REVERSE ENGINEERING MOBILE APPS:

HOW,
WHY,
AND WHAT NOW?

Michał Kałużny
maku@justmaku.org
@justMaku

About me:

- 15 years of experience as software developer
- First "Hello World" written in Atari Basic on an 8-bit Atari 65XLE
- Member of the World of Warcraft modding community and the original Pokemon Go Reverse Engineering team.
- Found vulnerabilities in Blizzard's Battle.net, Disney mobile apps and others

INTRODUCTION

- INTRODUCTION
- MAIN DISH
  - TOOLKIT
  - DECOMPILATION
  - CODE INJECTION
  - PREVENTION
- EPILOGUE

Reverse Engineering:

The reproduction of another manufacturer's product following detailed examination of its construction or composition.

Decompilation:

The reverse process of compilation i.e. creating high level language code from assembly.

At the basic level, it just requires understanding of the machine code and rewriting it into a higher level language.

Dynamic vs static analysis

DECOMPILATION/TOOLKIT

- **Static Analysis**
  - IDA Pro/Free
  - Hopper
  - Baksmali
  - JDGUI
- **Dynamic Analysis**
  - Frida
- **Packaging and Signing**
  - APKtool
  - CydiaImpactor

DECOMPILATION/CODE ACCESS

- **iOS**
  - Jailbroken device (via Clutch)
  - Public ad-hoc/enterprise releases
  - Cracked IPA repositories (iphonecake)
- **Android**
  - Rooted device
  - Public APK releases
  - Play Store mirrors (apkfind)

DECOMPILATION/CODE ACCESS

iOS vs Android vs Non-native

	ObjC	Swift	Java	Kotlin	C#	JS
Machine Code	ASM	ASM	Smali	Smali	IL	JS
Symbols	Partial	Partial	Full	Full	Full	Partial
Difficulty	Medium	Hard	Easy	Easy	Easy	Hard

Note: Default settings in release mode.

Swift Example

DECOMPILATION/CODE ACCESS

```
func hasPaidForApp() -> Bool {
    return UserDefaults().bool(forKey: "HasPaidForApp")
}
```

DECOMPILATION/CODE ACCESS

```
push    rbp                                                     ; CODE XREF=-[_TtC7DemoApp14ViewController hasPaidForApp]+31
mov     rbp, rsp
push    r13
sub     rsp, 0x28
mov     qword [rbp+var_10], r13
call    __TtSo12UserDefaultsCMa                                 ; type metadata accessor for __ObjC.UserDefaults
mov     r13, rax
call    __TtSo12UserDefaultsCABycfC                             ; __ObjC.UserDefaults.__allocating_init() -> __ObjC.UserDefaults
lea     rdi, qword [aHaspaidforapp_100003d26]                   ; "HasPaidForApp"
mov     ecx, 0xd
mov     esi, ecx
mov     edx, 0x1
mov     qword [rbp+var_18], rax
call    imp___stubs___TtSS2SBp21_builtinStringLiteral_Bw17utf8CodeUnitCountBi1_7isASCIItcfC   ; Swift.String.init(_builtinStringLiteral: Builtin.RawPointer, utf8CodeUnitCount: ...
mov     rdi, rax
mov     rsi, rdx
mov     rdx, rcx
mov     qword [rbp+var_20], rcx
call    imp___stubs___TtSS10FoundationE19_bridgeToObjectiveCSo8NSStringCyF   ; (extension in Foundation):Swift.String._bridgeToObjectiveC() -> __ObjC.NSString
mov     rdi, qword [rbp+var_20]                                 ; argument "instance" for method imp___stubs__swift_unknownRelease
mov     qword [rbp+var_29], rax
call    imp___stubs__swift_unknownRelease                       ; swift_unknownRelease
mov     rsi, qword [0x1000060e8]                                ; argument "selector" for method imp___stubs__objc_msgSend, @selector(boolForKey:)
mov     rdi, qword [rbp+var_18]                                 ; argument "instance" for method imp___stubs__objc_msgSend
mov     rdx, qword [rbp+var_29]
call    imp___stubs__objc_msgSend                               ; objc_msgSend
mov     rdi, qword [rbp+var_29]                                 ; argument "instance" for method imp___stubs__objc_release
mov     byte [rbp+var_29], al
call    imp___stubs__objc_release                               ; objc_release
mov     rdi, qword [rbp+var_18]                                 ; argument "instance" for method imp___stubs__objc_release
call    imp___stubs__objc_release                               ; objc_release
mov     al, rax
add     rsp, 0x28
pop     r13
pop     rbp
ret
```

DECOMPILATION/CODE ACCESS

```
int __T07DemoApp14ViewControllerC010hasPaidForBOSbyF() {
    type metadata accessor for __ObjC.UserDefaults();
    var_18 = __ObjC.UserDefaults.__allocating_init();
    stack[-48] = "HasPaidForApp"
    swift_unknownRelease(0xd);
    var_29 = [var_18 boolForKey:stack[-48]];
    [stack[-48] release];
    [var_18 release];
    rax = var_29;
    return rax;
}
```

Kotlin Example

DECOMPILATION/CODE ACCESS

```
fun hasPaidForApp(): Boolean {
    return false
}
```

DECOMPILATION/CODE ACCESS

```
.method public final hasPaidForApp()Z
    .locals 1

    .line 34
    const/4 v0, 0x0

    return v0
.end method
```

DECOMPILATION/CODE ACCESS

```
public final boolean hasPaidForApp()
{
    return false;
}
```

DEMO

Code obfuscation

- Android: ProGuard/DexGuard
- C#: ConfuserEx
- Objective-C: ios-class-guard
- *and many other commercial products*

Other techniques

- Inline "door opening" functions.
- Use lower level languages for mission critical code.

Unsecure token storage

- Possible leakage of information
  - i.e an attacker is able to contact your A/B test provider's API and receive information about running experiments
- Rate limiting
  - i.e an attacker may use your token to block your app from using the API
- Paid APIs
  - i.e an attacker may send enormous amount of request in your name that you will have to pay for

Secure token storage

- Always use:
  - Android KeyStore, not SharedPreferences
  - iOS Keychain, not UserDefaults
- Try not to ship them with binary
  - Push Notifications were used as a secure channel
- Proxy request through a server you control.

Secure token storage

EPILOGUE

Securing your app is a 10:1 effort.

For each 10 hours you spend "protecting" your code, it will take an experienced attacker around 1 hour to undo all that hard work.

EPILOGUE

Not all reverse engineering is bad

EPILOGUE

Want to learn more?

- https://beginners.re
- HackerOne
- CTFTime

Resistance is futile.

Q&A

@justMaku

maku@justmaku.org