Analysing iOS apps: road from AppStore to security analysis report

Egor Fominykh, Lenar Safin, Yaroslav Alexandrov
SmartDec

REcon, Brussels, 2017

What we do at SmartDec

- Decompilation, deobfuscation
  - x86/x64
  - ARM/AArch64
  - JVM, Android
  - Custom (VMs, less known archs, …)

- Code analysis (sources and binaries)
  - Manual static analysis
  - Pentesting
  - Analysis tools development

iTunes link

https://itunes.apple.com/us/app/balloonist-travellers-world/id1070769999?mt=8

Security report

```xml
<Bug>
    <RuleId>OBJC_NSLOG</RuleId>
    <RelativeSrc>smd-prod-m4-14.c</RelativeSrc>
    <LineFirst>1155</LineFirst>
    <LineLast>1155</LineLast>
</Bug>
<Bug>
    <RuleId>REFLECTION</RuleId>
    <RelativeSrc>smd-prod-m4-23.c</RelativeSrc>
    <LineFirst>274</LineFirst>
    <LineLast>274</LineLast>
</Bug>
<Bug>
    <RuleId>REFLECTION</RuleId>
    <RelativeSrc>smd-prod-m4-23.c</RelativeSrc>
    <LineFirst>279</LineFirst>
    <LineLast>279</LineLast>
</Bug>
```

Pseudocode

```
    objc_release(x8->MapSVGKOverlayRenderer::_layersByName)
} else {
    // bb_10005b018:
    x24 = "fillColor"
    x0 = x20
    x1 = x24
    [x8->MapSVGKOverlayRenderer::_layersByName fillColor]
    x23 = x0
    if (x22)==(1) {
        // bb_10005b034:
        x0 = x20
        x1 = x24
        [x8->MapSVGKOverlayRenderer::_layersByName fillColor]
        x2 = x0
    } else {
        // bb_10005b048:
        x2 = 0
    }
```

Plan

- Get an application binary
- Translate application binary into some IR
- Analyse IR for security flaws
- Translate IR into human-readable pseudocode

1:

Getting binary

A problem

Applications are encrypted. Decryption:

1. Launch an app on an iOS device.
2. iOS decrypts it and loads it to RAM.
3. Dump decrypted binary from RAM.

Jailbroken iOS device is needed.

Jailbreak

- SSH
- Bash
- Cydia Substrate (call/hook any method)
- Clutch

Approach

- Figure out chain of method calls / GUI decisions to initiate the download
- Figure out how to make needed GUI decisions programmatically, using Cydia Substrate

Main applications

- Springboard.app (GUI)
- AppStore.app

Process

1. Unlock device — SpringBoard
2. Uninstall all apps — SpringBoard
3. Open iTunes page — SpringBoard
4. Press GET button — AppStore
5. Sign in (detect sign in alert, fill login/password, press ok) — SpringBoard
6. Wait OPEN button — AppStore
7. Decrypt — Clutch

2:
Translation into IR

iOS application recovery challenges

- Lots of things to recover
  - Functions
  - Program CFG
  - Call site arguments and function signatures
  - Objective-C/Swift interfaces (even C++)
  - Data flow of the program

- AArch64
  - ARM32 is not supported anymore

Why LLVM?

- Nice and useful
- Bunch of algorithms
  - Alias Analysis
  - Dominators
  - Loops
  - Transformations and optimizations
- Pass Manager
- Ok for C-family apps

Ideas

- Fast automatic translation into LLVM
- Functions and function calls recovery
- CFG reconstruction
- Types and variables recovery
- Objective-C/Swift3 support

Architecture

Image parsing

- Unpacking Fat (Universal) binaries
- Mach-O
- Symbols
- Function starts
- Objective-C runtime (__objc_*)
- Swift virtual tables

CFG reconstruction

- Entry point
- Function starts
- Vtables
- Call sites
- __TEXT section inspection
- Tail calls and trampolines

Trampolines

```
imp___stubs__objc_release:
    adrp        x16, #0x100801000
    ldr         x16, [x16, #0x1d0]
    br          x16
  ; endp
```

```
                    _objc_release_ptr:
00000001008011d0    dq              _objc_release
```

Tail calls

```
-[TNDRSlidingPagedViewController handleGroupStatusBeganEditing:]:
    stp        x20, x19, [sp, #-0x20]!
    stp        x29, x30, [sp, #0x10]
    add        x29, sp, #0x10
    adrp       x8, #0x10099c000
    ldr        x1, [x8, #0x1e0]
    bl         imp___stubs__objc_msgSend
    mov        x29, x29
    bl         imp___stubs__objc_retainAutoreleasedReturnValue
    mov        x19, x0
    adrp       x8, #0x10099c000
    ldr        x1, [x8, #0x1e8]
    movz       w2, #0x0
    bl         imp___stubs__objc_msgSend
    mov        x0, x19
    ldp        x29, x30, [sp, #0x10]
    ldp        x20, x19, [sp]!, #0x20
    b          imp___stubs__objc_release
  ; endp
```

Interface recovery

- Objective-C interface
  - Classes
  - Protocols
  - Method names
  - Ivars
  - Demangling

- Swift interface
  - Vtables
  - Class hierarchy
  - Demangling

Objective-C runtime

```
@interface NSObject (iRate)
- (void)iRateDidOpenAppStore;   // IMP=0x00000001000199ac
- (_Bool)iRateShouldOpenAppStore;       // IMP=0x00000001000199a4
- (void)iRateUserDidRequestReminderToRateApp;   // IMP=0x00000001000199a0
- (void)iRateUserDidDeclineToRateApp;   // IMP=0x000000010001999c
- (void)iRateUserDidAttemptToRateApp;   // IMP=0x0000000100019998
- (void)iRateDidPromptForRating;        // IMP=0x0000000100019994
- (_Bool)iRateShouldPromptForRating;    // IMP=0x000000010001998c
- (void)iRateDidDetectAppUpdate;        // IMP=0x0000000100019988
- (void)iRateCouldNotConnectToAppStore:(id)arg1;        // IMP=0x0000000100019984
@end
```

Objective-C runtime

```
    { isa_addr: 0x00000001002bdf18, superclass_addr: 0x0000000000000000, cache_a
ddr: 0x0000000000000000, vtable_addr: 0x0000000000000000, data_addr: 0x000000010
0278d48, superclass_import: "OBJC_CLASS_$_NSObject", cache_import: "_objc_empty_
cache", data: { flags: 0x0, instanceStart: 0x8, instanceSize: 8, reserved: 0, iv
arLayout_addr: 0x0000000000000000, name_addr: 0x00000001001d4515, name: "GAIStri
ngUtil", baseMethods_addr: 0x0000000100278d28, baseProtocols_addr: 0x00000000000
00000, ivars_addr: 0x0000000000000000, weakIvarLayout_addr: 0x0000000000000000,
baseProperties_addr: 0x0000000000000000, methods: { entsize: 24, count: 1, data:
 [ { name_addr: 0x00000001001aa35e, name: "init", types_addr: 0x00000001001d5ec4
, types: "@16@0:8", imp_addr: 0x00000001000cedbc } ] } } }
```

Swift runtime

```
SWIFT CLASS TNDRSelectWelcomeViewController
  sub_00000001004687e8
  sub_000000010468b2c
  sub_0000000100469044
  sub_000000010046a1fc
  sub_000000010046a464
  sub_000000010046a4d4 (init)
SWIFT CLASS TNDRDialogRoundedBottomMaskView
  sub_000000010046b8ac
  sub_000000010046b9f0
  sub_000000010046ba88
  sub_000000010046bbb4 (init)
SWIFT CLASS TNDRPurchaseLogger
  sub_000000010047e540 (init)
```

Variables and types

- Memory object reconstruction
  - Temporary
  - Variables
  - Globals
  - Strings

- Types recovery
  - Interprocedural arguments recovery
  - Known function signatures
  - Objective-C signatures
  - WIP: arrays and structs (we already have done it for x86)

Objective-C function signatures parsing example

```
v56@0:8@16@24d32{_NSRange=QQ}40
void (i64, {}, {}, double, { i64, i64 })
----------------
@24@0:8q16
{} (i64, i64)
----------------
@32@0:8{CGPoint=dd}16
{} (i64, { double, double })
----------------
q32@0:8{CGPoint=dd}16
i64 (i64, { double, double })
----------------
v88@0:8^{__CTFramesetter=}16@24{?=qq}32{CGRect={CGPoint=dd}{CGSize=dd}}48^{CGContext=}80
void (i64, i1*, {}, { i64, i64 }, { { double, double }, { double, double } }, i1*)
----------------
v64@0:8^{__CTFrame=}16{CGRect={CGPoint=dd}{CGSize=dd}}24^{CGContext=}56
void (i64, i1*, { { double, double }, { double, double } }, i1*)
----------------
v64@0:8^{__CTFrame=}16{CGRect={CGPoint=dd}{CGSize=dd}}24^{CGContext=}56
void (i64, i1*, { { double, double }, { double, double } }, i1*)
```

LLVM generation

- Translation preserving semantics
- Simplification
  - DCE (dead code elimination)
  - MemProp
  - ConstProp
- CFG region analysis

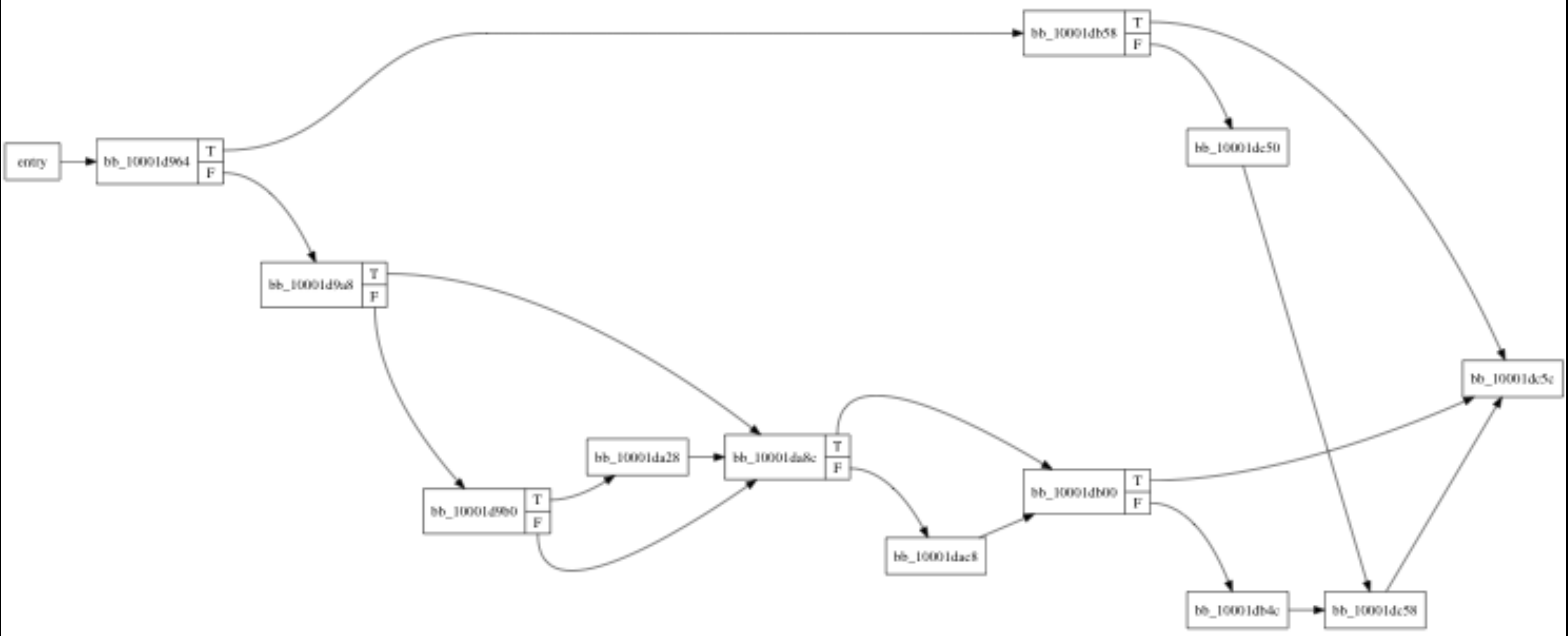
Example

```
adrp    x8, #0x1009b9000
ldr     x0, [x8, #0xdf0]
adrp    x8, #0x10099c000
ldr     x1, [x8, #0xf8]
bl      imp___stubs__objc_msgSend
mov     x29, x29
bl      imp___stubs__objc_retainAutoreleasedReturnValue
mov     x22, x0
ldr     x2, [x21, #0x20]
adrp    x8, #0x10099d000
ldr     x1, [x8, #0x690]
bl      imp___stubs__objc_msgSend
```

Example

```
bb_10001d9b0:                                         ; preds = %bb_10001d9a8
  %15 = ptrtoint [29 x i8]* @"OBJC_CLASS_$_TNDRCurrentUser" to i64
  store i64 %15, i64* %x0, !smd.objc-class !0, !smd.hidden !0
  %16 = ptrtoint [18 x i8]* @s_1006af362 to i64
  store i64 %16, i64* %x1, !smd.objc-sel !0
  %17 = ptrtoint [29 x i8]* @"OBJC_CLASS_$_TNDRCurrentUser" to i64
  %18 = call i64 @"TNDRCurrentUser::+ sharedCurrentUser"(i64 %17)
  %19 = ptrtoint [29 x i8]* @"OBJC_CLASS_$_TNDRCurrentUser" to i64
  %20 = call i64 @objc_retainAutoreleasedReturnValue(i64 %19)
  %21 = load i64, i64* %x0
  store i64 %21, i64* %x22
  %22 = ptrtoint [9 x i8]* @s_1006b44bd to i64
  store i64 %22, i64* %x1, !smd.objc-sel !0
  %23 = ptrtoint [29 x i8]* @"OBJC_CLASS_$_TNDRCurrentUser" to i64
  %24 = load i64, i64* %x2
  %25 = call i64 @"TNDRCurrentUser::+ setJobs:"(i64 %23, i64 %24)
```

Example



CFG for 'sub_10001d964' function

3, 4:

Vulnerabilities detection and results presentation

Pseudocode

LLVM to Objective-C/Swift-like pseudocode

(more accurate for Objective-C)

- Function names, signatures
- Statements
- Arguments
- Types
- Call sites
- Structural analysis (WIP)

Pseudocode

```
    objc_release(x8->MapSVGKOverlayRenderer::_layersByName)
} else {
    // bb_10005b018:
    x24 = "fillColor"
    x0 = x20
    x1 = x24
    [x8->MapSVGKOverlayRenderer::_layersByName fillColor]
    x23 = x0
    if (x22)==(1) {
        // bb_10005b034:
        x0 = x20
        x1 = x24
        [x8->MapSVGKOverlayRenderer::_layersByName fillColor]
        x2 = x0
    } else {
        // bb_10005b048:
        x2 = 0
    }
```

Analysis

- Pattern matching on LLVM (detects most of vulnerabilities)
- TBD: deep dataflow analysis (e.g., taint analysis)
- LLVM to pseudocode mapping (for results presentation)

Vulnerabilities: data transfer

Weak SSL

```
    if ((x01)==(0)) goto bb_1000b037c;
    x01 = [x20 sender];
    x01 = objc_retainAutoreleasedReturnValue(x01);
    x21 = x01;
    [x01 continueWithoutCredentialForAuthenticationChallenge:x20];
bb_1000b0350:
    x01 = x21;
bb_1000b0354:
    objc_release(x01);
```

Vulnerabilities: data transfer

No SSL

```
static const char *str_53 = "http://www.qq.com";
static const char *str_54 = "http://qzs.qq.com/";
static const char *str_55 = "http://ti.qq.com/favorite/favorite_error.html";
static const char *str_56 = "http://ti.qq.com/favorite/share_error.html";
static const char *str_57 = "http://itunes.apple.com/cn/app/id444934666";
```

Vulnerabilities: bad crypto

MD5, SHA1, 3DES, etc…

```
x0 = (x31)+(8)
x1 = (x31)+(104)
var x2 = x8
CC_SHA1_Update()
x0 = (x31)+(104)
w1 = 1
w2 = 1
```

Vulnerabilities: data storage

– Pasteboard usage

– NSLog

– Background mode

```
    x0 = x25;
    goto bb_10037ef34;
bb_10037eeb4:
    NSLog("2ñ*0åö¡ã1Y%ç");
    [x22 disConnectCurPeripheralForVerifyFailure:x2];
    x1 = x23;
    [x22 curPeripheral];
```

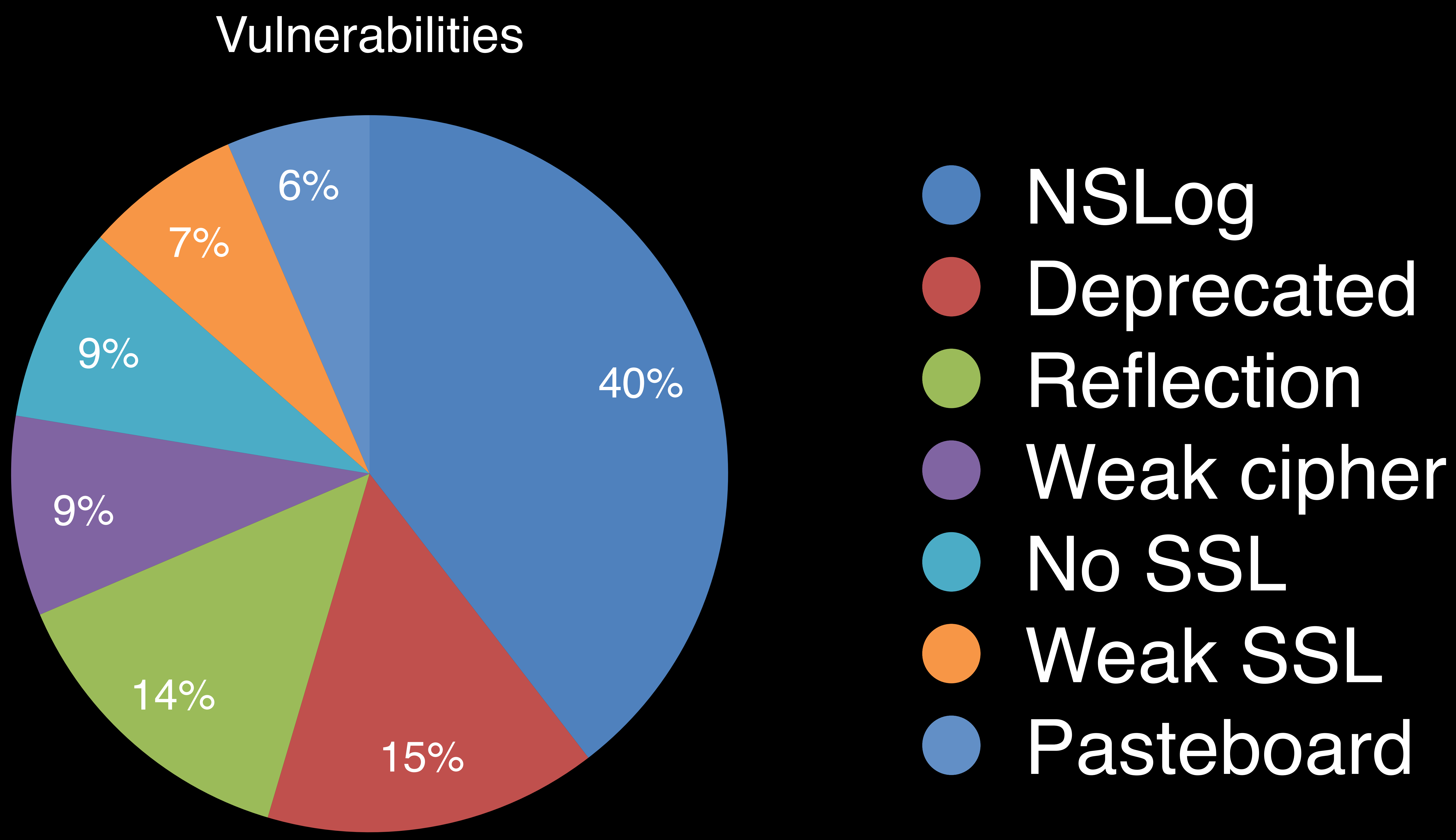
Vulnerabilities: reflection

```
bb_10031d804:
    x8 = (x31)+(16);
    x8 = (x31)+(15);
    x3 = (x31)+(24);
    [x22 performSelectorIfExists:x24 withArguments:x3];
    objc_release(x22);
    objc_release(x19);
    objc_release(x0);
    x8 = (x26)-(x8);
```

Vulnerabilities: TBD

- Unencrypted sensitive data storage in application directory
- Cache of network requests
- Data validation (SQLi, XSS, path manipulation, …)
- Weak jailbreak detection
- Authentication (2fa, password complexity, number of attempts)

Statistics: vulnerabilities



Conclusion

- Our toolset can:
  - Find vulnerabilities in iOS app using only its iTunes link
  - Present these vulnerabilities on pseudocode

- Future work:
  - Deep analysis (dataflow, etc.)
  - Less false positives
  - Objective-C/Swift decompilation

Questions?

alexandrov@smartdec.net

safin@smartdec.net