NESO Security Labs

Pentesting iOS Apps

Runtime Analysis and Manipulation

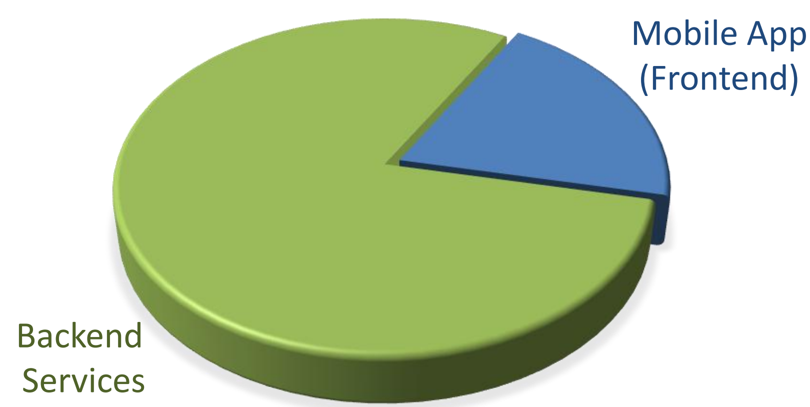
Andreas Kurtz

About

- PhD candidate at the Security Research Group, Department of Computer Science, University of Erlangen-Nuremberg
  - Security of mobile devices & mobile Apps
  - Dynamic analysis of iOS Apps
- Co-Founder of NESO Security Labs GmbH
  - Software security
  - Penetration testing, static code analysis

Pentesting iOS Apps

- Status quo: Focus on backend services
  - Well-known methodologies and techniques
  - Numerous tools available
- So far only little information on mobile App assessments
- Lack of tools



What this talk is about

- Introduction to the Objective-C Runtime
  – Backgrounds, techniques and tools for manipulating iOS Apps at runtime

- Use cases and impacts
  – Pentesters should be able to explore the attack surface of iOS Apps more efficiently
  – Developers might prefer to avoid client-side logic and security measures in the future

Objective-C Runtime

INTRODUCTION

Objective-C

- Provides a set of extensions to the C programming language

- Additions are mostly based on Smalltalk
  - Object-oriented
  - Messaging
  - Dynamic typing
  - Reflection

These concepts make Objective-C quite attractive from a hacking perspective

Objective-C

- Sample Code:

```
HelloWorld *hello = [[HelloWorld alloc] init];
[hello sayHello:@"DeepSec"];
```

```
- (void) sayHello: (NSString *) string {
    printf("Hello %s!", [string UTF8String]);
}
```

Objective-C Runtime

- Apps are linked to *libobjc.A.dylib*

```
# otool -L HelloWorld

HelloWorld:
/System/Library/Frameworks/Foundation.framework/Foundation
(compatibility version 300.0.0, current version 890.1.0)
/usr/lib/libobjc.A.dylib (compatibility version 1.0.0,
current version 228.0.0)
[..]
```

This library provides all runtime functionalities of the Objective-C Runtime

Objective-C Runtime

- Most important function: *objc_msgSend*

- Example

```
Class class = objc_getClass("HelloWorld");
id receiver = [[class alloc] init];
SEL selector = NSSelectorFromString(@"sayHello:");

objc_msgSend(theReceiver,theSelector,@"DeepSec");
```

Pointer to an instance of the class, whose method we want to call

Objective-C Runtime

- Most important function: *objc_msgSend*

- Example

```
Class class = objc_getClass("HelloWorld");
id receiver = [[class alloc] init];
SEL selector = NSSelectorFromString(@"sayHello:");

objc_msgSend(theReceiver,theSelector,@"DeepSec");
```

The selector of the method that handles the message

NESO Security Labs

Objective-C Runtime

- Most important function: *objc_msgSend*

- Example

```
Class class = objc_getClass("HelloWorld");
id receiver = [[class alloc] init];
SEL selector = NSSelectorFromString(@"sayHello:");
```

```
objc_msgSend(theReceiver,theSelector,@"DeepSec");
```

A variable argument list containing the arguments to the method

Static vs. Dynamic Analysis

- During static analysis, control flow is lost when *objc_msgSend* is called

- Characteristics of the Objective-C Runtime enables comprehensive dynamic analysis

Technique	Usage
▪ Intercept messages	▪ Trace internal control flow
▪ Send arbitrary messages to existing objects ▪ Rewrite implementations of arbitrary methods	▪ Manipulate internal state and processing logic of an iOS App

Backgrounds & Techniques

RUNTIME MANIPULATION

Starting Point

- Goal: Black box analysis of an arbitrary iOS App
  - Enterprise or AppStore App
  - Binary format (no source code available)

- Approach: Examine the iOS App on a jailbroken device
  - Removes the limitations imposed by Apple
  - Provides root access to the operating system
  - Enables the installation of additional software
  - Enables access to the Objective-C Runtime!

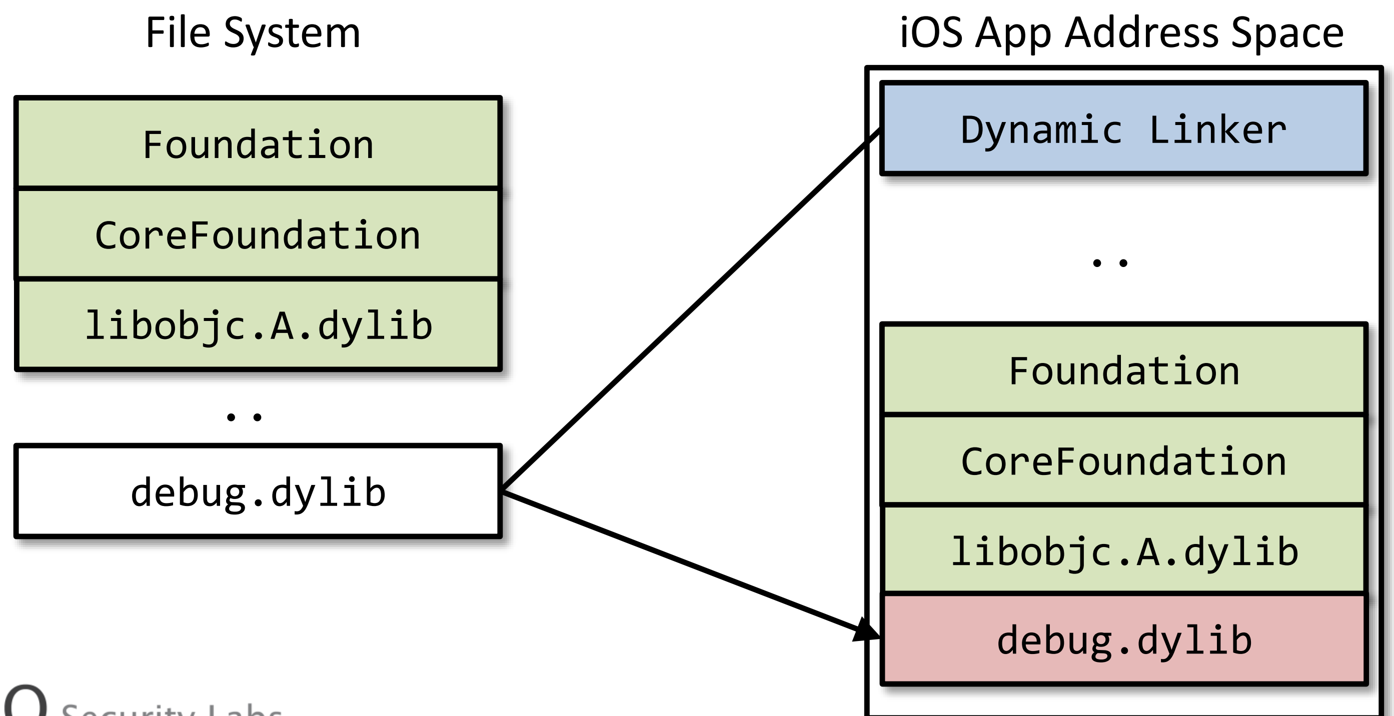
Runtime Manipulation

- Objective-C Runtime [1] offers a wide range of opportunities to manipulate existing iOS Apps

- Two different approaches
  - Injecting a static library with new functionalities
  - Injecting an interpreter for on-the-fly manipulations

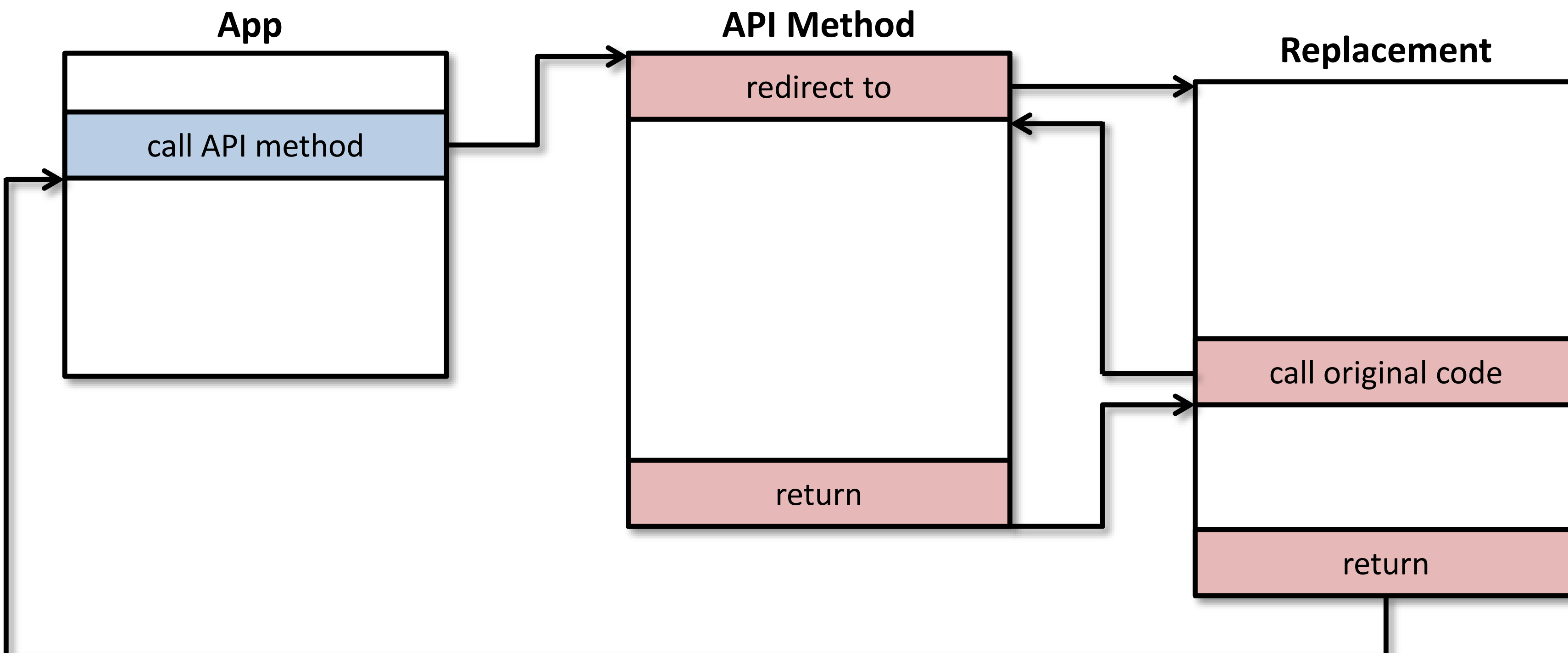
Dynamic Library Injection

- Advise the dynamic linker to load a dynamic shared library (*DYLD_INSERT_LIBRARIES)* [2]



Runtime Patching

- Replace existing methods and reroute program control during library initialization



Hooking in Practice

- MobileSubstrate [3]
  - *MobileLoader* loads 3rd-party patching code into the running application
  - *MobileHooker* is used to hook and replace system methods and functions

```
IMP MSHookMessage(Class class, SEL selector, IMP replacement, const
char* prefix);

void MSHookFunction(void* function, void* replacement, void**
p_original);
```

- Recommendation: Theos suite eases the development of MobileSubstrate extensions (Tweaks) [4]

Example: Fake Device Information

```
#include "substrate.h"
#import <Foundation/Foundation.h>

NSString *replaced_UIDevice_uniqueIdentifier() {
    return @"DeepSec";
}

__attribute__((constructor))
static void initialize() {
    MSHookMessage(objc_getClass("UIDevice"),
                  @selector(uniqueIdentifier),
                  (IMP)replaced_UIDevice_uniqueIdentifier,
NULL);
}
```

Runtime Manipulation

- Objective-C Runtime [1] offers a wide range of opportunities to manipulate existing iOS Apps

- Two different approaches
  - Injecting a static library with new functionalities ✓
  - Injecting an interpreter for on-the-fly manipulations

Cycript: Objective-JavaScript [5]

“A programming language designed to blend the barrier between Objective-C and JavaScript.”

- Injects a JavaScript interpreter into a running App
  - Based on MobileSubstrate
- Enables runtime manipulations in a flexible way [6], [7]

Example: Fake Device Information

- **Step 1:** Attach to the App process

```
# cycript -p <PID>
```

- **Step 2:** Determine the current UDID

```
cy# [[UIDevice currentDevice] uniqueIdentifier];
@"768f0c93a69276d190b6…"
```

Example: Fake Device Information

- **Step 3:** Replace the implementation of the API method

```
cy# UIDevice.messages['uniqueIdentifier'] =
    function() { return @"DeepSec"; }
```

- **Step 4:** Query the UDID again

```
cy# [[UIDevice currentDevice] uniqueIdentifier];
@"DeepSec"
```

Example: Fake Device Information

Example: Fake Device Information

- Example demonstrates the diverse possibilities of iOS runtime injection
- This might be useful in different scenarios
  - Apps that rely on hardware identifier for authentication
  - Apps that use binary or any proprietary protocols
- Easier to manipulate the App endpoint, compared to modifications at protocol-level

USE CASES

Advantages of Runtime Manipulation

- By using these techniques, running Apps can be extended with additional debugging and runtime tracing capabilities

- This assists security assessments of iOS Apps
  - Eases the discovery of vulnerabilities
  - Simplifies bypassing client-side limitations and restrictions

Evaluate Encryption Schemes

- Typical question: Which App methods are called after the “Login” button is pressed?
- Idea: Make use of dynamic analysis to reconstruct the control flow of an App
  - Use the results to navigate through static code
- Solution: Log all messages to *objc_msgSend*

The gdb way

```
(gdb) exec-file /var/mobile/Applications/<APP-EXECUTABLE>
Reading symbols for shared libraries . done
(gdb) attach <PID>
Attaching to program: `/private/var/mobile/Applications/...', process PID.
Reading symbols for shared libraries . done
Reading symbols for shared libraries ............................... done
Reading symbols for shared libraries + done
0x364d7004 in mach_msg_trap ()
(gdb) break objc_msgSend
Breakpoint 1 at 0x32ce2f68
(gdb) commands
Type commands for when breakpoint 1 is hit, one per line.
End with a line saying just "end".
>printf "-[%s %s]\n", (char *)class_getName($r0),$r1
>c
>end
(gdb) c
Continuing.
```

The gdb way

```
Breakpoint 1, 0x32ce2f68 in objc_msgSend ()
-[UIStatusBarServer _receivedStatusBarData:actions:]

Breakpoint 1, 0x32ce2f68 in objc_msgSend ()
-[UIStatusBar statusBarServer:didReceiveStatusBarData:withActions:]

Breakpoint 1, 0x32ce2f68 in objc_msgSend ()
-[UIStatusBar _currentComposedData]

Breakpoint 1, 0x32ce2f68 in objc_msgSend ()
-[UIStatusBar _currentComposedDataForStyl

Breakpoint 1, 0x32ce2f68 in objc_msgSend

-[UIStatusBarComposedData alloc]

[..]
```

Very noisy! All background activities of the runtime are shown as well.

App Tracing

- Preferred approach: Intercept messages to *objc_msgSend* within the runtime

- Apply filters with different granularity
  - Enumerate registered App classes and methods using the Objective-C Runtime API (*objc_getClassList*, *class_copyMethodList*, etc.)
  - Output a trace of only matching items

- Inspired by Aspective-C [8] and Subjective-C [9]

App Tracing

- Tricky part is to handle all parameters and to continue normal execution
  - Logging itself modifies CPU registers and the stack

- Current execution state has to be preserved
  - Allocate an alternate stack within heap memory
  - Backup `r0 - r3` and `lr` registers to alternate stack
  - Do the logging and filtering
  - Restore `r0 - r3` and `lr`
  - Continue execution

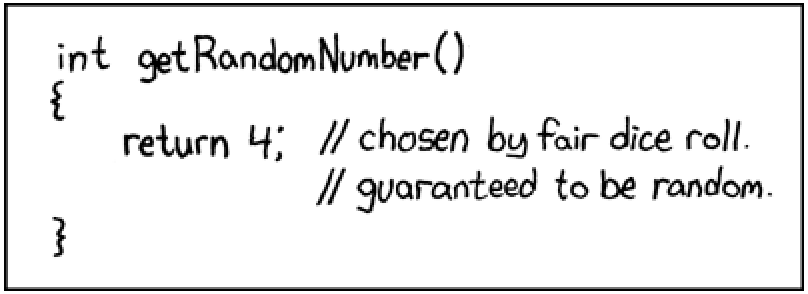
Sample Output

```
+ [SyncManager sharedSyncManager]
- [SyncManager init]
- [SyncManager setSynDocumentOpen:], args: 0
+ [DataModel setSynchManager:], args: <0x1102ce30>
+ [DataModel initFromFile]
+ [DataModel securityModelFilePath]
+ [DataModel securityModelFilePath]
+ [PBKDF2 getKeyForPassphrase:], args: <__NSCFConstantString 0x15e2e4: >
+ [CryptoUtils decrypt]
+ [DataModel sharedModel]
+ [CryptoUtils md5:], args: <__NSCFConstantString 0x15dea4: >
+ [DataModel sharedModel]
```

Encryption scheme is based on a hardcoded key within the App

Sample Output

```
+ [SyncManager sharedSyncManager]
- [SyncManager init]
- [SyncManager setSynDocumentOpen:], args: 0
+ [DataModel setSynchManager:], args: <0x1102ce30>
+ [DataModel initFromFile]
+ [DataModel securityModelFilePath]
+ [DataModel securityModelFilePath]
+ [PBKDF2 getKeyForPassphrase:], args: <__NSCFConstantString 0x15e2e4: >
+ [CryptoUtils decrypt]
+ [DataModel sharedModel]
+ [CryptoUtils md5:], args: <__NSCFConstantString 0x15dea4: >
+ [DataModel sharedModel]
```



Advantages of Runtime Manipulation

- The ability to manipulate Apps at runtime strikes out new paths
  - Discover weak/missing encryption
  - Bypassing client-side restrictions
  - Execution of hidden functionality, which was not supposed to be accessible
  - Unlock additional features and premium content
  - Dump copyright-protected content
  - Etc.

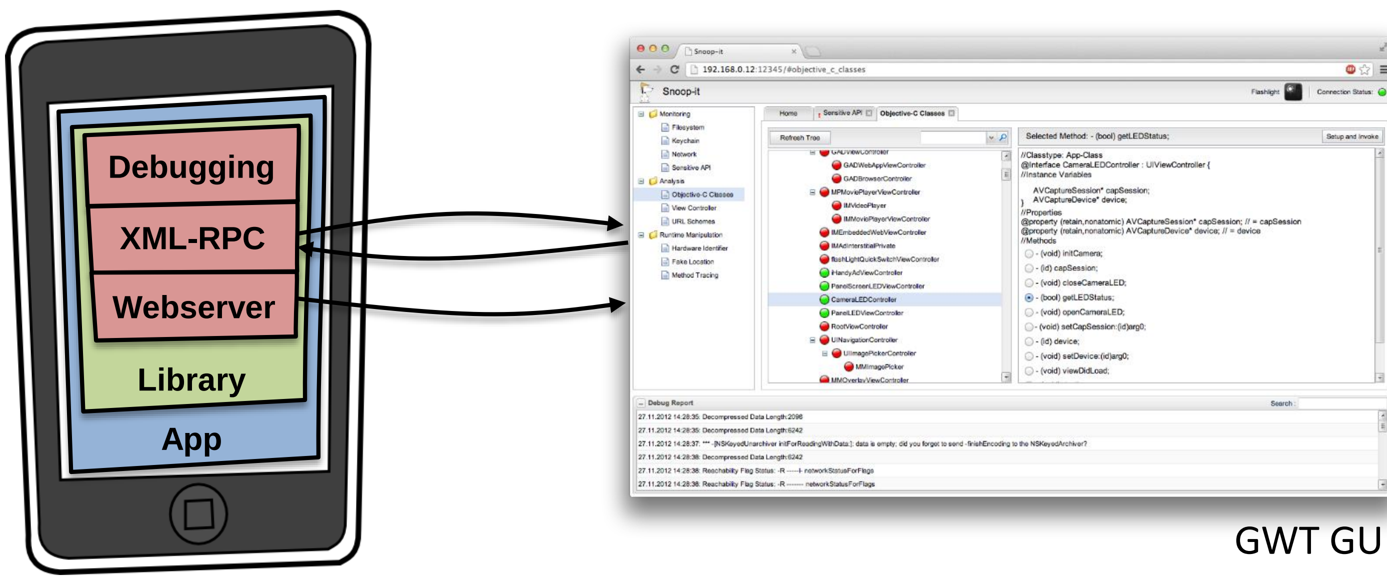
Lack of Tools

“Security will not get better until tools for practical exploration of the attack surface are made available”

- Josh Wright

Closing the Gap

- Retrofitting existing apps with debugging and runtime tracing capabilities



GWT GUI

Introducing *Snoop-it*

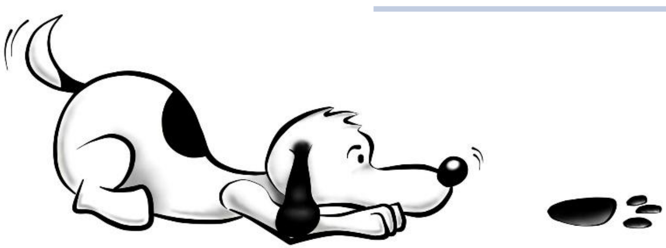
- A tool to assist security assessments and dynamic analysis of iOS Apps

Features

Monitoring	File system access (print data protection classes)
	Keychain access
	HTTP(S) connections
	Access to sensitive API (address book, photos etc.)
	Debug outputs
	Tracing App internals (objc_msgSend)

Features

Analysis / Manipulation	Fake hardware identifier (UDID, Wireless MAC, etc.)
	Fake location/GPS data
	Explore and force display of available ViewControllers
	List custom URL schemes
	List available Objective-C classes, objects and methods
	Invoke and replace arbitrary methods at runtime



Features

Other

- Simple installation and configuration
- Easy to use graphical user interface
- Plenty of filter and search options
- Detailed description of the XML-RPC web service interface
- Freely available at the end of this year

Getting Started

- There's an App for That!™

Getting Started

- There's an App for That!™

  ➊ Open the *Snoop-it Configuration* App

Getting Started

- There's an App for That!™

  ➊ Open the *Snoop-it Configuration* App

  ➋ Select Apps (System/Cydia/AppStore) to analyze

Getting Started

- There's an App for That!™

  ❶ Open the *Snoop-it Configuration* App

  ❷ Select Apps (System/Cydia/AppStore) to analyze

  ❸ Adjust settings (GUI, Authentication, …)

Getting Started

- There's an App for That!™

❶ Open the *Snoop-it Configuration* App

❷ Select Apps (System/Cydia/AppStore) to analyze

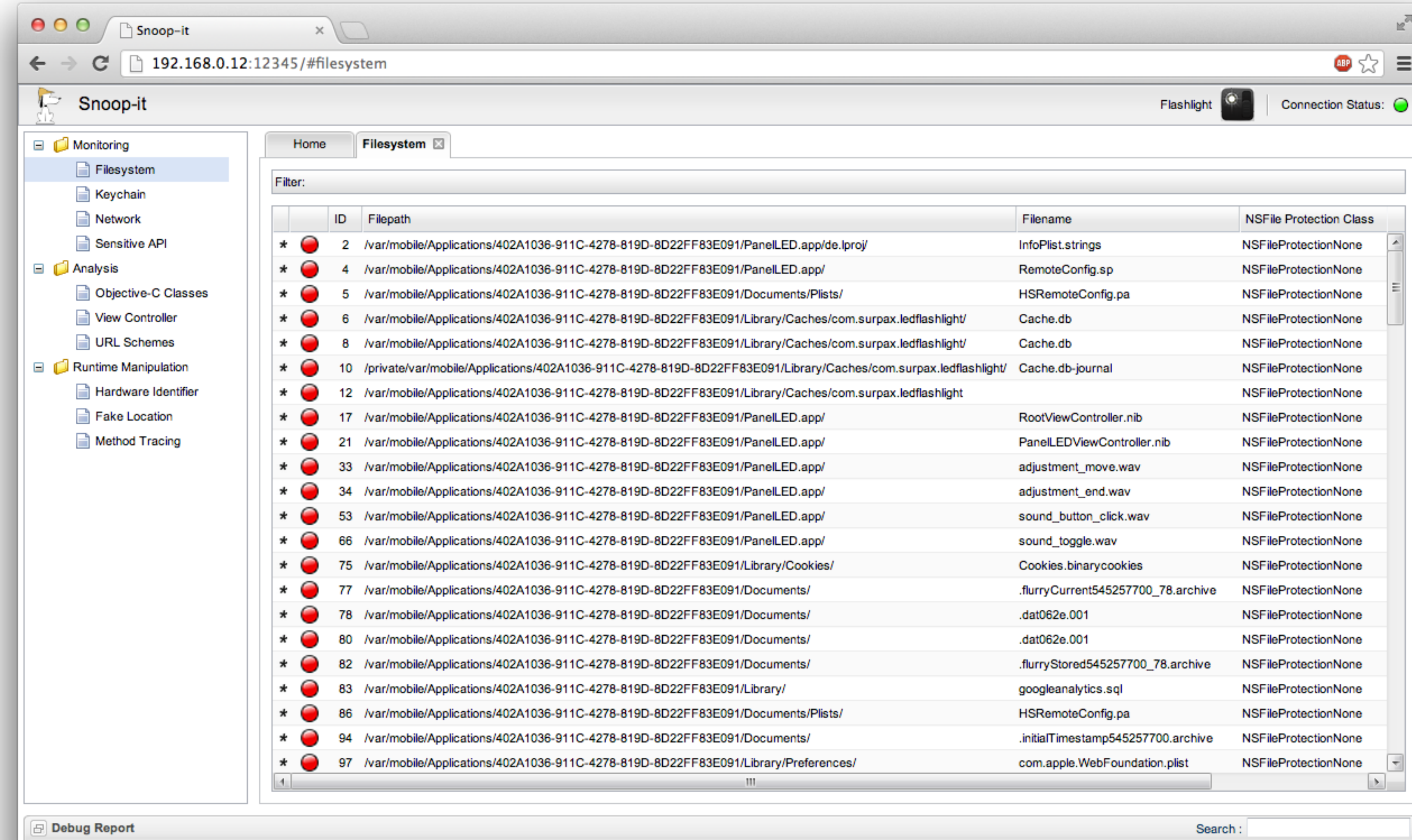
❸ Adjust settings (GUI, Authentication, …)

❹ Run App & point your browser to the *Snoop-it* web interface

DEMO

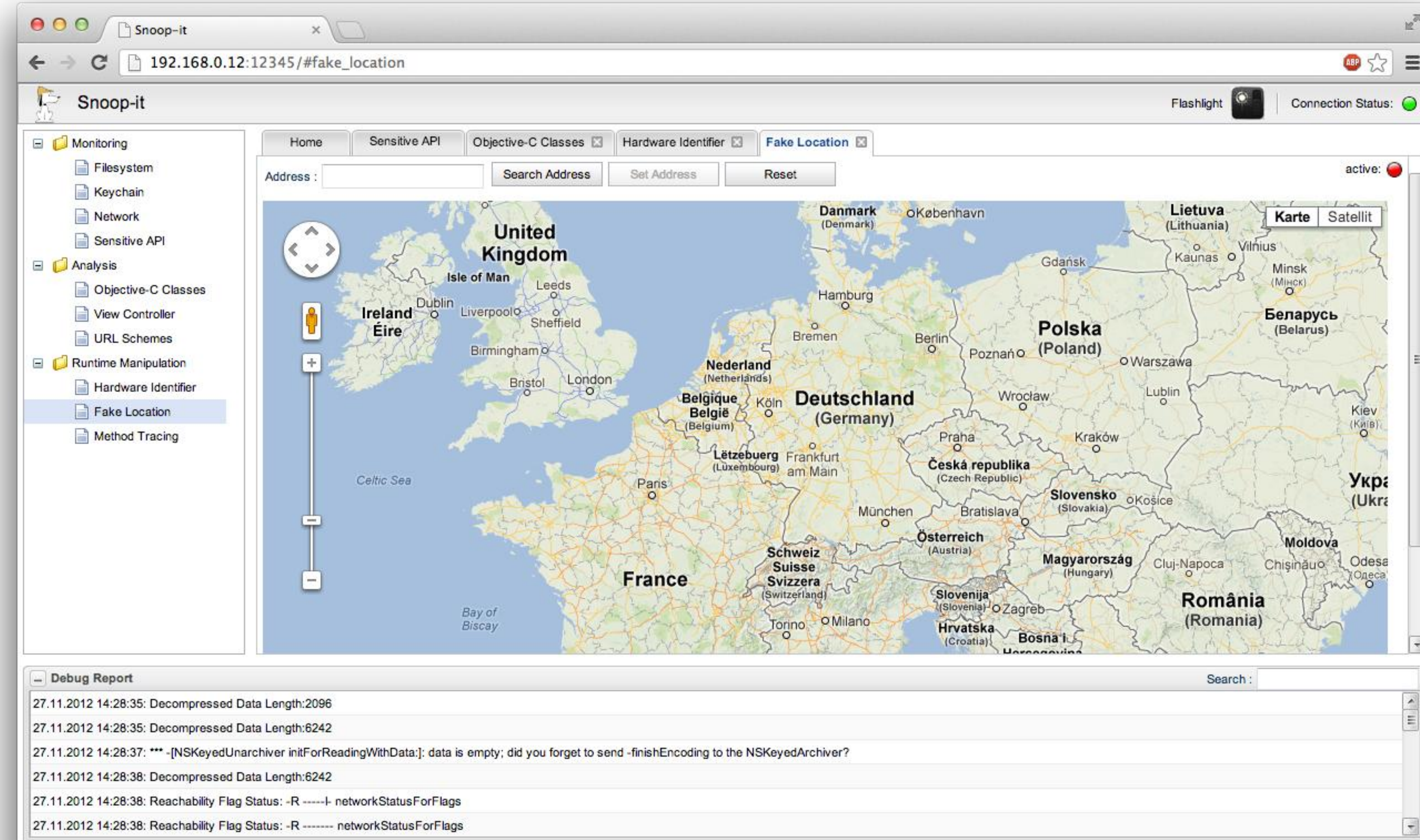
Please follow me on Twitter (@aykay)
to stay up-to-date with the latest news on *Snoop-it*

Filesystem Monitor

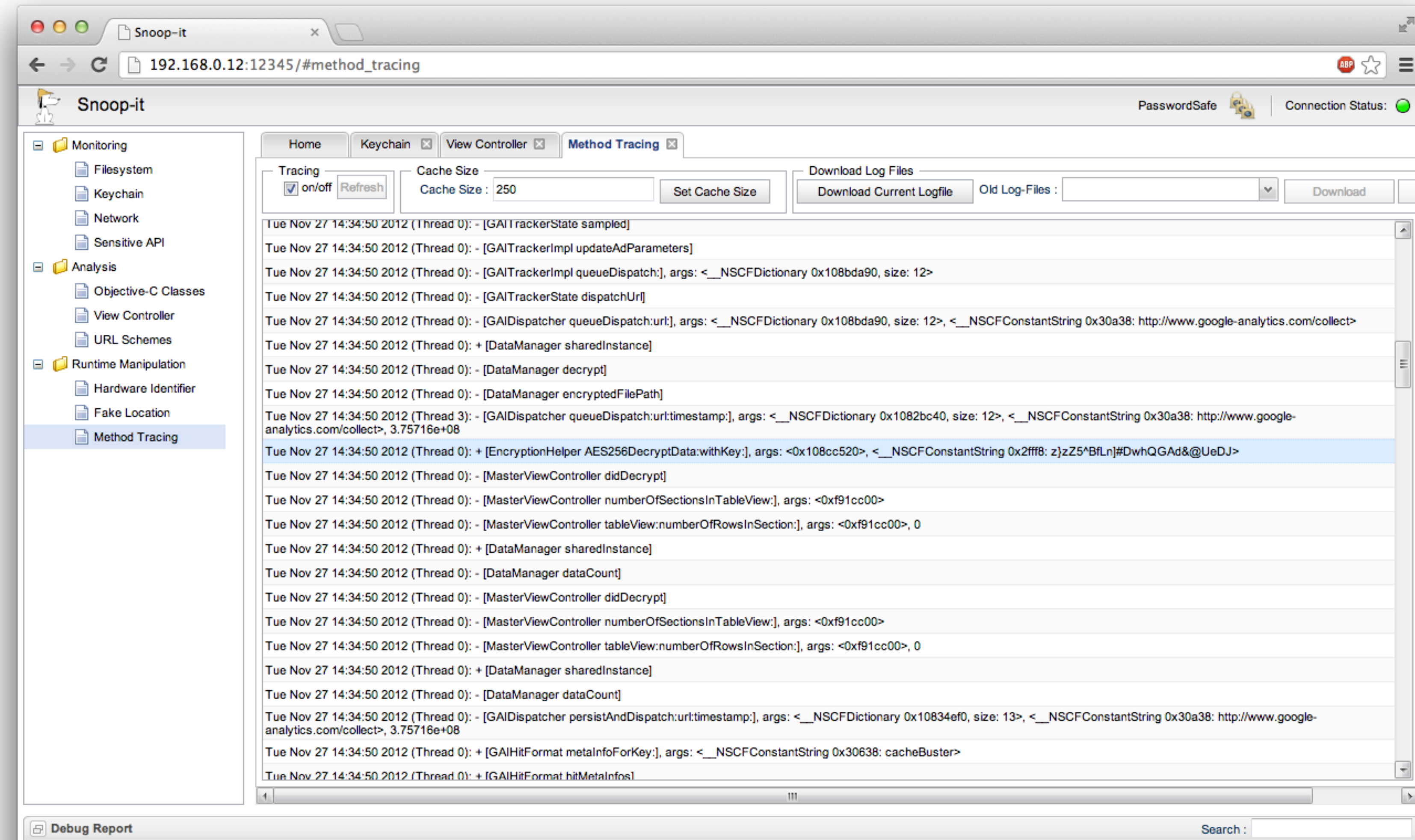


		ID	Filepath	Filename	NSFile Protection Class
*		2	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/PanelLED.app/de.lproj/	InfoPlist.strings	NSFileProtectionNone
*		4	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/PanelLED.app/	RemoteConfig.sp	NSFileProtectionNone
*		5	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Documents/Plists/	HSRemoteConfig.pa	NSFileProtectionNone
*		6	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Library/Caches/com.surpax.ledflashlight/	Cache.db	NSFileProtectionNone
*		8	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Library/Caches/com.surpax.ledflashlight/	Cache.db	NSFileProtectionNone
*		10	/private/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Library/Caches/com.surpax.ledflashlight/	Cache.db-journal	NSFileProtectionNone
*		12	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Library/Caches/com.surpax.ledflashlight		NSFileProtectionNone
*		17	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/PanelLED.app/	RootViewController.nib	NSFileProtectionNone
*		21	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/PanelLED.app/	PanelLEDViewController.nib	NSFileProtectionNone
*		33	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/PanelLED.app/	adjustment_move.wav	NSFileProtectionNone
*		34	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/PanelLED.app/	adjustment_end.wav	NSFileProtectionNone
*		53	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/PanelLED.app/	sound_button_click.wav	NSFileProtectionNone
*		66	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/PanelLED.app/	sound_toggle.wav	NSFileProtectionNone
*		75	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Library/Cookies/	Cookies.binarycookies	NSFileProtectionNone
*		77	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Documents/	.flurryCurrent545257700_78.archive	NSFileProtectionNone
*		78	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Documents/	.dat062e.001	NSFileProtectionNone
*		80	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Documents/	.dat062e.001	NSFileProtectionNone
*		82	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Documents/	.flurryStored545257700_78.archive	NSFileProtectionNone
*		83	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Library/	googleanalytics.sql	NSFileProtectionNone
*		86	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Documents/Plists/	HSRemoteConfig.pa	NSFileProtectionNone
*		94	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Documents/	.initialTimestamp545257700.archive	NSFileProtectionNone
*		97	/var/mobile/Applications/402A1036-911C-4278-819D-8D22FF83E091/Library/Preferences/	com.apple.WebFoundation.plist	NSFileProtectionNone

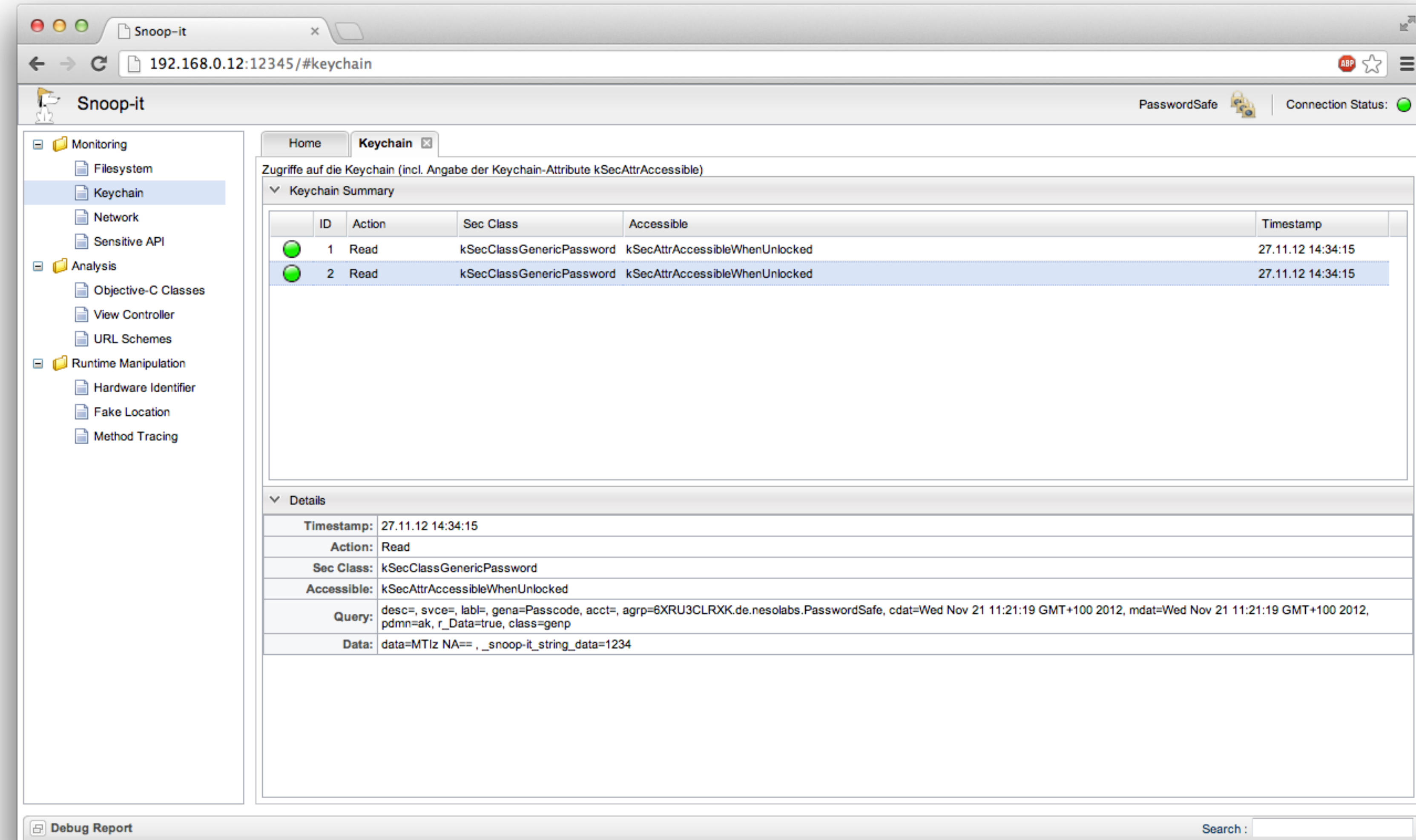
Location Faker



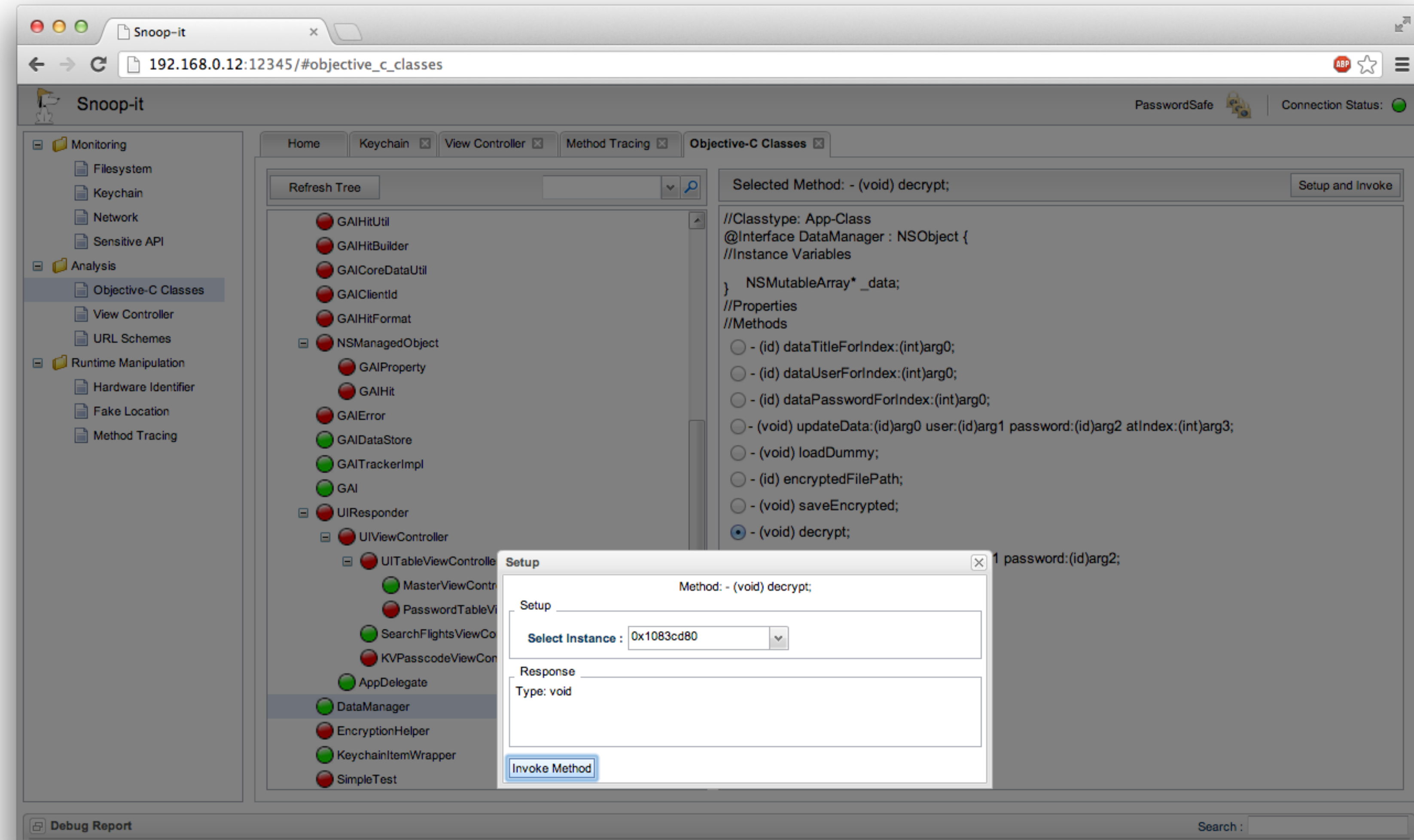
App Tracing


Snoop-it
192.168.0.12:12345/#method_tracing
Snoop-it
PasswordSafe
Connection Status:
Monitoring
Filesystem
Keychain
Network
Sensitive API
Analysis
Objective-C Classes
View Controller
URL Schemes
Runtime Manipulation
Hardware Identifier
Fake Location
Method Tracing
Home
Keychain
View Controller
Method Tracing
Tracing
on/off
Refresh
Cache Size
Cache Size : 250
Set Cache Size
Download Log Files
Download Current Logfile
Old Log-Files :
Download
Tue Nov 27 14:34:50 2012 (Thread 0): - [GAITrackerImpl updateAdParameters]
Tue Nov 27 14:34:50 2012 (Thread 0): - [GAITrackerImpl queueDispatch:], args: <__NSCFDictionary 0x108bda90, size: 12>
Tue Nov 27 14:34:50 2012 (Thread 0): - [GAITrackerState dispatchUrl]
Tue Nov 27 14:34:50 2012 (Thread 0): - [GAIDispatcher queueDispatch:url:], args: <__NSCFDictionary 0x108bda90, size: 12>, <__NSCFConstantString 0x30a38: http://www.google-analytics.com/collect>
Tue Nov 27 14:34:50 2012 (Thread 0): + [DataManager sharedInstance]
Tue Nov 27 14:34:50 2012 (Thread 0): - [DataManager decrypt]
Tue Nov 27 14:34:50 2012 (Thread 0): - [DataManager encryptedFilePath]
Tue Nov 27 14:34:50 2012 (Thread 3): - [GAIDispatcher queueDispatch:url:timestamp:], args: <__NSCFDictionary 0x1082bc40, size: 12>, <__NSCFConstantString 0x30a38: http://www.google-analytics.com/collect>, 3.75716e+08
Tue Nov 27 14:34:50 2012 (Thread 0): + [EncryptionHelper AES256DecryptData:withKey:], args: <0x108cc520>, <__NSCFConstantString 0x2fff8: z}zZ5^BfLn]#DwhQGAd&@UeDJ>
Tue Nov 27 14:34:50 2012 (Thread 0): - [MasterViewController didDecrypt]
Tue Nov 27 14:34:50 2012 (Thread 0): - [MasterViewController numberOfSectionsInTableView:], args: <0xf91cc00>
Tue Nov 27 14:34:50 2012 (Thread 0): - [MasterViewController tableView:numberOfRowsInSection:], args: <0xf91cc00>, 0
Tue Nov 27 14:34:50 2012 (Thread 0): + [DataManager sharedInstance]
Tue Nov 27 14:34:50 2012 (Thread 0): - [DataManager dataCount]
Tue Nov 27 14:34:50 2012 (Thread 0): - [MasterViewController didDecrypt]
Tue Nov 27 14:34:50 2012 (Thread 0): - [MasterViewController numberOfSectionsInTableView:], args: <0xf91cc00>
Tue Nov 27 14:34:50 2012 (Thread 0): - [MasterViewController tableView:numberOfRowsInSection:], args: <0xf91cc00>, 0
Tue Nov 27 14:34:50 2012 (Thread 0): + [DataManager sharedInstance]
Tue Nov 27 14:34:50 2012 (Thread 0): - [DataManager dataCount]
Tue Nov 27 14:34:50 2012 (Thread 0): - [GAIDispatcher persistAndDispatch:url:timestamp:], args: <__NSCFDictionary 0x10834ef0, size: 13>, <__NSCFConstantString 0x30a38: http://www.google-analytics.com/collect>, 3.75716e+08
Tue Nov 27 14:34:50 2012 (Thread 0): + [GAIHitFormat metaInfoForKey:], args: <__NSCFConstantString 0x30638: cacheBuster>
Debug Report
Search :

Keychain Monitor



Runtime Manipulation

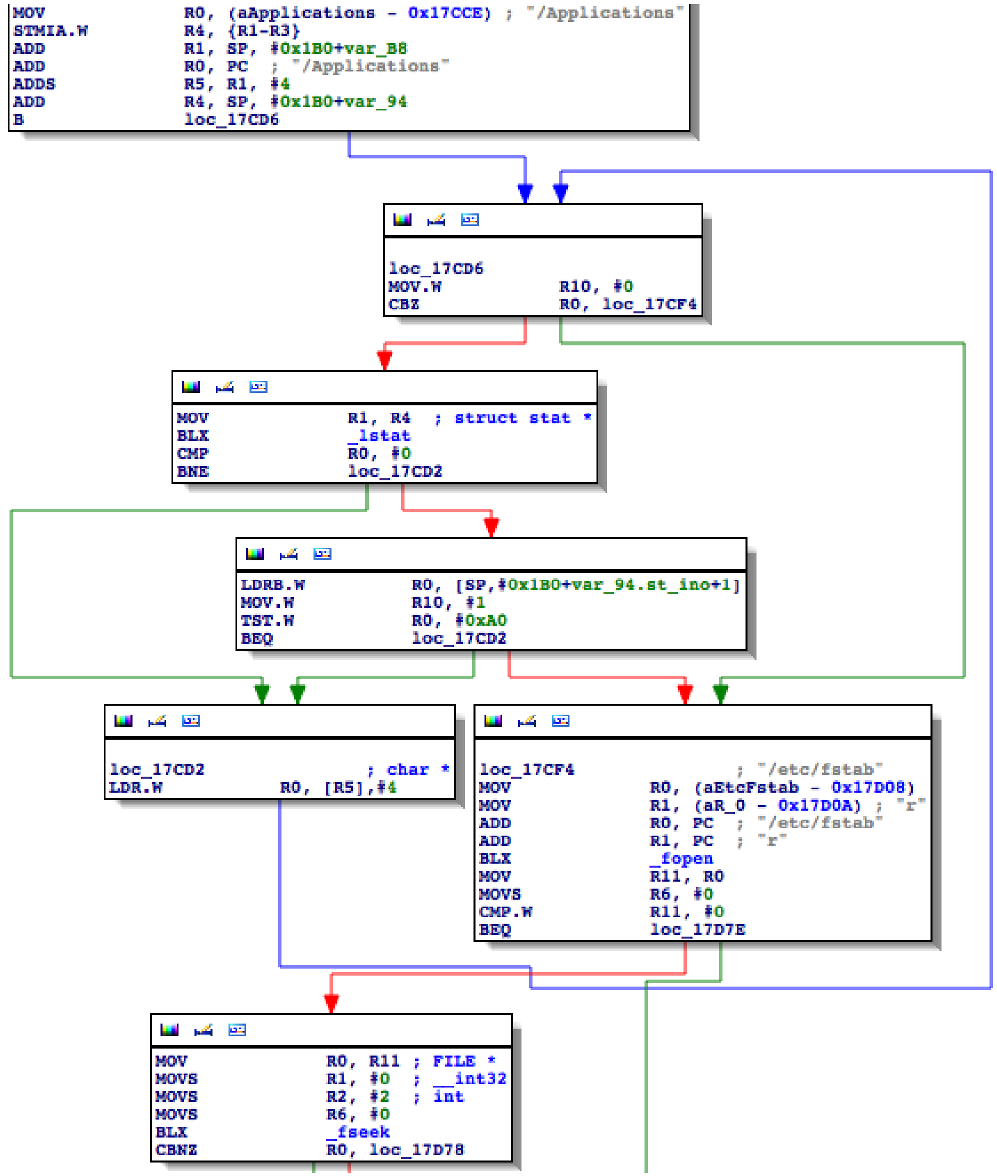


Jailbreak Detection

- Purpose: Verification of platform integrity

- Common checks
  - Suspicious files and directories
  - File system permissions
  - Mount options
  - Symbolic links
  - Dynamic shared libraries
  - SSH Loopback
  - Sandbox integrity (fork)

Jailbreak Detection



Jailbreak Detection

- In order to assess the security of an iOS App, at first the jailbreak detection mechanisms have to be bypassed
  - Binary / Run-time patching to remove all checks (specific, time-consuming)

    ```
    Delegate.messages['isJailbroken'] =
                    function() { return NO; }
    ```

  - Intercept system calls to simulate an unmodified execution environment (generic)

Jailbreak Detection Bypass

- *Snoop-it* supports generic bypass of the most common jailbreak detection mechanisms
  - Simple configuration switch in the Configuration App

Bypassing Jailbreak Detection

DEMO

Securing the Runtime

- Minimum of data/logic on the client-side
- Preferred use of C, at least for security-critical implementations
  - Inline Functions
  - Obfuscation

At least try to,
it's worth a shot.

- Advanced Jailbreak Detection
- Runtime Integrity Checks (*dladdr()*[10])

Summary

- Runtime Analysis and Manipulation facilitates both, *dynamic* and *static* analysis of iOS Apps
- Attack surface of iOS Apps can be explored more efficiently

When in doubt, Snoop-it out!

Acknowledgements

- Thanks to
  - Markus Troßbach *(University of Heidelberg)*
  - Sebastian Stocker *(University of Heidelberg)*
  - Christoph Settgast *(University of Erlangen)*
  - Andreas Weinlein *(University of Erlangen)*
  - Francesca Serpi *(University of Milan)*

References

[1] Objective C Runtime Reference
http://developer.apple.com/library/mac/#documentation/Cocoa/Reference/ObjCRuntimeRef/Reference/reference.html

[2] dyld - the dynamic link editor (DYLD_INSERT_LIBRARIES)
http://developer.apple.com/library/mac/#documentation/Darwin/Reference/Manpages/man1/dyld.1.html

[3] Mobile Substrate
http://iphonedevwiki.net/index.php/MobileSubstrate

[4] Theos
http://iphonedevwiki.net/index.php/Theos

[5] Cycript
http://www.cycript.org

References

[6] Cycript Overview
http://iphonedevwiki.net/index.php/Cycript

[7] Cycript Tips
http://iphonedevwiki.net/index.php/Cycript_Tricks

[8] Aspective-C by saurik
http://svn.saurik.com/repos/menes/trunk/aspectivec/AspectiveC.mm

[9] Subjective-C by KennyTM~
http://networkpx.blogspot.de/2009/09/introducing-subjective-c.html

[10] dladdr - find the image containing a given address
http://developer.apple.com/library/Mac/#documentation/Darwin/Reference/ManPages/man3/dladdr.3.html

NESO Security Labs

Weipertstraße 8-10 · 74076 Heilbronn
☎ +49 (7131) 7669-540
info@nesolabs.de
www.nesolabs.de