Beginners Guide to Developing for a Jailbroken iOS Platform

Priya Rajagopal
Twitter: @rajagp
Blog: http://www.priyaontech.com
CocoaHeads, Jan 2012

Jailbreaking is Legal

(..at least in the US)

Why develop for a jailbroken platform?

- Develop run-time patches (.dylibs) that can be automatically loaded and shared across apps
  - Link with third part dylibs (eg- BTStack)
- Hook into "system" apps and control platform behavior
  - Eg. Mobile Safari, Springboard
- Utilize features not exposed through SDK's public APIs to build something really cool

Why develop for a jailbroken platform?

- More control over the platform
  - Terminal window, ssh, scp, rm etc. It's a unix system.
- Don't need an Apple developer's license
  - Self signed apps, pseudo signed apps
- You don't even need a Mac
  - You can even develop on the phone (Cool!)
- Options :
  - Distribute through Cydia
  - Internal Enterprise apps
  - Personal use. If you can't find it, you can build it!

Tethered vs. Untethered Jailbreak

- Tethered
  - You need to tether your device to your PC to reboot it. Quite inconvenient
- Untethered
  - You don't need to tether your device to your PC to reboot it.
- Partial Untethered
  - Tethered but you can reboot untethered to enable minimal functionality

Jailbreak Software

(If its not free, it's a scam)

- RedSn0w (Mac /Windows)
- Jailbreakme.com (Web)
- PwnageTool (Mac)
- GreenPois0n (Mac/Windows)

Status of iOS Jailbreak

- iOS 5.0.1 for A4 devices: Untethered jailbreak available from RedSn0w
  - http://cydiahelp.com/jailbreak-5.0.1-untethered-iphone-4-3gs-ipod-touch-4g-3g-ipad-with-redsn0w-0.9.10b1-tutroial/
  - iPhone4S and iPad2 coming soon
- iOS 4.3.3: Last untethered jailbreak

Basic Apps/packages to install on your JB phone

Cydia – App Distribution center for jailbroken Apps

- Jay Freeman aka “Saurik”

- OpenSSH
- SBSettings
- syslogd
- syslog toggler
- Mobile Terminal
  - Download it from a source http://YourCydiaRepo.org via Cydia

SHSH Blobs

- Signature Hashes associated with your firmware
  - Unique to a device
- During upgrade/restore, Apple signature servers verify the signatures
- With every new release, Apple stops signing old versions
- Save your SHSH blobs if you want to restore to an older version
- Cydia now automatically saves them
  - Can also use TinyUmbrella
- To Restore to older version of firmware
  - TinyUmbrella or
  - Use iTunes , point to Cydia's signature servers

Mobile Substrate

- *"… is the de facto framework that allows 3rd-party developers to provide run-time patches ("MobileSubstrate extensions") to system functions"*
  - From Jay Freeman ("Father of Cydia")
- Mobile Substrate Extensions a.k.a **Tweaks**
- MobileHooker
  - Hooking system functions (Obj-C, C/C++)
  - *MSHookMessageEx()*
  - *MSHookFunction()*
- MobileLoader
  - Loads using DYLD_INSERT_LIBRARIES env. var
  - Can specify filters

More Mobile Substrate…

- Safe Mode Operation
  - All tweaks will be disabled if a tweak crashes SpringBoard
- MobileSubstrate installed via Cydia

http://iphonedevwiki.net/index.php/MobileSubstrate

A note on class-dump

- Command line utility that generates Obj-C declarations for classes, categories and protocols from Mach-O files
- You can use it to generate header file declarations for private headers, private frameworks, system apps etc.

```
./class-dump -H /Developer/Platforms/
iPhoneOS.platform/Developer/SDKs/iPhoneOS5.0.sdk/
System/Library/CoreServices/SpringBoard.app/
SpringBoard -r --sdk-ios 5.0 -o SpringBoard
```

- http://www.codethecode.com/projects/class-dump/

Development Options

- Xcode
  - Mac
- Theos
  - Mac, Linux, iOS

Development Option- XCode

- Can build self-signed apps
- Can install and debug via Xcode
  - Fairly Complicated Setup. Could not get app to install with Xcode 4/iOS5
- Can manually install the app
  - You rely on syslog logging
- Could build Mobile Substrate extensions
  - "MobileSubstrate dylib" Template was available for Xcode 3 (from Skylar EC)
  - Template not available for XCode4

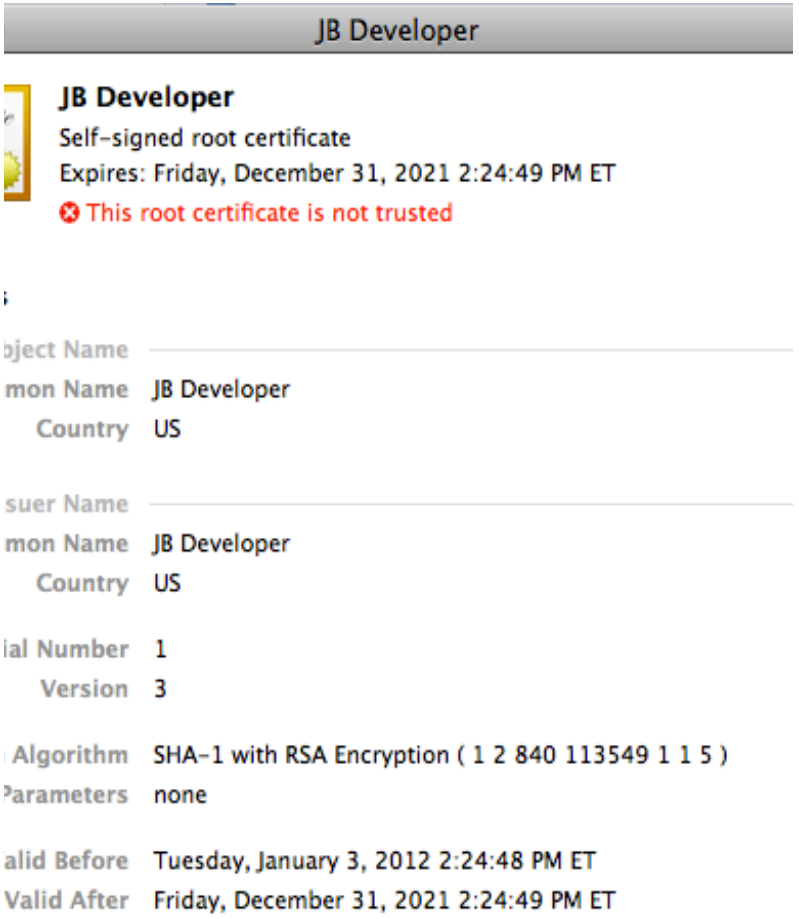
Development Option – Theos

- "*Theos is a cross-platform suite of development tools for managing, developing, and deploying iOS software without the use of Xcode*"
- Creator: D Howett
- Can develop on Linux , Mac or iOS
- Project templates via NIC.pl
- A build system with automatic packaging support (ready for Cydia distribution)
  - Can build "pseudo signed" apps with ldid
- Automatic installation of apps onto device
  - No debugging facility. Rely on syslogs
- Can (easily) build mobile substrate extensions
- Preferred Option

Building self-signed app with XCode

- Generate self signed certificate
  - KeyChain->Certificate Assistant
- Instruct Xcode to use code signing procedures in XCCodeSignContext instead of the more restrictive XCiPhoneOSCodeSignContext

```
sudo /usr/bin/sed -i .bak 's/
XCiPhoneOSCodeSignContext/
XCCodeSignContext/' /Developer/
Platforms/iPhoneOS.platform/
Info.plist
```



Building self-signed app with XCode

- Update Project Build Settings

Building self-signed app with XCode

- Build the app (Do not Install it)

Installing self-signed app (..that was built with Xcode)

- Manual (From terminal)
  - scp <myApp.app> root@<iPhone>:/Applications
  - Respring the phone
    - SBToggler
  - Debugging
    - Enable syslog on phone
    - /var/log/syslog
- Via Xcode
  - Follow series of steps to enable entitlements for debugging
    - Blog post: http://networkpx.blogspot.com/2009/09/compiling-iphoneos-31-apps-with-xcode.html
  - Never got it working with Xcode 4.2

Uninstalling App

- ssh root@<iPhone>
- cd /Applications
- rm –rf MyApp.app
- Repring

Using Private Headers with XCode

- Get the headers
  - Option1:Download the headers
    - https://github.com/nst/iOS-Runtime-Headers/tags
  - Option2: Generate with class-dump
    - The generated header files have some spurious #imports that need to be removed

```
sudo sed -i.old '/NSObject\.h/ d' *.h
```

- Copy the headers into the appropriate Frameworks folder
  - /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS<sdk>.sdk/System/Library/PrivateFrameworks/<Framework>/Headers folder
    - You would have to create “Headers” folder
  - /Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS<sdk>.sdk/System/Library/Frameworks/<Framework>/Headers folder

- Add the framework to your project
  - Link Binary With Libraries build phase (“Add Other”)

- Demo : Creating self signed app with XCode

Theos : Steps to set up Development Environment

1) Install the iOS SDK & Xcode
2) Install MacPorts (package mgmt. system)
   – http://www.macports.org/install.php

3) Setup Theos
(Run Cmds from a terminal window)

- Create the installation directory

```
mkdir /theos
export theos = /opt/theos
```

- Check out the theos src

```
cd $THEOS
svn co http://svn.howett.net/svn/theos/trunk $THEOS
```

- Install ldid – “pseudo code signing tool”

```
cd $THEOS/bin
curl -s http://dl.dropbox.com/u/3157793/ldid > $THEOS/bin/ldid;
chmod +x $THEOS/bin/ldid
```

4) Install Private headers

- Download private headers for private frameworks for 3.X from https://github.com/rpetrich/iphoneheaders/archives/master
  - You can also generate the headers for missing frameworks using class-dump
- Copy the headers into include folder

```
cd $THEOS/include
cp -r  ~/Downloads/<headers folder>/* .
```

- Some system files may be missing : So do a manual copy

```
cp /System/Library/Frameworks/IOSurface.framework/Headers/IOSurfaceAPI.h $THEOS/include/IOSurface/.
```

5)Install dpkg

- Needed to create .deb packages

```
sudo port install dpkg
```

Building & Running an App With Theos

- `export SDKVERSION=<sdk version>`
- Run the "New Instance Creator" (NIC)
  - A perl script that allows you to create projects based on templates

```
$THEOS/bin/nic.pl
```

- Select "Application" template. Fill in the basic stuff
- Build and Install

```
make package
export THEOS_DEVICE_IP = <IPAddress of your JB phone>
make install
```

Demo of Simple App with Theos

Mobile Substrate Extensions with Theos

- Very simple with Theos
  - Template via nic.pl
- Logos
  - Preprocessor directives
  - %hook;
  - %orig;
- Logify
  - Logs methods within specified header file

A Comparison

Without Logos

```
IMP original_activateAlertItem_;

void replaced_activateAlertItem_
(SBAlertItemsController* self,SEL _cmd,id item)
{
       Class controller = objc_getClass
("SBSMSAlertItem");
        if (![item isKindOfClass:controller])
        {
            original_activateAlertItem_
(self,_cmd,item);
        }
}

extern "C" void initialize();
extern "C" void initialize()
{
       Class controller = objc_getClass
("SBAlertItemsController");
        MSHookMessageEx(controller,@selector
(activateAlertItem:),(IMP)
replaced_activateAlertItem_,(IMP*)
&original_activateAlertItem_);
}
```

With Logos

```
%hook SBAlertItemsController

-(void)activateAlertItem:(id)item
{
    %log;
    if (![item isKindOfClass:%c
(SBSMSAlertItem)])
    {
          %orig;
    }
}
%end
```

Demo : Simple Tweak Using Theos

“With Great Power Comes Great Responsibility…
So Please Code Responsibly”

Thank you!

Twitter:@rajagp