iOS App Auditing

Nils Schmidt, 22.02.15

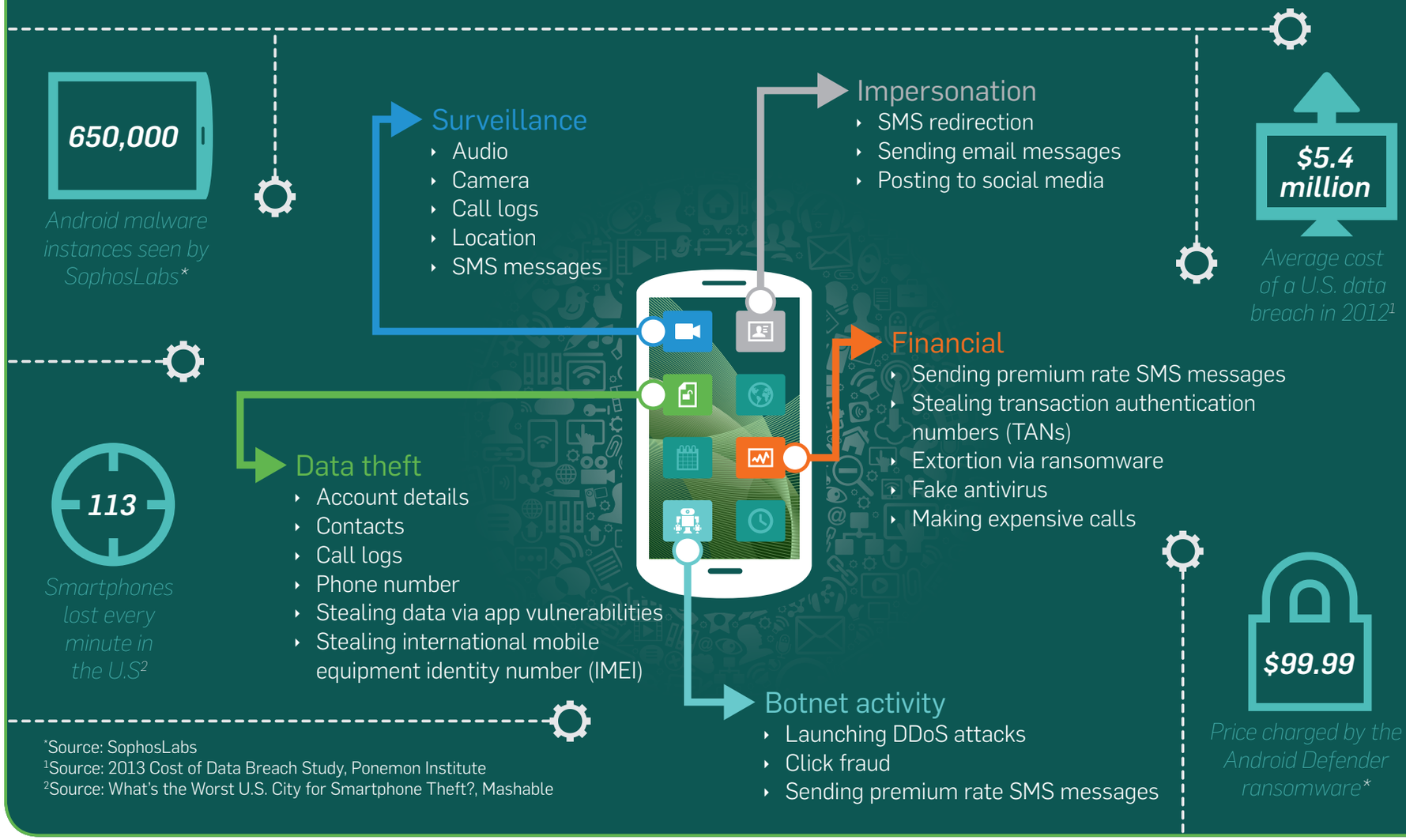
Outline

- Threats / introduction
- Reverse engineering
  - ARM
  - Objective-C
  - Mach-O
- Audit

Threats

Fig. 4 Anatomy of a Hacked Mobile Device: How a hacker can profit from your smartphone

Your Android smartphone may look innocent. But when compromised by malware, it can illegally watch and impersonate you, participate in dangerous botnet activities, capture your personal data, and even steal your money.



*Source: SophosLabs
[1]Source: 2013 Cost of Data Breach Study, Ponemon Institute
[2]Source: What's the Worst U.S. City for Smartphone Theft?, Mashable

ARM

- Reduced Instruction Set Computer (RISC)
- ARM 32/64 Bit
- many General Purpose Registers (GPRs)
  - r0 - r15
  - r7: ptr to current stack frame
  - r13/sp: top of the stack
  - r14/lr: holds return address
  - r15/pc: next instruction address

ARM

- Jumps: b[l][x]
- l (link) => return to caller after function call
- x: exchange instruction set
  - ARM <-> Thumb

Objective-C

- Strict superset of C
- New: Swift
- Calling methods vs. sending messages
  - **Java**: objPtr.method(param1, param2)
  - **Objective-C**:
    - id objc_msgSend(id theReceiver, SEL theSelector, …)
    - objc_msgSend(objPtr, @selector(method:param2Name:), param1, param2)
    - Calling conventions: r0:self, r1: selector, r2-r3, stack

ARM

```
- (void)callWrapper{
  [self objCFunction:1 arg2:2 arg3:3] ;}
```

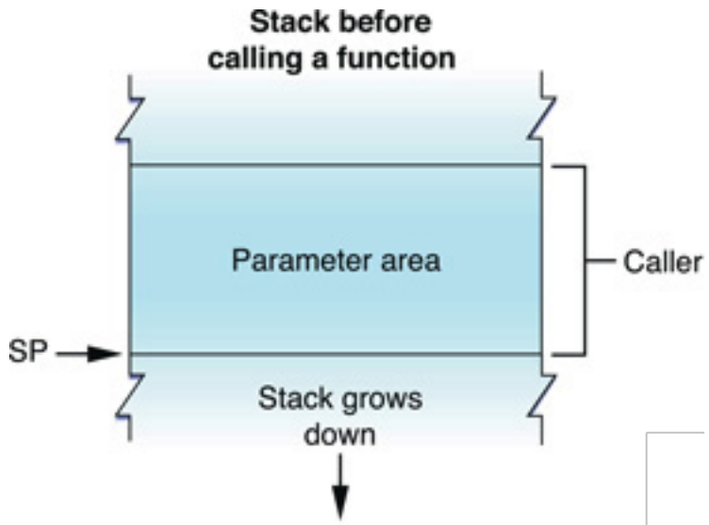
ARM

```
- (void)callWrapper{
   [self objCFunction:1 arg2:2 arg3:3] ;}
```

```
-[AppDelegate callWrapper]:

// function prologue
push {r7, lr}         // save old base pointer, return addr
mov  r7, sp           // frame pointer = stack pointer
sub  sp, #0x4         // allocate additional space on stack

// body
movw r1, #0x3402      // higher r1 = 0x3402
movs r2, #0x3         // r2 = 3
movt r1, #0x0         // lower r1 = 0x0
str  r2, [sp, #0x4]   // store 3rd argument on stack
add  r1, pc           // r1 = addr of
selector(objCFunction:arg2:arg3:)
movs r2, #0x1         // r2 = 1
movs r3, #0x2         // r3 = 2
ldr  r1, [r1]         // r1 = selector(objCFunction:arg2:arg3:)
// [self(r0) objCFunction:1(r2) arg2:2(r3) arg3:3(sp+4)]
blx  imp___symbolstub1__objc_msgSend

// function epilogue
add  sp, #0x4         // deallocate stack space
pop  {r7, pc}         // restore old frame pointer & retaddr
```

ARM

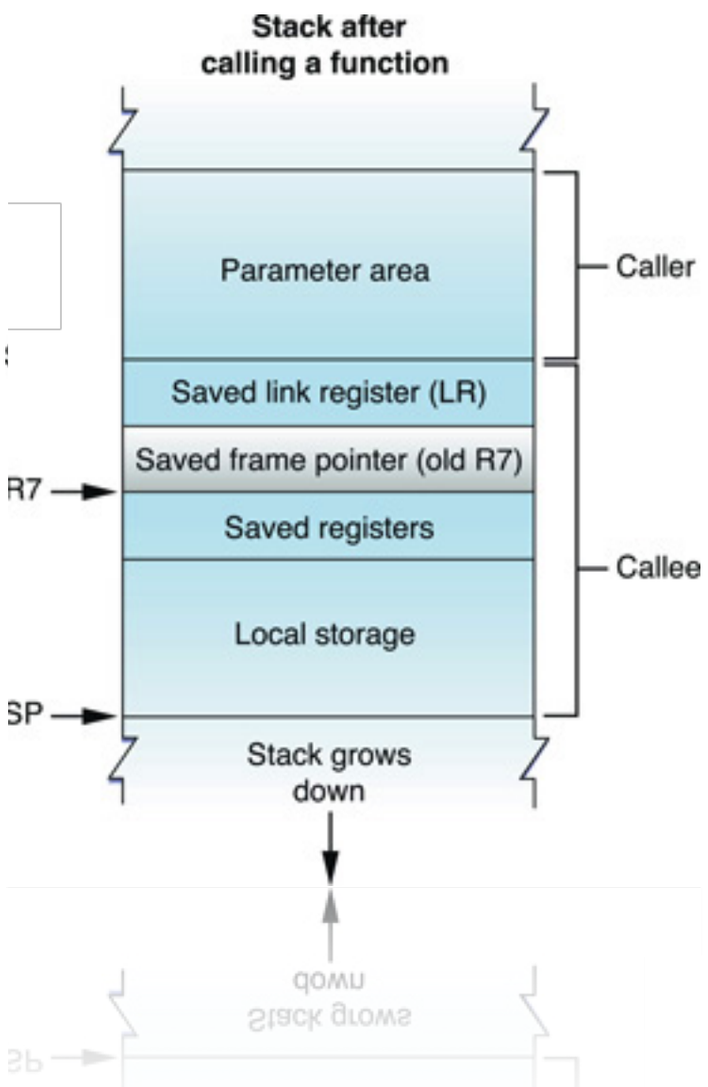
```
- (void)callWrapper{
   [self objCFunction:1 arg2:2 arg3:3] ;}
```




```
-[AppDelegate callWrapper]:

// function prologue
push {r7, lr}         // save old base pointer, return addr
mov  r7, sp           // frame pointer = stack pointer
sub  sp, #0x4         // allocate additional space on stack

// body
movw r1, #0x3402      // higher r1 = 0x3402
movs r2, #0x3         // r2 = 3
movt r1, #0x0         // lower r1 = 0x0
str  r2, [sp, #0x4]   // store 3rd argument on stack
add  r1, pc           // r1 = addr of
selector(objCFunction:arg2:arg3:)
movs r2, #0x1         // r2 = 1
movs r3, #0x2         // r3 = 2
ldr  r1, [r1]         // r1 = selector(objCFunction:arg2:arg3:)
// [self(r0) objCFunction:1(r2) arg2:2(r3) arg3:3(sp+4)]
blx  imp___symbolstub1__objc_msgSend

// function epilogue
add  sp, #0x4         // deallocate stack space
pop  {r7, pc}         // restore old frame pointer & retaddr
```

ARM

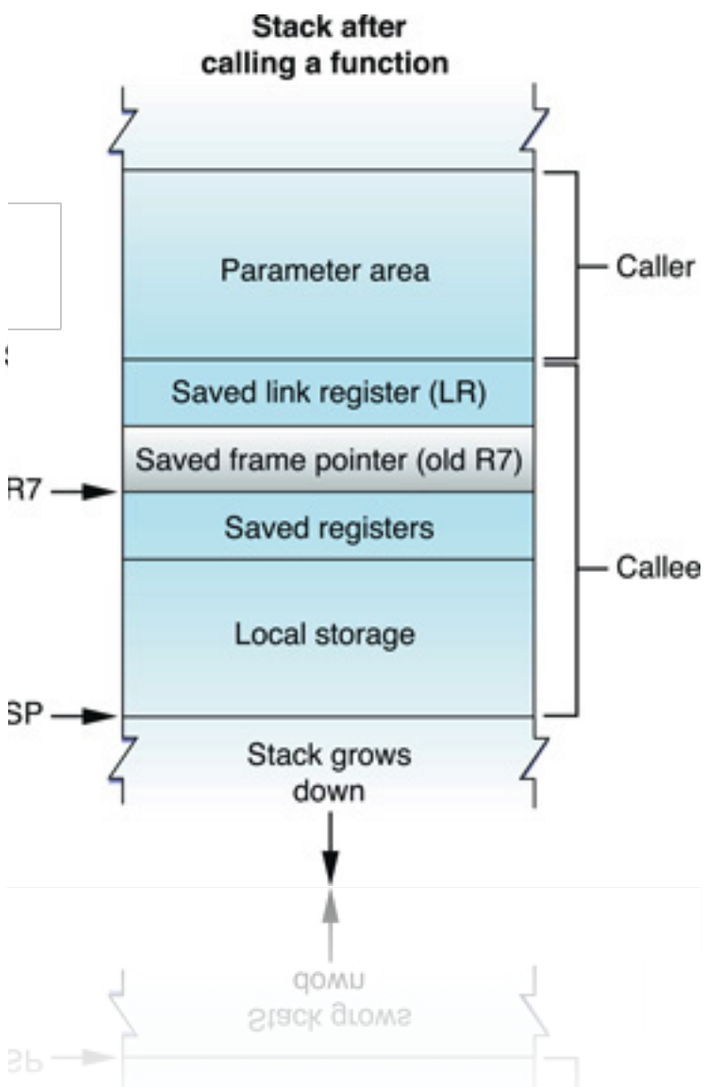
```
- (void)callWrapper{
   [self objCFunction:1 arg2:2 arg3:3] ;}
```




```
-[AppDelegate callWrapper]:

// function prologue
push {r7, lr}          // save old base pointer, return addr
mov  r7, sp            // frame pointer = stack pointer
sub  sp, #0x4          // allocate additional space on stack

// body
movw r1, #0x3402       // higher r1 = 0x3402
movs r2, #0x3          // r2 = 3
movt r1, #0x0          // lower r1 = 0x0
str  r2, [sp, #0x4]    // store 3rd argument on stack
add  r1, pc            // r1 = addr of
selector(objCFunction:arg2:arg3:)
movs r2, #0x1          // r2 = 1
movs r3, #0x2          // r3 = 2
ldr  r1, [r1]          // r1 = selector(objCFunction:arg2:arg3:)
// [self(r0) objCFunction:1(r2) arg2:2(r3) arg3:3(sp+4)]
blx  imp___symbolstub1__objc_msgSend

// function epilogue
add  sp, #0x4          // deallocate stack space
pop  {r7, pc}          // restore old frame pointer & retaddr
```

ARM

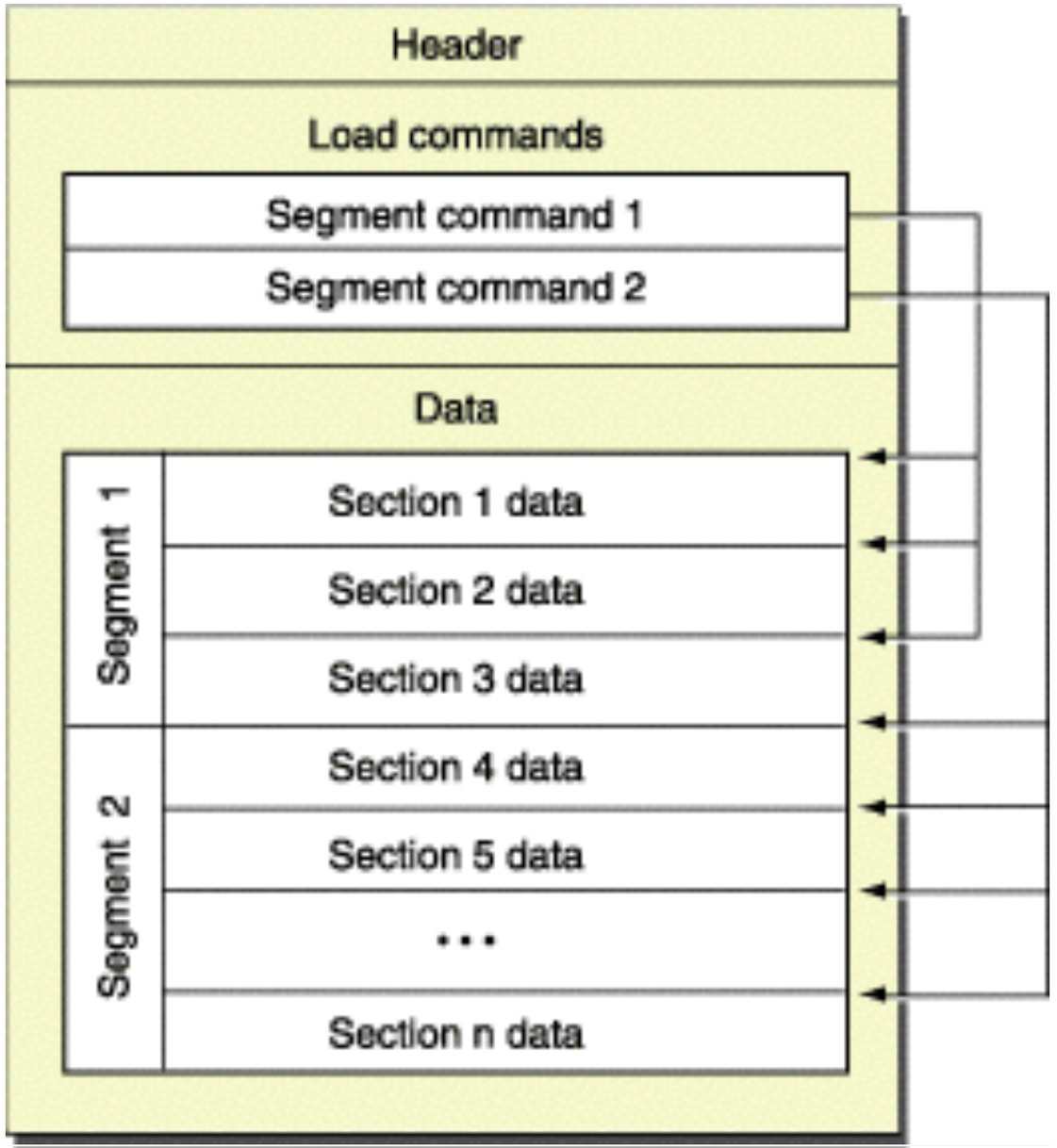
```
- (void)callWrapper{
   [self objCFunction:1 arg2:2 arg3:3] ;}
```




```
-[AppDelegate callWrapper]:

// function prologue
push {r7, lr}          // save old base pointer, return addr
mov  r7, sp            // frame pointer = stack pointer
sub  sp, #0x4          // allocate additional space on stack

// body
movw r1, #0x3402       // higher r1 = 0x3402
movs r2, #0x3          // r2 = 3
movt r1, #0x0          // lower r1 = 0x0
str  r2, [sp, #0x4]    // store 3rd argument on stack
add  r1, pc            // r1 = addr of
selector(objCFunction:arg2:arg3:)
movs r2, #0x1          // r2 = 1
movs r3, #0x2          // r3 = 2
ldr  r1, [r1]          // r1 = selector(objCFunction:arg2:arg3:)
// [self(r0) objCFunction:1(r2) arg2:2(r3) arg3:3(sp+4)]
blx  imp___symbolstub1__objc_msgSend

// function epilogue
add  sp, #0x4          // deallocate stack space
pop  {r7, pc}          // restore old frame pointer & retaddr
```

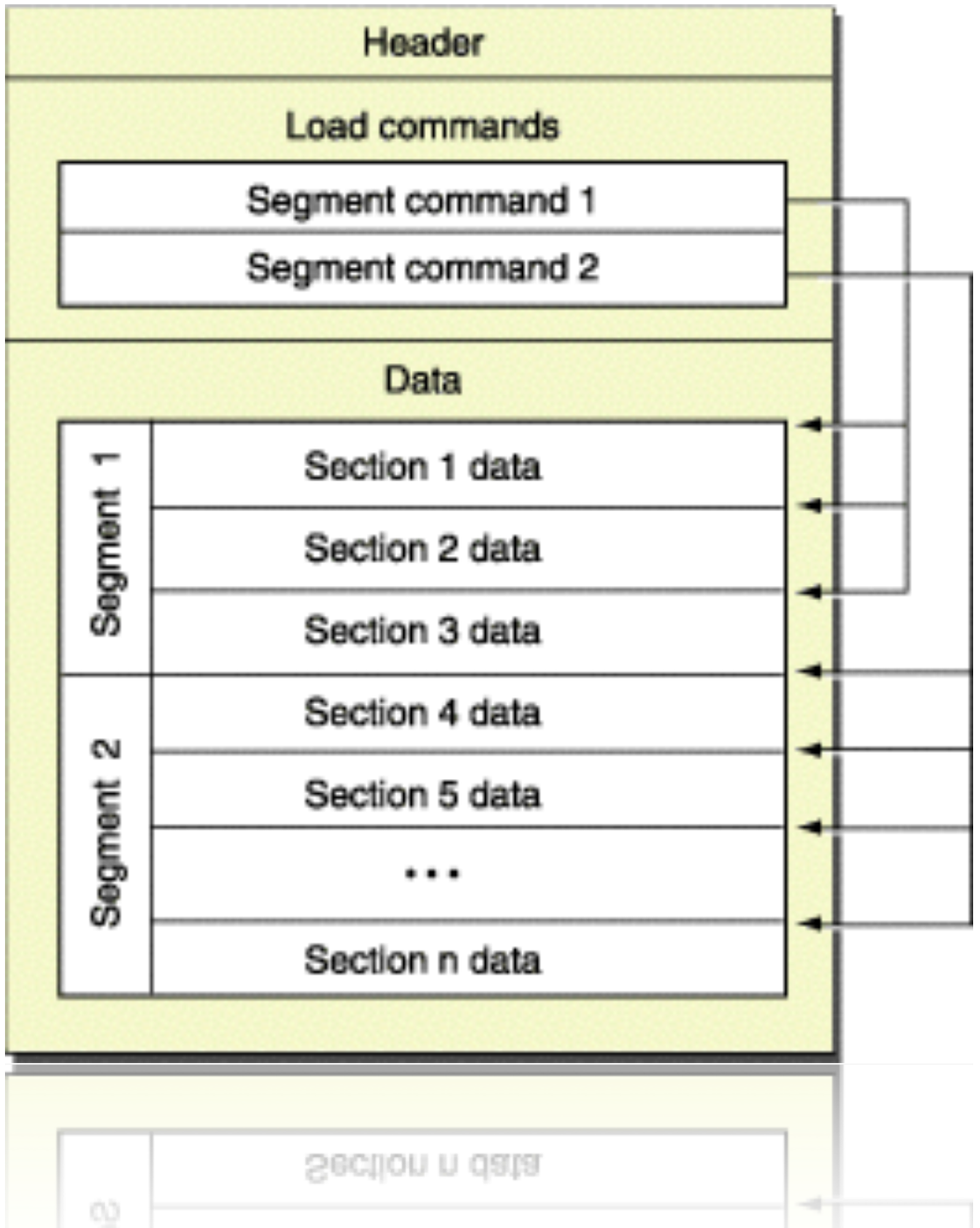
Mach-O

- FAT binaries
- Header
- Load commands
- Segments & Sections



Mach-O

- **__TEXT**
  - __text: executable code
  - __cstring, __const: ro data
- **__DATA**:
  - __data: writable data
  - __nl_symbol_ptr
- **__OBJC**:
  - __objc_classlist, __objc_classref, …



__OBJC Segment

$ class-dump Facebook.decrypted

__OBJC Segment

$ class-dump Facebook.decrypted

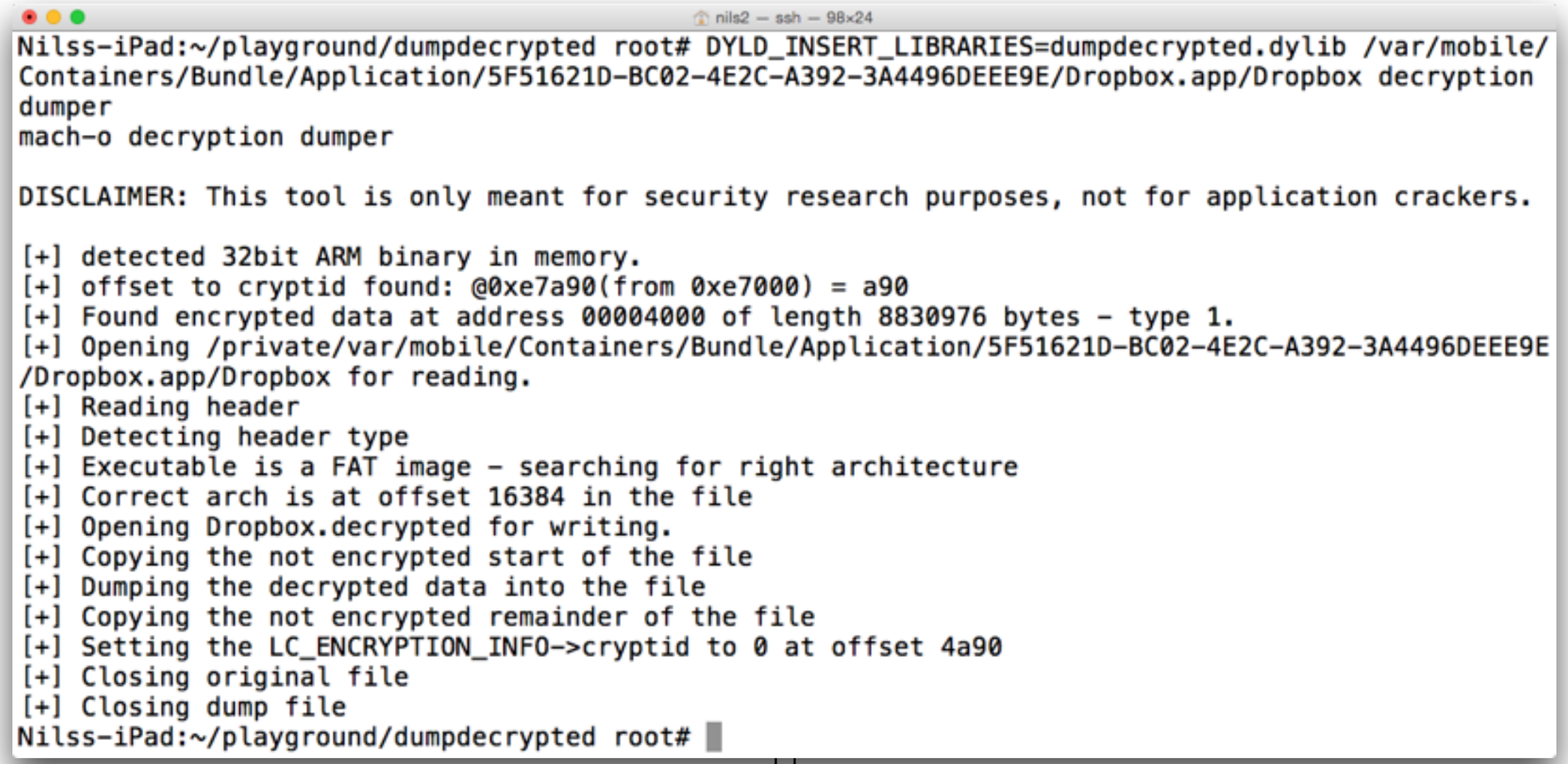
class-dump_facebook_part.m — UNREGISTERED

```
@interface FBCredentialStoreUpdateManager : NSObject <FBCredentialStoreUpdateDelegate>
{
    BOOL _shouldStartNewUpdate;
    BOOL _accountStoreChangedWhileInactive;
    FBUserSession *_session;
    ACAccountType *_fbAccountType;
    ACAccount *_fbAccount;
    FBCredentialStoreUpdate *_currentUpdate;
}

@property(nonatomic) BOOL accountStoreChangedWhileInactive; // @synthesize accountStoreChangedWhileIn
@property(nonatomic) BO**OL shouldStartNewUpdate; // @synthesize shouldStartNewUpdate=_shouldStartNew
@property(retain, nonatomic) FBCredentialStoreUpdate *currentUpdate; // @synthesize currentUpdate=_cu
@property(retain, nonatomic) ACAccount *fbAccount; // @synthesize fbAccount=_fbAccount;
@property(readonly, nonatomic) ACAccountType *fbAccountType; // @synthesize fbAccountType=_fbAccountT
@property(retain, nonatomic) FBUserSession *session; // @synthesize session=_session;
- (void).cxx_destruct;
- (void)_scheduleNextUpdateForResult:(unsigned int)arg1;
- (void)credentialStoreUpdateDidFinish:(id)arg1 withResult:(unsigned int)arg2;
- (void)_appDidBecomeActive:(id)arg1;
- (void)_accountStoreChanged:(id)arg1;
- (void)_beginNewUpdateWithCurrentAccount;
- (void)updateSystemAccount;
- (BOOL)_isScheduledTimeElapsed;
- (void)_setRescheduleInterval:(double)arg1;
- (void)dealloc;
- (id)initWithSession:(id)arg1;
- (id)init;

// Remaining properties
@property(readonly, copy) NSString *debugDescription;
@property(readonly, copy) NSString *description;
@property(readonly) unsigned int hash;
@property(readonly) Class superclass;

@end

@interface FBUsernameRequest : FBSessionNetworkerRequest
```

Credential — Find | Find Prev | Find All

FairPlay

- Applications are encrypted
- Jailbroken device + d*umpdecrypted*



Audit

- Common pitfalls in iOS applications
  - Classic C Attacks / Memory corruption
  - Data storage: Data Protection API, NSUserDefaults, Pasteboard, SQL
  - Non persistent data: Keyboard cache, Logging
  - Transport Security
  - IPC
  - UIWebView/XSS

Format Strings

- Format datatypes according to format string
- Problem:
  - Missing format string
  - Input from untrusted source
  - Attacker can control format function
  - Dump memory, DoS, Exploit

```c
// input from untrusted source
char input[200];

...

// Wrong way:
printf(input);

// Correct way:
printf("%s", input);
```

Exploit: "123\xde\xf3\xff\xbf....\xdc\xf3\xff\xbf%x.%x.%x.%x.%x.%49096d.%hn%12925d %hn\x90\x90\x90\x90" + payload

Format Strings

- Countermeasures:
  - Prefer Objective-C over C
  - Xcode static code analyzer
- Detection:
  - Check format functions
  - “%” in argument at format string position ?
  - # function arguments correct ?
- Format functions:
  - C: [f|sp|sn|as|d|v|vf|vs|vsn|va|vd]printf, syslog
  - Objective-C:
    - NSLog()
    - [NSString stringWithFormat:]
    - [NSString initWithFormat:]
    - [NSMutableString appendFormat:]
    - [NSAlert informativeTextWithFormat:]
    - [NSPredicate predicateWithFormat:]
    - [NSException format:]
    - NSRunAlertPanel

Buffer Overflow

- Write beyond buffer
- No proper bounds checking
- Check for “unsafe” functions:
  - strcat, strncat -> strlcat
  - strcpy, strncpy -> strlcpy
  - sprintf -> asprintf
  - vsprintf -> vsnprintf, vaprintf
  - gets -> fgets

```
char buf[BUFSIZE];
gets(buf);
```

```
char buf[12];
strcpy(buf, user_input);
```

Data Storage

- NSUserDefaults
- Data Protection API
- Keychain
- Pasteboard
- SQL

NSUserDefaults

- Defaults system
- Convenient way of storing data/preferences
- Stored in preference file: */private/var/mobile/Containers/Bundle/Application/<UUID>/Library/Preferences/<BundleIdentifier>.plist*
- Property list (*.plist*)
  - Apple's Binary XML
  - $ *plutil -convert xml1 de.nachtmaar.SecureObjectiveCIOs.plist*
- Don't store credentials or sensitive data here!
  - No encryption (*NSFileProtectionNone*)
  - Accesible via: backups, MDM,

NSUserDefaults

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://
www.apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4 <dict>
5  <key>passwordKey</key>
6  <string>password</string>
7 </dict>
8 </plist>
```

```
[[NSUserDefaults standardUserDefaults] setObject:@"password" forKey:@"passwordKey"];
[[NSUserDefaults standardUserDefaults] synchronize];
```

```
[[NSUserDefaults standardUserDefaults] objectForKey:@"passwordKey"];
```

NSUserDefaults

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://
www.apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4 <dict>
5  <key>passwordKey</key>
6  <string>password</string>
7 </dict>
8 </plist>
```

```
[[NSUserDefaults standardUserDefaults] setObject:@"password" forKey:@"passwordKey"];
[[NSUserDefaults standardUserDefaults] synchronize];
```

```
[[NSUserDefaults standardUserDefaults] objectForKey:@"passwordKey"];
```

UIPasteboard

- Implements copy & paste functionality
- Data exchange between apps
- General pasteboard vs. application pasteboards
- Persistent vs. non persistent
- Often used as migration from free to paid version
- **Deactivate copy & paste for sensitive text fields !**

```
  UIPasteboard *sharedPasteboard = [UIPasteboard pasteboardWithName:@"SharedPasteboard"
create:YES];
  sharedPasteboard.persistent = YES;
  sharedPasteboard.string = @"Hello, world";
```


Source: Manuel Binna - iOS Application Security

UIPasteboard

- Implements copy & paste functionality
- Data exchange between apps
- General pasteboard vs. application pasteboards
- Persistent vs. non persistent
- Often used as migration from free to paid version
- **Deactivate copy & paste for sensitive text fields !**

Source: Manuel Binna - iOS Application Security

UIPasteboard

- Implements copy & paste functionality
- Data exchange between apps
- General pasteboard vs. application pasteboards
- Persistent vs. non persistent
- Often used as migration from free to paid version
- **Deactivate copy & paste for sensitive text fields !**

```
  UIPasteboard *sharedPasteboard = [UIPasteboard pasteboardWithName:@"SharedPasteboard"
create:NO];
  NSString *receivedString = sharedPasteboard.string;
  NSLog(@"Received the string '%@' from the other application.",
        receivedString);
```

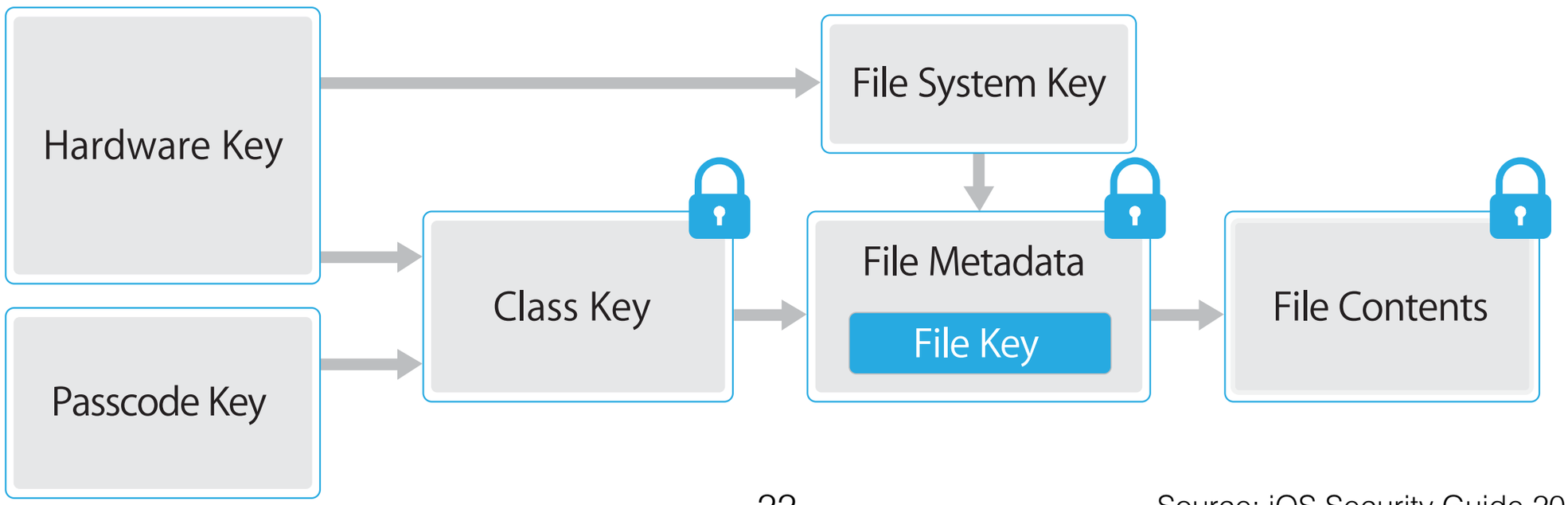

Source: Manuel Binna - iOS Application Security

Data Protection

- Dedicated AES crypto engine, unique device ID 256 Bit AES key compiled into processor
- Fast wipe
- Data Protection: additional layer of encryption, levels:
  - Complete protection
  - Protected unless open
  - Until first user authentication
  - No protection

Data Protection

- AES 256 Bit per-file key
- Each level has a distinct class key
    - Protected with passcode/touchID for some classes
    - wraps the per-file key
        - stored in file's metadata
- Decryption: decrypt file's metadata, unwrap per-file key



Source: iOS Security Guide 2014, Apple. Inc

Data Protection

- API usage through classes:
  - NS(Mutable)Data
    - Selector “*writeToFile:options:error*”
  - NSFilemanager

```
typedef NS_OPTIONS(NSUInteger, NSDataWritingOptions) {
    NSDataWritingFileProtectionNone                          = 0x10000000,
    NSDataWritingFileProtectionComplete                      = 0x20000000,
    NSDataWritingFileProtectionCompleteUnlessOpen            = 0x30000000,
    NSDataWritingFileProtectionCompleteUntilFirstUserAuthentication = 0x40000000
};
```

Data Protection

- API usage through classes:
  - NS(Mutable)Data
  - NSFilemanager
    - Selector “*createFileAtPath:contents:attributes*“
    - attributes: dictionary with keys and values

```
FOUNDATION_EXPORT NSString * const NSFileProtectionKey NS_AVAILABLE_IOS(4_0);
FOUNDATION_EXPORT NSString * const NSFileProtectionNone NS_AVAILABLE_IOS(4_0);
FOUNDATION_EXPORT NSString * const NSFileProtectionComplete NS_AVAILABLE_IOS(4_0);
FOUNDATION_EXPORT NSString * const NSFileProtectionCompleteUnlessOpen
NS_AVAILABLE_IOS(5_0);
FOUNDATION_EXPORT NSString * const
NSFileProtectionCompleteUntilFirstUserAuthentication NS_AVAILABLE_IOS(5_0);
```

Steal Data

- Why should I encrypt data ?????
- Retrieve files via Apple File Communication protocol (AFC)
- Malicious dock station
    - Retrieve application ressources
        - Preference files -> credentials ?
- Initiate backup
    - Logs, sqlite databases, files
- No user authorization except unlocking with passcode/ touchID
- Autosync option set on computer side

Source: Mathieu Renard - Practical iOS Apps hacking

SQL Injection

- No input validation
- Parameterized Statements
- Detection: strings command

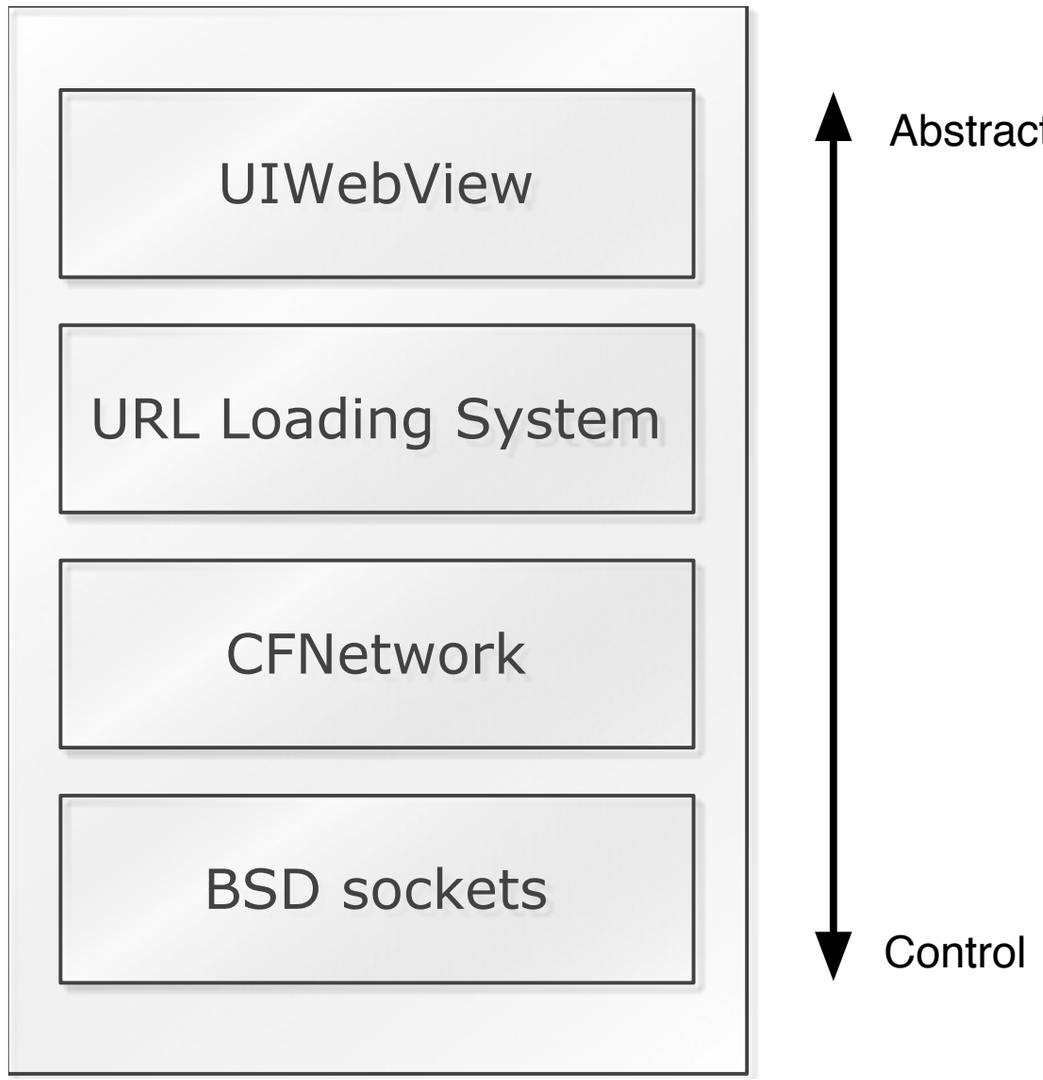
```
1 sqlite3 *db;
2 char sqlbuf[256], *err;
3 sqlite3_open("sample.db", &db);
4 snprintf(sqlbuf, sizeof(sqlbuf), "SELECT * FROM table
WHERE user = %s", attackerControlled);
5 sqlite3_exec(db, sqlbuf, NULL, NULL, &err);
```

Non Persistent Data

- Keyboard cache
  - Used for autocorrection feature
  - Turn it off for sensitive fields!
- App transitions
  - Zoom out and in effect needs app screenshot
  - Taken if app changes from foreground -> background
- Logging
  - Can be viewed with Xcode
  - Conditional compilation: debug vs. release mode

Transport Security

- several network APIs
- Abstraction vs. control
- URL Loading System:
  - Supports standard protocols like HTTP(s), FTP(s), …
- CFNetwork: low-level framework for network communication



URL Loading System

- By default proper SSL/TLS handling
- But people turn it off!
- Debugging Code

URL Loading System

NSURLConnectionDelegate prior iOS 8:

```
- (BOOL)connection:(NSURLConnection *)connection
canAuthenticateAgainstProtectionSpace:(NSURLProtectionSpace *)protectionSpace {
 return [protectionSpace.authenticationMethod
isEqualToString:NSURLAuthenticationMethodServerTrust];
}
```

```
- (void)connection:(NSURLConnection *)connection
didReceiveAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge {
    [challenge.sender useCredential:[NSURLCredential
credentialForTrust:challenge.protectionSpace.serverTrust]
forAuthenticationChallenge:challenge];
}
```

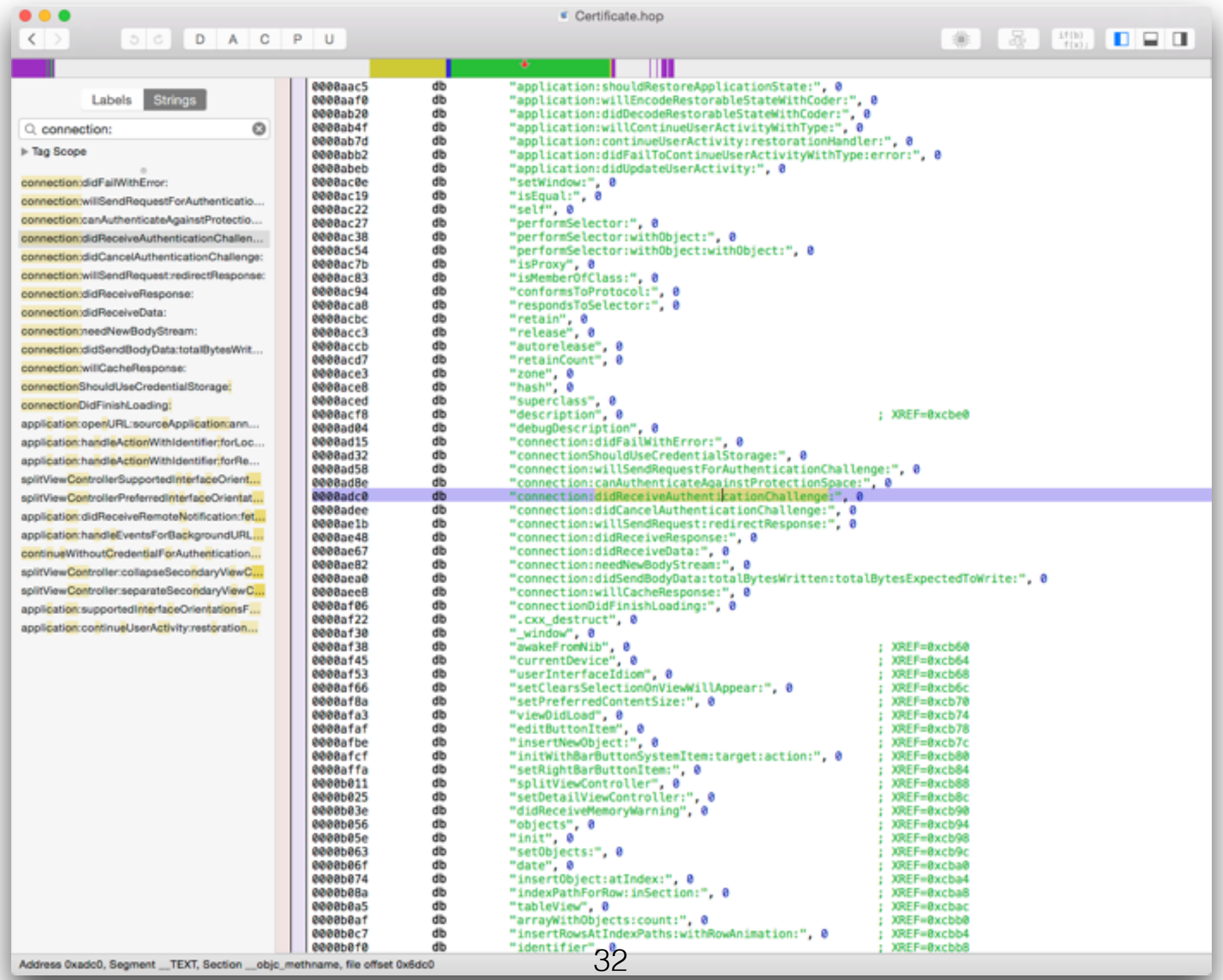
URL Loading System

Since iOS 8:

```
- (void)connection:(NSURLConnection *)connection didReceiveAuthenticationChallenge:
(NSURLAuthenticationChallenge *)challenge{
  [challenge.sender useCredential:
   [NSURLCredential
credentialForTrust:challenge.protectionSpace.serverTrust]forAuthenticationChallenge:chall
enge];
}
```

- Detection:
    - **__TEXT, __nl_symbol_ptr**: Check for *NSURLAuthenticationMethodServerTrust*
    - **__OBJC, __objc_methname** :
        - "*connection:canAuthenticateAgainstProtectionSpace:*"
        - "*didReceiveAuthenticationChallenge:*"

URL Loading System



CFNetwork

- Finer grained control
- Based on BSD sockets
- Read and write bytes (a)synchronously to/from a stream
- Possibility to influence certificate validation, force specific SSL/TLS versions

Source: [Binna]

CFNetwork

```
NSURL *theURL = [NSURL URLWithString:@"https://www.example.com"];
CFHTTPMessageRef httpRequest =
CFHTTPMessageCreateRequest(kCFAllocatorDefault, CFSTR("GET"),
(CFURLRef)theURL, kCFHTTPVersion1_1);

// Create stream to retrieve HTTP response
CFReadStreamRef readStream =
CFReadStreamCreateForHTTPRequest(kCFAllocatorDefault, httpRequest);
```

```
// Configure SSL/TLS security settings
NSDictionary *tlsSettings =
  [NSDictionary dictionaryWithObjectsAndKeys:
    (id)kCFStreamSocketSecurityLevelNegotiatedSSL,
    (id)kCFStreamSSLLevel,
    (id)kCFBooleanFalse,
    (id)kCFStreamSSLValidatesCertificateChain,
    nil];
```

```
CFReadStreamSetProperty(readStream,
                        kCFStreamPropertySSLSettings,
                        tlsSettings);

// read from stream
```


Source: [Binna]

CFNetwork

```
NSURL *theURL = [NSURL URLWithString:@"https://www.example.com"];
CFHTTPMessageRef httpRequest =
CFHTTPMessageCreateRequest(kCFAllocatorDefault, CFSTR("GET"),
(CFURLRef)theURL, kCFHTTPVersion1_1);

// Create stream to retrieve HTTP response
CFReadStreamRef readStream =
CFReadStreamCreateForHTTPRequest(kCFAllocatorDefault, httpRequest);

// Configure SSL/TLS security settings
NSDictionary *tlsSettings =
  [NSDictionary dictionaryWithObjectsAndKeys:
    (id)kCFStreamSocketSecurityLevelNegotiatedSSL,
    (id)kCFStreamSSLLevel,
    (id)kCFBooleanFalse,
    (id)kCFStreamSSLValidatesCertificateChain,
    nil];

CFReadStreamSetProperty(readStream,
                        kCFStreamPropertySSLSettings,
                        tlsSettings);

// read from stream
```

```
const CFStringRef
kCFStreamPropertySSLSettings;
const CFStringRef
kCFStreamSSLValidatesCertificateChain,
kCFStreamSSLPeerName,
kCFStreamSSLCertificates,
kCFStreamSSLIsServer;

const CFStringRef kCFStreamSSLLevel;
const CFStringRef
kCFStreamSocketSecurityLevelNone,
kCFStreamSocketSecurityLevelSSLv2,
kCFStreamSocketSecurityLevelSSLv3,
kCFStreamSocketSecurityLevelTLSv1,
kCFStreamSocketSecurityLevelNegotiatedSSL;
```

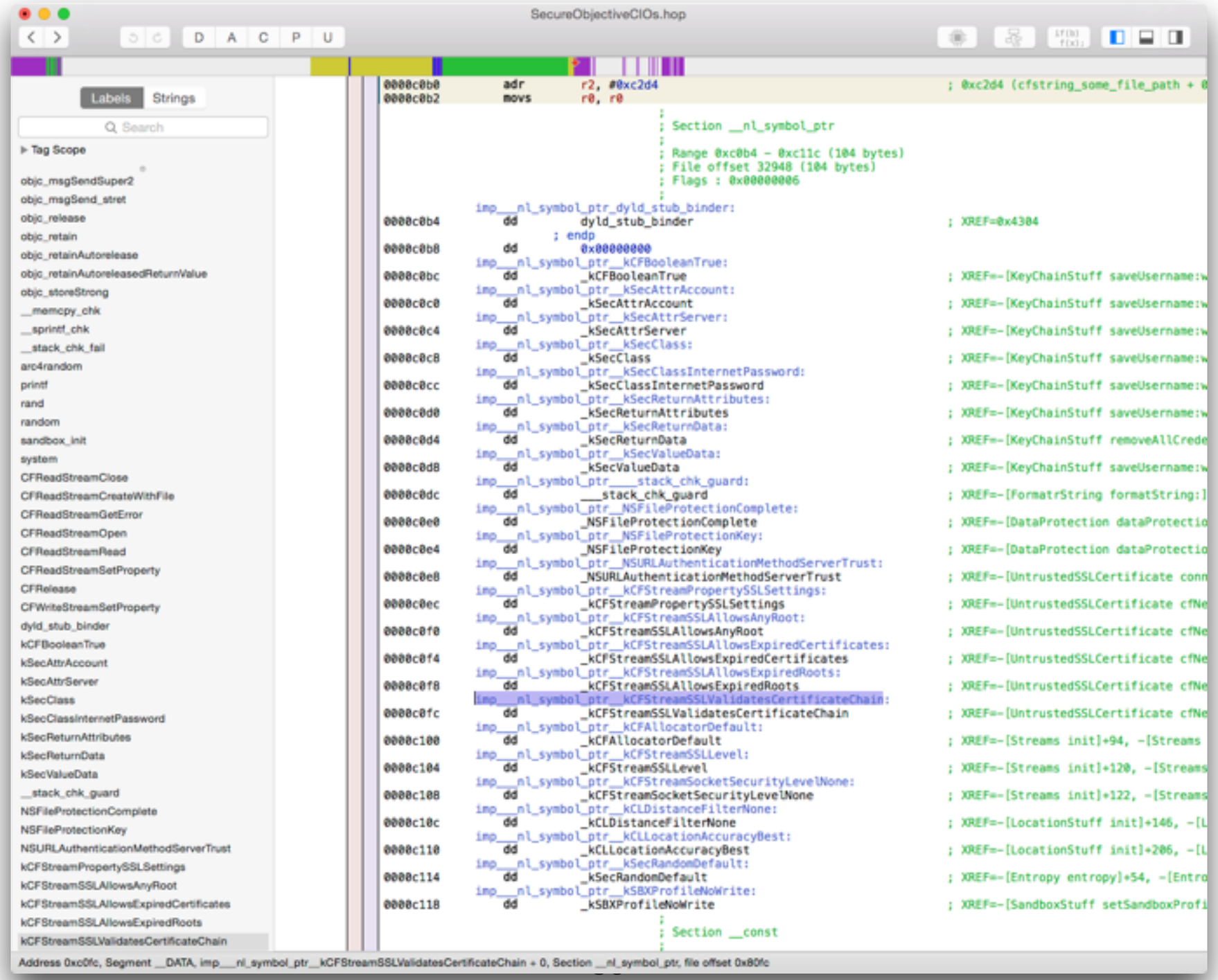

Source: [Binna]

CFNetwork

- Toll-free-bridging between *CFStreams* and ***NSStreams*** (Objective-C API)
- same constants, different name
  - prefix: *NSStreamSocketSecurityLevel*
- Check selector "*setProperty:forKey:*"

Source: [Binna]

CFNetwork


SecureObjectiveCIOs.hop
D A C P U
Labels
Strings
Search
Tag Scope
objc_msgSendSuper2
objc_msgSend_stret
objc_release
objc_retain
objc_retainAutorelease
objc_retainAutoreleasedReturnValue
objc_storeStrong
__memcpy_chk
__sprintf_chk
__stack_chk_fail
arc4random
printf
rand
random
sandbox_init
system
CFReadStreamClose
CFReadStreamCreateWithFile
CFReadStreamGetError
CFReadStreamOpen
CFReadStreamRead
CFReadStreamSetProperty
CFRelease
CFWriteStreamSetProperty
dyld_stub_binder
kCFBooleanTrue
kSecAttrAccount
kSecAttrServer
kSecClass
kSecClassInternetPassword
kSecReturnAttributes
kSecReturnData
kSecValueData
__stack_chk_guard
NSFileProtectionComplete
NSFileProtectionKey
NSURLAuthenticationMethodServerTrust
kCFStreamPropertySSLSettings
kCFStreamSSLAllowsAnyRoot
kCFStreamSSLAllowsExpiredCertificates
kCFStreamSSLAllowsExpiredRoots
kCFStreamSSLValidatesCertificateChain
0000c0b0 adr r2, #0xc2d4 ; 0xc2d4 (cfstring_some_file_path + 0
0000c0b2 movs r0, r0
; Section __nl_symbol_ptr
; Range 0xc0b4 - 0xc11c (104 bytes)
; File offset 32948 (104 bytes)
; Flags : 0x00000006
imp___nl_symbol_ptr_dyld_stub_binder:
0000c0b4 dd dyld_stub_binder ; XREF=0x4304
; endp
0000c0b8 dd 0x00000000
imp___nl_symbol_ptr__kCFBooleanTrue:
0000c0bc dd _kCFBooleanTrue ; XREF=-[KeyChainStuff saveUsername:w
imp___nl_symbol_ptr__kSecAttrAccount:
0000c0c0 dd _kSecAttrAccount ; XREF=-[KeyChainStuff saveUsername:w
imp___nl_symbol_ptr__kSecAttrServer:
0000c0c4 dd _kSecAttrServer ; XREF=-[KeyChainStuff saveUsername:w
imp___nl_symbol_ptr__kSecClass:
0000c0c8 dd _kSecClass ; XREF=-[KeyChainStuff saveUsername:w
imp___nl_symbol_ptr__kSecClassInternetPassword:
0000c0cc dd _kSecClassInternetPassword ; XREF=-[KeyChainStuff saveUsername:w
imp___nl_symbol_ptr__kSecReturnAttributes:
0000c0d0 dd _kSecReturnAttributes ; XREF=-[KeyChainStuff saveUsername:w
imp___nl_symbol_ptr__kSecReturnData:
0000c0d4 dd _kSecReturnData ; XREF=-[KeyChainStuff removeAllCrede
imp___nl_symbol_ptr__kSecValueData:
0000c0d8 dd _kSecValueData ; XREF=-[KeyChainStuff saveUsername:w
imp___nl_symbol_ptr____stack_chk_guard:
0000c0dc dd ___stack_chk_guard ; XREF=-[FormatrString formatString:]
imp___nl_symbol_ptr__NSFileProtectionComplete:
0000c0e0 dd _NSFileProtectionComplete ; XREF=-[DataProtection dataProtectio
imp___nl_symbol_ptr__NSFileProtectionKey:
0000c0e4 dd _NSFileProtectionKey ; XREF=-[DataProtection dataProtectio
imp___nl_symbol_ptr__NSURLAuthenticationMethodServerTrust:
0000c0e8 dd _NSURLAuthenticationMethodServerTrust ; XREF=-[UntrustedSSLCertificate conn
imp___nl_symbol_ptr__kCFStreamPropertySSLSettings:
0000c0ec dd _kCFStreamPropertySSLSettings ; XREF=-[UntrustedSSLCertificate cfNe
imp___nl_symbol_ptr__kCFStreamSSLAllowsAnyRoot:
0000c0f0 dd _kCFStreamSSLAllowsAnyRoot ; XREF=-[UntrustedSSLCertificate cfNe
imp___nl_symbol_ptr__kCFStreamSSLAllowsExpiredCertificates:
0000c0f4 dd _kCFStreamSSLAllowsExpiredCertificates ; XREF=-[UntrustedSSLCertificate cfNe
imp___nl_symbol_ptr__kCFStreamSSLAllowsExpiredRoots:
0000c0f8 dd _kCFStreamSSLAllowsExpiredRoots ; XREF=-[UntrustedSSLCertificate cfNe
imp___nl_symbol_ptr__kCFStreamSSLValidatesCertificateChain:
0000c0fc dd _kCFStreamSSLValidatesCertificateChain ; XREF=-[UntrustedSSLCertificate cfNe
imp___nl_symbol_ptr__kCFAllocatorDefault:
0000c100 dd _kCFAllocatorDefault ; XREF=-[Streams init]+94, -[Streams
imp___nl_symbol_ptr__kCFStreamSSLLevel:
0000c104 dd _kCFStreamSSLLevel ; XREF=-[Streams init]+120, -[Streams
imp___nl_symbol_ptr__kCFStreamSocketSecurityLevelNone:
0000c108 dd _kCFStreamSocketSecurityLevelNone ; XREF=-[Streams init]+122, -[Streams
imp___nl_symbol_ptr__kCLDistanceFilterNone:
0000c10c dd _kCLDistanceFilterNone ; XREF=-[LocationStuff init]+146, -[L
imp___nl_symbol_ptr__kCLLocationAccuracyBest:
0000c110 dd _kCLLocationAccuracyBest ; XREF=-[LocationStuff init]+206, -[L
imp___nl_symbol_ptr__kSecRandomDefault:
0000c114 dd _kSecRandomDefault ; XREF=-[Entropy entropy]+54, -[Entro
imp___nl_symbol_ptr__kSBXProfileNoWrite:
0000c118 dd _kSBXProfileNoWrite ; XREF=-[SandboxStuff setSandboxProfi
; Section __const
Address 0xc0fc, Segment __DATA, imp___nl_symbol_ptr__kCFStreamSSLValidatesCertificateChain + 0, Section __nl_symbol_ptr, file offset 0x80fc

Transport Security

- Dangers?
- MitM-Attacks
  - Intercepting traffic -> Steal sensitive information
  - Inject/modify code/data
- Protocol issues:
  - SSL 3.0: Poodle
  - TLS 1.0: Beast
  - Steal HTTPS cookies

Source: [Binna]

IPC

- Simple IPC mechanism
- Allows data exchange, launch other applications
- Register scheme
- Info.plist: *CFBundleURLTypes -> CFBundleURLSchemes*
- e.g.: Facebook, “fb“ => “fb://action?parameter1=value1&parameter2=value2”
- Encode transactions in URL
- As always: thorough input validation
- Ask for authorization first!

IPC

```
- (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL
*)url{
  // trigger action
  return YES;
}
```

Since iOS 4.2:

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
sourceApplication:(NSString *)sourceApplication annotation:
(id)annotation{
  // trigger action
  return YES;
}
```

IPC

IPC handling

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
sourceApplication:(NSString *)sourceApplication annotation:
(id)annotation{
  // trigger action
  return YES;
}
```

IPC call

```
  [[UIApplication sharedApplication]
   openURL:
    [NSURL URLWithString:@"other-app://action?parameter1=value1&parameter2=value2"]
   ];
```

IPC

IPC handling

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
sourceApplication:(NSString *)sourceApplication annotation:
(id)annotation{
  // trigger action
  return YES;
}
```

IPC call

```
  [[UIApplication sharedApplication]
   openURL:
    [NSURL URLWithString:@"other-app://action?parameter1=value1&parameter2=value2"]
   ];
```

```
<iframe src="fb://profile/"></iframe>
```

IPC

IPC handling

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
sourceApplication:(NSString *)sourceApplication annotation:
(id)annotation{
  // trigger action
  return YES;
}
```

com.mobile.safari

IPC call

```
  [[UIApplication sharedApplication]
   openURL:
    [NSURL URLWithString:@"other-app://action?parameter1=value1&parameter2=value2"]
   ];
```

```
<iframe src="fb://profile/"></iframe>
```

IPC

decrypted — bash — 105×31

```
nachtmaars-MacBook-Pro:decrypted nils$ strings Facebook.decrypted |  grep "^fb://"
fb://profile
fb://profile?id=%@
fb://profile?%@=%@
fb://profile?id=%@&%@=%@
fb://profile?id=%@&%@=%@&%@=%@
fb://album?id=%@
fb://group?id=%@
fb://photo?%@
fb://group?id=%@&object_id=%@&view=permalink
fb://groupPhotos?id=%@
fb://%@?%@
fb://story?%@
fb://page_about?id=%@
fb://page_reviews?id=%@
fb://page_friend_likes_and_visits?id=%@&should_show_visits_first=%d
fb://page_post_insights?page_id=%@&story_id=%@
fb://page?id=%@
fb://page?id=%@&source=notification&notif_type=%@
fb://page?id=%@&source=%@&source_id=%@
fb://page?id=%@&showreactionoverlay=%d
fb://event?id=%@
fb://event?id=%@&post_id=%@
fb://eventguestlist?event_id=%@
fb://topic/%@
fb://hashtag/%@
fb://codegenerator
fb://messaginglist
fb://messaging?tid=%@
fb://messaging?id=%@&%@
fb://messaging/new?id=%@&name=%@&isPage=%d
```

IPC

- Malicious website containing:

<iframe src=“skype://1408555555?call/“**></iframe>**

- Credentials cached in app?

Source: Nitesh Dhanjani - NEW AGE APPLICATION ATTACKS AGAINST APPLE’s iOS

IPC

- Malicious website containing:

<iframe src=“skype://1408555555?call/“**></iframe>**

- Credentials cached in app?

Source: Nitesh Dhanjani - NEW AGE APPLICATION ATTACKS AGAINST APPLE’s iOS

UIWebView

- In-app browser
- Renders web pages, PDFs, images, …
- Executes JavaScript -> Danger of XSS
- Often used as GUI
  - Additional JavaScript <-> Objective-C bridge
  - XSS attacks can be much more severe

UIWebView

```
1  let myvar = "<script>alert('XSS!');</script>"
2  let js = "var myvar=\"" + myvar + "\";"
3  webView.stringByEvaluatingJavaScriptFromString(js)
4
5  webView.loadRequest(
6   NSURLRequest(
7     URL: NSURL(
8       fileURLWithPath: NSBundle.mainBundle().pathForResource("index", ofType:
"html")!
9     )!
10   )
11 )
```

```
1 <html>
2  <head></head>
3  <body>
4     <p>
5     Cross-Site Scripting in UIWebView: </p>
6     <p>
7     This is an example of XSS: <script>document.write(myvar);</script>
8     </p>
9  </body>
10 </html>
```

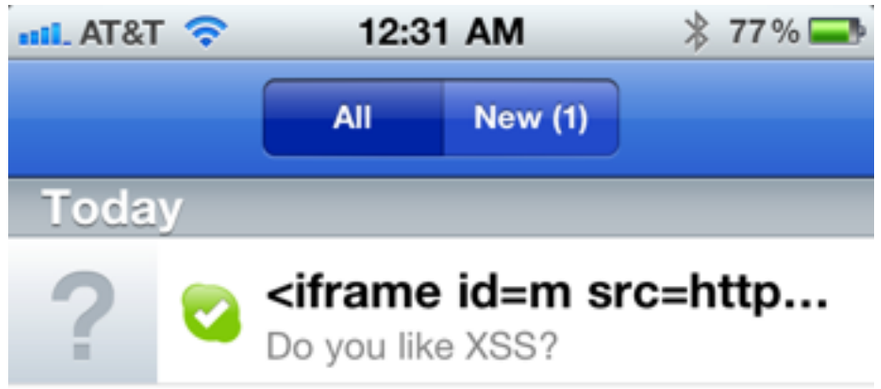
UIWebView

```
1  let myvar = "<script>alert('XSS!');</script>"
2  let js = "var myvar=\"" + myvar + "\";"
3  webView.stringByEvaluatingJavaScriptFromString(js)
4
5  webView.loadRequest(
6   NSURLRequest(
7     URL: NSURL(
8       fileURLWithPath: NSBundle.mainBundle().pathForResource("index", ofType:
"html")!
9     )!
10   )
11 )
```

```
1 <html>
2  <head></head>
3  <body>
4     <p>
5     Cross-Site Scripting in UIWebView: </p>
6     <p>
7     This is an example of XSS: <script>document.write(myvar);</script>
8     </p>
9  </body>
10 </html>
```

UIWebView

- Skype <= 3.01
- Rendered local HTML page
- Failed to validate full name
- Steal user's address book
  - Upload to external server



What’s left ?

- Encryption
  - hardcoded keys
  - ECB Mode
- Weak PRNG’s -> arc4random, rand, random
- Push notifications -> Certificate’s public available at server-side?

Conclusion

- Input validation, input validation, input validation !