

# iDecompile: Writing an iOS Decompiler

Laurie Kirk



# whoami

- Laurie Kirk
- Reverse Engineer
- Specialize in cross-platform malware with a focus on mobile threats
- Run YouTube channel  
[@lauriewired](https://www.youtube.com/@lauriewired)



@lauriewired



# DECOMPILERS



# THOUSANDS OF APPS



# LITTLE RESEARCH

# It's Hard.



IPA



Mach-O

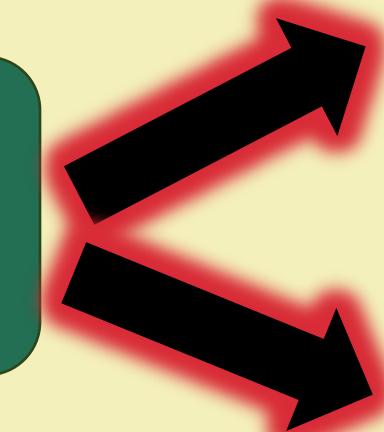


Decompiler

lipo

plutil

otool



**Goal:**  
Simplify iOS Analysis

# OUTCOME

- Platform-Agnostic iOS RE
- Reduce manual analysis
- Increase iOS app research

# MALIMITE



# IPA FILES

- Main application file for iOS
- Contains code, libs, permissions, and resources

**Extract** the payload

**Decode** the resources

**Locate** the main Mach-O executable

**Find Entry** in binary

*Important*  
resources are neglected.

# FOUR PILLARS

Permissions

Resources

Executable

Libraries /  
Frameworks

# **PLAN:**

Analyze each pillar on multiple platforms

# Pillar 1



Permissions

*Understand*  
application capabilities

# **PRIMARY PERMISSION LOCATIONS**

**Info.plist**

**embedded.mobileprovision**

# INFO.plist

- Permissions granted by the user at runtime
- Example: camera, location

# EMBEDDED.MOBILEPROVISION

- Entitlements granted by the system
- Example: iCloud, push notifications

# PROPERTY LISTS

- Binary or XML formats
- More than just the Info.plist!

# XML PLISTS ARE SIMPLE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>CFBundleName</key>
    <string>MyApp</string>
    <key>CFBundleIdentifier</key>
    <string>com.example.myapp</string>
    <key>CFBundleVersion</key>
    <string>1.0</string>
    <key>CFBundleShortVersionString</key>
    <string>1.0.0</string>
    <key>NSCameraUsageDescription</key>
    <string>This app requires access to the camera to take photos.</string>
    <key>NSLocationWhenInUseUsageDescription</key>
    <string>This app uses your location to provide recommendations.</string>
    <key>UIBackgroundModes</key>
    <array>
        <string>location</string>
        <string>audio</string>
    </array>
</dict>
</plist>
```

Binary **plists** are a custom  
Apple format.

# DECODING LIBRARIES

Mac

Linux

Windows



plutil

libplist

plistlib

Unique  
tool per platform?

# PROPRIETARY PLIST REVERSED

```
private static int determineType(InputStream is, int offset) throws IOException {
    int index = offset;
    int readLimit = index + 1024;
    if (is.markSupported()) {
        is.mark(readLimit);
    }

    is.skip(offset);
    int b;
    ByteOrderMarkReader bomReader = new ByteOrderMarkReader();
    boolean bom = true;
    //Skip any possible whitespace at the beginning of the file
    do {
        if (++index > readLimit) {
            is.reset();
            return determineType(is, readLimit);
        }
        b = is.read();
        //Check if we are reading the Unicode byte order mark (BOM) and skip it
        bom &= bomReader.readByte(b);
    } while (b != -1 && (b == ' ' || b == '\t' || b == '\r' || b == '\n' || b == '\f' || bom));
}
```

```
bplist00?•#.....  
••  
..... ! "#$%&'/0&1235>?  
@AB@CDEFHIJKLMNOPHQRUVX\_\_BuildMachineOS  
undleInfoDictionaryVersion\CFBundleName  
ypes_\_CFBundleVersionZDTCompiler_\_DTP  
plicationQueriesSchemes_\_LSRequiresIP  
oneUsageDescription_\_NSPhotoLibraryAdd
```



```
"MinimumOSVersion": "11.1",  
"UIStatusBarStyle": "UIStatusBarStyleDefault",  
"CFBundleIdentifier": "fun.Animoji",  
"DTXcodeBuild": "9C40b",  
"NSMicrophoneUsageDescription": "I need access to the microphone to record audio",  
"CFBundleExecutable": "AnimojiStudio",  
"LSRequiresiPhoneOS": true,  
"NSPhotoLibraryAddUsageDescription": "I need access to your photo library to save your video",  
"BuildMachineOSBuild": "17C88",  
"CFBundlePackageType": "APPL",  
"LSApplicationQueriesSchemes": [  
    "spotify-action"
```

# EMBEDDED.MOBILPROVISION

- Part property list
- Part PKCS #7 signature
  - Signed by Apple

Less popular,  
less research

# Part plist

```
<key>TimeToLive</key>
<integer>222</integer>
<key>UUID</key>
<string>749f009f-a7ac-4b9a-83fb-15fd3c6b210e</string>
<key>Version</key>
<integer>1</integer>
</dict>
</plist>00
00•00••••••0
• *0H00
••••0b1•0 ••U•••US1•0•••U•
•
Apple Inc.1&0$•U••••Apple Certification Authority1•0•••U••
Apple Root CA0••
070412174328Z•
220412174328Z0y1•0 ••U•••US1•0•••U•
•
Apple Inc.1&0$•U••••Apple Certification Authority1-0+••U•••$Apple iPhone Certification Authority0•"0
• *0H00
•••••0•
•••••0•••0•
•••••0•••0••[F00!00•!p(E`\••
dc000i00T00[•N•/0k30DL0K0 000[00dir••••••••a0*x00Y^0i0d•0•000PF 0•0尔
m00000AN00e0•Rz00•0n•0•U00)XI•0G0•43000D]p0,0*070••000l08UV05+X0D0&000fJ0w0=c0)0Z0•000 N0•5e00
```

Part  
certificate

Extract and decode plist

```
embedded.mobileprovision x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
3 <plist version="1.0">
4 <dict>
5   <key>AppIDName</key>
6   <string>AnimojiStudio</string>
7   <key>ApplicationIdentifierPrefix</key>
8   <array>
9     <string>8C7439RJLG</string>
10  </array>
11  <key>CreationDate</key>
12  <date>2017-11-14T14:43:41Z</date>
13  <key>Platform</key>
14  <array>
15    <string>iOS</string>
16  </array>
17  <key>DeveloperCertificates</key>
18  <array>
19    <data>
20      MIIFoTCCBImgAwIBAgIILokx/b0wKe0wDQYJKoZIhvcNAQE
21    </data>
22  </array>
23  <key>Entitlements</key>
24  <dict>
25    <key>keychain-access-groups</key>
26    <array>
27      <string>8C7439RJLG.*</string>
28    </array>
29    <key>get-task-allow</key>
```

# Pillar 2



Resources

Supporting **resources** are spread throughout the IPA

Various proprietary formats.

# PLISTS

- Sometimes encoded
- Metadata, identifiers, URL schemes

# Plists



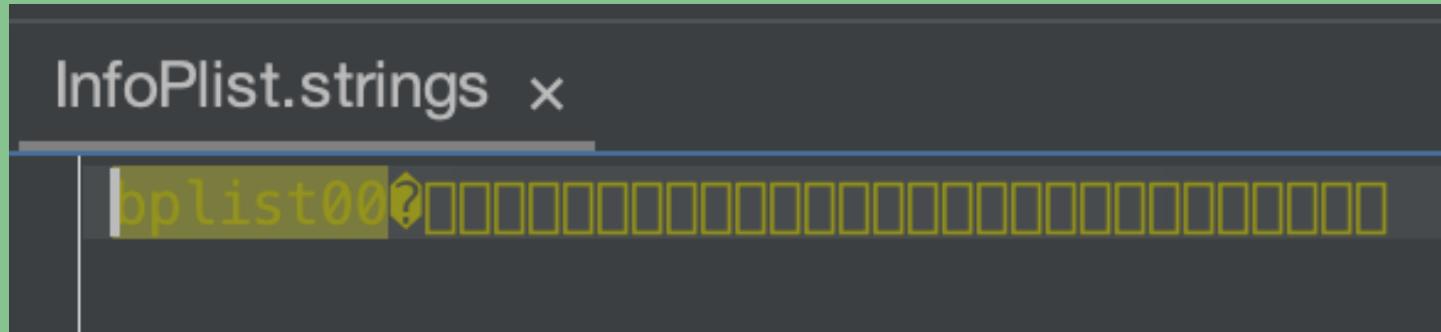
# LOCALIZED STRINGS

- User-facing text
- Labels, error messages, dev strings

# LOCALIZED STRING FILES

- Stored in .strings
- Multiple languages bundled into IPA

This looks *familiar*....



Always has been

Wait, it's all plists?



# Localized Strings



# THE RESOURCES NEVER END

- Nib
- Xib
- Storyboard

Parse *strings* from encoded files.

## Resource Strings

| Value   | File                | Type  |
|---|---------------------|-------|
| "CFBundleShortVersionString": "1.0",                              | ObjectAL-Info.plist | plist |
| "NSPrincipalClass": "",   | ObjectAL-Info.plist | plist |
| "CFBundleIdentifier": "org.stenerud.\${PRODUCT_NAME}:rfc1034...", | ObjectAL-Info.plist | plist |
| "CFBundleName": "\${PRODUCT_NAME}",                               | ObjectAL-Info.plist | plist |
| "CFBundleInfoDictionaryVersion": "6.0",                           | ObjectAL-Info.plist | plist |
| "CFBundleExecutable": "\${EXECUTABLE_NAME}",                      | ObjectAL-Info.plist | plist |
| "CFBundleSignature": "????"                                       | ObjectAL-Info.plist | plist |
| "rules": {  | ResourceRules.plist | plist |
| ".*": true,   | ResourceRules.plist | plist |
| "ResourceRules.plist": {  | ResourceRules.plist | plist |
| "weight": 100.0,  | ResourceRules.plist | plist |
| "omit": true  | ResourceRules.plist | plist |
| "Info.plist": {   | ResourceRules.plist | plist |
| "weight": 10.0,   | ResourceRules.plist | plist |
| "omit": true  | ResourceRules.plist | plist |

# Pillar 3



Executable

# INFO.plist

- Defines the main executable binary

```
<key>CFBundleDisplayName</key>
<string>Flappy Bird</string>
<key>CFBundleExecutable</key>
<string>flap</string>
```

# ACCOUNT FOR UNIVERSAL BINARIES

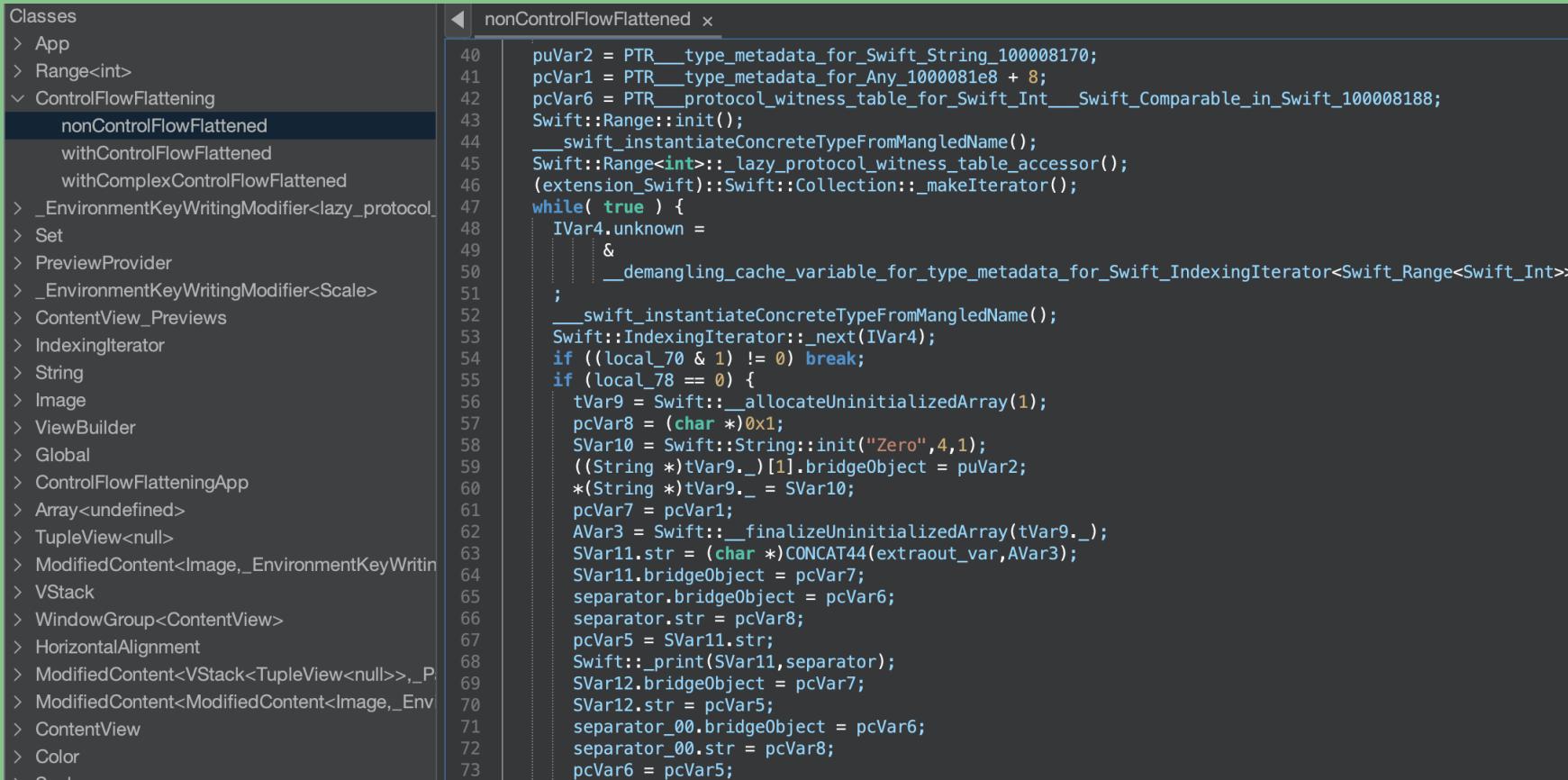
| <b>Field</b> | <b>Size</b> | <b>Description</b>                         |
|--------------|-------------|--|
| magic        | 4 bytes     | Magic number (indicates fat binary type).  |
| nfat_arch    | 4 bytes     | Number of architectures in the binary.     |
| cputype      | 4 bytes     | CPU type (e.g., arm64, x86_64).            |
| cpusubtype   | 4 bytes     | CPU subtype (e.g., armv7, arm64e).         |
| offset       | 4 bytes     | Offset to the Mach-O binary for this arch. |
| size         | 4 bytes     | Size of the Mach-O binary for this arch.   |
| align        | 4 bytes     | Alignment of the binary (power of 2).      |

# EXTRACT ARCH SLICE FOR DECOMP

```
laurie@Mac AirRec.app % otool -hv AirRec
AirRec (architecture armv7):
Mach header
    magic  cputype cpusubtype  caps      filetype ncmds sizeofcmds
    flags
    MH_MAGIC      ARM          V7  0x00      EXECUTE    31      3520
    NOUNDEFS DYLDLINK TWOLEVEL WEAK_DEFINES BINDS_TO_WEAK PIE
AirRec (architecture arm64):
Mach header
    magic  cputype cpusubtype  caps      filetype ncmds sizeofcmds
    flags
MH_MAGIC_64      ARM64        ALL  0x00      EXECUTE    31      4048
    NOUNDEFS DYLDLINK TWOLEVEL WEAK_DEFINES BINDS_TO_WEAK PIE
```



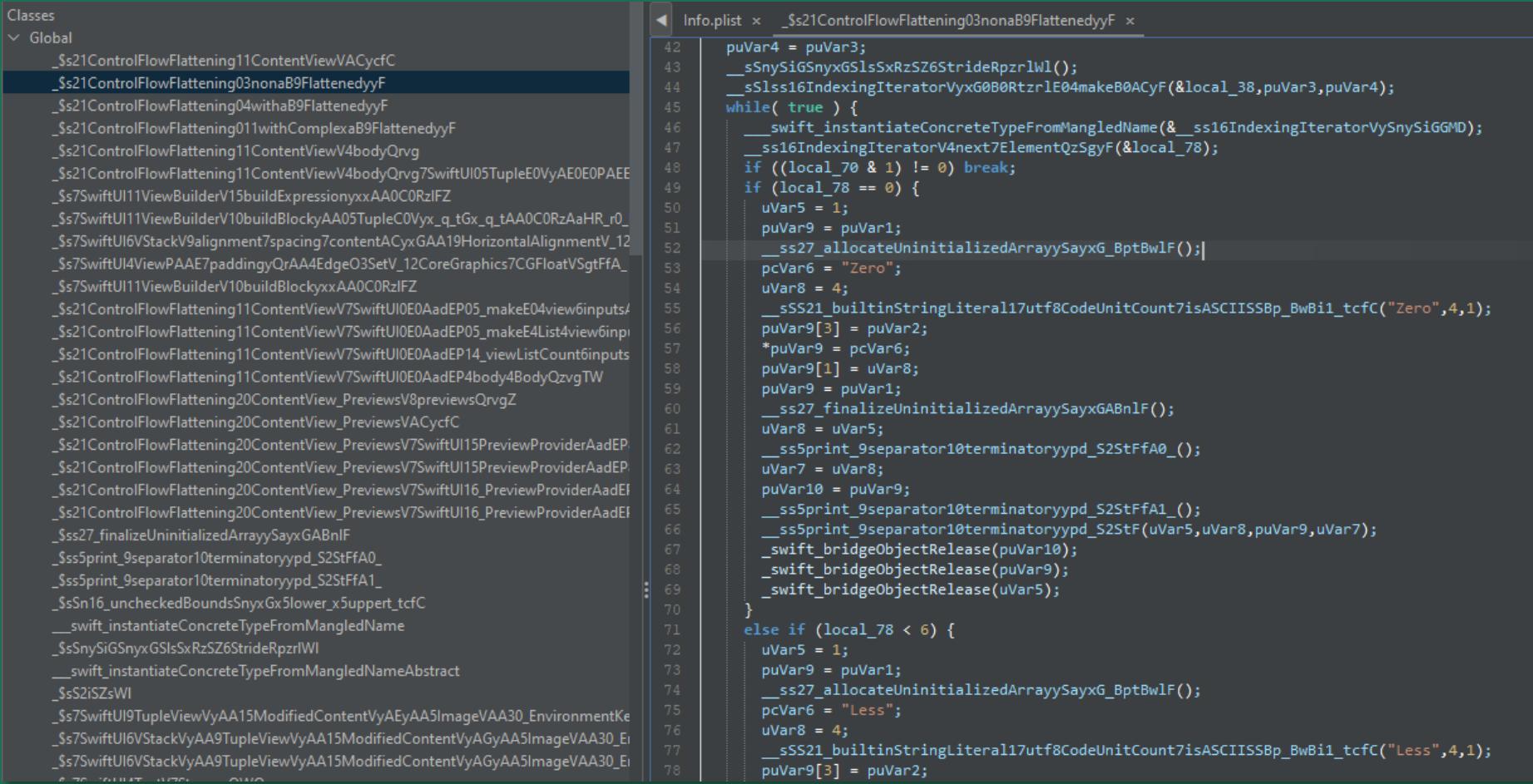
# LOOKS GREAT ON MAC!



The screenshot shows the Xcode interface with the sidebar on the left displaying a list of classes under the 'Classes' section. The 'nonControlFlowFlattened' class is selected and highlighted in blue. The main editor area on the right contains the source code for this class, which is heavily obfuscated. The code includes numerous memory addresses (e.g., puVar2, pcVar1, pcVar6) and Swift type metadata pointers (e.g., Swift::Range<int>, Swift::String). The code appears to be generated by the Swift optimizer or linker, showing how complex control flow flattening is handled at the assembly level.

```
40  puVar2 = PTR__type_metadata_for_Swift_String_100008170;
41  pcVar1 = PTR__type_metadata_for_Any_1000081e8 + 8;
42  pcVar6 = PTR__protocol_witness_table_for_Swift_Int__Swift_Comparable_in_Swift_100008188;
43  Swift::Range::init();
44  __swift_instantiateConcreteTypeFromMangledName();
45  Swift::Range<int>::_lazy_protocol_witness_table_accessor();
46  (extension_Swift)::Swift::Collection::_makeIterator();
47  while( true ) {
48      IVar4.unknown =
49          &
50          __demangling_cache_variable_for_type_metadata_for_Swift_IndexingIterator<Swift_Range<Swift_Int>>
51          ;
52          __swift_instantiateConcreteTypeFromMangledName();
53  Swift::IndexingIterator::_next(IVar4);
54  if ((local_70 & 1) != 0) break;
55  if (local_78 == 0) {
56      tVar9 = Swift::__allocateUninitializedArray(1);
57      pcVar8 = (char *)0x1;
58      SVar10 = Swift::String::init("Zero",4,1);
59      ((String *)tVar9._)[1].bridgeObject = puVar2;
60      *(String *)tVar9._ = SVar10;
61      pcVar7 = pcVar1;
62      AVar3 = Swift::__finalizeUninitializedArray(tVar9._);
63      SVar11.str = (char *)CONCAT44(extraout_var,AVar3);
64      SVar11.bridgeObject = pcVar7;
65      separator.bridgeObject = pcVar6;
66      separator.str = pcVar8;
67      pcVar5 = SVar11.str;
68      Swift::_print(SVar11,separator);
69      SVar12.bridgeObject = pcVar7;
70      SVar12.str = pcVar5;
71      separator_00.bridgeObject = pcVar6;
72      separator_00.str = pcVar8;
73      pcVar6 = pcVar5;
```

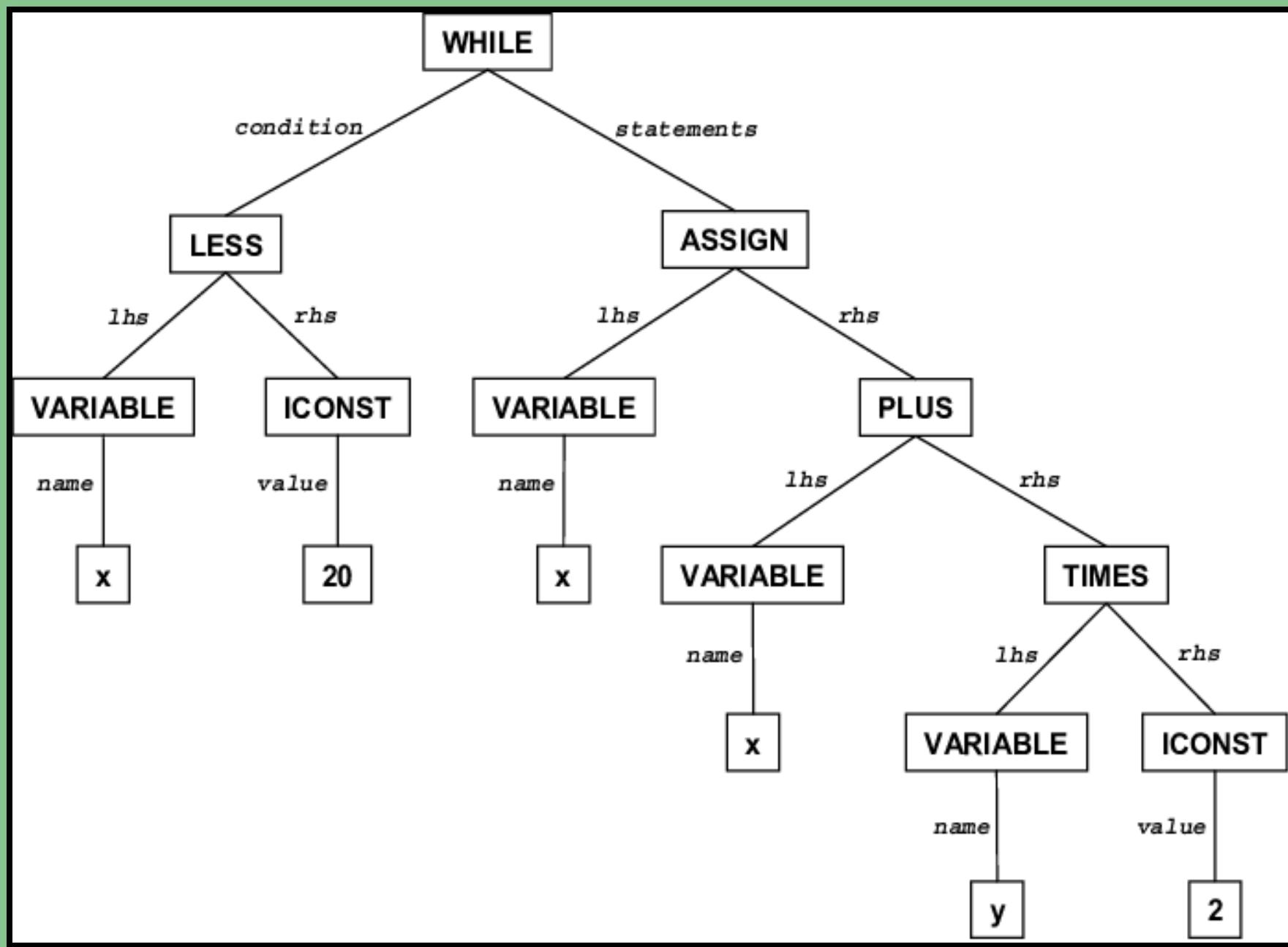
# NOT SO GREAT ELSEWHERE...



The image shows a screenshot of the Xcode IDE. On the left, there is a 'Classes' sidebar with a tree view. Under the 'Global' section, several classes are listed, including `$_$21ControlFlowFlattening03nonaB9FlattenedyyF`, which is highlighted. To the right of the sidebar is a large code editor window displaying a C++ file named `Info.plist x _$s21ControlFlowFlattening03nonaB9FlattenedyyF x`. The code contains numerous calls to Swift bridge functions like `__swift_instantiateConcreteTypeFromMangledName` and `__swift_bridgeObjectRelease`, along with various local variables and control structures. The code is heavily annotated with comments and includes several `__ss27_allocateUninitializedArrayySayxG_BptBwlF()` calls.

```
42 puVar4 = puVar3;
43 __SnySiGSnyxGSlsSxRzSZ6StrideRpzrlWl();
44 __SlsSs16IndexingIteratorVyxG0B0RtzrlE04makeB0ACyF(&local_38,puVar3,puVar4);
45 while( true ) {
46     __swift_instantiateConcreteTypeFromMangledName(&__ss16IndexingIteratorVySnySiGGMD);
47     __ss16IndexingIteratorV4next7ElementQzSgyF(&local_78);
48     if ((local_70 & 1) != 0) break;
49     if (local_78 == 0) {
50         uVar5 = 1;
51         puVar9 = puVar1;
52         __ss27_allocateUninitializedArrayySayxG_BptBwlF();
53         pcVar6 = "Zero";
54         uVar8 = 4;
55         __sSS21_builtinStringLiteral17utf8CodeUnitCount7isASCIISBp_BwBil_tcfC("Zero",4,1);
56         puVar9[3] = puVar2;
57         *puVar9 = pcVar6;
58         puVar9[1] = uVar8;
59         puVar9 = puVar1;
60         __ss27_finalizeUninitializedArrayySayxGABnlF();
61         uVar8 = uVar5;
62         __ss5print_9separator10terminatorypd_S2StFFa0_();
63         uVar7 = uVar8;
64         puVar10 = puVar9;
65         __ss5print_9separator10terminatorypd_S2StFFa1_();
66         __ss5print_9separator10terminatorypd_S2StFFa1_();
67         __swift_bridgeObjectRelease(puVar10);
68         __swift_bridgeObjectRelease(puVar9);
69         __swift_bridgeObjectRelease(uVar5);
70     }
71     else if (local_78 < 6) {
72         uVar5 = 1;
73         puVar9 = puVar1;
74         __ss27_allocateUninitializedArrayySayxG_BptBwlF();
75         pcVar6 = "Less";
76         uVar8 = 4;
77         __sSS21_builtinStringLiteral17utf8CodeUnitCount7isASCIISBp_BwBil_tcfC("Less",4,1);
78         puVar9[3] = puVar2;
```

Parse code to  
**reconstruct**  
classes



# MODIFIED C++ GRAMMAR

- Handle Swift namespace declarations
- Tolerate incorrectly formatted code

s21ControlFlowFlattening03nonaB9FlattenedyyF



void ControlFlowFlattening::nonControlFlowFlattened (void)

Classes

> ControlFlowFlattening

▽ SwiftUI

ViewBuilderbuildExpression

ViewBuilderbuildBlockTuple

VStackalignmentspacingcontentHorizontalAlignmentCoreGraphicsCGFloat

ViewpaddingEdgeSetCoreGraphicsCGFloat

ViewBuilderbuildBlock

TupleViewModifiedContentImage\_EnvironmentKeyWritingModifierScaleColorText

VStackTupleViewModifiedContentImage\_EnvironmentKeyWritingModifierScaleColorText

VStackTupleViewModifiedContentImage\_EnvironmentKeyWritingModifierScaleColorText

TextStorage

ModifiedContentVStackTupleViewImage\_EnvironmentKeyWritingModifierScaleColorText

ModifiedContentVStackTupleViewImage\_EnvironmentKeyWritingModifierScaleColorText

ModifiedContentVStackTupleViewImage\_EnvironmentKeyWritingModifierScaleColorText

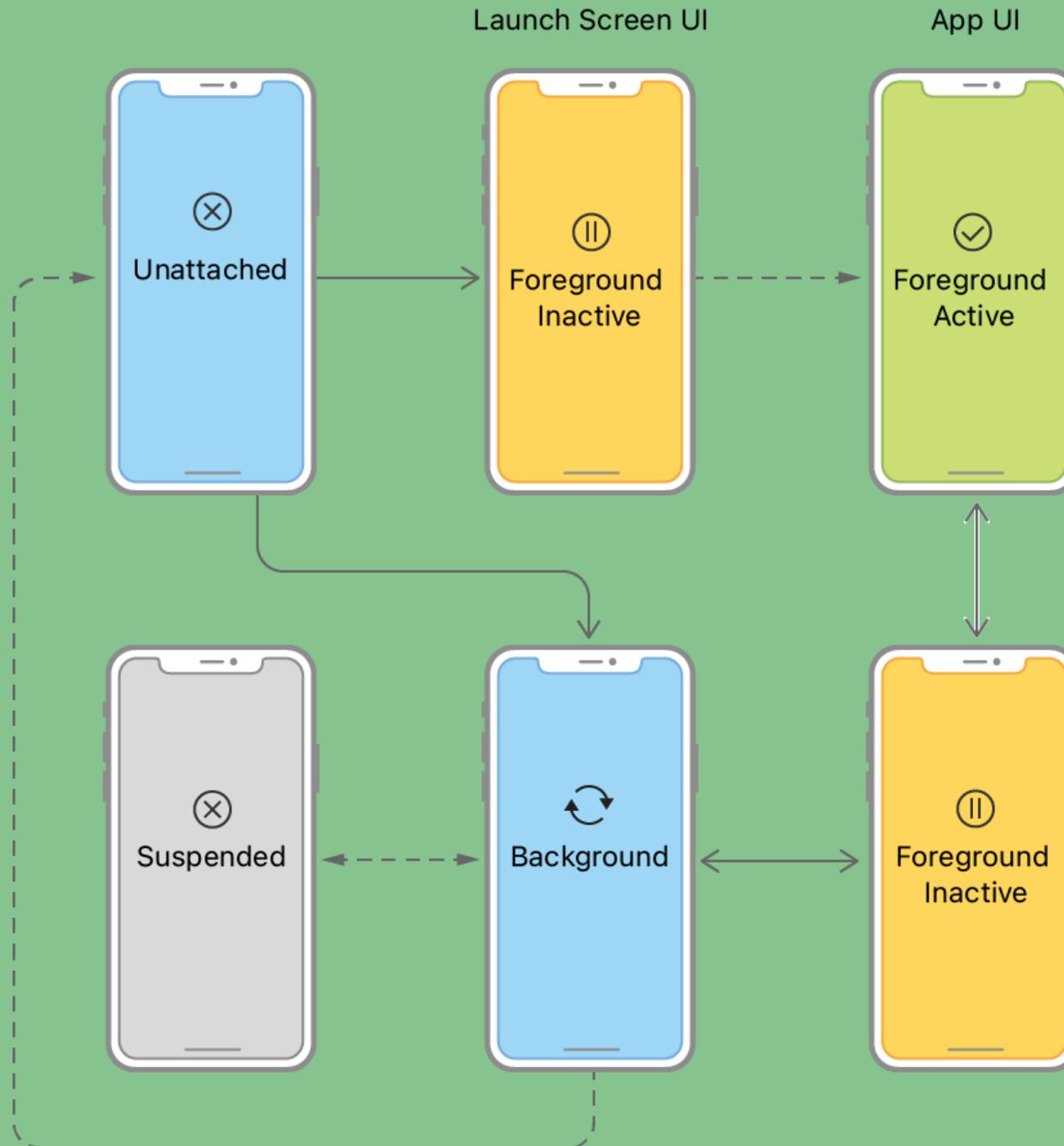
TextStorage

Image

ModifiedContentImage\_EnvironmentKeyWritingModifierScaleViewaM

\_EnvironmentKeyWritingModifierImageScaleView

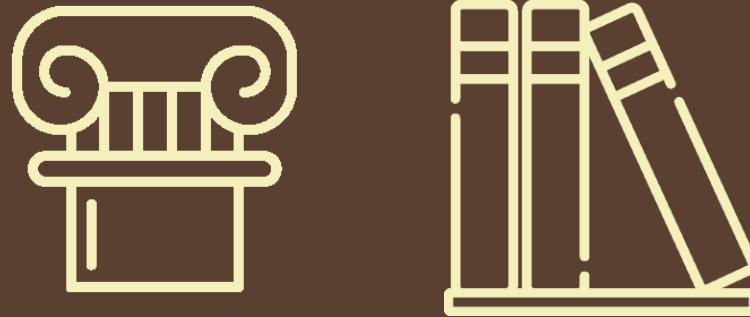
Where do we start **analysis**?



# COMMON ENTRYPPOINTS

- applicationDidFinishLaunchingWithOptions
- applicationDidBecomeActive
- applicationDidEnterBackground
- applicationWillTerminate

# Pillar 4



Libraries and Frameworks

Why are iOS apps so

LARGE ?

# Libraries and frameworks are often bundled.

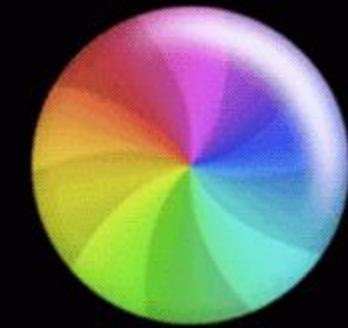
## Classes

- > ASAppearance
- > ASColorWell
- > ASPuppetView
- > AppDelegate
- > ColorSheetPresentationController
- > ColorSheetViewController
- > ErrorViewController
- > Global
- > KaraokeFlowController
- > PuppetCollectionViewCell
- > PuppetSelectionViewController
- > RecordingCoordinator
- > RecordingFlowController
- > RecordingSettingsViewController
- > RecordingStatusViewController
- > RecordingViewController
- > SPTAlbum
- > SPTArtist
- > SPTAudioStreamingController
- > SPTAuth
- > SPTAuthViewController
- > SPTBrowse
- > SPTCircularBuffer
- > SPTConnectButton
- > SPTCoreAudioController



# HANDS ON:

Wait for this app  
to decompile



How can we

*distinguish*

libraries?

## Audio and Video

[AVFoundation](#)[Audio](#)[AirPlay](#)

# AVFoundation

AVFoundation is the full featured framework for working with time-based audiovisual media on iOS, macOS, watchOS and tvOS. Using AVFoundation, you can easily play, create, and edit QuickTime movies and MPEG-4 files, play HLS streams, and build powerful media functionality into your apps.

# READ DOCUMENTATION, NOT DECOMPILE

[AVFoundation](#) / Media reading and writing

API Collection

## Media reading and writing

Read images from video, export to alternative formats, and perform sample-level reading and writing of media data.

### Topics

#### Media export

 [Exporting video to alternative formats](#)

Convert an existing movie file to a different format.

`class AVAssetExportSession`

An object that exports assets in a format that you specify using an export preset.

# RUNTIME METHODS

```
:local_38 = PTR_PTR_1000af8f0;
local_40 = param_1;
objc_msgSendSuper2(&local_40,PTR_s_viewDidLoad_1000ad278);
puVar1 = PTR_s_view_1000ad2a8;
objc_msgSend(param_1,PTR_s_view_1000ad2a8);
uVar2 = _objc_retainAutoreleasedReturnValue();
objc_msgSend(PTR_OBJC_CLASS__UIColor_1000af460,PTR_s whiteColor_1000ad5d8);
uVar3 = _objc_retainAutoreleasedReturnValue();
objc_msgSend(uVar2,PTR_s_setBackgroundColor__1000ad5e0,uVar3);
objc_release(uVar3);
objc_release(uVar2);
objc_msgSend(param_1,puVar1);
uVar2 = _objc_retainAutoreleasedReturnValue();
objc_msgSend(uVar2,PTR_s_setOpaque__1000ad5e8,1);
objc_release(uVar2);
objc_msgSend(param_1,PTR_s_navigationItem_1000ade38);
uVar2 = _objc_retainAutoreleasedReturnValue();
objc_msgSend(uVar2,PTR_s_setLargeTitleDisplayMode__1000ae4b8,2);
objc_release(uVar2);
objc_msgSend(param_1,PTR_sSetTitle__1000ad5f0,&cf_Record);
return;
```

# RUNTIME METHODS

## Objective-C

- `objc_msgSend`
- `objc_release`

## Swift

- `swift_demangle`
- `swift_allocObject`

```
Class cls = swift_getClassByName("MyApp.MyClass");
void *instance = swift_allocObject(cls, sizeof(MyClass), alignof(MyClass));
```



```
let instance = MyClass()
```

# Pillar 1



Permissions ✓

# Pillar 2



Resources ✓

# Pillar 3



Executable ✓

# Pillar 4



Libraries and Frameworks ✓

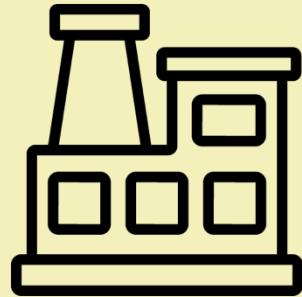
One more thing...

```
Swift::Range<int>::_lazy_protocol_witness_table_accessor();
(extension_Swift)::Swift::Collection::_makeIterator();
while( true ) {
    IVar4.unknown =
        &
        __demangling_cache_variable_for_type_metadata_for_Swift_IndexingIterator<Swift_Range<Swift_Int>>;
    __swift_instantiateConcreteTypeFromMangledName();
    Swift::IndexingIterator::_next(IVar4);
    if ((local_70 & 1) != 0) break;
    if (local_78 == 0) {
        tVar9 = Swift::__allocateUninitializedArray(1);
        pcVar8 = (char *)0x1;
        SVar10 = Swift::String::init("Zero",4,1);
        ((String *)tVar9._)[1].bridgeObject = puVar2;
        *(String *)tVar9._ = SVar10;
        pcVar7 = pcVar1;
        AVar3 = Swift::__finalizeUninitializedArray(tVar9._);
        SVar11.str = (char *)CONCAT44(extraout_var,AVar3);
        SVar11.bridgeObject = pcVar7;
        separator.bridgeObject = pcVar6;
        separator.str = pcVar8;
        pcVar5 = SVar11.str;
        Swift::_print(SVar11,separator);
        SVar12.bridgeObject = pcVar7;
        SVar12.str = pcVar5;
        separator_00.bridgeObject = pcVar6;
        separator_00.str = pcVar8;
        pcVar6 = pcVar5;
        pcVar8 = pcVar7;
        Swift::_print(SVar12,separator_00);
        SVar13.bridgeObject = pcVar5;
        SVar13.str = SVar11.str;
        separator_01.bridgeObject = pcVar6;
        separator_01.str = pcVar7;
        Swift::_print(SVar13,separator_01);
        _swift_bridgeObjectRelease(pcVar8);
        _swift_bridgeObjectRelease(pcVar7);
        _swift_bridgeObjectRelease(SVar11.str);
    }
}
```

# APPROXIMATE CODE IN TARGET LANGUAGES



C-Pseudocode



LLM Processing



Swift Approximation

```
// Class: ControlFlowFlattening
// Function: nonControlFlowFlattened

func nonControlFlowFlattened() {
    var local78: Int = 0 // Placeholder for local_78
    var local70: UInt8 = 0 // Placeholder for local_70
    let puVar2 = typeMetadataForSwiftString() // Placeholder for puVar2
    let pcVar1 = typeMetadataForAny() + 8 // Placeholder for pcVar1
    let pcVar6 = protocolWitnessTableForSwiftIntComparable() // Placeholder for pcVar6
    // Assuming the necessary Swift constructs replacing the low-level operations in the C-like code:
    while true {
        // This is a placeholder next.
        // The .next() method should be properly adjusted according to the actual iterable used in Swift
        local70 &= 1
        if local70 != 0 {
            break
        }

        if local78 == 0 {
            let tVar9 = allocateUninitializedArray(1) // Placeholder for tuple allocation
            let sVar10 = "Zero"
            print(sVar10)
            // Releasing memory if needed or using ARC in Swift, this can be omitted.
        } else if local78 < 6 {
            let tVar9 = allocateUninitializedArray(1) // Placeholder for tuple allocation
            let sVar10 = "Less"
            print(sVar10)
            // Releasing memory if needed or using ARC in Swift, this can be omitted.
        } else {
            let tVar9 = allocateUninitializedArray(1) // Placeholder for tuple allocation
            let sVar10 = "Greater"
            print(sVar10)
            // Releasing memory if needed or using ARC in Swift, this can be omitted.
        }
    }
}
```

# Introducing Malimite

The screenshot shows the Malimite application window titled "ControlFlowFlattening.ipa". The main pane displays the "Info.plist" file content, which includes various metadata keys such as CFBundleDevelopmentRegion, MinimumOSVersion, CFBundleIdentifier, DTXcodeBuild, CFBundleExecutable, LSRequiresiPhoneOS, BuildMachineOSBuild, UIApplicationSupportsIndirectInputEvents, CFBundlePackageType, UISupportedInterfaceOrientations~iphone, DTCompiler, CFBundleName, CFBundleSupportedPlatforms, UISupportedInterfaceOrientations~ipad, DTPlatformBuild, UIApplicationSceneManifest, UISceneConfigurations, UIApplicationSupportsMultipleScenes, CFBundleInfoDictionaryVersion, DTSDKBuild, DTXcode, CFBundleVersion, DTSDKName, UIDeviceFamily, DTPlatformVersion, UILaunchScreen, CFBundleShortVersionString, and DTPlatformName. The "File Analysis Report" section at the bottom provides details about the file: File: ControlFlowFlattening.ipa, Size: 18 KB, Type: Single Architecture Mach-O, Mach-O Analysis, Language: Swift.

```
CFBundleDevelopmentRegion": "en",
"MinimumOSVersion": "16.4",
"CFBundleIdentifier": "com.testing.ControlFlowFlattening",
"DTXcodeBuild": "14E300c",
"CFBundleExecutable": "ControlFlowFlattening",
"LSRequiresiPhoneOS": true,
"BuildMachineOSBuild": "22D68",
"UIApplicationSupportsIndirectInputEvents": true,
"CFBundlePackageType": "APPL",
"UISupportedInterfaceOrientations~iphone": [
    "UIInterfaceOrientationPortrait",
    "UIInterfaceOrientationLandscapeLeft",
    "UIInterfaceOrientationLandscapeRight"
],
"DTCompiler": "com.apple.compilers.llvm.clang.1_0",
"CFBundleName": "ControlFlowFlattening",
"CFBundleSupportedPlatforms": [
    "iPhoneSimulator"
],
"UISupportedInterfaceOrientations~ipad": [
    "UIInterfaceOrientationPortrait",
    "UIInterfaceOrientationPortraitUpsideDown",
    "UIInterfaceOrientationLandscapeLeft",
    "UIInterfaceOrientationLandscapeRight"
],
"DTPlatformBuild": "20E238",
"UIApplicationSceneManifest": {
    "UISceneConfigurations": {},
    "UIApplicationSupportsMultipleScenes": true
},
"CFBundleInfoDictionaryVersion": "6.0",
"DTSDKBuild": "20E238",
"DTXcode": "1431",
"CFBundleVersion": "1",
"DTSDKName": "iphonesimulator16.4",
"UIDeviceFamily": [
    1,
    2
],
"DTPlatformVersion": "16.4",
"UILaunchScreen": {
    "UILaunchScreen": {}
},
"CFBundleShortVersionString": "1.0",
"DTPlatformName": "iphonesimulator"
```

# FEATURES

- Multi-Platform
    - Mac, Windows, Linux
  - Direct support for IPA files
  - Auto decodes iOS resources
  - Avoids lib code decompilation
  - Reconstructs Swift classes
  - Built-in LLM method translation

The image shows a Mac OS X desktop environment. In the foreground, there is a large window for 'Malimite - Analysis' showing Swift code. The code is annotated with various colors (yellow, blue, green) and numbers (local\_28, local\_30, bVar1, bVar2, puVar1, puVar3, pcVar4) to highlight specific parts of the program flow. A modal dialog box is open over the main window, titled 'Variable References', which lists the references for the variable 'PTR\_\_sSSN\_100008170'. The modal includes tabs for 'Type', 'Variable', 'Function', and 'Line', with the 'Line' tab selected. The background of the desktop shows the Mac OS X Dock at the bottom, featuring icons for Finder, Safari, Mail, and other applications.

# DEMO:

## Analyzing iOS Apps with Malimite



Now go decompile some iOS Applications!

You should be able to solve this.

# THANK YOU!



<https://github.com/LaurieWired/Malimite>