

# CNIT 128

# Hacking Mobile Devices



## 3. Attacking iOS Apps Part 2

# Topics: Part 1

- Introduction to Transport Security
- Identifying Insecure Storage
- Patching iOS Applications with Hopper

# Topics: Part 2

- Attacking the iOS Runtime
- Understanding Interprocess Communication
- Attacking Using Injection

# **Attacking the iOS Runtime**

# The Runtime

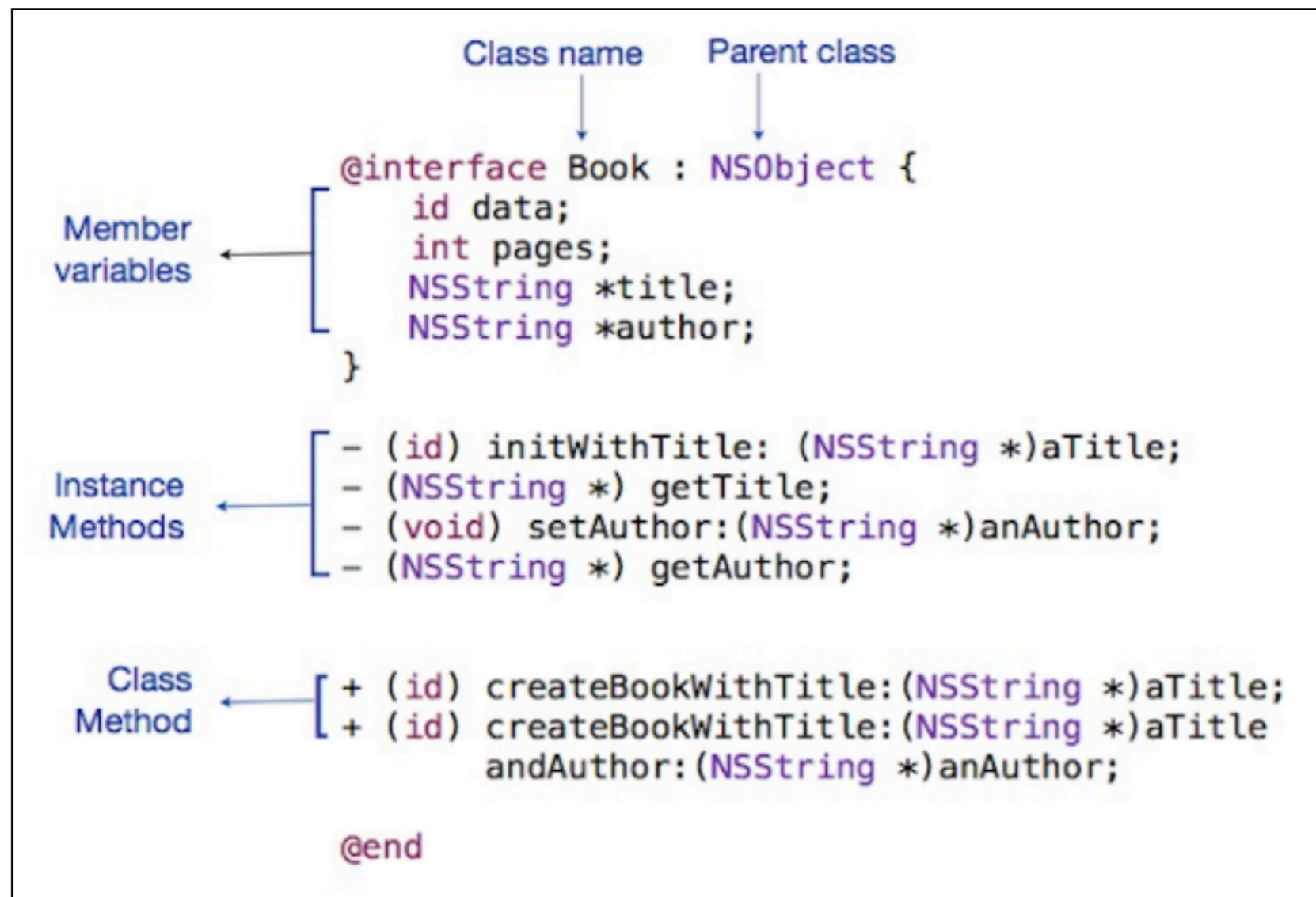
- Objective-C and Swift defer many decisions
  - From compile-and-link time
  - To runtime
- By using **reflection**
  - Apps modify their own behavior at runtime
  - Dynamically load new classes
  - Change method implementations

# Understanding Objective-C and Swift

- Object-oriented languages
- **Objects** encapsulate data in the form of **classes**
  - **Classes** contain
    - Instance variables
    - Methods
    - Properties

# Interface File

- Defines a class structure
  - Image from <https://blog.teamtreehouse.com/an-introduction-to-objective-c>

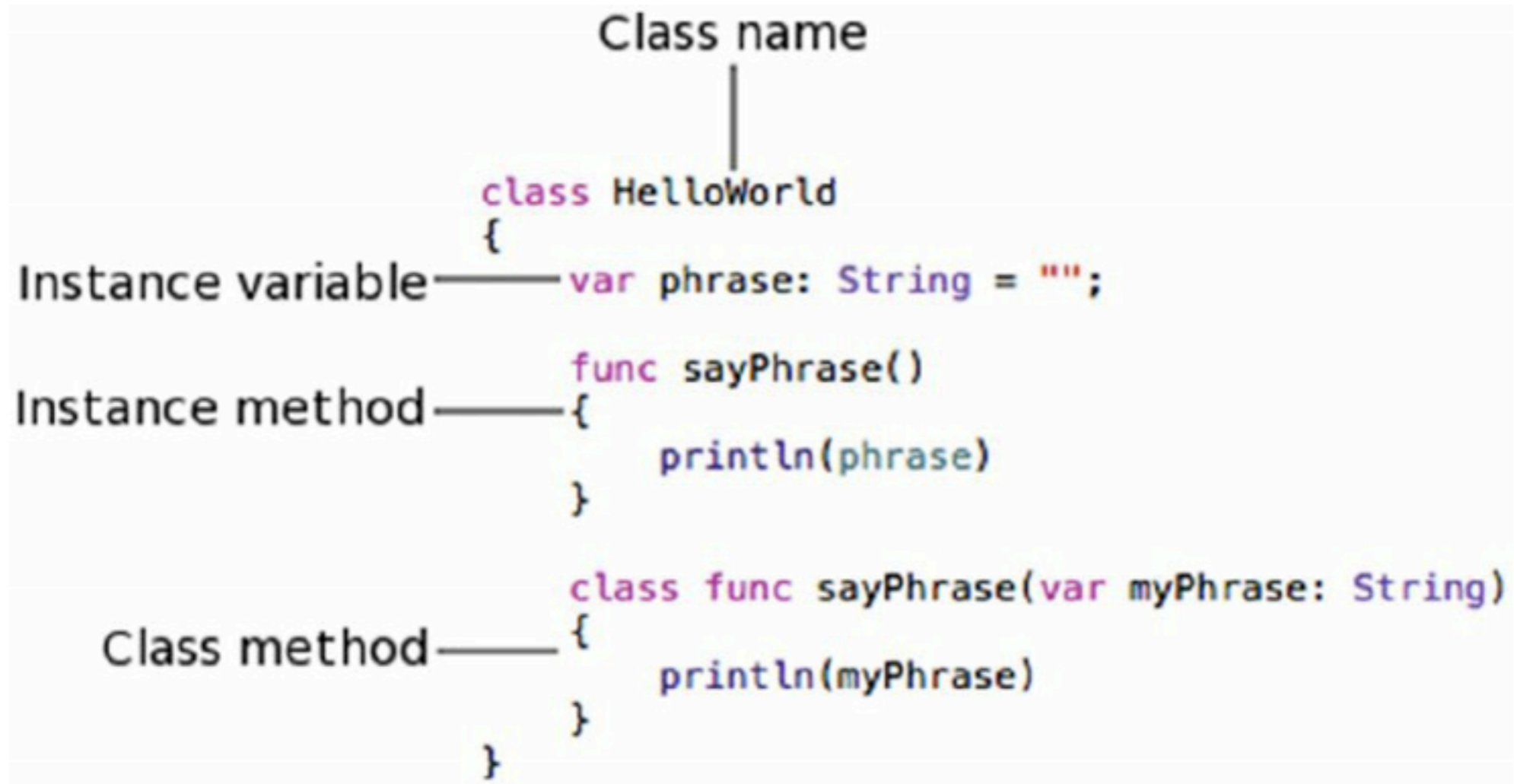


# Methods

- **Instance methods** can only be invoked
  - After creating an **instance** of the **class**
- **Class methods** can be invoked
  - Without actually creating an **instance** of the **class**



# Swift Class



**Figure 3.18** A breakdown of Swift class

# Instrumenting the iOS Runtime

- Tracing, debugging, or otherwise profiling the execution of an app at runtime
- Examples:
  - Bypassing jailbreak detection
  - Stealing encryption keys
  - Force-loading view controllers
  - Attacking local authentication
  - Pivoting to internal networks
  - Demonstrating the risks of malware
  - Inspecting a custom encryption protocol

# Instrumenting Objective-C

- Objective-C is by far easiest to instrument
- To invoke a function
  - Pass it a message
  - Through the runtime's **objc\_msgSend()** function
- To instrument it, simulate calls to **objc\_msgSend()**

# Method Swizzling

- Replace the implementation of a method at runtime
- A **class** maintains a **dispatch table**
  - With a map of **selectors** to **implementations**
  - **Selector**: name of **method**
  - **Implementation**: pointer to function
- Replacing pointers achieves swizzling

# Instrumenting Swift

- Swift uses direct function calls and vtable lookups
- Requires more effort to instrument

# Cydia Substrate

- Runtime manipulation framework
  - Created by saurik
  - Can instrument apps on iOS
  - Inherent in most jailbreaks
  - Pre-installed with Cydia

# Tweaks

- Also called **substrate extensions**
- Developed using the Cydia Substrate C API
- Compiled as dynamic libraries
- Placed in **/Library/MobileSubstrate/  
DynamicLibraries**
- Loaded into an app by **MobileLoader**

# Filters

- Prevent your extension being loaded into every new process
- **Filters** are plist files
  - In binary plist, XML, or JSON format
  - Name is same as your tweak, with **.plist** file extension

```
-rw--r-xr-- x 1 root  staff 1485584 Oct 24 16:12 mdsectweak.dylib*
```

```
-rw-r-r- 1 root  staff 304 Oct 24 16:12 mdsectweak.plist
```



```
<?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>Filter</key>
    <dict>
        <key>Bundles</key>
        <array>
            <string>com.mdsec.lab1-1a</string>
        </array>
    </dict>
</dict>
</plist>
```

mdsectweak.  
plist

Filters by  
bundle  
identifier

# Tweak Development Environments

- iOSOpenDev
  - Limited to OS X
- **Theos**
  - Works on iOS, OS X, and Linux
  - Recommended
- Captain Hook
  - Dated, limited to OS X

# Key Functions

- MSHookFunction
- MSFindSymbol
- MSGetImageByName
- MSHookMessageEx

# Key Functions

- MSHookFunction
  - Hooks native C or C++ code functions
  - Uses a trampoline to divert the execution flow to a replacement function
- MSFindSymbol
  - Finds symbols by name
  - Not possible with stripped apps

# Key Functions

- `MSGetImageByName`
  - Loads a dynamic library
  - If it's not already loaded
- `MSHookMessageEx`
  - Implements method swizzling for functions that inherit from **NSObject**

# Example

```
1: #include <substrate.h>
2: #include <sys/stat.h>
3:
4: static int (*oldStat)(const char *path, struct stat *buf);
5:
6: int newStat(const char *path, struct stat *buf)
7: {
8:     NSLog(@"Stat hooked - checking for bash");
9:     if (strcmp(path, "/bin/bash") == 0)
10:         return ENOENT;
11:
12:     return oldStat(path, buf);
13: }
14:
15: MSInitialize {
16:     MSHookFunction(stat, newStat, &oldStat);
17: }
```

- Line 4: **oldStat** points to original **stat()** function, which shows a file's status
- Lines 6-13: Replacement stat
  - Says **/bin/bash** does not exist

# Example

```
1: #include <substrate.h>
2: #include <sys/stat.h>
3:
4: static int (*oldStat)(const char *path, struct stat *buf);
5:
6: int newStat(const char *path, struct stat *buf)
7: {
8:     NSLog(@"Stat hooked - checking for bash");
9:     if (strcmp(path, "/bin/bash") == 0)
10:         return ENOENT;
11:
12:     return oldStat(path, buf);
13: }
14:
15: MSInitialize {
16:     MSHookFunction(stat, newStat, &oldStat);
17: }
```

- Line 15: **MSInitialize** loads its contents first when the app loads
- Line 16: **MSHookFunction** has three arguments
  - Symbol to replace, new function, old function

# Cycript

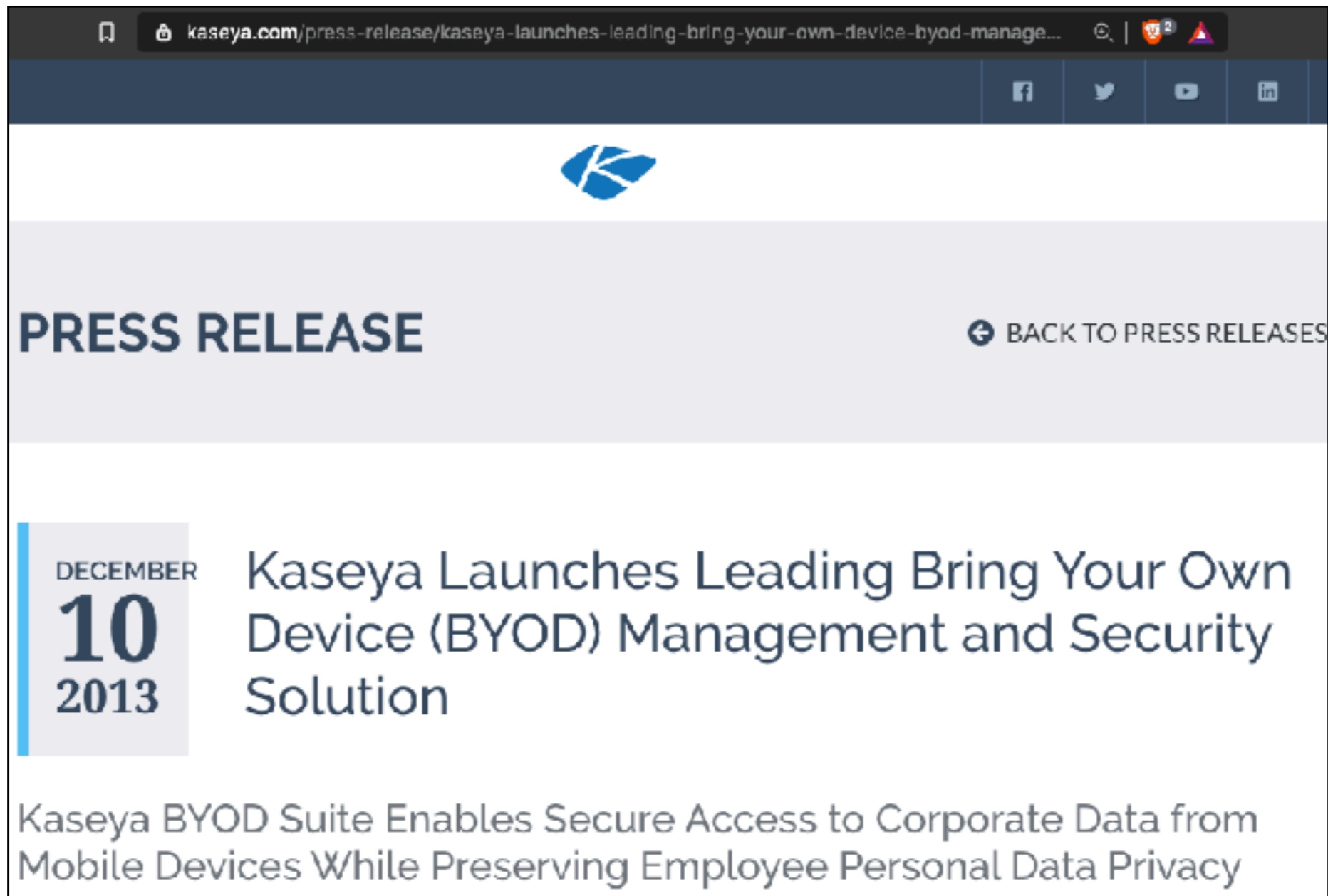
- A runtime instrumentation tool for iOS apps
  - Blends JavaScript and Objective-C
  - Can access and manipulate objects in a running app
- Able to
  - Brute-force local authentication
  - Steal encryption keys from populated objects
  - Force loading of view containers



# Pivoting to Internal Networks





- **BYOD** (Bring Your Own Device)
- **MDM** (Mobile Device Management)
- Apps that let you connect to company resources from a phone
- If vulnerable, allow an attacker into the internal network


# Kaseya BYOD



The image is a screenshot of a web browser displaying a press release from Kaseya. The browser's address bar shows the URL: [kaseya.com/press-release/kaseya-launches-leading-bring-your-own-device-byod-manage...](http://kaseya.com/press-release/kaseya-launches-leading-bring-your-own-device-byod-manage...). The page features the Kaseya logo at the top center. Below the logo, the text "PRESS RELEASE" is displayed in a large, bold, dark blue font. To the right of this text is a link labeled "BACK TO PRESS RELEASES" with a left-pointing arrow icon. The main content of the press release is dated "DECEMBER 10 2013" and has the headline "Kaseya Launches Leading Bring Your Own Device (BYOD) Management and Security Solution". Below the headline is a sub-headline: "Kaseya BYOD Suite Enables Secure Access to Corporate Data from Mobile Devices While Preserving Employee Personal Data Privacy".

[kaseya.com/press-release/kaseya-launches-leading-bring-your-own-device-byod-manage...](#)



**PRESS RELEASE** [← BACK TO PRESS RELEASES](#)

DECEMBER  
**10**  
2013

**Kaseya Launches Leading Bring Your Own Device (BYOD) Management and Security Solution**

Kaseya BYOD Suite Enables Secure Access to Corporate Data from Mobile Devices While Preserving Employee Personal Data Privacy

# Attacking Kaseya BYOD

- Kaseya gateway provides service to internal resources
  - Can be accessed by **Kaseya Secure Browser**
  - With no further authentication
- Compromise of mobile device exposes internal resources

# Instrumentation with Frida

- Frida is a standalone instrumentation framework
  - Does not use Substrate
- No modification to the device required
  - Other than running the **frida-server** binary
- Controlled by a client over USB or the network

# Dynamic Linker

- In Linux, the **LD\_PRELOAD** environment variable
  - Dynamically loads a library into a process
- In Mac OS or iOS, use **DYLD\_INSERT\_LIBRARIES**

```
launchctl setenv DYLD_INSERT_LIBRARIES "/usr/lib/isjailbroken.dylib"
```

# **Understanding Interprocess Communication**

# Sandbox

- iOS apps run in an isolated sandbox
- Interprocess communication is prohibited
- Exceptions
  - Pasteboard
  - Registered protocol handlers
  - Application extensions

# Attacking Protocol Handlers

- To open the App Store app, use
  - **itms-apps://itunes.apple.com/app/id<num>**
- You can define a custom URL scheme in your app's **Info.plist** file, such as
  - **myvoip://dialer/?call=123**



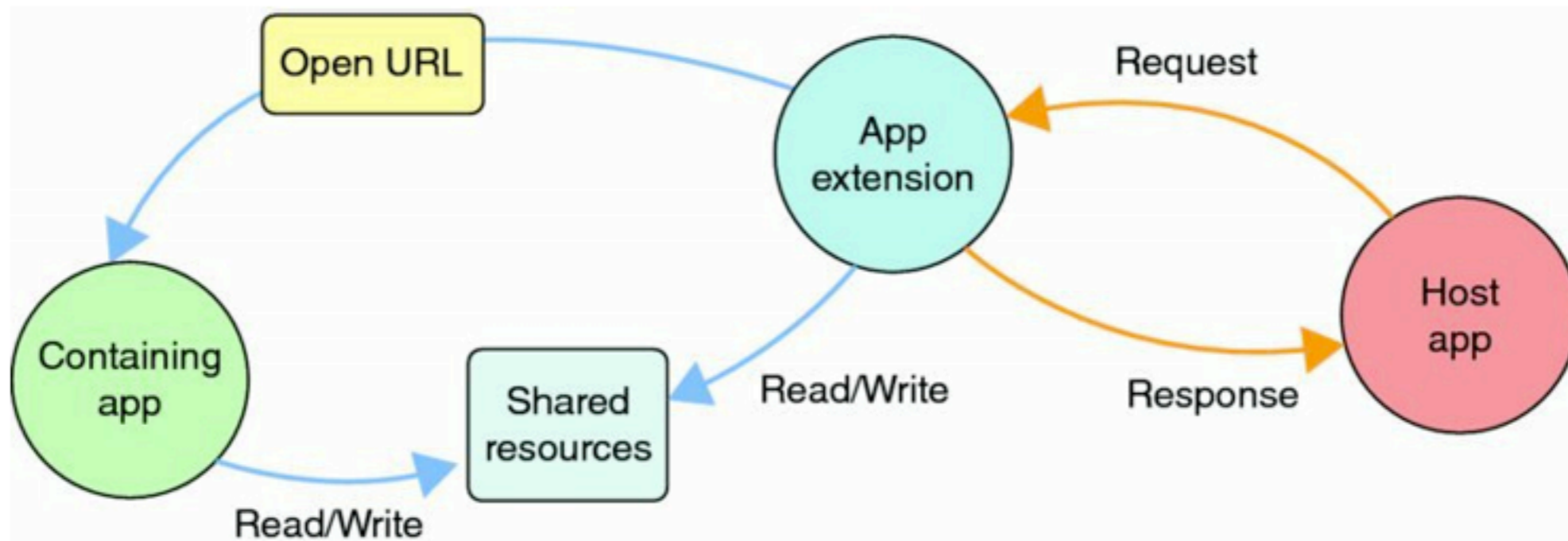
# Automatic Loading

- In an iframe on a web page
  - `<iframe src="myvoip://dialer/?call=0044906123123 "></iframe>`
- This happened with Skype

# Application Extensions

- Some are pre-defined by Apple
  - **Today** -- extend the Today view of the notification center
  - **Share** -- to share content with other apps
  - **Custom Keyboard**

# Application Extensions



**Figure 3.24** An app extension can indirectly communicate and share resources with the containing app.

# 1 Password

- Uses an extension so other apps can query credentials, such as Twitterific
- A malicious app could request credentials for any domain

# Attacking Using Injection

# iOS Entry Points

- Web applications
- URL schemes
- File types
- AirDrop
- iBeacons
- Bluetooth
- Wi-Fi
- Pasteboards
- Application extensions

# Injecting into UIWebViews

- UIWebView renders web content from
  - HTML
  - PDF
  - RTF
  - Office documents
  - iWork documents
- Build on WebKit, like Safari and MobileSafari

# UIWebView

- Supports JavaScript
  - Cannot be disabled
- XSS attacks are possible
  - Can steal content, such as the Address Book



# Skype XSS

- Skype iOS app allowed script injection into a user's full name
- Could access the local file system
- And upload the address book

# Injecting into Client-Side Data Stores

- SQLite databases
  - Vulnerable to SQL injection
- Exposes data, but not usually command injection

# Injecting into XML

- "Billion Laughs" DoS attack
  - Multiple nested XML entities
  - Expanding them uses excessive resources
- If parsing of external entities is allowed
  - Could be used to attack web apps on the local network

# Injection into File-Handling Routines

- User can control a filename
- Directory traversal attacks
- Example:
  - Joe can upload a profile pic to **Documents/joe/joepic.png**
  - Joe can change the filename to
    - **../jane/janepic.png** to read or write to another user's folder

**Kahoot!**