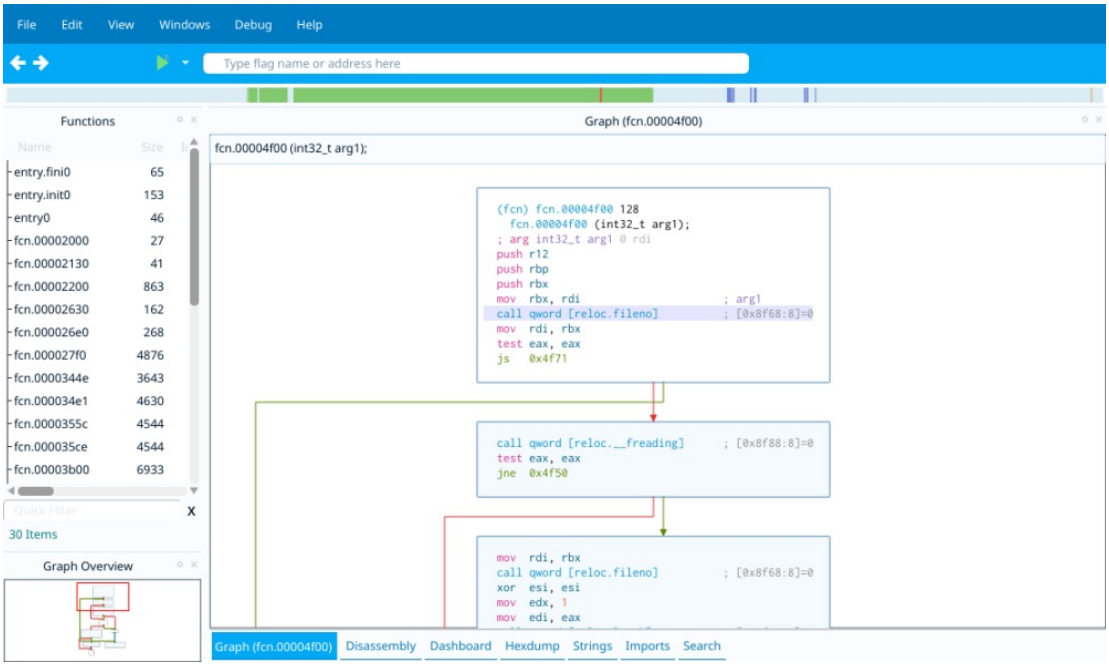
```
$ radare2 -h
Usage: r2 [-ACdfLMnNqStuvwzX] [-P patch] [-p prj] [-a arch] [-b bits] [-i file]
          [-s addr] [-B baddr] [-m maddr] [-c cmd] [-e k=v] file|pid|-|--|=
 --            run radare2 without opening any file
 -             same as 'r2 malloc://512'
 =             read file from stdin (use -i and -c to run cmds)
 -=            perform =| command to run all commands remotely
 -0            print \x00 after init and every command
 -2            close stderr file descriptor (silent warning messages)
 -a [arch]     set asm.arch
 -A            run 'aaa' command to analyze all referenced code
 -b [bits]     set asm.bits
 -B [baddr]    set base address for PIE binaries
 -c 'cmd..'    execute radare command
 -C            file is host:port (alias for -c+=http://%s/cmd/)
 -d            debug the executable 'file' or running process 'pid'
 -D [backend]  enable debug mode (e cfg.debug=true)
 -e k=v        evaluate config var
 -f            block size = file size
 -F [binplug]  force to use that rbin plugin
 -h, -hh       show help message, -hh for long
 -H ([var])    display variable
 -i [file]     run script file
 -I [file]     run script file before the file is opened
 -k [OS/kern]  set asm.os (linux, macos, w32, netbsd, ...)
 -l [lib]      load plugin file
 -L            list supported IO plugins
 -m [addr]     map file at given address (loadaddr)
 -M            do not demangle symbol names
 -n, -nn       do not load RBin info (-nn only load bin structures)
 -N            do not load user settings and scripts
 -q            quiet mode (no prompt) and quit after -i
 -Q            quiet mode (no prompt) and quit faster (quickLeak=true)
 -p [prj]      use project, list if no arg, load if no file
 -P [file]     apply rapatch file and quit
 -r [rarun2]   specify rarun2 profile to load (same as -e dbg.profile=X)
 -R [rr2rule]  specify custom rarun2 directive
 -s [addr]     initial seek
 -S            start r2 in sandbox mode
```

```
-t            load rabin2 info in thread
-u            set bin.filter=false to get raw sym/sec/cls names
-v, -V        show radare2 version (-V show lib versions)
-w            open file in write mode
-x            open without exec-flag (asm.emu will not work), See io.exec
-X            same as -e bin.usextr=false (useful for dyldcache)
-z, -zz       do not load strings or load them even in raw
```

R2Pipe

- `R2Pipe`
  - 是什么：`R2Pipe`是一个可以调用`radare2`的`Python`脚本库
  - 示例代码
    - https://github.com/radareorg/radare2-r2pipe/tree/master/python/examples

Cutter

- `Cutter`
  - 一句话描述：（radare2的fork版）Rizin的GUI版本
    - Free and Open Source RE Platform powered by radare2
    - Cutter is the official UI for radare2 for Linux, macOS and Windows, it's written in C++ and uses the Qt
  - 支持多平台
    - `Linux`
    - `Mac`
    - `Windows`
  - 实现细节
    - `C++` 语言写的
    - 前端：`QT`
  - 截图
    - 
    - 

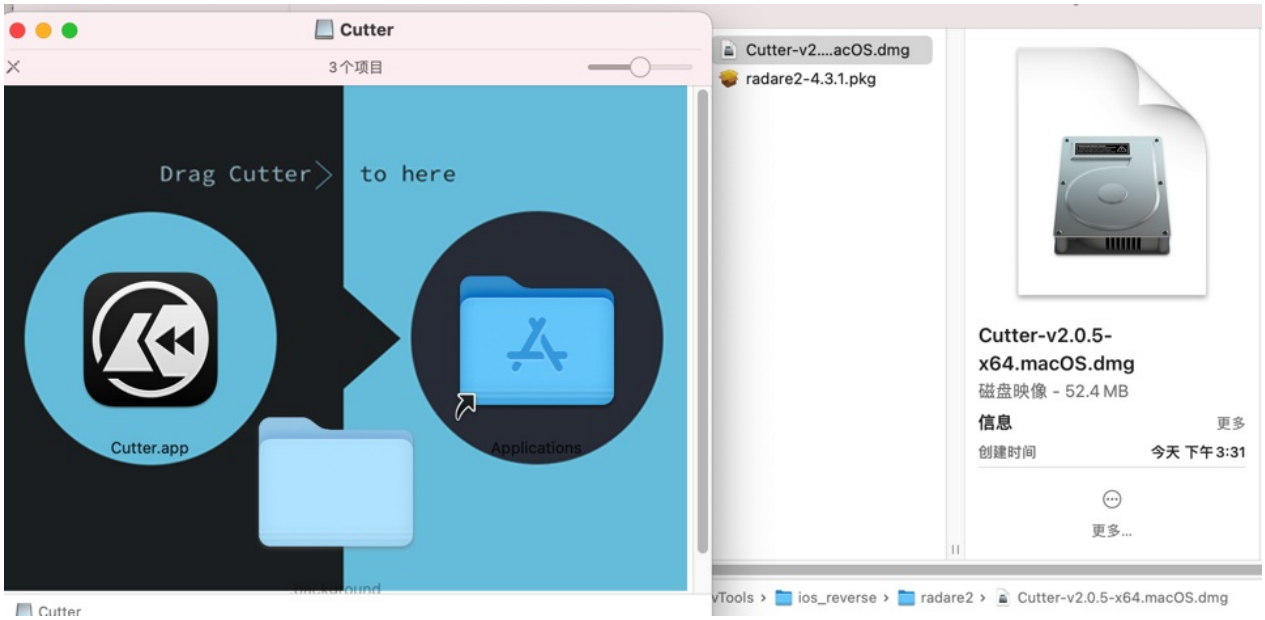
  - 特点
    - 内置Ghidra decompiler

        - 无需额外安装Java
            - 用C++实现的，目的是性能更好
    - 核心功能和特点
        - 开源 Open Source
            - Completely FREE and licensed under GPLv3
        - Decompiler
            - Native integration of Ghidra's decompiler in Cutter releases
        - Graph View
            - Fully featured graph view as well as mini-graph for fast navigation
        - Debugger
            - Multiplatform native and remote debugger for dynamic analysis
        - Disassembly
            - Linear disassembly view
        - Hex Editor
            - View and modify any file with a rich and powerful Hex View
        - Python Scripting Engine
            - Quickly write python scripts to automate tasks
        - Plugins
            - Use Native or Python plugins to extend Cutter's core functionality
        - Binary Patching
            - Add, remove and modify bytes and instructions
        - Emulation
            - Great for automation, crypto algorithms and malware analysis
        - Theme Editor
            - Fully featured theme editor for easy and user-friendly customization of Cutter
        - Modern & Customizable UI
            - Built using Qt C++ and design best practices
        - Integrated Radare2 Console
        - Multi Language
        - Binary Searching
        - Types & Structs
        - Syntax Highlighting
        - STDIO Redirection
        - Remote Debugging
        - Kernel Debug
        - Graph Overview
    - 资料
        - 官网
            - Cutter
                - https://cutter.re
        - GitHub
            - radareorg/cutter: Free and Open Source Reverse Engineering Platform powered by radare2
                - https://github.com/rizinorg/cutter
            - rizinorg/rizin: UNIX-like reverse engineering framework and command-line toolset.
                - https://github.com/rizinorg/rizin

安装Cutter

Cutter官网点击Download，此处下载到：（之前某旧版本）

`Cutter-v2.0.5-x64.macOS.dmg`



双击，即可安装。

打开后，即可看到主界面：

- Cutter欢迎界面

- Cutter的关于

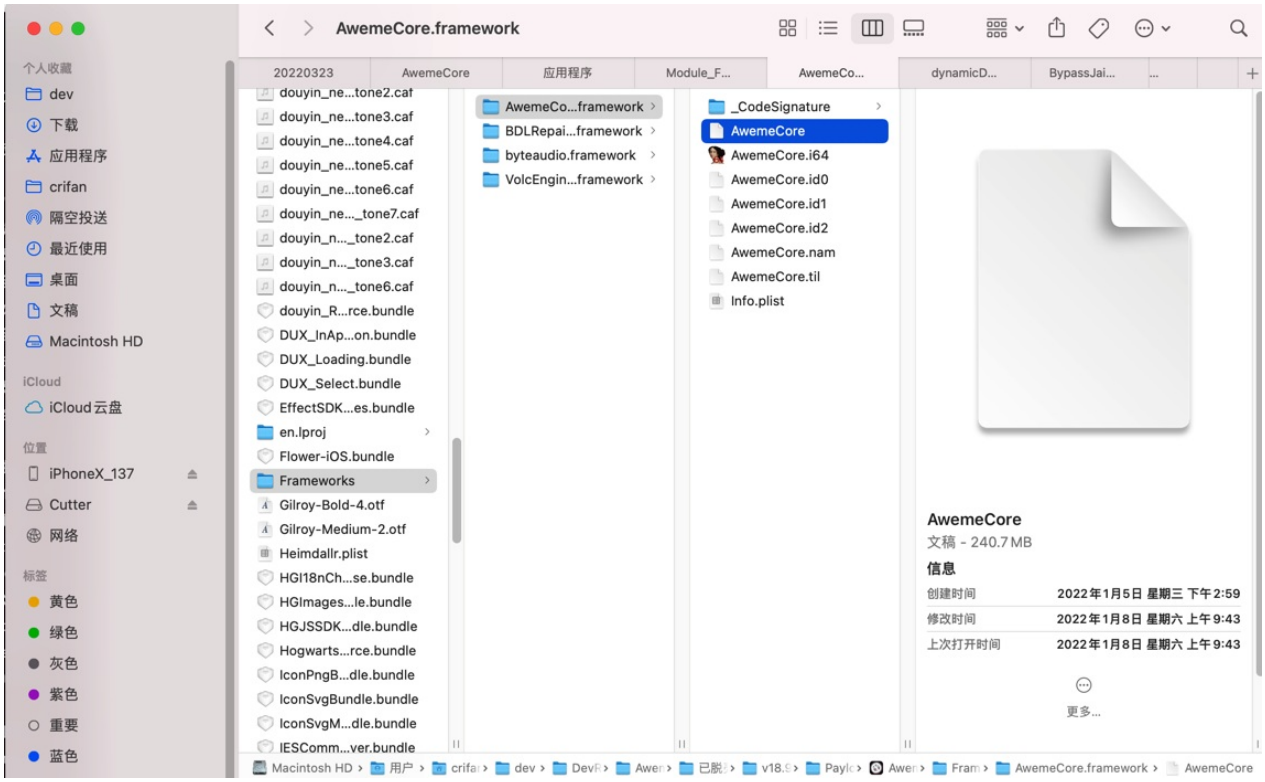
- 。

Cutter使用举例

Cutter使用举例：AwemeCore

打开页面：

把要逆向的文件 `AwemeCore` 拖动过去：


AwemeCore.framework
个人收藏
dev
下载
应用程序
crifan
隔空投送
最近使用
桌面
文稿
Macintosh HD
iCloud
iCloud云盘
位置
iPhoneX_137
Cutter
网络
标签
黄色
绿色
灰色
紫色
重要
蓝色
20220323
AwemeCore
应用程序
Module_F...
AwemeCo...
dynamicD...
BypassJai...
douyin_ne...tone2.caf
douyin_ne...tone3.caf
douyin_ne...tone4.caf
douyin_ne...tone5.caf
douyin_ne...tone6.caf
douyin_ne..._tone7.caf
douyin_n..._tone2.caf
douyin_n..._tone3.caf
douyin_n..._tone6.caf
douyin_R...rce.bundle
DUX_InAp...on.bundle
DUX_Loading.bundle
DUX_Select.bundle
EffectSDK...es.bundle
en.lproj
Flower-iOS.bundle
Frameworks
Gilroy-Bold-4.otf
Gilroy-Medium-2.otf
Heimdallr.plist
HGI18nCh...se.bundle
HGImages...le.bundle
HGJSSDK...dle.bundle
Hogwarts...rce.bundle
IconPngB...dle.bundle
IconSvgBundle.bundle
IconSvgM...dle.bundle
IESComm...ver.bundle
AwemeCo...framework
BDLRepai...framework
byteaudio.framework
VolcEngin...framework
_CodeSignature
AwemeCore
AwemeCore.i64
AwemeCore.id0
AwemeCore.id1
AwemeCore.id2
AwemeCore.nam
AwemeCore.til
Info.plist
AwemeCore
文稿 - 240.7 MB
信息
创建时间 2022年1月5日 星期三 下午2:59
修改时间 2022年1月8日 星期六 上午9:43
上次打开时间 2022年1月8日 星期六 上午9:43
更多...
Macintosh HD › 用户 › crifa › dev › DevF › Awer › 已脱 › v18.5 › Payl › Awer › Fram › AwemeCore.framework › AwemeCore

加载选项：


加载选项
程序：
脱壳/v18.9.0/Payload/Aweme.app/Frameworks/AwemeCore.framework/AwemeCore
分析：已启用
级别: 自动分析 (aaa)
None
自动
Auto Exp
高级
以写入模式加载 (-w)
不加载二进制文件信息 (-n)
使用虚拟地址
导入 demangled 符号
高级选项
Cancel
确定

点击展开 `高级选项` 看看：

各个参数选项：

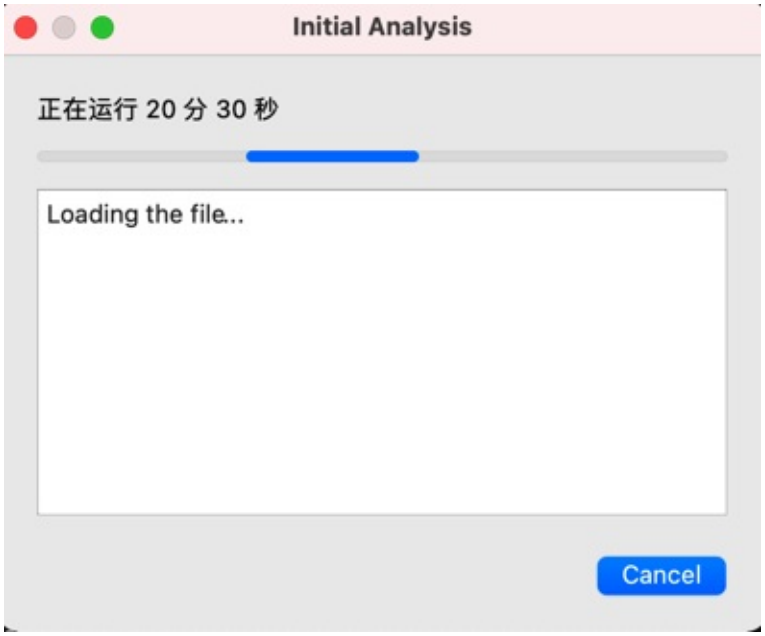
- `architecture`：支持很多类型

- bits

- Endianness
- kernel
- format

-

开始加载和分析：



运行了几个小时，都没结束。效率太低。那算了，最终放弃。

静态分析心得

此处整理iOS逆向期间的静态分析反面的心得。

二进制中找不到类

当前二进制找不到函数时，找别的二进制看看：

概述：

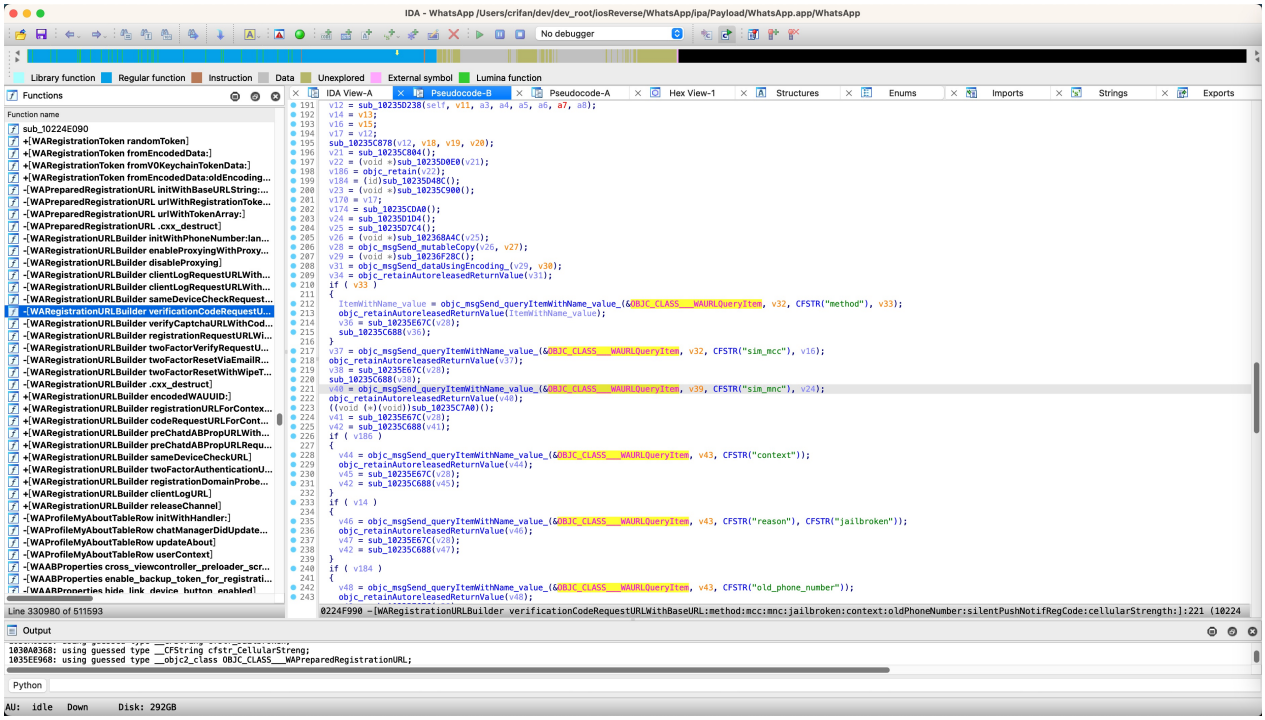
- iOS逆向期间，如果当前二进制中找不到某个类和函数时，可以去试试：找其他二进制
  - 以及可以从加载动态库依赖中，确认是否是依赖于别的二进制
    - 对应的 `Load Command` 中是否有 `LC_LOAD_DYLIB` ，去加载（同一个iOS的app内部的其他的）二进制文件
    - 对应的 `library` 中是否有别的二进制文件
  - 以及二进制查看函数
    - 一个是 `U = Undefined` ：是当前二进制未定义的，需要从别的二进制导入的
    - 一个是 `D = Defined` ：是当前二进制实现的类和函数

WhatsApp找不到WAURLQueryItem，但别的二进制SharedModules中可以找到

iOS的app `WhatsApp` 中的主要二进制是：

- `WhatsApp`
  -

去逆向期间，发现 `IDA` 伪代码中：




```
id __cdecl -[WARegistrationURLBuilder verificationCodeRequestURLWithBaseURL:method:mcc:
mnc:jailbroken:context:oldPhoneNumber:silentPushNotifRegCode:cellularStrength:](
        WARegistrationURLBuilder *self,
        SEL a2,
        id a3,
        id a4,
        id a5,
        id a6,
        bool a7,
        id a8,
        id a9,
        id a10,
        id a11)
{
...
    v46 = objc_msgSend_queryItemWithName_value_(&OBJC_CLASS___WAURLQueryItem, v43, CFSTR
("reason"), CFSTR("jailbroken"));
```

其中有：

- `OBJC_CLASS___WAURLQueryItem`
  - iOS的类：`WAURLQueryItem`

但是，此处WhatsApp的主要二进制 `WhatsApp` 中，竟然找不到 `WAURLQueryItem`

具体现象是：

`WhatsApp` 发现类 `WAURLQueryItem` 是未定义，需要额外引入的，且另外需要加载的库文件有 `SharedModules.framework/SharedModules`

- WhatsApp

（1）导出的的头文件中

- 搜不到：`WAURLQueryItem`

（2）导出的字符串等资源中

- 能搜到`WAURLQueryItem`的字眼

具体搜索到的内容有

```
WhatsApp_jtool2_S_symbol.txt:
  8954                  U _OBJC_CLASS_$_WATooltipView
  8955                  U _OBJC_CLASS_$_WATooltipView
  8956:                 U _OBJC_CLASS_$_WAURLQueryItem
  8956                  U _OBJC_CLASS_$_WAUUID
  8957                  U _OBJC_CLASS_$_WAUUID

WhatsApp_nm.txt:
  8954                  U _OBJC_CLASS_$_WATooltipView
  8955:                 U _OBJC_CLASS_$_WAURLQueryItem
  8956                  U _OBJC_CLASS_$_WAUUID
```

```
  WhatsApp_otool_oV.txt:
    1050534  0000000103417970 0x0 _OBJC_CLASS_$_NSURLComponents
    1050535: 0000000103417978 0x0 _OBJC_CLASS_$_WAURLQueryItem
    1050536  0000000103417980 0x0 _OBJC_CLASS_$_WACacheQueue

  WhatsApp_rabin2_i_imports.txt:
    8957  8953  ----------  NONE OBJC_CLASS         WATooltipView
    8958: 8954  ----------  NONE OBJC_CLASS         WAURLQueryItem
    8959  8955  ----------  NONE OBJC_CLASS         WAUUID
```

- 具体分析：
  - `WhatsApp_jtool2_S_symbol.txt` 中的
    - `8956: U _OBJC_CLASS_$_WAURLQueryItem`
      - 相关含义：U=Undefined=未定义
  - `WhatsApp_rabin2_i_imports.txt` 中的
    - `8958: 8954 ---------- NONE OBJC_CLASS WAURLQueryItem`
      - 中有WAURLQueryItem：表示WAURLQueryItem是属于（从外部）import进来的
- -> 表示此类：WAURLQueryItem，是从外部导入import进来的，当前二进制（WhatsApp）中是没有此类的具体实现的

（3）后记：在依赖的导入的Library库中，也能找到，是依赖于：库 `SharedModules` 的

具体现象是：

导出的字符串资源文件中，可以找到：

```
13 个结果 - 8 文件

WhatsApp_jtool2_L_library.txt:
  20      /System/Library/Frameworks/CoreBluetooth.framework/CoreBluetooth (compatibili
ty version 1.0.0, current version 1.0.0)
  21      /System/Library/Frameworks/CoreBluetooth.framework/CoreBluetooth (compatibili
ty version 1.0.0, current version 1.0.0)
  22:     @rpath/SharedModules.framework/SharedModules (compatibility version 1.0.0, cu
rrent version 1.0.0)
  22      @rpath/WARCDManager.framework/WARCDManager (compatibility version 0.0.0, curr
ent version 0.0.0)
  23      @rpath/WARCDManager.framework/WARCDManager (compatibility version 0.0.0, curr
ent version 0.0.0)

WhatsApp_jtool2_l_list.txt:
  88  LC 32: LC_LOAD_DYLIB            /System/Library/Frameworks/CoreBluetooth.framewo
rk/CoreBluetooth
  89: LC 33: LC_LOAD_DYLIB            @rpath/SharedModules.framework/SharedModules
  90  LC 34: LC_LOAD_WEAK_DYLIB       @rpath/WARCDManager.framework/WARCDManager

...

WhatsApp_otool_l.txt:
  Load command 33
            cmd LC_LOAD_DYLIB
        cmdsize 72
          name @rpath/SharedModules.framework/SharedModules (offset 24)
```

```
     time stamp 2 Thu Jan  1 08:00:02 1970
        current version 1.0.0
  compatibility version 1.0.0

...

WhatsApp_rabin2_l_libraries.txt:
  21  /System/Library/Frameworks/CoreBluetooth.framework/CoreBluetooth
  22: @rpath/SharedModules.framework/SharedModules
  23  @rpath/WARCDManager.framework/WARCDManager

...
```

具体解释是：

- `WhatsApp_jtool2_L_library.txt`
  - `22: @rpath/SharedModules.framework/SharedModules (compatibility version 1.0.0, current version 1.0.0)`
    - 表示要，当前二进制 `WhatsApp` 加载时，所依赖的，还要额外加载的库是
      - `SharedModules.framework/SharedModules`
- `WhatsApp_jtool2_l_list.txt`
  - `89: LC 33: LC_LOAD_DYLIB @rpath/SharedModules.framework/SharedModules`
    - 表示还要额外加载一个 Dylib=动态库文件：`SharedModules.framework/SharedModules`
- `WhatsApp_otool_l.txt`
  - `Load command 33`：是有一个Load Command
  - `cmd LC_LOAD_DYLIB`：command类型是要加载 `Dylib` =动态库
  - `name @rpath/SharedModules.framework/SharedModules (offset 24)`：具体要加载的库文件是：`SharedModules.framework/SharedModules`

`SharedModules` 中找到了类 `WAURLQueryItem` 的定义

最后是从同一个iOSapp：`WhatsApp` 的：

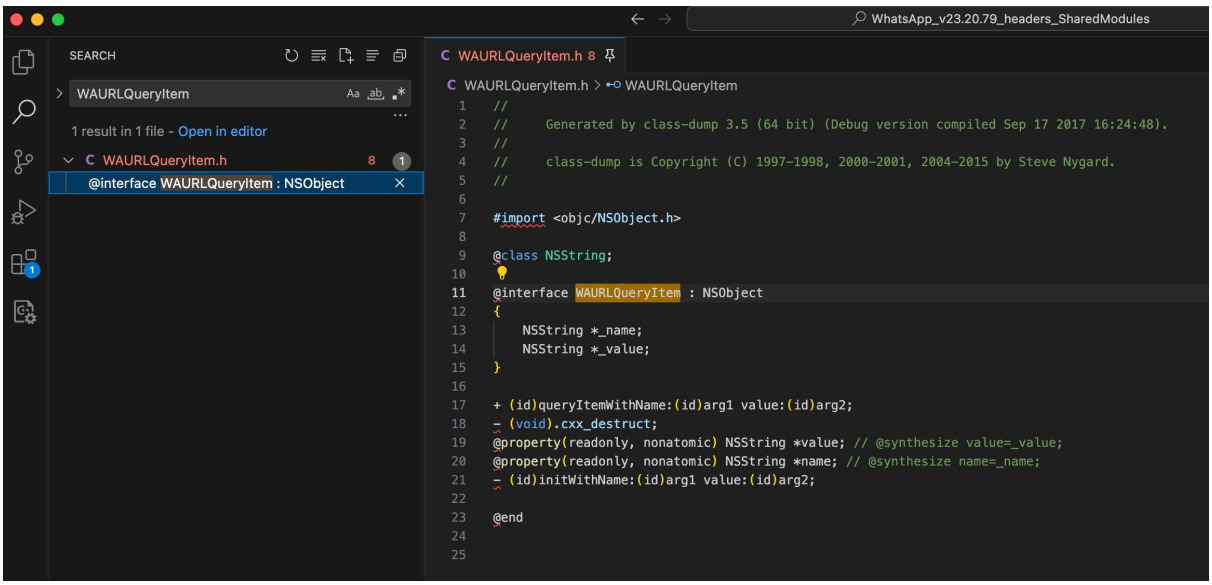
- 另外一个核心二进制 `SharedModules`

-

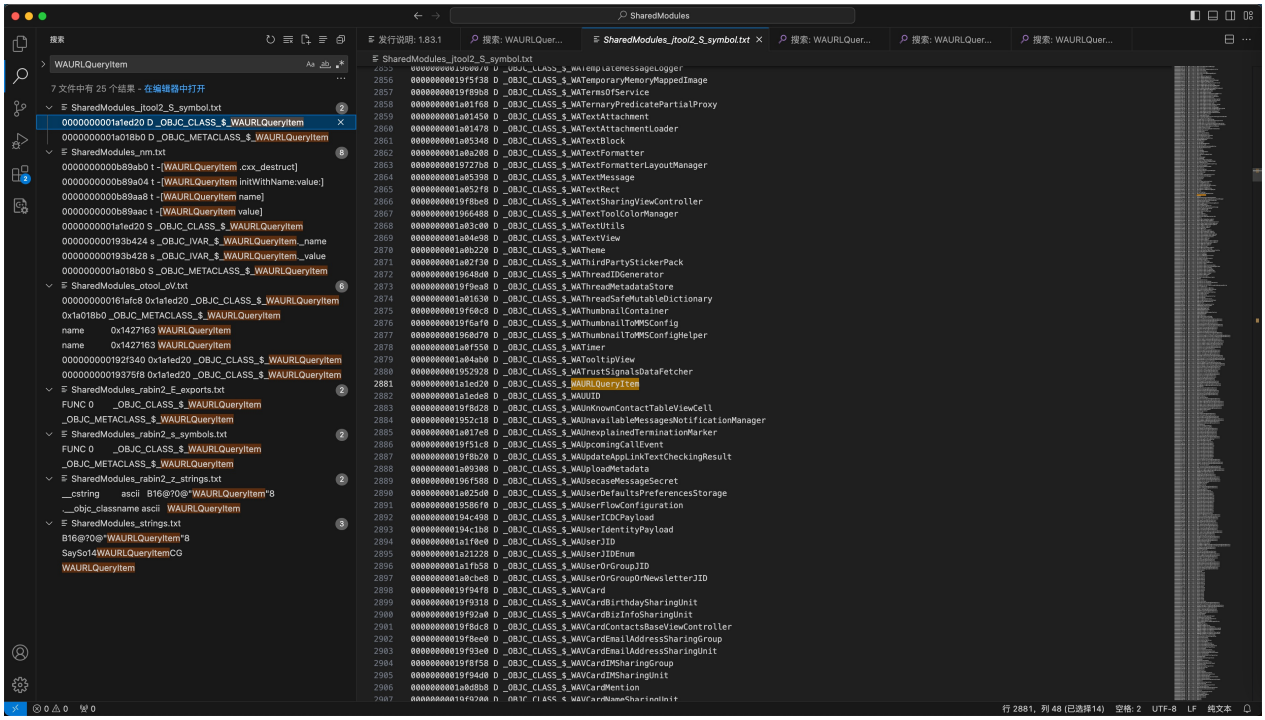
中找到了：类 `WAURLQueryItem` 的具体实现

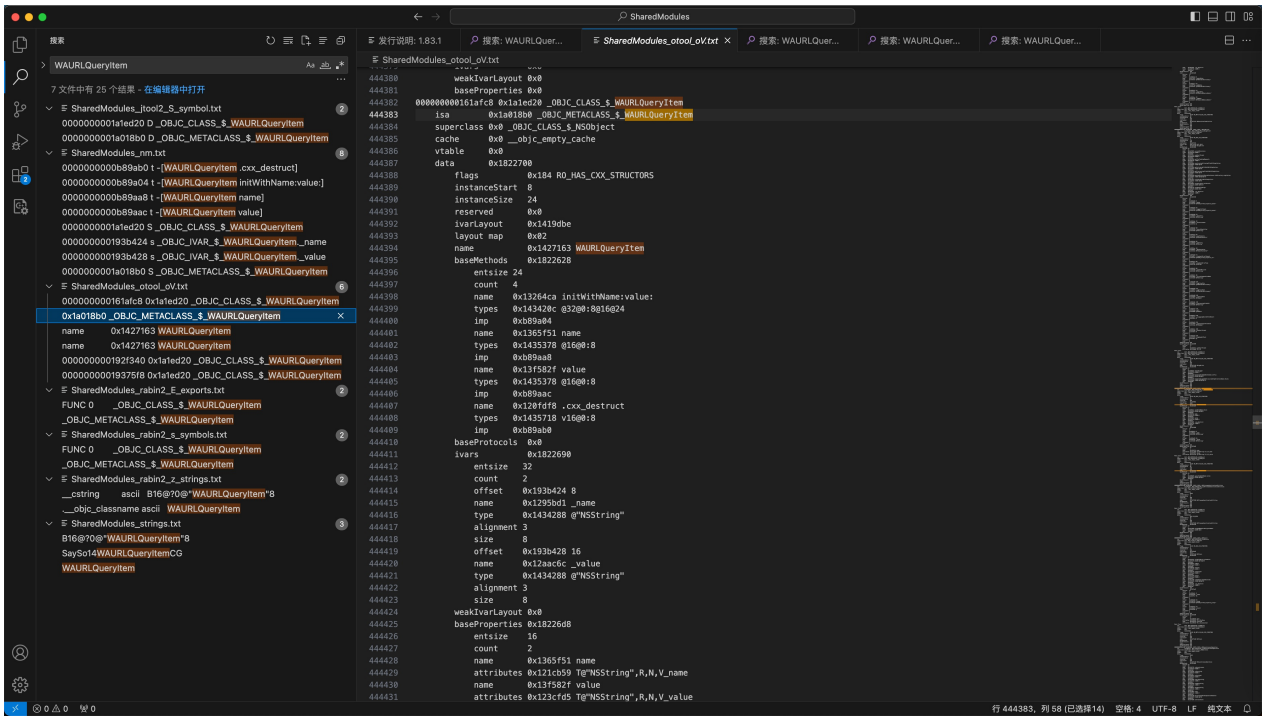
具体现象是：

（1）SharedModules的导出头文件中，能搜到：类WAURLQueryItem

- 

（2）以及SharedModules导出的字符串资源中，也能找到：类WAURLQueryItem





25 个结果 - 7 文件

SharedModules_jtool2_S_symbol.txt:

```
  2880  0000000001952928 D _OBJC_CLASS_$_WATrustSignalsDataFetcher
  2881: 0000000001a1ed20 D _OBJC_CLASS_$_WAURLQueryItem
  2882  0000000001a1ede8 D _OBJC_CLASS_$_WAUUID

  6993  0000000001a26170 D _OBJC_METACLASS_$_WATrustSignalsDataFetcher
  6994: 0000000001a018b0 D _OBJC_METACLASS_$_WAURLQueryItem
  6995  0000000001a1ecf8 D _OBJC_METACLASS_$_WAUUID
```

SharedModules_nm.txt:

```
    25025  00000000000ac75c t -[WATrustSignalsDataFetcher init]
```

```
   25026: 0000000000b89ab0 t -[WAURLQueryItem .cxx_destruct]
   25027: 0000000000b89a04 t -[WAURLQueryItem initWithName:value:]
   25028: 0000000000b89aa8 t -[WAURLQueryItem name]
   25029: 0000000000b89aac t -[WAURLQueryItem value]
   25030  0000000000ee293c t -[WAUUID .cxx_destruct]

   92477  0000000001952928 S _OBJC_CLASS_$_WATrustSignalsDataFetcher
   92478: 0000000001a1ed20 S _OBJC_CLASS_$_WAURLQueryItem
   92479  0000000001a1ede8 S _OBJC_CLASS_$_WAUUID

  101272  0000000001a261b8 d _OBJC_IVAR_$_WATrustSignalsDataFetcher.workQueue
  101273: 0000000000193b424 s _OBJC_IVAR_$_WAURLQueryItem._name
  101274: 0000000000193b428 s _OBJC_IVAR_$_WAURLQueryItem._value
  101275  0000000000193dc64 s _OBJC_IVAR_$_WAUUID._creationDate

  112634  0000000001a26170 D _OBJC_METACLASS_$_WATrustSignalsDataFetcher
  112635: 0000000001a018b0 S _OBJC_METACLASS_$_WAURLQueryItem
  112636  0000000001a1ecf8 S _OBJC_METACLASS_$_WAUUID

SharedModules_otool_oV.txt:
  444381          baseProperties 0x0
  444382: 0000000000161afc8 0x1a1ed20 _OBJC_CLASS_$_WAURLQueryItem
  444383:     isa        0x1a018b0 _OBJC_METACLASS_$_WAURLQueryItem
  444384      superclass 0x0 _OBJC_CLASS_$_NSObject

  444393          layout map     0x02
  444394:         name           0x1427163 WAURLQueryItem
  444395          baseMethods    0x1822628

  444442          ivarLayout     0x0
  444443:         name           0x1427163 WAURLQueryItem
  444444          baseMethods    0x18225c0

  672642  000000000192f338 0x1a02120 _OBJC_CLASS_$_WAStashedMessage
  672643: 000000000192f340 0x1a1ed20 _OBJC_CLASS_$_WAURLQueryItem
  672644  000000000192f348 0x1a16aa8 _OBJC_CLASS_$_WAPBPeerMessage

  676826  00000000019375f0 0x1a01ba8 _OBJC_CLASS_$_WALogWriter
  676827: 00000000019375f8 0x1a1ed20 _OBJC_CLASS_$_WAURLQueryItem
  676828  0000000001937600 0x1a01928 _OBJC_CLASS_$_WAFuture

SharedModules_rabin2_E_exports.txt:
  34621  34617 0x01952928 0x01952928 GLOBAL FUNC 0        _OBJC_CLASS_$_WATrustSignalsD
ataFetcher
  34622: 34618 0x01a1ed20 0x01a1ed20 GLOBAL FUNC 0        _OBJC_CLASS_$_WAURLQueryItem
  34623  34619 0x01a1ede8 0x01a1ede8 GLOBAL FUNC 0        _OBJC_CLASS_$_WAUUID

  38734  38730 0x01a26170 0x01a26170 GLOBAL FUNC 0        _OBJC_METACLASS_$_WATrustSign
alsDataFetcher
  38735: 38731 0x01a018b0 0x01a018b0 GLOBAL FUNC 0        _OBJC_METACLASS_$_WAURLQueryI
tem
  38736  38732 0x01a1ecf8 0x01a1ecf8 GLOBAL FUNC 0        _OBJC_METACLASS_$_WAUUID

SharedModules_rabin2_s_symbols.txt:
  34621  34617 0x01952928 0x01952928 GLOBAL FUNC 0        _OBJC_CLASS_$_WATrustSignalsD
ataFetcher
```

```
  34622: 34618 0x01a1ed20 0x01a1ed20 GLOBAL FUNC 0        _OBJC_CLASS_$_WAURLQueryItem
  34623  34619 0x01a1ede8 0x01a1ede8 GLOBAL FUNC 0        _OBJC_CLASS_$_WAUUID

  38734  38730 0x01a26170 0x01a26170 GLOBAL FUNC 0        _OBJC_METACLASS_$_WATrustSign
alsDataFetcher
  38735: 38731 0x01a018b0 0x01a018b0 GLOBAL FUNC 0        _OBJC_METACLASS_$_WAURLQueryI
tem
  38736  38732 0x01a1ecf8 0x01a1ecf8 GLOBAL FUNC 0        _OBJC_METACLASS_$_WAUUID

SharedModules_rabin2_z_strings.txt:
   18365  13101 0x0114de06 0x0114de06 34   35   5.__TEXT.__cstring         ascii   mms-
base-task//finish/finishing/%@
   18366  13101 0x0114de06 0x0114de06 34   35   5.__TEXT.__cstring         ascii   mms-
base-task//finish/finishing/%@
   18367: 13102 0x0114de29 0x0114de29 24   25   5.__TEXT.__cstring         ascii   B16@
?0@"WAURLQueryItem"8

   18367  13103 0x0114de42 0x0114de42 7    8    5.__TEXT.__cstring         ascii   _nc_
cat
   18368  13103 0x0114de42 0x0114de42 7    8    5.__TEXT.__cstring         ascii   _nc_
cat

  114694  2073  0x0142715e 0x0142715e 4    5    12.__TEXT.__objc_classname ascii   HTML
  114695  2073  0x0142715e 0x0142715e 4    5    12.__TEXT.__objc_classname ascii   HTML
  114696: 2074  0x01427163 0x01427163 14   15   12.__TEXT.__objc_classname ascii   WAUR
LQueryItem
  114696  2075  0x01427172 0x01427172 13   14   12.__TEXT.__objc_classname ascii   Data
Detection
  114697  2075  0x01427172 0x01427172 13   14   12.__TEXT.__objc_classname ascii   Data
Detection

SharedModules_strings.txt:
   17521  mms-base-task//finish/finishing/%@
   17522: B16@?0@"WAURLQueryItem"8
   17523  _nc_cat

  112316  ySo11WAMessageIDCSo0A0C_G
  112317: SaySo14WAURLQueryItemCG
  112318  SDySo11WADeviceJIDC

  114720  HTML
  114721: WAURLQueryItem
  114722  DataDetection
```

具体解释是：

- `SharedModules_jtool2_S_symbol.txt`
  - 中的
    - `2881: 0000000001a1ed20 D _OBJC_CLASS_$_WAURLQueryItem`
    - `6994: 0000000001a018b0 D _OBJC_METACLASS_$_WAURLQueryItem`
  - 其中：
    - **D=** `Defined` = 已定义 = 有定义
    - `_OBJC_CLASS_` = 是ObjC的class类

        - `_OBJC_METACLASS_` = 是ObjC的meta class=元类
- `SharedModules_nm.txt`
    - `25027: 0000000000b89a04 t -[WAURLQueryItem initWithName:value:]`
        - **t=** `Text symbol, local (static)` = 本地的static的文本符号
            - 函数名： -[WAURLQueryItem initWithName:value:]
    - `92478: 0000000001a1ed20 S _OBJC_CLASS_$_WAURLQueryItem`
        - **S=** `Section symbol, global` = 全局的节的符号，此处ObjC的类WAURLQueryItem
    - `101273: 000000000193b424 s _OBJC_IVAR_$_WAURLQueryItem._name`
        - **t=** `Text symbol, local (static)` = 本地的static的文本符号
            - ObjC类WAURLQueryItem的ivar变量名_name
    - `112635: 0000000001a018b0 S _OBJC_METACLASS_$_WAURLQueryItem`
        - **S=** `Section symbol, global` = 全局的节的符号，此处是ObjC的MetaClass元类WAURLQueryItem
- `SharedModules_otool_oV.txt`
    - `444382: 000000000161afc8 0x1a1ed20 _OBJC_CLASS_$_WAURLQueryItem`
        - 是有ObjC的类：WAURLQueryItem
- `SharedModules_rabin2_E_exports.txt`
    - 中的
        - `34622: 34618 0x01a1ed20 0x01a1ed20 GLOBAL FUNC 0 _OBJC_CLASS_$_WAURLQueryItem`
        - `38735: 38731 0x01a018b0 0x01a018b0 GLOBAL FUNC 0 _OBJC_METACLASS_$_WAURLQueryItem`
    - 表示：
        - 有ObjC的普通类WAURLQueryItem 和ObjC的元类WAURLQueryItem，要导出export （供别处调用）
- `SharedModules_rabin2_s_symbols.txt`
    - 中的
        - `34622: 34618 0x01a1ed20 0x01a1ed20 GLOBAL FUNC 0 _OBJC_CLASS_$_WAURLQueryItem`
        - `38735: 38731 0x01a018b0 0x01a018b0 GLOBAL FUNC 0 _OBJC_METACLASS_$_WAURLQueryItem`
    - 表示
        - 有global全局的类：ObjC普通类和元类：WAURLQueryItem的 s=symbol符号
- `SharedModules_rabin2_z_strings.txt`
    - `114696: 2074 0x01427163 0x01427163 14 15 12.__TEXT.__objc_classname ascii WAURLQueryItem`
        - 表示：WAURLQueryItem是个Text节（Section）的ObjC的类名 `__objc_classname`
- `SharedModules_strings.txt`
    - 中的
        - `17522: B16@?0@"WAURLQueryItem"8`
        - `112317: SaySo14WAURLQueryItemCG`
        - `114721: WAURLQueryItem`
    - 表示
        - 有类WAURLQueryItem相关的字符串（类名）和函数名等内容

附录

下面列出相关参考资料。

参考资料

- 【已解决】class-dump导出砸壳后抖音ipa的头文件为空
- 【记录】用jtool查看抖音二进制信息
- 【记录】静态分析Mask的动态库：MaskPro.dylib
- 【记录】静态分析Mask的动态库：Mask.dylib
- 【记录】静态分析黑豹动态库zzzzHeiBaoLib.dylib
- 【记录】用radare2查看抖音二进制信息
- 【记录】用rabin2查看抖音AwemeCore二进制的信息
- 【已解决】Mac M2 Max中安装和使用jtool2
- 【规避解决】Mac M2 Max中jtool2运行崩溃：killed
- 【记录】用strings查看dylib库中包含的字符串
- 【记录】研究黑豹dylib文件zzzzHeiBaoLib.dylib中字符串和反越狱相关内容
- 【记录】研究佐罗dylib文件zorro.dylib中字符串和反越狱相关内容
- 【记录】用Cutter查看分析抖音AwemeCore二进制文件信息
- 【未解决】iOS逆向WhatsApp：-[WARegistrationURLBuilder verificationCodeRequestURLWithBaseURL:method:mcc:mnc:jailbroken:context:oldPhoneNumber:silentPushNotifRegCode:cellularStrength:]
- 【已解决】iOS逆向WhatsApp：寻找类WAURLQueryItem的定义
- 【已解决】nm输出的函数符号类型T、S、U、s的含义
- 【已解决】iOS逆向WhatsApp：静态分析WhatsApp二进制
- 【整理】Mac中2个版本的ldid：brew版和iOSOpenDev版
- 【已解决】修改iOS二进制权限工具：ldid
- 【记录】写shell脚本自动化导出安卓ELF的so库文件的静态字符串等资源内容
- 【记录】静态分析黑豹二进制HeiBao
- 【未解决】静态分析符号函数字符串：Aweme
- 【未解决】静态分析符号函数字符串：AwemeCore
- 【已解决】iOS逆向：mobileactivationd导出头文件支持Swift
- 【记录】iOS逆向改机激活：mobileactivationd导出头文件
- 【已解决】Mac M2 Max中获取lechium/classdumpios的二进制classdumpc
- 【已解决】iOS逆向：paradiseduo/dsdump导出Swift头文件信息
- 【已解决】iOS逆向：导出头文件信息对比：dsdump vs class-dump
-
- iOS逆向开发
- iOS逆向开发：砸壳ipa
-
- OS Internals: (newosxbook.com)
- IOS逆向初探 · 浮萍's Blog (fuping.site)
- iOS逆向之Reveal、Hopper、MachOView等逆向工具的安装使用 - 简书 (jianshu.com)
- Mach-O Programming Topics Introduction
- IOS逆向初探 · 浮萍's Blog (fuping.site)
- iOS逆向之Reveal、Hopper、MachOView等逆向工具的安装使用 - 简书 (jianshu.com)
- poboke/Class-Decompile: Class Decompile is a python script for Hopper Disassembler. This script can export pseudo code of the classes. (github.com)

- 【OSG】jtool - Taking the O out of otool(1)-iOS安全-看雪论坛-安全社区|安全招聘|bbs.pediy.com
- JTool2 - Taking the O out of otool - squared (newosxbook.com)
- Program Sections - The Official Radare2 Book
- Strings - The Official Radare2 Book
- Libraries - The Official Radare2 Book
- Symbols - The Official Radare2 Book
- Exports - The Official Radare2 Book
- Imports - The Official Radare2 Book
- Entrypoint - The Official Radare2 Book
- File Identification - The Official Radare2 Book
- Rabin2 - The Official Radare2 Book
- Introduction - The Official Radare2 Book
- git.saurik.com Git - ldid.git/summary
- xerub/ldid: Unofficial fork from saurik git repository git://git.saurik.com/ldid.git (github.com)
-