

iPhone development

Past, Present, and Future

Nate True

Inventor, Founder of creations.net

O'Reilly Emerging Technologies Conference 2008

Who is Nate True?

- Christian
- Computer Science graduate from the University of Washington
- Inventor
- iPhone developer

Presentation Outline

- First a little history
- The state of the SDKs
- Setting up the hacker SDK
- Compiling your first program
- A bit about iPhone UI
- Your first iPhone UI program

In the beginning...

- The first dev team met on OSX86.hu
- To unlock the iPhone

Complications

- Without service, iPhones were unusable
- Those who would unlock their phones could not get in to hack them!

Attack vectors

- The phones
- The downloadable firmware

Firmware attacks

- The IPSW file was in ZIP format
- The ramdisk wasn't encrypted
- It contained the encryption key for the main disk
- Main disk could be extracted
- iPhone is like Apple TV - based on OSX

Phone attacks

- Phone running a service called AFC
- Allows file access to the Media area only
- This is known as a “chroot jail”
- Hence the term “jailbreak”

Phone attacks

- Then the iTunes phone restore process was analyzed and documented
- It could be manipulated!

The first jailbreak

- A few files are dropped into the Media area
- These:
 - Reconfigure AFC to access the whole filesystem
 - Make the entire phone writable
- Then the restore process is hijacked to copy them over the originals

This is where I came in

- The first jailbreak release was for Mac
- Help needed reverse-engineering the functions for the Windows version
- So I helped

iPhone hacking terms

- Jailbreak
 - To reconfigure or duplicate the AFC service to access the whole phone
 - More recently, to simply install Installer.app no matter what the method

iPhone hacking terms

- Activate
 - To bypass the “Activate iPhone” screen
 - Either by activating with AT&T or by patching lockdown

iPhone hacking terms

- Unlock
 - To allow the iPhone to accept any SIM card
 - Requires modifying the baseband firmware
 - And additional lockdown patches

Native iPhone apps

- Once the iPhone had been jailbroken, this became a primary focus
- And it's why you're here today!

Two SDKs

- The official Apple SDK
- The hacker's SDK

Apple's SDK

- As yet unreleased, rumors everywhere
- Likely:
 - Large barrier to entry
 - Very enterprise-focused
 - iTunes store distribution
 - Highly profitable

The hacker SDK

- Been out for months
- Difficult to set up
- Free software
- Poor documentation
- Apps are likely to be compatible with Apple SDK

Concerning openness

- No official documentation
- One learns by studying others' code
- Open source is essential here

App distribution

- Apps were hard to install through AFC
- Permissions, etc
- Lupinglade's Installer.app changed that
- An excellent solution

How Installer works

- Installer has a list of repositories
- “Community Sources” is a default list
- Users can add others by URL though

How Installer works

- An app is sent to a repository maintainer
- They make packages and add them to their repository
- Users find the application this way

Making money

- This probably brought most of you here
 - Donations
 - Selling your software
 - Ads and sponsorship
 - Contract work
 - Employment

iPhone Donationware

- Many people are willing to donate
- That is, if they love your app
- Nag screens can increase donation rates
- But they decrease user satisfaction

Selling iPhone software

- Most popular with businesses
- Generally as trialware
- Paypal works on iPhone (big plus!)
- However:
 - Many users reject trialware immediately
 - So you have to prove you're worth the \$

Ads and sponsorship

- Many look down upon adware
- But free is good
- User interest problems
- Sponsorships are hard to find

Contract work

- Simple
 - You write an app you care about
 - Company pays you to make an app they care about
- Sometimes you can negotiate a commission on actual sales
- Sometimes you don't want to

Employment

- Also simple
- Make an app you care about
- Get a job offer from a company
- Great if you don't want your own business

Tutorial time!

- Let's set up the iPhone toolchain
- Requirements:
 - A Mac

Software

- Xcode DMG
- iPhone Toolchain V0.5 DMG
- iPhone 1.1.4 firmware filesystem
- Headers patch archive
- Sample projects
- These are all on the DVDs I will pass out

Extracting the firmware

- `sudo mkdir -p /usr/local/arm-apple-darwin`
- `sudo chmod 777 /usr/local/arm-apple-darwin`
- `cd /usr/local/arm-apple-darwin`
- `tar -xvzf /path/to/heavenly.tgz`

Installing Xcode

- Mount the DMG
- Run `XcodeTools.mpkg`

Installing the toolchain

- Mount the Toolchain DMG
- Run the installer
- Run Ooh shiny

Patching the headers

- As-is, the toolchain doesn't work with latest Leopard
- So extract the fixed headers
 - `cd /usr/local/arm-apple-darwin/arm-apple-darwin`
 - `tar -xvzf /path/to/include.tgz`

Building

- Extract and open HelloConsole.xcodeproj
- Build it!

```
baladoux:helloconsole natetrue$ file helloconsole  
helloconsole: Mach-O executable arm
```

Copying to your phone

- A few ways to do this
 - AFP (from Installer)
 - Mount as share from your Mac
 - iPHUC or similar
 - Uses AFC to copy it over
 - SSH/SFTP

Running it

```
# chmod +x helloconsole  
# ./helloconsole  
Hello world!  
# █
```

Boring

- Console apps are great and all
- But that's not what the iPhone is famous for
- Time for some UIKit basics

Objective C

- All iPhone app programming is (currently) done in Objective C
- Just like Mac apps
- The calling convention is very Scheme-like
- Memory management is the hardest part about it

Memory Management in Objective C

- Objects have a reference count
- Can be set to autorelease at the end of the message loop
- When allocated with `[class alloc]` they do not autorelease
- When allocated with `[class classWithExampleParameter: parameter]` they do

Memory Management in Objective C

- This is why so many Mac applications have memory leaks
- To set an object to autorelease, use [object autorelease]
- To release an object immediately use [object release]
- To keep an object use [object retain]

What is UIKit?

- UIKit is one of the iPhone frameworks
- Implements most of the iPhone UI

Other frameworks

- Celestial
 - Controls audio/video playback
- MusicLibrary
 - Lets you query the iPod database
- GraphicsServices
 - Does lots of neat things but is poorly documented

UIKit classes to know

- UIApplication
 - An application is a subclass of this
- UIWindow
 - Each application needs at least one
- UIView
 - Most controls are subclasses of this

Anatomy of an iPhone application

- UIApplication::applicationDidFinishLaunching
 - UIWindow::alloc
 - UIView::alloc
 - UIWindow::setContentView(UIView)
 - UIView::addSubview(all controls)

Building SampleApp

- I don't know who wrote SampleApp
- But it's a great sample app
- Building is the same
 - Open the Xcode project
 - Build it

Installing SampleApp

- The build process makes SampleApp.app
- This must be copied to /Applications on your iPhone
- Then you need to +x the SampleApp executable
- And restart SpringBoard
 - killall SpringBoard

Learning more about iPhone development

- Grab your favorite open-source project and examine it
- Great ones are:
 - NESapp by Jon Zdziarski
 - ApolloIM by Alex Schaefer
 - Erica Sadun's various utilities

Question time

- Please ask your questions presently