Hardening Mobile Apps

SecAppDev 2015
Ken van Wyk, @KRvW

Leuven, Belgium
23-27 February 2015

KRvW Associates, LLC
Ken van Wyk, ken@krvw.com, @KRvW

SecureApplication Development
secappdev.org

Topics covered

- Problems
  - ObjC run-time
  - Static analysis
  - Dynamic analysis
- Hardening
  - Configuration
  - Architecture
  - Hardening tips

ObjC Run-time is flawed

- Unlike C "functions" are not called
  - Messages are passed
  - Objects dynamically allocated
- Within process space, dynamic tampering also possible
  - Message traffic
  - Objects

Biggest risks

- Information harvesting
  - Lost/stolen device
  - Personal data
- Reverse engineering apps
  - Locate and exploit weaknesses

Reverse engineering

- Attacker wants to learn how your app works
  - Deep internal details
- Attacker wants to attempt to trick your app into misbehaving
  - Tamper with runtime
- How? Jailbroken device and some free tools
  - And a *lot* of time

WARNING
OUR NEIGHBORS ARE WATCHING TO REPORT ANY SUSPICIOUS ACTIVITY TO OUR LOCAL LAW ENFORCEMENT AGENCY

Prerequisite tools and env

- Mac with OS X and Xcode
- Jailbroken device
  - evasi0n works great
- Cydia and friends
  - Cydia installed with evasi0n
  - Shell access
    - OpenSSH - install with Cydia
  - Debugger
    - gdb - install with Cydia

Bare minimum essentials

Analysis techniques

- Static analysis
  - Observe attributes of the executable, app files
  - Yes, encrypted (app store) apps too
- Dynamic analysis
  - Run the app and learn how it works
- Tampering
  - Trick the run-time env

Static analysis

- Any binary can be examined
  - Usually reveal a map to classes, objects, text, symbols, etc.
- Common tools
  - otool
  - class-dump-z
  - nm

- Examples
  - Linked libs, methods
    - otool -L appname
    - otool -l appname
  - List of classes
    - class-dump-z appname
  - Symbol table
    - nm appname

It’s C underneath the hood

Beneath that nice OOP ObjC layer lies a C foundation

- Pretty much everything in ObjC can be done in C
  - Primitives for doing all the OO stuff
  - *objc_msgSend()*, *objc_getClass()* are prime examples

This matters to us when analyzing statically or dynamically

Encrypted binaries too

- Basic process
  - Use app loader to decrypt
  - Calculate memory offsets
  - Store process to disk
    - dd is your friend
    - Will also need plutil and gdb
- HOWTO available
  - http://www.mandalorian.com/2013/05/decrypting-ios-binaries/

Let’s take a look…

Dynamic analysis

What can we learn from observing it running?

- A lot
- All those messages
- Memory contents
- CPU registers

You don't have anything to hide, right?

Attacking a running app

- Man in the app (MITA)
  - The most dangerous form of on-host dynamic attack
  - Internal access to everything
- That ObjC run-time messaging architecture is going to haunt us

A few more tools

For these, you'll want

- gdb
- Cycript (see slide)
- Network proxy (e.g., Burpsuite)
- SSLstrip (optional)

Message eavesdropping

Use gdb to build a simple but effective message eavesdropper

Example

```
gdb -q -p PID
break obj_msgSend
commands
x/a $r0
x/s $r1
c
```

Cycript

"Cycript allows developers to explore and modify running applications on either iOS or Mac OS X using a hybrid of Objective-C++ and JavaScript syntax through an interactive console that features syntax highlighting and tab completion"
— From http://www.cyript.org

It is an amazing utility for dynamically probing a running app

Fun with Cycript

Basics

```
# cycript
cy# var myString = [[ NSString alloc ]
cy> initWithString: @“Hello world”];
“Hello world”
cy# [ myString length ];
11
```

Combination of JavaScript and ObjC syntax gives amazing capabilities

Cycript (2)

PhotoVault examples (from Zdziarski)

```
# cycript -p PID
cy# var app = [UIApplication sharedApplication];
“<UIApplication: 0x22f050>”
cy# [ app openURL: [ NSURL URLWithString:
cy> @“http://www.secappdev.org”]];
1
cy# app.networkActivityIndicatorVisible = YES
```

Cycripting for fun and profit

- Break client-side logic
  - Alter PINs, booleans, semaphores
  - Replace methods
- Probe running app data
  - Can be verbose, but you get everything in an object

```
    cy# function appls(a){ var x={};
for(i in *a){ try{ x[i] = (*a)[i]; }
catch(e){}} return x; }
    cy# appls(object);
```

Client-side logic

You didn't think you could trust client-side logic, did you?

Network eavesdropping

- Can MITM all HTTP and HTTPS traffic
  - Coffee shop attack is easy to implement for testing an app
- Tools and techniques
  - Proxy, like Burpsuite
  - SSLstrip
  - APN attack (cellular data)

Tampering

- Now let's go beyond mere observation
- Replace existing methods
  - Change address in gdb
  - Dynamic linker attack
    - Put your library in DYLD_INSERT_LIBRARIES
- Automate dynamic linking
  - MobileSubstrate

Nothing is what it appears

Now we can change the entire universe your app runs in

(If this doesn't seem bad, go watch The Matrix)

Resources

Hacking and Securing iOS Applications, Jonathan Zdziarski, O'Reilly, 2012

Evasi0n, popular jailbreaking tool, http://www.evad3rs.com/

Hardening

User actions and client configurations

Architectural considerations

Hardening tips

But remember, nothing is perfect.

User actions and configurations

Strong passcodes help

MDMs can manage configurations of entire fleets

LOADING ZONE
15 MINUTE
AUTHORISED VEHICLES EXCEPTED
7-10 AM
AUTHORISED VEHICLES EXCEPTED
10 AM - 10 PM
7-10 PM
TAXIS EXCEPTED WHEN PICKING UP/ DROPPING OFF PASSENGERS
TAXI ZONE
ALL OTHER TIMES
10 PM - 7 AM

Architectural considerations

Design choices make a huge difference

- Client cannot be trusted
  - Sensitive data
  - Sensitive functions
  - Security controls
- Client should provide presentation layer
  - Minimal functionality
  - Processing should be server

Hardening tips

- Non-obvious names
  - Obfuscate functional purpose
- Disable debugging

```
#define DENY_DEBUG 31
ptrace(DENY_DEBUG,0,0,0);
```

- Complicate disassembly
  - Compiler optimizer
  - Strip symbols

Hardening tips (2)

- Sensitive code
  - On server, but…
  - Write in C or ASM
  - Compile + link in-line
  - Expand loops manually
- Force your attacker to single step through
- Don't give away anything

Hardening (3)

- Data storage
  - Encrypt
    - DataProtection API for consumer grade
    - Keys on server
  - Common Crypto Lib
- Secure file wiping
- SQLite data wiping
  - Update before delete

Tamper detection

How do we know?

- Run-time integrity checks
  - Memory offsets of sensitive objects
- Sandbox integrity
  - Attempt to fork
  - Size and checksum of */etc/fstab*
  - Symbolic links in */Applications*
  - Common jailbreak files and apps
    - */Applications/Cydia.app*
- Honeypots in app

There ain't a horse that can't be rode or a man that can't be throwed.

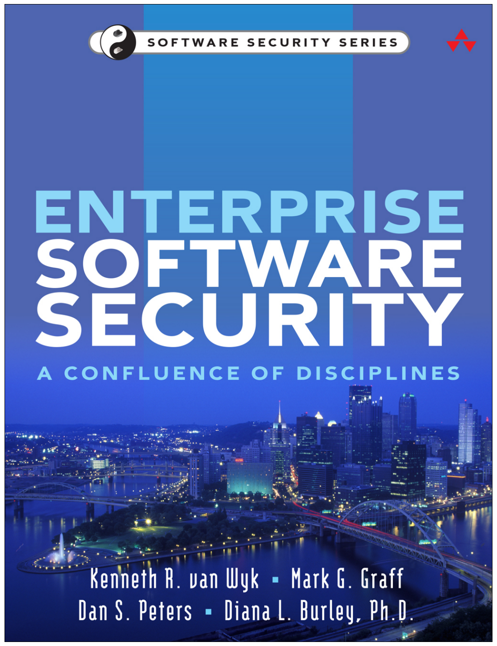
Tamper response

What to do?

- Remote wipe
- Phone home
- Log everything
- Wipe user data, keys
- Disable network access
- Et cetera

Kenneth R. van Wyk
KRvW Associates, LLC

Ken@KRvW.com
http://www.KRvW.com
@KRvW


SOFTWARE SECURITY SERIES
ENTERPRISE SOFTWARE SECURITY
A CONFLUENCE OF DISCIPLINES
Kenneth R. van Wyk • Mark G. Graff
Dan S. Peters • Diana L. Burley, Ph.D.