

Swift Reversing in 2024

It’s not so bad :)

Christopher Lopez - @L0psec

% id -un

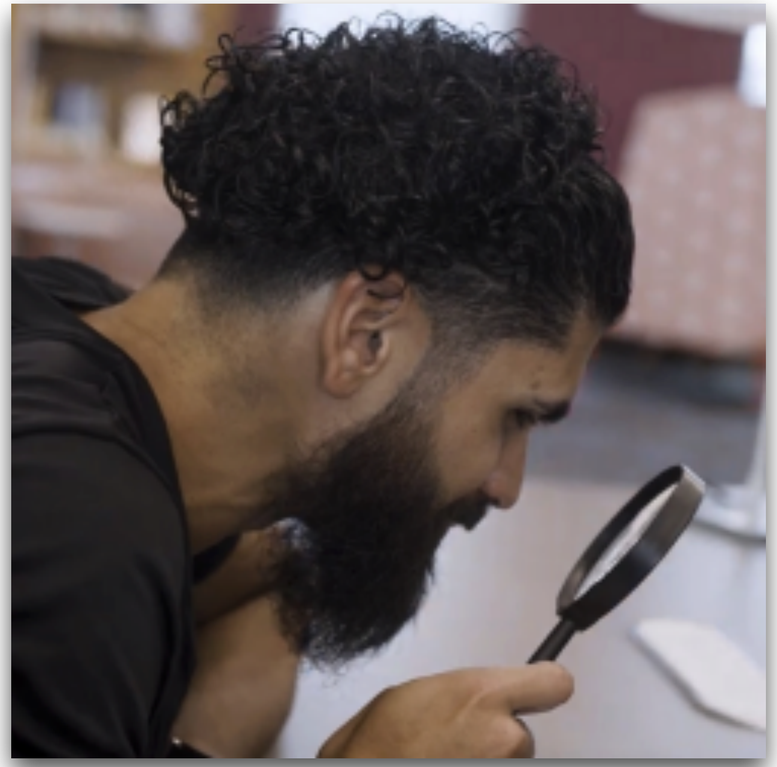


Christopher Lopez (@L0psec)

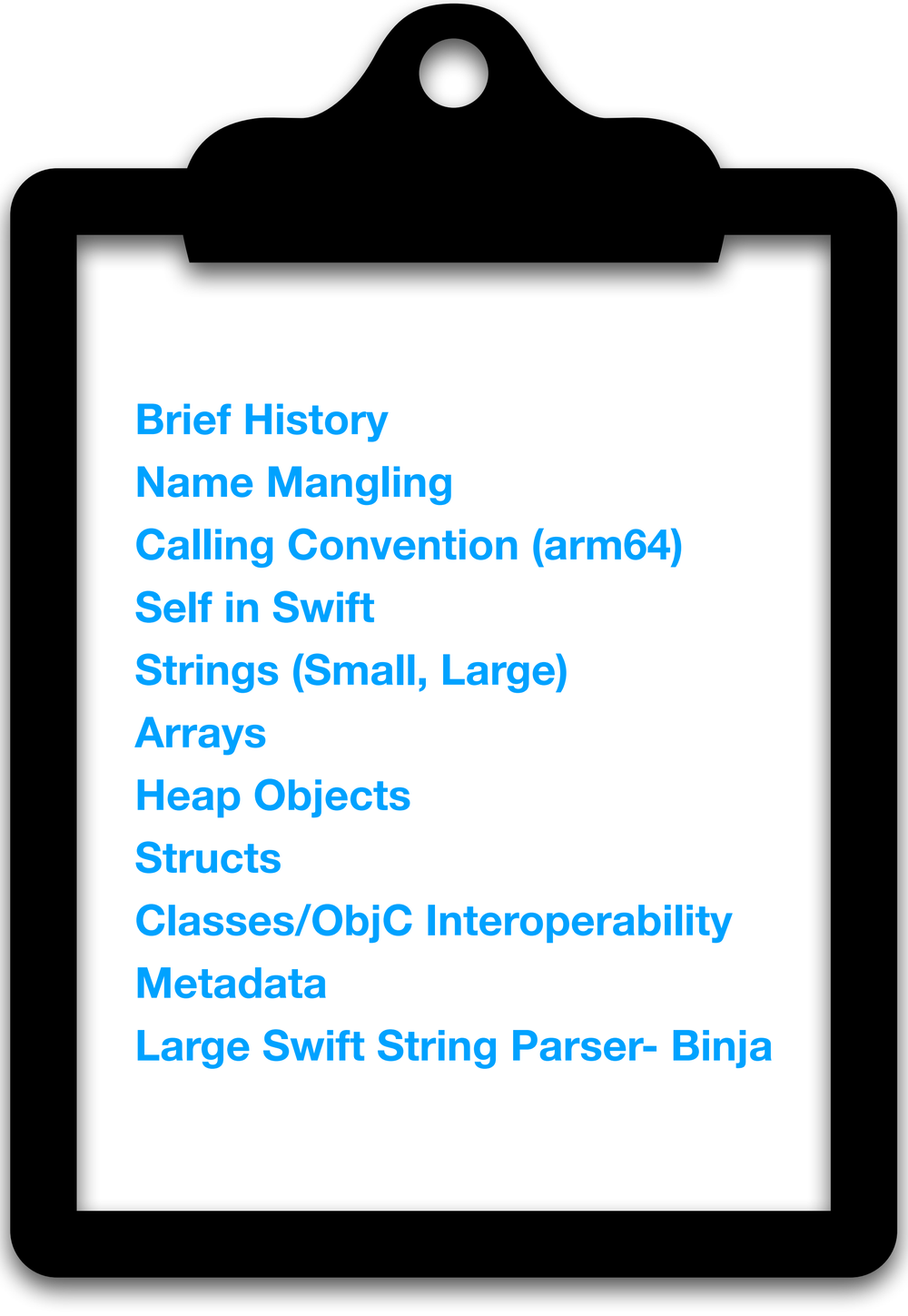
Security Researcher @ Kandji

Reversing, BJJ, Running, Video Games

YT = Reversing Videos
https://www.youtube.com/@L0psec



Agenda


Brief History
Name Mangling
Calling Convention (arm64)
Self in Swift
Strings (Small, Large)
Arrays
Heap Objects
Structs
Classes/ObjC Interoperability
Metadata
Large Swift String Parser- Binja

Why Should We Care??

Blog > Threat Intelligence > TodoSwift ...

TodoSwift Disguises Malware Download Behind Bitcoin PDF

Blog > Threat Intelligence > InfoSteale...

InfoStealer Uses SwiftUI, OpenDirectory API to Capture Passwords

Blog > Threat Intelligence > Another PD...

Another PDF Viewer - Is It Malicious?

Increase in Malware Written in Swift

elastic security labs

Stage 2

The Stage 2 binary (**.pd**) is compiled in Swift and operates based on command-line arguments. The binary expects a C2 URL to be provided as the first parameter when executed. Upon execution, it invokes the **downAndExec** function, which is responsible for preparing a POST HTTP request. To initiate this request, the binary sets the User-Agent string as **mozilla%2F4.0 (compatible; msie 8.0; windows nt 5.1; trident%2F4.0)** and includes the string **pw** in the body of the HTTP request.

First research on JOKERSPY

An overview of JOKERSPY, which was discovered in June 2023 and used custom and open source macOS tools to exploit a cryptocurrency exchange in Japan.

A Little History & Details



Introduced in 2014

Swift 5 - Stable ABI

Currently on Swift 6 - Sept 2024

Designed to be *safe

Open Source (mostly)

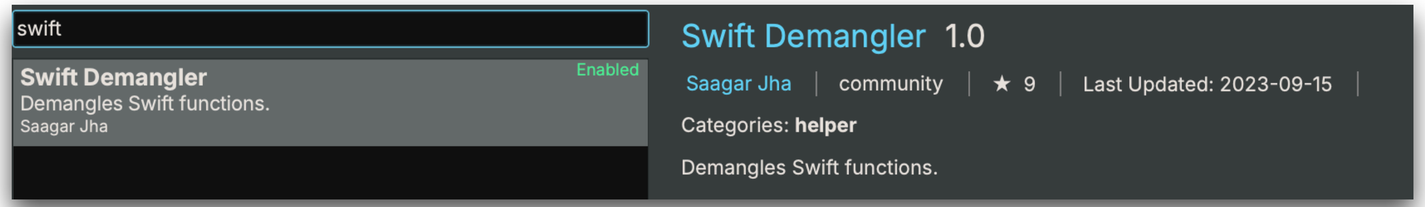
STDLIB in Swift, Runtime in C++

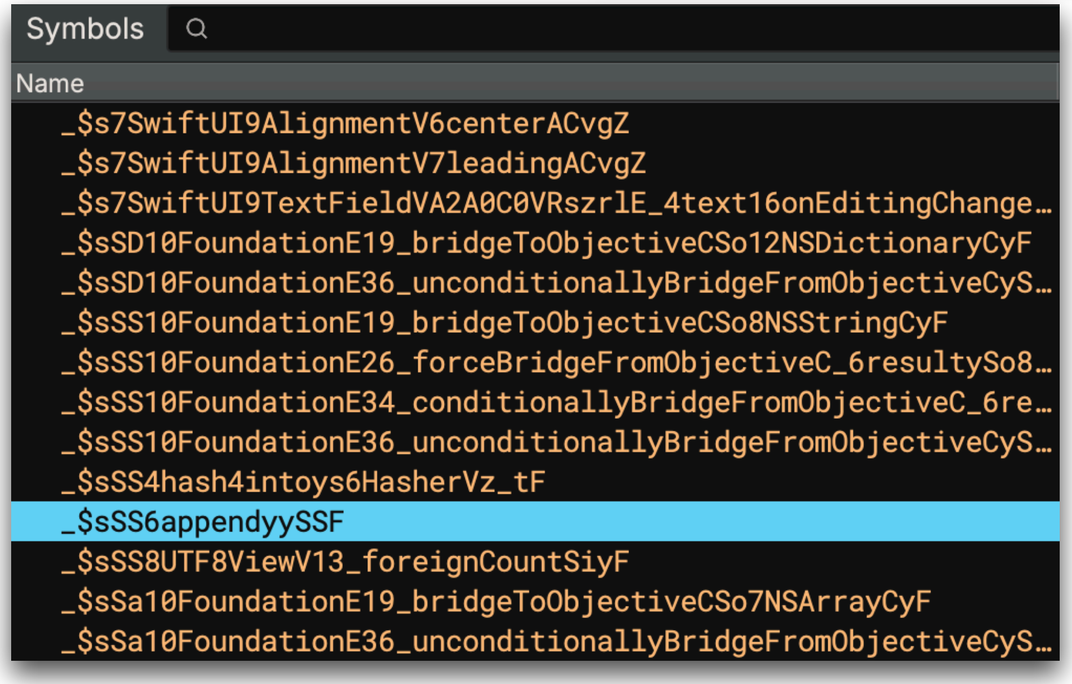
Swift Mangling

/Library/Developer/CommandLineTools/usr/bin/swift-demangle

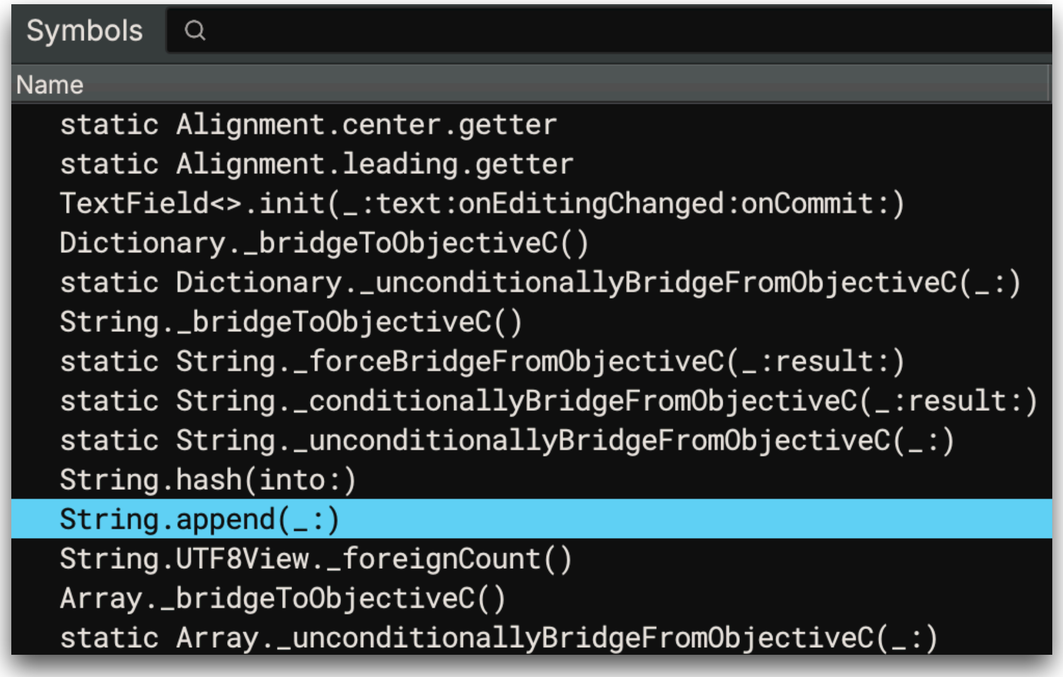
First thing a lot of reversers see

“The Swift compiler emits mangled names into binary images to encode references to types for runtime instantiation and reflection. In a binary, these mangled names may embed pointers to runtime data structures in order to more efficiently represent locally-defined types.”









https://github.com/swiftlang/swift/blob/main/docs/ABI/Mangling.rst

https://github.com/LaurieWired/iOS_Reverse_Engineering/blob/main/SwiftNameDemangler.py

Swift Calling Convention arm64

Register	Special	Purpose	C++	ObjC	Swift
`x0`		Integer argument 1 (1st return value)	`this`	`self`	
`x1`		Integer argument 2 (2nd return value)		`_cmd`	
`x2` - `x7`		Integer arguments 3-8 (3rd-8th return values)			
`x8`		Indirect result location register			
`x16`, `x17`	`ip0`, `ip1`	Scratch registers (used by dyld, can be used freely otherwise)			
`x18`		RESERVED **DO NOT USE**			
`x19`		Callee-saved register			
`x20`		Callee-saved register			`self`
`x21`		Callee-saved register			Error return
`x22`		Callee-saved register			Async context
`x23` - `x28`		Callee-saved registers			
`x29`	`fp`	Frame pointer			
`x30`	`lr`	Link register			
`sp`		Stack pointer			
`v0` - `v7`		Floating point/SIMD arguments 1-8 (also for return)			
`v8` - `v15`		Callee-saved registers (**lower 64-bits only**)			



https://github.com/swiftlang/swift/blob/f8e0980efecc14a712b16640b7072025b9ee7186/docs/ABI/CallConvSummary.rst

TodoSwift:
f1b3ce96462027644f9caa314d3da745dab139ee1cb14fe508234e76bd686f93

Swift Self in X20

```
100005650   f70301aa    mov     x23, x1
100005654   f80300aa    mov     x24, x0
```

Move X0 and X1
to registers X24 and X23

X0 + X1 = Swift String:
/tmp/GoogleMsgStatus.pdf

“file://“ initialized, saved
on stack and moved to X20

```
1000056dc   a8273ba9    stp     x8, x9, [x29, #-0x50] {var_58} {file://}  {'file://'}  {0xe700000000000000}
1000056e0   b44301d1    sub     x20, x29, #0x50 {file://}
1000056e4   e00318aa    mov     x0, x24
1000056e8   e10317aa    mov     x1, x23
1000056ec   e5400094    bl      String.append(_:)
```

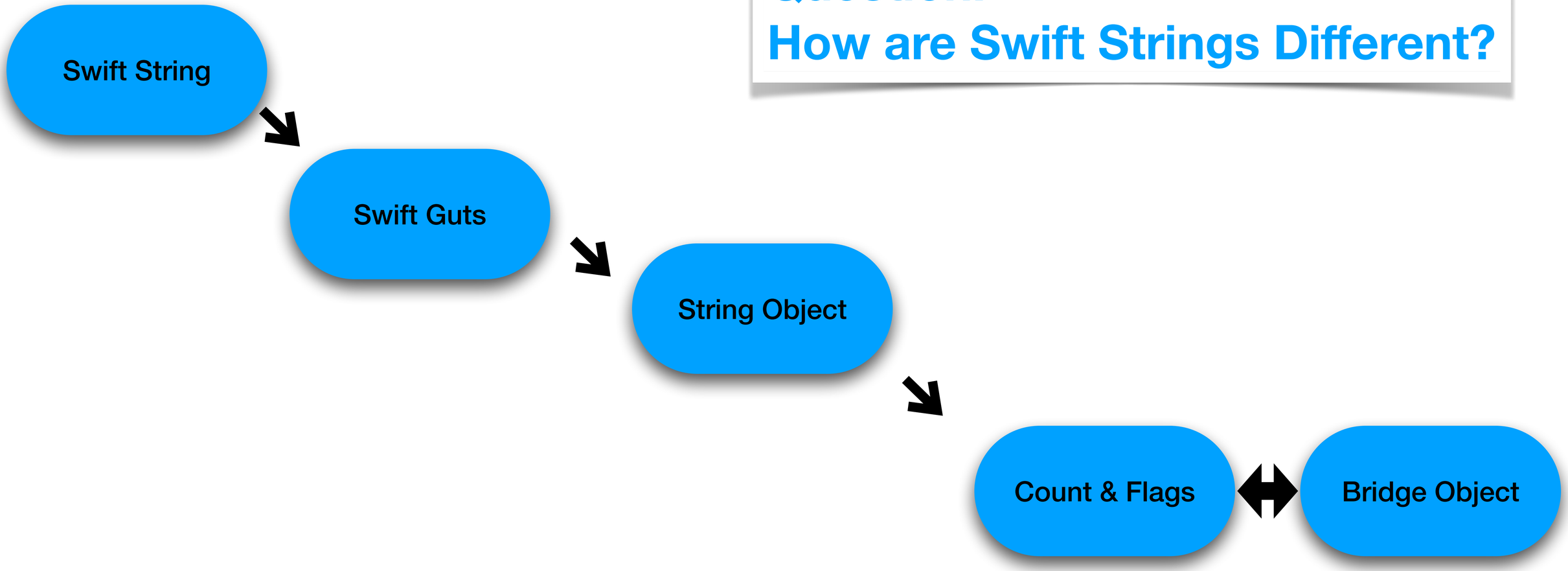
Question:
String.append() to WHAT????

X20 = String(“file://”)
X0+X1 = String(/tmp/GoogleMsgStatus.pdf”)

X20.append(X0, X1)
Result = “file:///tmp/GoogleMsgStatus.pdf”

Swift Strings

Not like C Strings :(

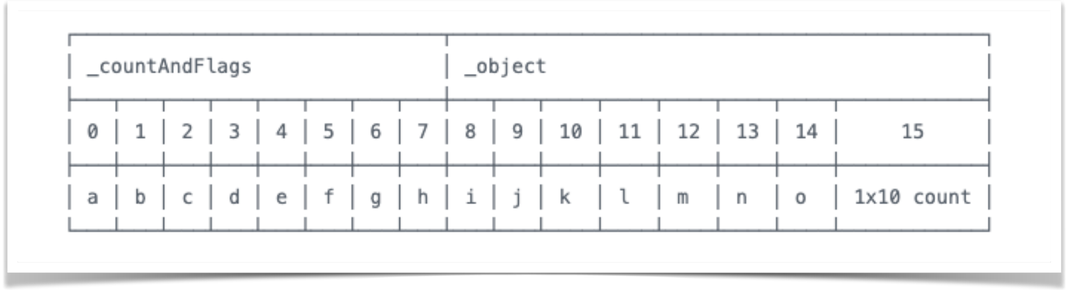


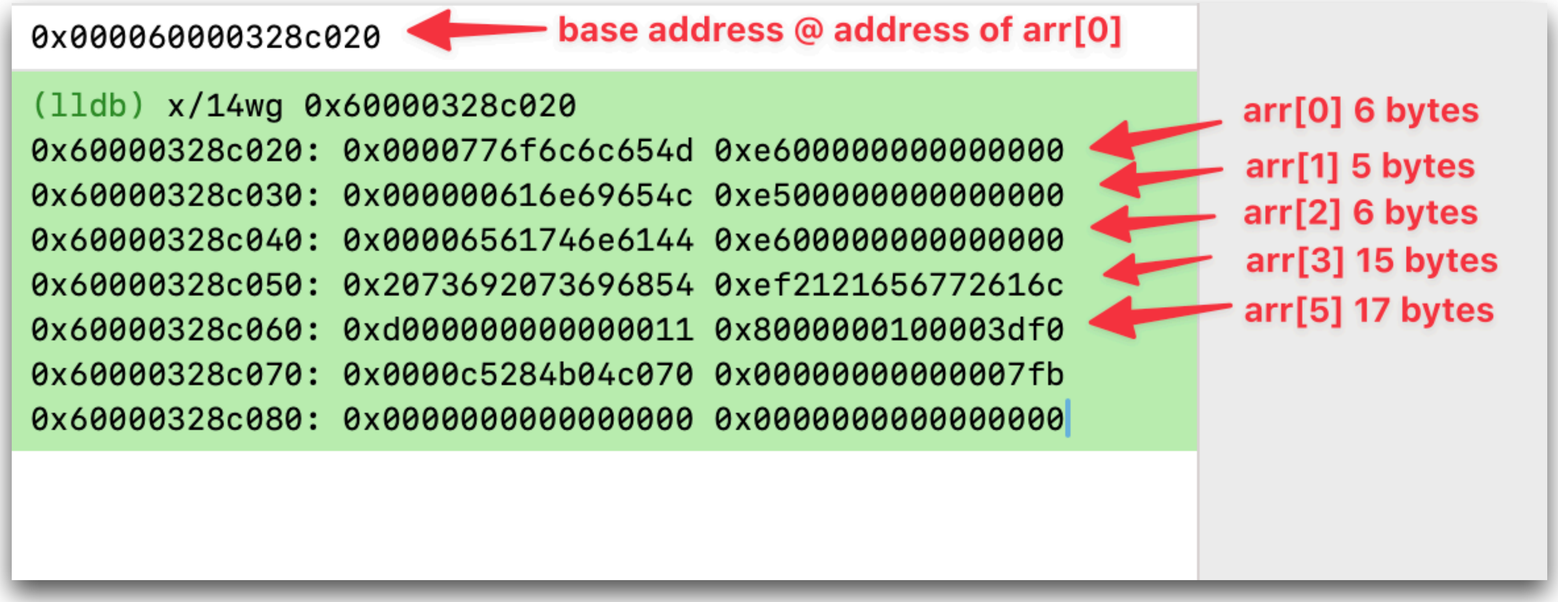
Question:
How are Swift Strings Different?

https://github.com/swiftlang/swift/blob/main/stdlib/public/core/String.swift

String Objects Encoding - Small

0xe(size) <=15 bytes long





Small Swift strings contain the size in the countAndFlags after the 0xe discriminator

Swiftbucket:
3b6f30369a4ee8bf9409d141b6d1b3fb4286c34984b5de005ed7431df549b17e

String Objects Encoding - Large

Structs (_countAndFlagsBits , _object)

```
URL.init(fileURLWithPath:)(0xd000000000000013, 0x8000000100003cc0)
Data.write(to:options:)(x22, 0, arg1, arg2)
(*(x26 + 8))(x22, x0)
specialized static Main.run_shell(_:)(0xd00000000000002f, 0x8000000100003ce0)
specialized static Main.run_shell(_:)(0xd000000000000053, 0x8000000100003d10)
specialized static Main.run_shell(_:)(0xd00000000000002c, 0x8000000100003d70)
```



```
// /Users/Shared/1.zip
URL.init(fileURLWithPath:)(0xd000000000000013, 0x8000000100003cc0)
Data.write(to:options:)(x22, 0, arg1, arg2)
(*(x26 + 8))(x22, x0)
// unzip -o -d /Users/Shared "/Users/Shared/1.zip"
specialized static Main.run_shell(_:)(0xd00000000000002f, 0x8000000100003ce0)
// chmod +x "/Users/Shared/Internal PDF
// Viewer.app/Contents/MacOS/Internal PDF Viewer"
specialized static Main.run_shell(_:)(0xd000000000000053, 0x8000000100003d10)
// open "/Users/Shared/Internal PDF Viewer.app"
specialized static Main.run_shell(_:)(0xd00000000000002c, 0x8000000100003d70)
```

b63	b62	b61	b60	b59	b58:48	b47:0
ASCII	NFC	native	tail	UTF8	reserved	count

Question:
How do we calculate the correct address of the string??

```
@inlinable @inline(__always)
  internal static var isASCIIMask: UInt64 { return 0x8000_0000_0000_0000 }
```

```
@usableFromInline
internal var _countAndFlagsBits: UInt64

@usableFromInline
internal var _object: Builtin.BridgeObject
```

Calculating String Objects - Large

Add NativeBias (+ 0x20)

```
extension _StringObject {
  @inlinable @inline(__always)
  internal static var nativeBias: UInt {
#if _pointerBitWidth(_64)
    return 32
#elseif _pointerBitWidth(_32)
    return 20
```

```
                                    _countAndFlagsBits              _object
URL.init(fileURLWithPath:)(0xd000000000000013, 0x8000000100003cc0)
```

```
>>> nativeBias = 0x20
>>> bv.read(0x100003cc0 + nativeBias, 0x13)
b'/Users/Shared/1.zip'
```

```
                                    _countAndFlagsBits              _object
specialized static Main.run_shell(_:)(0xd000000000000002f, 0x8000000100003ce0)
```

```
>>> bv.read(0x100003ce0 + nativeBias, 0x2f)
b'unzip -o -d /Users/Shared "/Users/Shared/1.zip"'
```

```
                                    _countAndFlagsBits              _object
specialized static Main.run_shell(_:)(0xd000000000000053, 0x8000000100003d10)
```

```
>>> bv.read(0x100003d10 + nativeBias, 0x53)
b'chmod +x
"/Users/Shared/Internal PDF Viewer.app/Contents/
MacOS/Internal PDF Viewer"'
```

```
                                    _countAndFlagsBits              _object
specialized static Main.run_shell(_:)(0xd000000000000002c, 0x8000000100003d70)
```

```
>>> bv.read(0x100003d70 + nativeBias, 0x2c)
b'open "/Users/Shared/Internal PDF Viewer.app"'
```

https://github.com/swiftlang/swift/blob/4df55a12cdb6905c33af655474d0f5e7b73f3d73/stdlib/public/core/StringObject.swift#L449

SwiftBucket_stage2:
c7f4aa77be7f7afe9d0665d3e705dbf7794bc479bb9c44488c7bf4169f8d14fe

Arrays

Contiguous Arrays

```
public static var arguments: [String] {
  get {
    _arguments
  }
```

```
100002de4  72110094   bl      static CommandLine.arguments.getter
100002de8  080840f9   ldr     x8, [x0, #0x10]
100002dec  1f0900f1   cmp     x8, #0x2
100002df0  e32c0054   b.lo    0x10000338c

100002df4  f40300aa   mov     x20, x0
100002df8  080043a9   ldp     x8, x0, [x0, #0x30]
100002dfc  f5b40410   adr     x21, url
100002e00  1f2003d5   nop
100002e04  a80200a9   stp     x8, x0, [x21]  {url}  {data_10000c4a0}
```

Args Return = X0
X8 = X0 + 0x10

X8 = X0 + 0x30
X0 = X0 + 0x38

X8 = url

Questions:

What is at X0 + 0x10?

What is at X0 + 0x30?

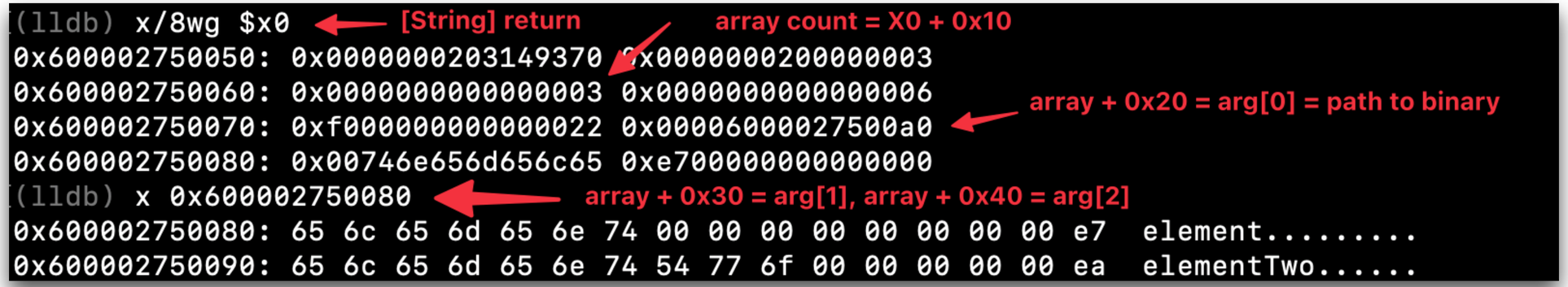
What is at X0 + 0x38?

https://github.com/swiftlang/swift/blob/main/stdlib/public/core/CommandLine.swift#L109

https://github.com/swiftlang/swift/blob/main/stdlib/public/core/ContiguousArray.swift

Arrays Cont.

Metadata, Count, Elements

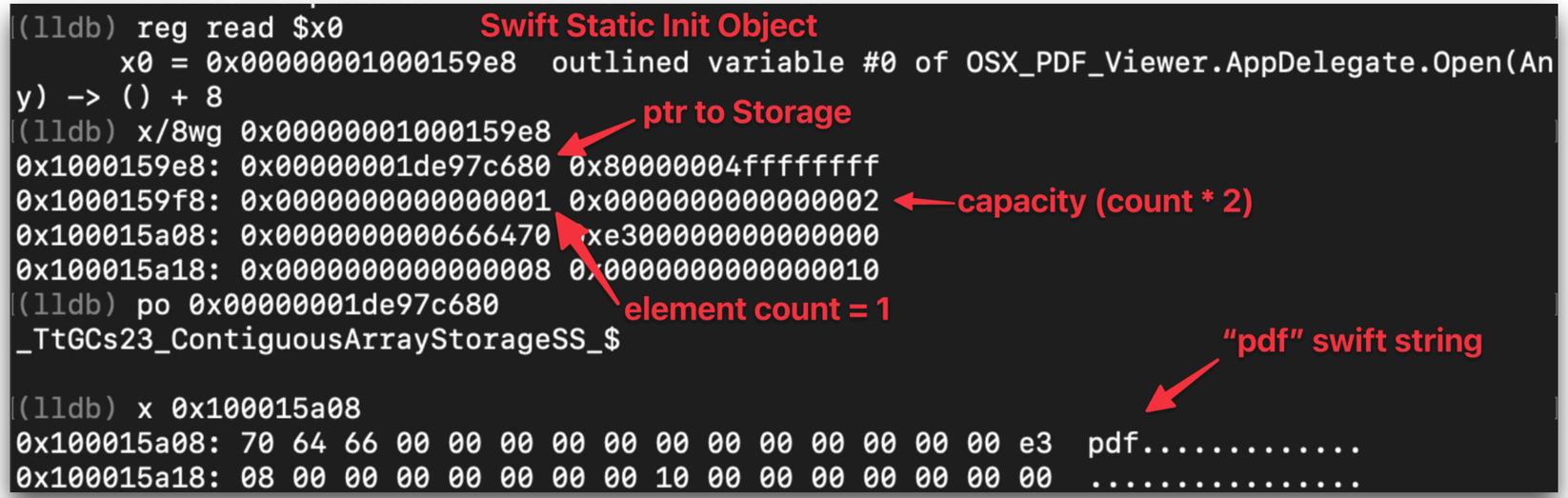


X0 + 0x0 = metadata
X0 + 0x10 = Element Count
X0 + 0x20 = array[0] - First Element
X0 + 0x30 = array[1] - Second Element
X0 + 0x40 = array[2] - Third Element

What about X0 + 0x38?
X0 + 0x30 = Swift String
Swift Strings are structs
X0 + 38 is part of the String

Arrays Cont.

Additional Example



X0 + 0x10 = Element Count
X0 + 0x20 = array[0] First Element
X0 + 0x30 = array[1] Second Element

PDFViewer
6c925e2a39e8312c704575af4ad7fe75f161e73f92ffda6b9abd3663b6c789d4

Runtime Heap Objects

```
HeapObject *

swift::swift_initStaticObject(HeapMetadata const *metadata,

                              HeapObject *object)
```

```
10000c5d0  c0a70410   adr     x0, demangling cache variabl...ta for _ContiguousArrayStorage<UInt8>
10000c5d4  1f2003d5   nop
10000c5d8  44fdff97   bl      ___swift_instantiateConcreteTypeFromMangledName
10000c5dc  e19b0410   adr     x1, data_100015958
10000c5e0  1f2003d5   nop
10000c5e4  dd030094   bl      _swift_initStaticObject
```

```
(lldb) reg read $x0          Key Swift Init Object
      x0 = 0x0000000100015958  outlined variable #0 of OSX_PDF_Viewer.decryptData(encrypt
edData: Swift.Optional<Foundation.Data>) -> Foundation.Data + 8
(lldb) x/8wg 0x0000000100015958
0x100015958: 0x00000001de97cc40 0x80000004ffffffff
0x100015968: 0x0000000000000064 0x00000000000000c8   <- capacity (count * 2)
0x100015978: 0xc01339d69de16c29 0x990c1044d15e94b5
0x100015988: 0xd6d0a00b47b44c68 0x8aa65cb6dcf3d875
(lldb) po 0x00000001de97cc40          <- count = 0x64 (100 bytes)
_TtGCs23_ContiguousArrayStorageVs5UInt8_$

(lldb) x 0x100015978          <- Key
0x100015978: 29 6c e1 9d d6 39 13 c0 b5 94 5e d1 44 10 0c 99  )l...9....^.D...
0x100015988: 68 4c b4 47 0b a0 d0 d6 75 d8 f3 dc b6 5c a6 8a  hL.G....u....\..
```

Question:
What is the result of _swift_initStaticObject?

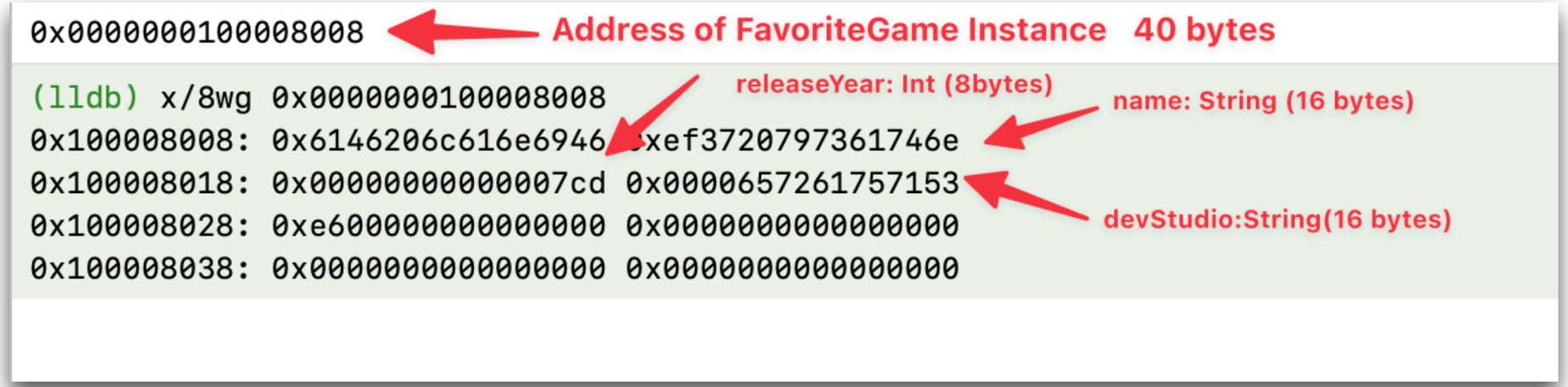
https://belkadan.com/blog/2020/08/Swift-Runtime-Heap-Objects/

Structs in Memory

```swift
struct Game {
    var name: String
    var releaseYear: Int
    var devStudio: String
    func printDesc() {
        print("\(name) was released on \(releaseYear) and developed by \(devStudio).")
    }
}
var favoriteGame = Game(name: "Final Fantasy 7", releaseYear: 1997, devStudio: "Square")
withUnsafePointer(to: &favoriteGame) { print("\($0)") } //prints address of instance

print(MemoryLayout.size(ofValue: favoriteGame)) //prints 40
```

Like C in memory Usage



Classes - interoperability with ObjC

Every Swift Class = Objective-C Class

```swift
import Foundation

class Car {
    var make: String
    var model: String
    var year: Int
    func printDescription() {
        print("My favorite car is a \(year) \(make) \(model).")
    }
    init(make: String, model: String, year: Int) {
        self.make = make
        self.model = model
        self.year = year
    }
}

let favoriteCar = Car(make: "BMW", model: "M5", year: 1988)
favoriteCar.printDescription()
```

Questions:

What is the type metadata accessor?

What is at the offset of favoriteCar + 176

```
1000033a8    int64_t _main()

1000033bc        type metadata accessor for Car()      <- Metadata, We will cover soon :)
1000033d8        int32_t var_34 = 1
1000033e0        int64_t BMW
1000033e0        int64_t x1                            <- Init a small Swift String = "BMW"
1000033e0        BMW, x1 = String.init(_builtinStringLiteral:utf8CodeUnitCount:isASCII:)("BMW", 3, 1)
100003404        int64_t M5
100003404        int64_t x1_1                          <- Init another small Swift String = "M5"
100003404        M5, x1_1 = String.init(_builtinStringLiteral:utf8CodeUnitCount:isASCII:)("M5", 2, 1)
100003428        favoriteCar = Car.__allocating_init(make:model:year:)(BMW, x1, M5, x1_1, 1988)
100003438        (*(*favoriteCar + 176))()             <- favoriteCar = Car class
10000344c        return 0                              <- What is at this Offset???
```

Class Metadata Accessor?

```
100003450  // _$s17swiftClassExample3CarCMa
100003450    id type metadata accessor for Car()

100003470      return _objc_opt_self(obj: &cls__TtC17swiftClassExample3Car)
```

Returns pointer to Class Data Backed by Objective-C ???

```
100008120  full type metadata for Car:
100008120  00 00 00 00 00 00 00 00                      ........
                                                        Pointer to de initalize function
100008128  void* data_100008128 = Car.__deallocating_deinit
                                                        Value Witness Table for Type
100008130  int64_t (* const value witness table for Builtin.NativeObject)() = value witness table for Builtin.NativeObject
100008138  struct objc_class_t cls__TtC17swiftClassExample3Car =
100008138  {
100008138      struct objc_class_t* isa = metaclass for Car
100008140      void* super = _OBJC_CLASS_$__TtCs12_SwiftObject   Swift Object (hidden "Root" object like NSObject)
100008148      void* cache = __objc_empty_cache
100008150      void* vtable = 0x0
100008158      struct objc_class_ro_t* data = &data_1000080b0->flags+2.72 {1000080b2}
                                                        Pointer to class instance variables, etc.
100008160  }
```

Class Metadata - __objc_const

```
1000080b0  struct objc_class_ro_t cls_ro__TtC17swiftClassExample3Car =
1000080b0  {
1000080b0      uint32_t flags = 0x80
1000080b4      uint32_t start = 0x10
1000080b8      uint32_t size = 0x38
1000080bc      uint32_t reserved = 0x0
1000080c0      void* ivar_layout = 0x0
1000080c8      char* name = clsName__TtC17swiftClassExample3Car {"_TtC17swiftClassExample3Car"}
1000080d0      struct objc_method_list_t* methods = 0x0
1000080d8      struct objc_protocol_list_t* protocols = 0x0
1000080e0      struct objc_ivar_list_t* ivars = ivar_list__TtC17swiftClassExample3Car
1000080e8      void* weak_ivar_layout = 0x0
1000080f0      void* properties = 0x0
1000080f8  }
```



```
100008048  struct objc_ivar_list_t ivar_list__TtC17swiftClassExample3Car =
100008048  {
100008048      uint32_t entsize = 0x20
1000804c       uint32_t count = 0x3
100008050  }
100008050  struct objc_ivar_t ivar_make =
100008050  {
100008050      uint32_t* offset = direct field offset for Car.make
100008058      char* name = ivarName_make {"make"}
100008060      char* type = ivarType_year
100008068      uint32_t alignment = 0x3
1000806c       uint32_t size = 0x10
100008070  }
100008070  struct objc_ivar_t ivar_model =
100008070  {
100008070      uint32_t* offset = direct field offset for Car.model
100008078      char* name = ivarName_model {"model"}
100008080      char* type = ivarType_year
100008088      uint32_t alignment = 0x3
1000808c       uint32_t size = 0x10
100008090  }
100008090  struct objc_ivar_t ivar_year =
100008090  {
100008090      uint32_t* offset = direct field offset for Car.year
100008098      char* name = ivarName_year {"year"}
1000080a0      char* type = ivarType_year
1000080a8      uint32_t alignment = 0x3
1000080ac      uint32_t size = 0x8
1000080b0  }
```

Class metadata points to Objective-C class information including methods, ivars, etc.

Class - What is at (favoriteCar + 176)?

```
swiftClassExample`main:
->  0x100003424 <+124>: adrp    x8, 5
    0x100003428 <+128>: str     x0, [x8, #0x1f8]
    0x10000342c <+132>: ldr     x20, [x8, #0x1f8]
    0x100003430 <+136>: ldr     x8, [x20]
(lldb) reg read $x0
      x0 = 0x0000600000eec000
(lldb) x/8wg 0x0000600000eec000
0x600000eec000: 0x0000000100008138 0x0000000000000003
0x600000eec010: 0x0000000000574d42 0xe300000000000000
0x600000eec020: 0x000000000000354d 0xe200000000000000
0x600000eec030: 0x00000000000007c4 0x0000000000000000
(lldb) po 0x0000000100008138
swiftClassExample.Car
```

init return

Car class Metadata

make, model, year

```
100003424  280000b0   adrp    x8, metacls_ro__TtC17swiftClassExample3Car
100003428  00fd00f9   str     x0, [x8, #0x1f8]  {favoriteCar}
10000342c  14fd40f9   ldr     x20, [x8, #0x1f8]  {favoriteCar}
100003430  880240f9   ldr     x8, [x20]
100003434  085940f9   ldr     x8, [x8, #176]
100003438  00013fd6   blr     x8
```

Variable

Load 64 bits from the Class

printDescription() method

Branch to Register (address in X8)

```
100008138  struct objc_class_t cls__TtC17swiftClassExample3Car =
100008138  {
100008138      struct objc_class_t* isa = metaclass for Car
100008140      void* super = _OBJC_CLASS_$__TtCs12_SwiftObject
100008148      void* cache = __objc_empty_cache
100008150      void* vtable = 0x0
100008158      struct objc_class_ro_t* data = &data_1000080b0->flags+2.72 {1000080b2}
100008160  }

100008160  02 00 00 00 00 00 00 00-38 00 00 00 07 00 00 00  ........8.......
100008170  d8 00 00 00 18 00 00 00                          ........

100008178  void* data_100008178 = nominal type descriptor for Car

100008180  00 00 00 00 00 00 00 00-10 00 00 00 00 00 00 00  ................
100008190  20 00 00 00 00 00 00 00-30 00 00 00 00 00 00 00   .......0.......

1000081a0  void* data_1000081a0 = Car.make.getter
1000081a8  void* data_1000081a8 = Car.make.setter
1000081b0  void* data_1000081b0 = Car.make.modify
1000081b8  void* data_1000081b8 = Car.model.getter
1000081c0  void* data_1000081c0 = Car.model.setter
1000081c8  void* data_1000081c8 = Car.model.modify
1000081d0  void* data_1000081d0 = Car.year.getter
1000081d8  void* data_1000081d8 = Car.year.setter
1000081e0  void* data_1000081e0 = Car.year.modify
1000081e8  void* data_1000081e8 = Car.printDescription()
1000081f0  void* data_1000081f0 = Car.__allocating_init(make:model:year:)
```

Address Class metadata points to = 0x100008138

0x100008138 + 0xb0 = 0x1000081e8

8 bytes loaded from the instance of class to register X8. This is the metadata pointer which points to the Class information. This is then used to get the method for this class at the offset + 176 (or 0xb0) = Car.printDescription()

Metadata - Type Accessors

The Swift runtime keeps a metadata record for every type used in a program, including every instantiation of generic types.

Symbols — type metadata accessor

Name	Address
type metadata accessor for AppDelegate	0x100009fa4
type metadata accessor for URL?	0x10000a09c
type metadata accessor for NSPopUpButton	0x10000bcd8
type metadata accessor for NSError	0x10000c7b0
type metadata accessor for Notification	0x10000d0cc
type metadata accessor for __DataStorage	0x10000d0fc
type metadata accessor for URL	0x10000d174
type metadata accessor for Data.RangeReference	0x10000d180
type metadata accessor for Data.Iterator	0x10000d1b0
type metadata accessor for String.Encoding	0x10000d210
type metadata accessor for Optional	0x10000d2b8

```
100003984  fc250094   bl      type metadata accessor for URL
100003988  f60300aa   mov     x22, x0
10000398c  1c805ff8   ldur    x28, [x0, #-0x8]
100003990  882340f9   ldr     x8, [x28, #0x40]
```

Question:
What is at X0 - 0x8?

Metadata - Type Accessors Cont.

```
(lldb) reg read $x0
      x0 = 0x00000001df8ff4b0  Foundation`type metadata for Foundation.URL
(lldb) x/8wg  0x00000001df8ff4b0
0x1df8ff4b0: 0x0000000000000200 0x0000000189849340
0x1df8ff4c0: 0x0000000000000000 0x0000000188e1146c
0x1df8ff4d0: 0x0000000188e0c57c 0x0000000188e1146c
0x1df8ff4e0: 0x0000000188e1146c 0x0000000188e1146c
(lldb) x/8wg 0x1df8ff4b0-0x8
0x1df8ff4a8: 0x00000001dfdd67f8 0x0000000000000200
0x1df8ff4b8: 0x0000000189849340 0x0000000000000000
0x1df8ff4c8: 0x0000000188e1146c 0x0000000188e0c57c
0x1df8ff4d8: 0x0000000188e1146c 0x0000000188e1146c
(lldb) image lookup -a 0x00000001dfdd67f8
      Address: libswiftCore.dylib[0x00000001d86aa7f8] (libswiftCore.dylib.__AUTH_CONST._
const + 74800)
      Summary: libswiftCore.dylib`value witness table for Builtin.UnknownObject
```

Return of Metadata Call

0x200 = Struct type, URL = Struct

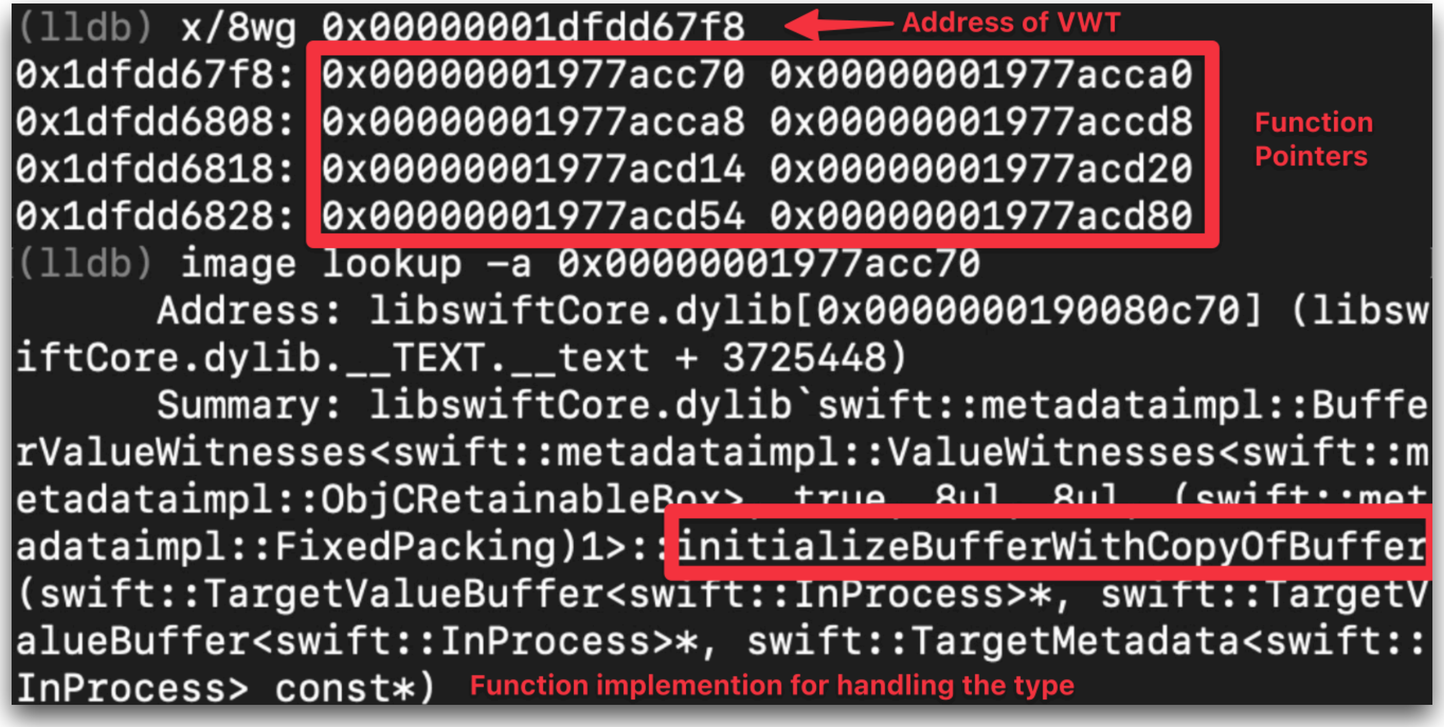
Subtract 0x8 (a pointer length)

Address of Value Witness Table

Question:
What is at X0 - 0x8?

X0 = metadata pointer for the URL type
(X0 - 0x8) = Value Witness Table (Function pointers for Type)

Metadata - Value Witness Tables



Value Witness tables contain function pointers relevant to the Type

Must Watch - Understanding Swift Performance: https://developer.apple.com/videos/play/wwdc2016/416/?time=1474

TodoSwift:
f1b3ce96462027644f9caa314d3da745dab139ee1cb14fe508234e76bd686f93

Parsing Large Swift Strings

With Binary Ninja

```
[ScriptingProvider] 0x100004084 b'isLogicalDeleted'
[ScriptingProvider] 0x100004274 b'logicalDeletedAt'
[ScriptingProvider] 0x1000041d4 b'logicalDeletedAt'
[ScriptingProvider] 0x100005000 b'/tmp/GoogleMsgStatus.pdf'
[ScriptingProvider] 0x100005010 b'/tmp/GoogleMsgStatus.pdf'
[ScriptingProvider] 0x10000503c b'/tmp/NetMsgStatus'
[ScriptingProvider] 0x100005050 b'/tmp/NetMsgStatus'
[ScriptingProvider] 0x10000d5ac b'Binding<PresentationMode>'
[ScriptingProvider] 0x10000def0 b'isLogicalDeleted == false'
```

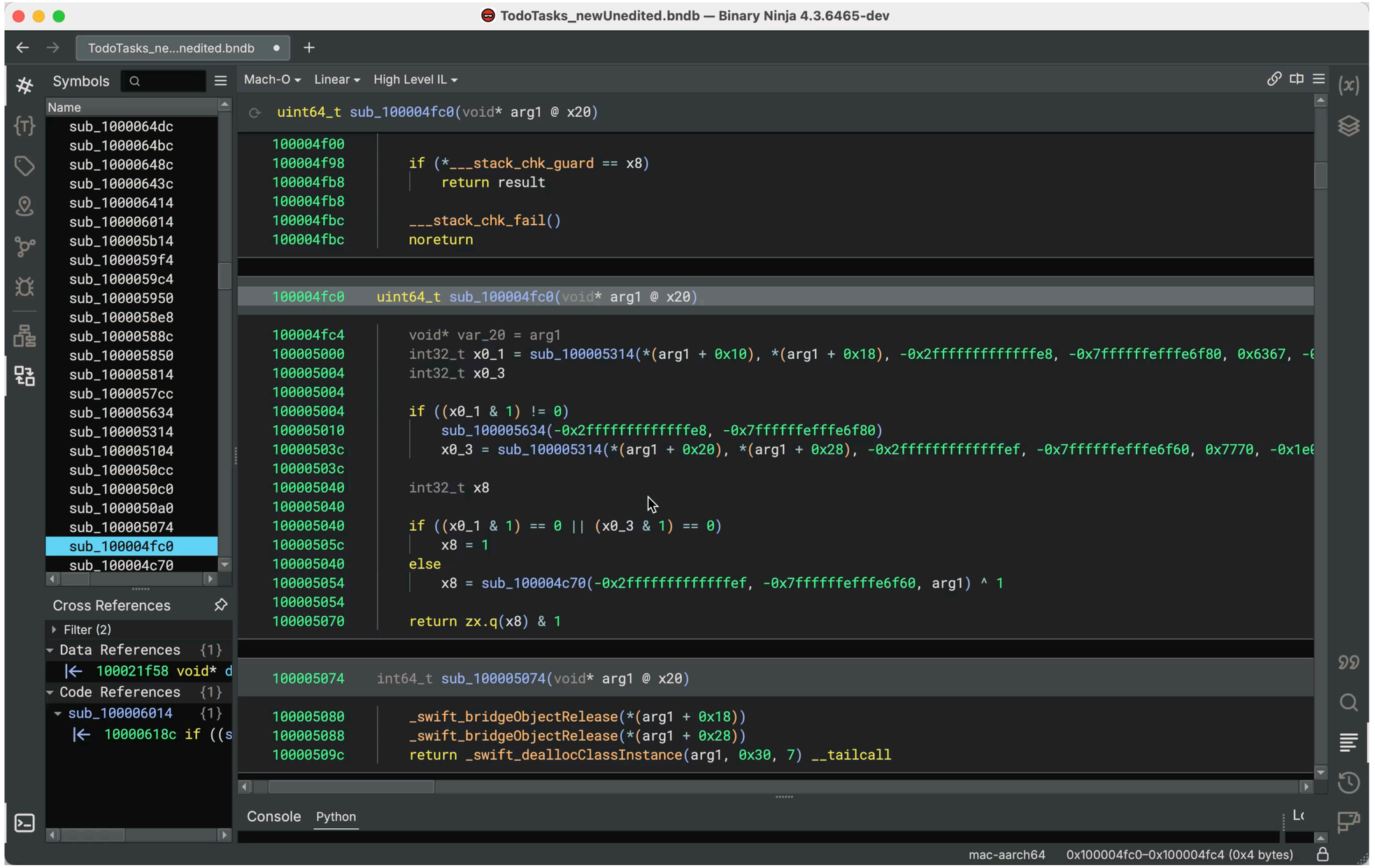
Addresses and Large Swift Strings

Tags and Comments added at the Call Site

```
! 100005010        // /tmp/GoogleMsgStatus.pdf
! 100005010        sub_100005634(-0x2fffffffffffffe8, -0x7ffffffefffe6f80)
! 10000503c        // /tmp/NetMsgStatus
! 10000503c        x0_3 = sub_100005314(*(arg1 + 0x20), *(arg1 + 0x28), -0x2fffffffffffffef,
```

https://github.com/l0psec/Binja_swift_large_string_parser/blob/main/swift_large_string_parser.py

Swift String Parsing Video



Resources

https://github.com/swiftlang/swift/blob/main/docs/Lexicon.md

Understanding Swift Performance: https://developer.apple.com/videos/play/wwdc2016/416/?time=1474

https://www.mikeash.com/pyblog/friday-qa-2014-07-18-exploring-swift-memory-layout.html

Mike Ash Swift Memory Layout Youtube: https://www.youtube.com/watch?v=ERYNyrfXjlg

https://knight.sc/reverse%20engineering/2019/07/17/swift-metadata.html

https://github.com/OpenSwiftUIProject/OpenSwiftUI

https://derekselander.github.io/dsdump/#swift

https://www.elastic.co/de/security-labs/inital-research-of-jokerspy

https://www.kandji.io/blog/another-pdf-viewer-is-it-malicious

https://www.kandji.io/blog/infostealer-swiftui-opendirectory-api-capture-verify-passwords

https://www.kandji.io/blog/todoswift-disguises-malware-download-behind-bitcoin-pdf

Special Thanks

Threat Intel Team @ Kandji :)

Noar :)

Alden

Open Source Swift Code

Mike Ash

Scott Knight

Derek Selander

