

# Hack.lu

Mobile application testing

# Android application testing

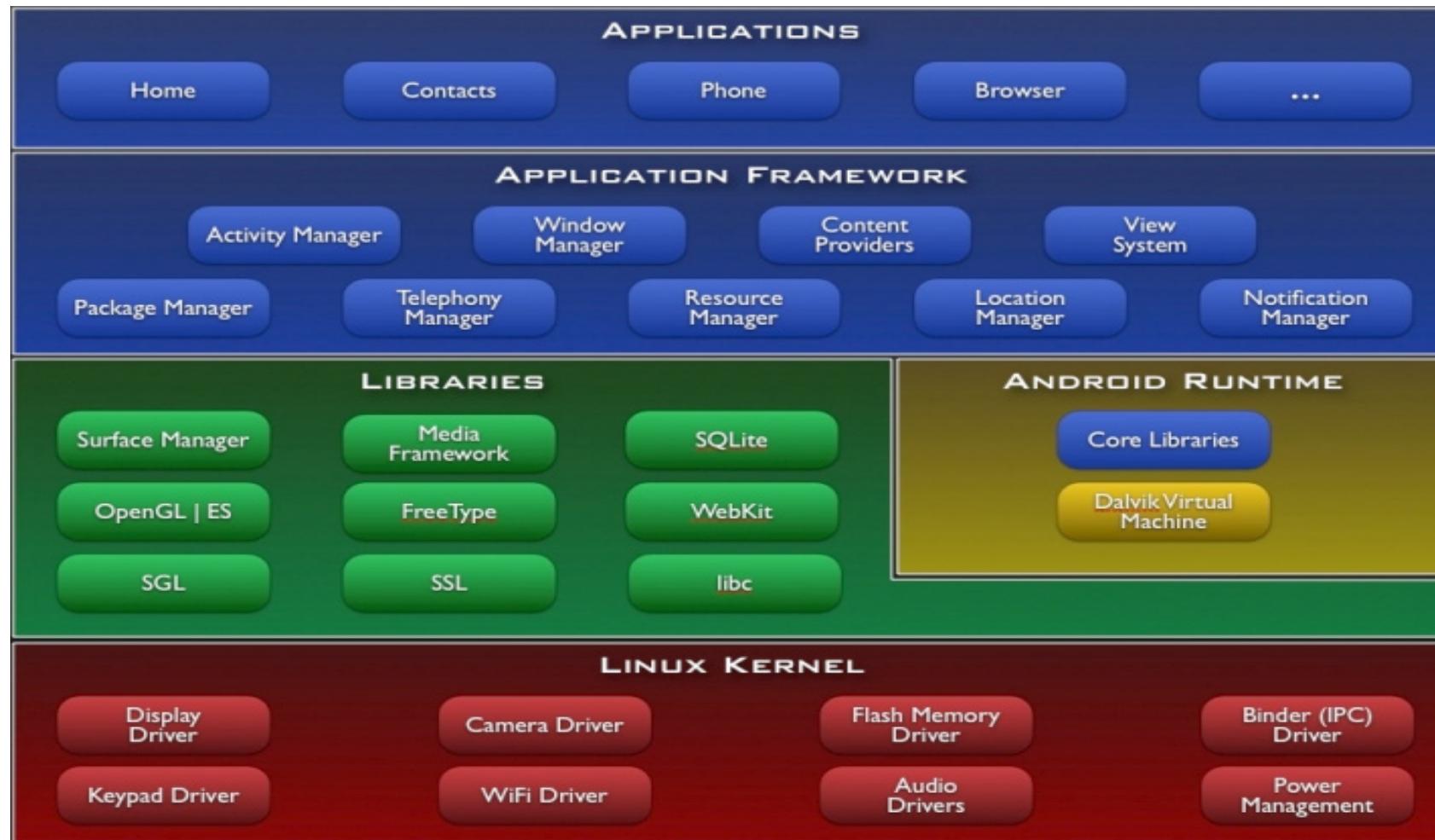


anDROID

# Android application testing

- If you want to do it yourself:
  - Android SDK: <http://developer.android.com/sdk/index.html#download>
  - APKTool: <https://code.google.com/p/android-apktool/downloads/list>
  - JD-GUI: <http://jd.benow.ca/>
  - Dex2Jar: <https://code.google.com/p/dex2jar/downloads/list>
  - Burp Proxy: <http://portswigger.net/burp/download.html>
  - Andriller: <http://android.saz.lt/cgi-bin/download.py>
  - Python 3.0: <http://python.org/download/releases/3.0/>
  - AFLogical: <https://github.com/viaforensics/android-forensics>
  - SQLite Browser: <http://sourceforge.net/projects/sqlitebrowser/>
  - Drozer: <https://www.mwrinfosecurity.com/products/drozer/community-edition/>
  - Lobotomy: <https://github.com/LifeForm-Labs/lobotomy>

# Android application testing



# Android application testing

- But first....
- Prepare your device:
  - Enable USB debugging
  - Unlock bootloader
  - Root it



# Android application testing

- Rooting your device means installing linux software that grants normal users root privileges
  - Allows you to bypass security
  - Is different for each device
  - May break your device and will void your warranty!



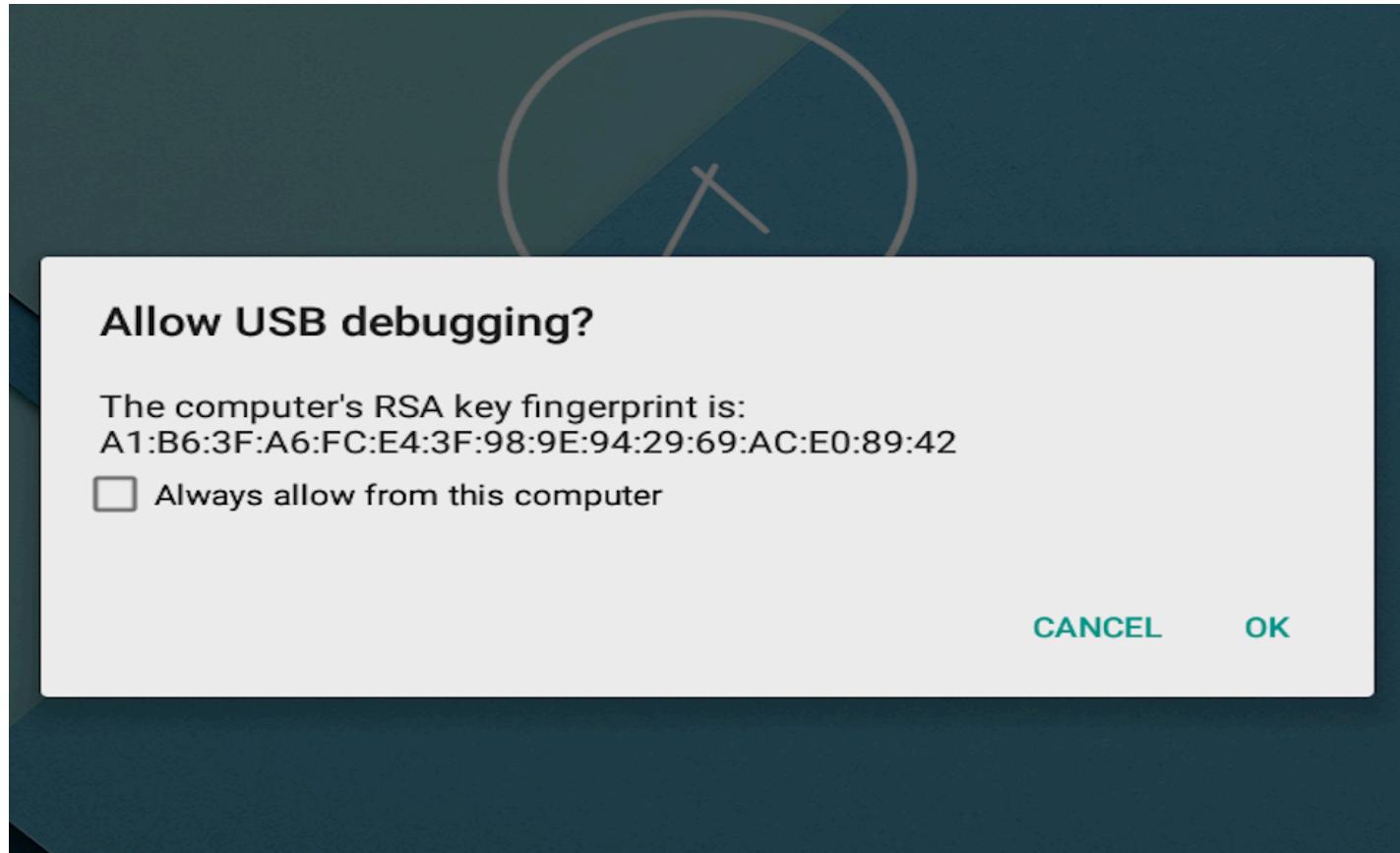
# Android application testing

- ADB (Android Debugging Bridge) is one of the first tools to use
- Is part of the platformtools part of the SDK

```
$ adb devices
List of devices attached
emulator-5554    device
```

- If you connect with ADB for the first time, you need to accept the certificate of the computer on the device

# Android application testing



# Android application testing

## Android boot process:

- The boot process is essential in securing Android.
- If you can intervene you can inject and manipulate the Android Platform.
- (*This is actually necessary when rooting an Android device*).
- For Nexus 7 (2012): <http://www.ibtimes.co.uk/how-root-nexus-7-2012-wi-fi-nexus-10-stock-android-5-0-lollipop-via-cf-auto-root-1474771>

# Android application testing

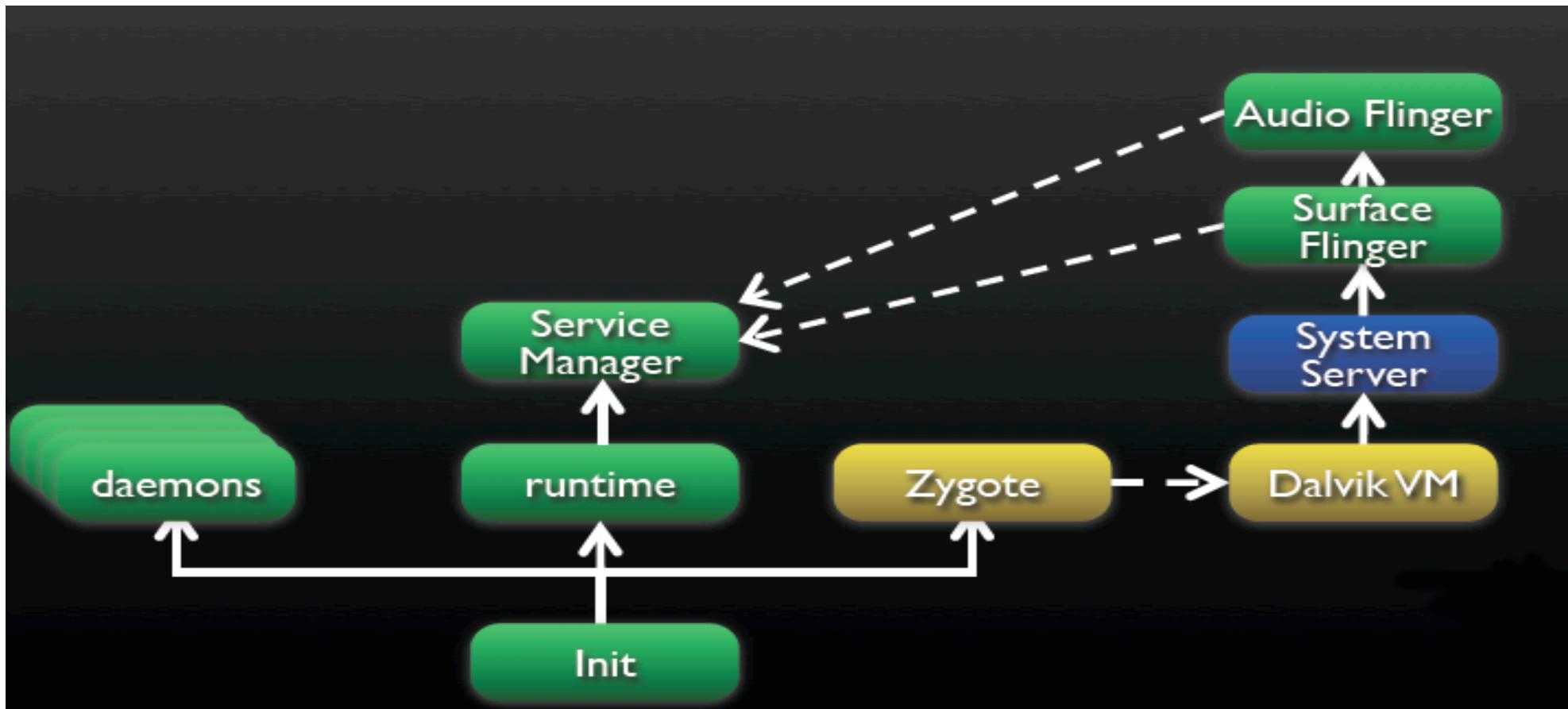
Android boot process:

1. BIOS starts the device (hardware).
2. The BIOS loads the kernel.
3. The kernel creates the first process:
  - Init
  - Mount directories, reads init.rc and other startup files
4. init will start other processes (adb, vold etc.)
  - In /system there is a build.prop file that also contains additional startup and device information

# Android application testing

- Android boot process
  - Once all system processes are started, init will startup Zygote
  - Zygote is responsible for creating Dalvik Virtual machines and loading libraries etc.
  - Applications are spawned from Zygote

# Android application testing



# Android application testing

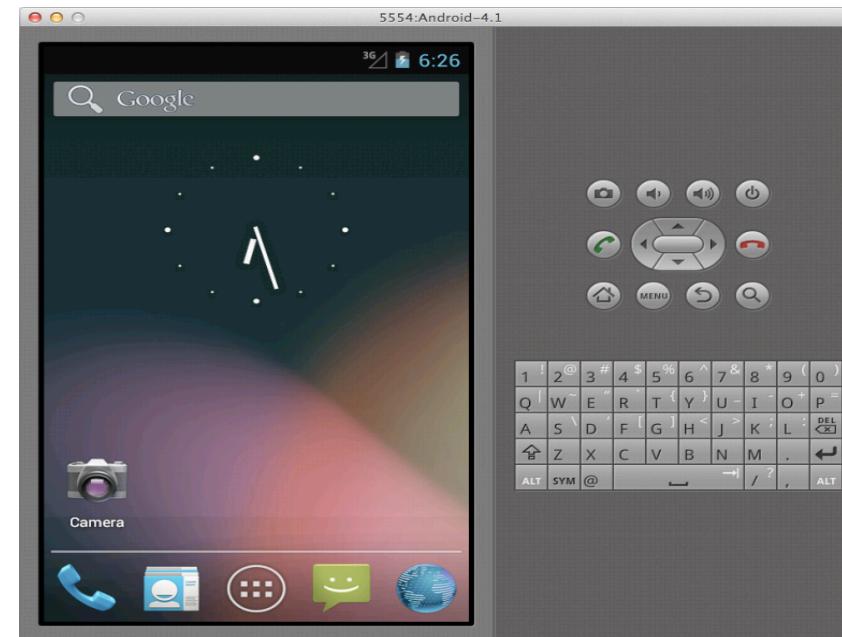
- To create your testing environment (examples on Kali):
  - Install the android-sdk (make sure to use the proper java version!):
    - apt-get install openjdk-7-jdk
    - apt-get install android-sdk
  - If you're on a 64 bit machine (older kali):
    - apt-get install lib32ncurses5 lib32stdc++6

# Android application testing

- Use android manager to install platform-tools and the SDK at least
- On Kali this is all installed in /usr/share/android-sdk
- Important subdirectories: tools and platform-tools
- Add them to you PATH so you can find all the commands
- Don't forget to select the proper java if you have more installed:
  - root@kali:~# update-java-alternatives -l
  - java-1.6.0-openjdk-amd64 1061 /usr/lib/jvm/java-1.6.0-openjdk-amd64
  - java-1.7.0-openjdk-amd64 1051 /usr/lib/jvm/java-1.7.0-openjdk-amd64

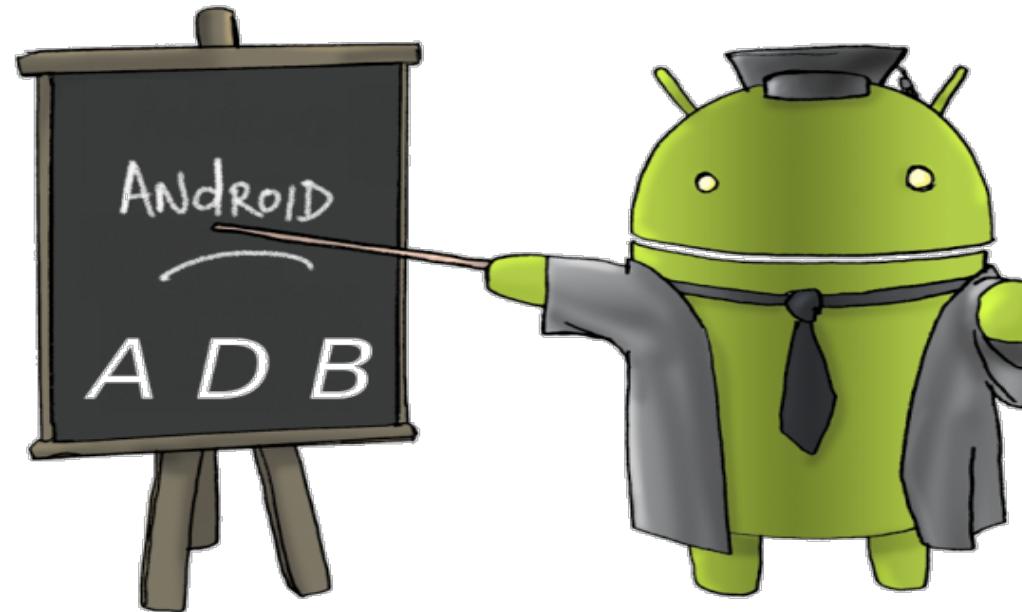
# Android application testing

- You can also install Android emulators and virtual devices.
- In my experience this does not provide:
  - a reliable test environment
  - better to buy a Nexus phone and use that as a testbed.



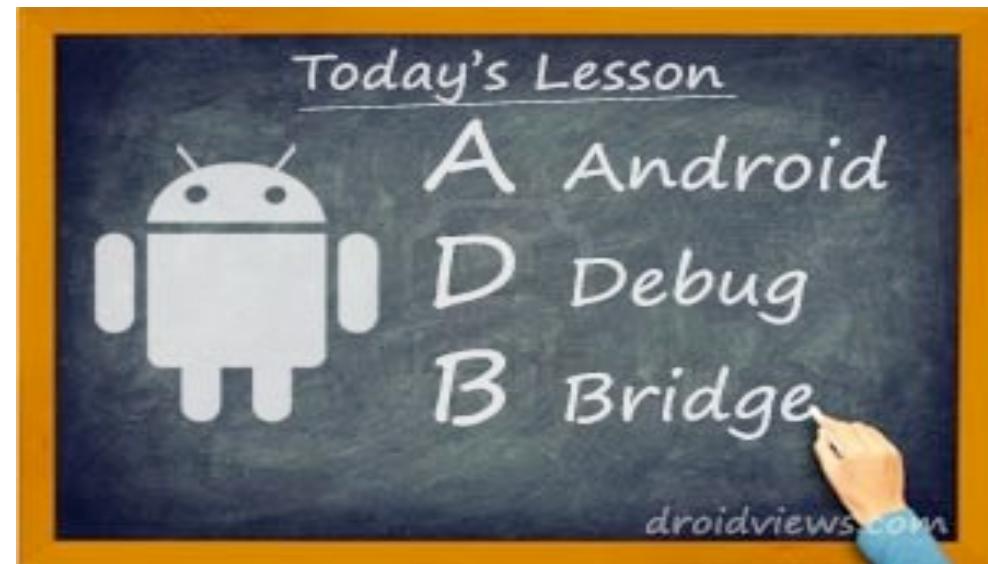
# Android application testing

- adb really is your friend....

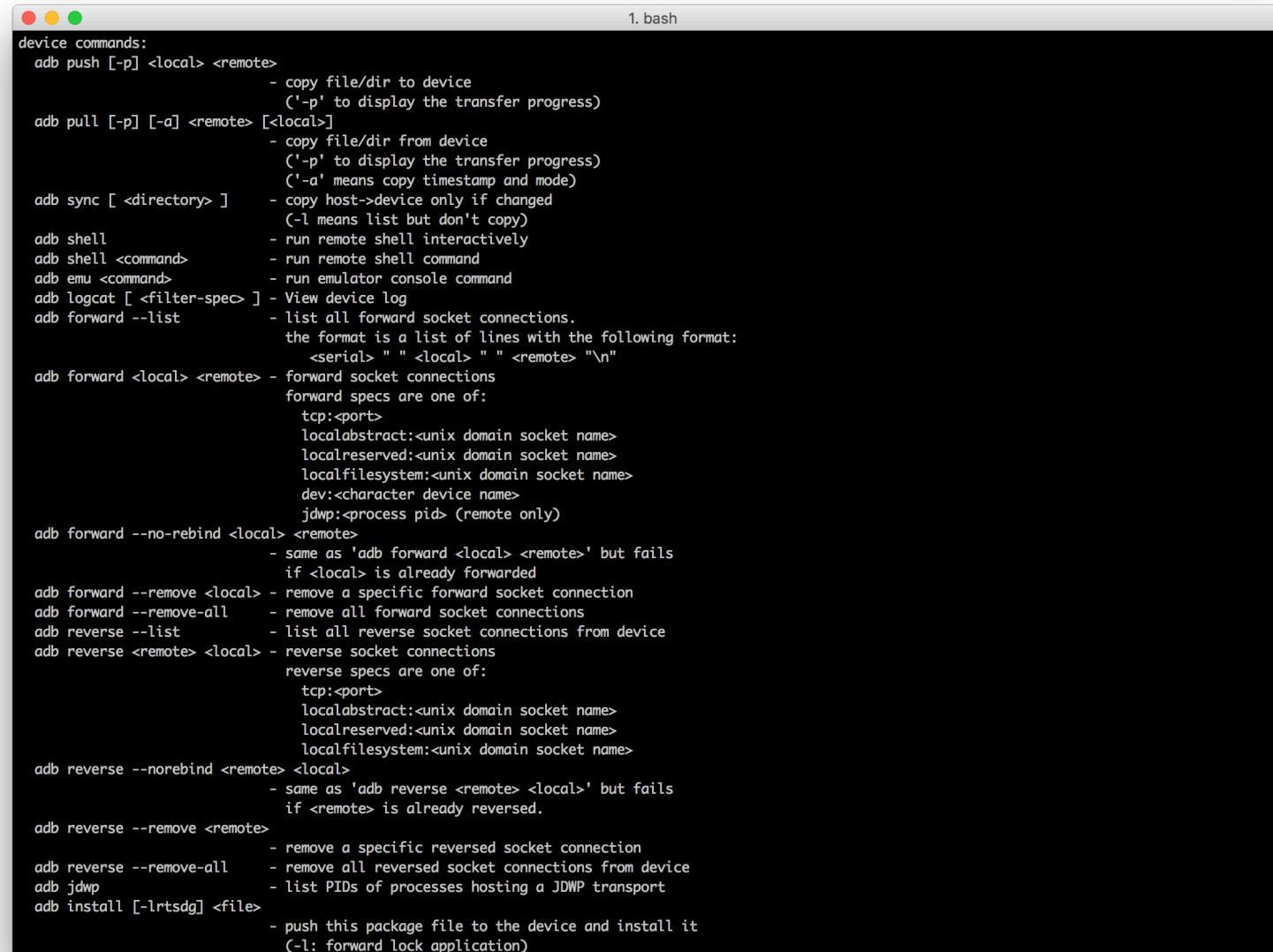


# Android application testing

- adb push and pull allow you to put and get files
- adb install allows you to install apk (package) files
- adb backup allows you to make backups
- Etc.



# Android application testing



The screenshot shows a terminal window titled "1. bash" displaying a list of Android Debug Bridge (adb) commands. The commands are organized into sections with descriptions:

- device commands:**
  - adb push [-p] <local> <remote>**: copy file/dir to device ('-p' to display the transfer progress)
  - adb pull [-p] [-a] <remote> [<local>]**: copy file/dir from device ('-p' to display the transfer progress) ('-a' means copy timestamp and mode)
  - adb sync [<directory> ]**: copy host->device only if changed (-l means list but don't copy)
  - adb shell**: run remote shell interactively
  - adb shell <command>**: run remote shell command
  - adb emu <command>**: run emulator console command
  - adb logcat [<filter-spec> ]**: View device log
  - adb forward --list**: list all forward socket connections.  
the format is a list of lines with the following format:  
<serial> " " <local> " " <remote> "\n"
  - adb forward <local> <remote>**: forward socket connections  
forward specs are one of:
    - tcp:<port>
    - localabstract:<unix domain socket name>
    - localreserved:<unix domain socket name>
    - localfilesystem:<unix domain socket name>
    - dev:<character device name>
    - jdwp:<process pid> (remote only)
  - adb forward --no-rebind <local> <remote>**: same as 'adb forward <local> <remotes>' but fails if <local> is already forwarded
  - adb forward --remove <local>**: remove a specific forward socket connection
  - adb forward --remove-all**: remove all forward socket connections
  - adb reverse --list**: list all reverse socket connections from device
  - adb reverse <remote> <local>**: reverse socket connections  
reverse specs are one of:
    - tcp:<port>
    - localabstract:<unix domain socket name>
    - localreserved:<unix domain socket name>
    - localfilesystem:<unix domain socket name>
  - adb reverse --norebind <remote> <local>**: same as 'adb reverse <remote> <local>' but fails if <remote> is already reversed.
  - adb reverse --remove <remote>**: remove a specific reversed socket connection
  - adb reverse --remove-all**: remove all reversed socket connections from device
  - adb jdwp**: list PIDs of processes hosting a JDWP transport
  - adb install [-lrtsgd] <file>**: push this package file to the device and install it (-l: forward lock application)

# Android application testing

- adb shell command allows you to access the device and browse around

```
File Edit View Search Terminal Help
u0_a44 3874 130 1308028 60676 ffffffff 00000000 S com.google.android.apps.docs.editor.s
des
u0_a70 3958 130 1392940 94000 ffffffff 00000000 S com.google.android.gm
u0_a32 4059 130 1292000 43140 ffffffff 00000000 S com.google.android.calendar
u0_a2 4085 130 1199008 29140 ffffffff 00000000 S com.android.providers.calendar
root 4331 2 0 0 ffffffff 00000000 S kworker/0:0
u0_a39 4332 130 1198876 25648 ffffffff 00000000 S com.google.android.deskclock
root 4670 2 0 0 ffffffff 00000000 S kworker/u:0
root 4680 2 0 0 ffffffff 00000000 S kworker/u:3
root 4681 2 0 0 ffffffff 00000000 S kworker/u:4
u0_a91 4693 130 1262780 50308 ffffffff 00000000 S com.safetyculture.iauditor
root 4713 2 0 0 ffffffff 00000000 S kworker/u:5
u0_a22 4808 130 1219204 37380 ffffffff 00000000 S com.android.systemui:screenshot
root 4838 2 0 0 ffffffff 00000000 S kworker/u:6
root 5000 2 0 0 ffffffff 00000000 S kworker/0:2
system 5005 130 1196424 26912 ffffffff 00000000 S android:ui
u0_a40 5024 130 1217592 44036 ffffffff 00000000 S com.android.documentsui
u0_a6 5043 130 1196428 25476 ffffffff 00000000 S com.android.externalstorage
root 5286 2 0 0 ffffffff 00000000 S flush-0:20
shell 5406 133 1116 528 c0076820 40412b80 S /system/bin/sh
root 5412 2 0 0 ffffffff 00000000 S migration/1
root 5413 2 0 0 ffffffff 00000000 S kworker/1:0
root 5414 2 0 0 ffffffff 00000000 S ksoftirqd/1
root 5415 2 0 0 ffffffff 00000000 S watchdog/1
root 5416 2 0 0 ffffffff 00000000 S migration/2
root 5417 2 0 0 ffffffff 00000000 S kworker/2:0
root 5418 2 0 0 ffffffff 00000000 S ksoftirqd/2
root 5419 2 0 0 ffffffff 00000000 S watchdog/2
root 5420 2 0 0 ffffffff 00000000 S migration/3
root 5421 2 0 0 ffffffff 00000000 S kworker/3:0
root 5422 2 0 0 ffffffff 00000000 S ksoftirqd/3
root 5423 2 0 "the quietest sound you are able to hear"
root 5424 2 0 0 ffffffff 00000000 S kworker/1:1
root 5425 2 0 0 ffffffff 00000000 S kworker/3:1
shell 5426 5406 2424 716 00000000 403a03c0 R ps
root 5427 2 0 0 ffffffff 00000000 S kworker/2:1
shell@grouper:/ $ █
```

# Android application testing

```
root@kali: ~
File Edit View Search Terminal Help
init
init.environ.rc
init.grouper.rc
init.grouper.usb.rc
init.rc
init.trace.rc
init.usb.rc
init.zygote32.rc
mnt
proc
property_contexts
root
sbin
sdcard
seapp_contexts
selinux_version
sepolicy
service_contexts
storage
sys
system
ueventd.grouper.rc
ueventd.rc
vendor
shell@grouper:/ $ ls
```

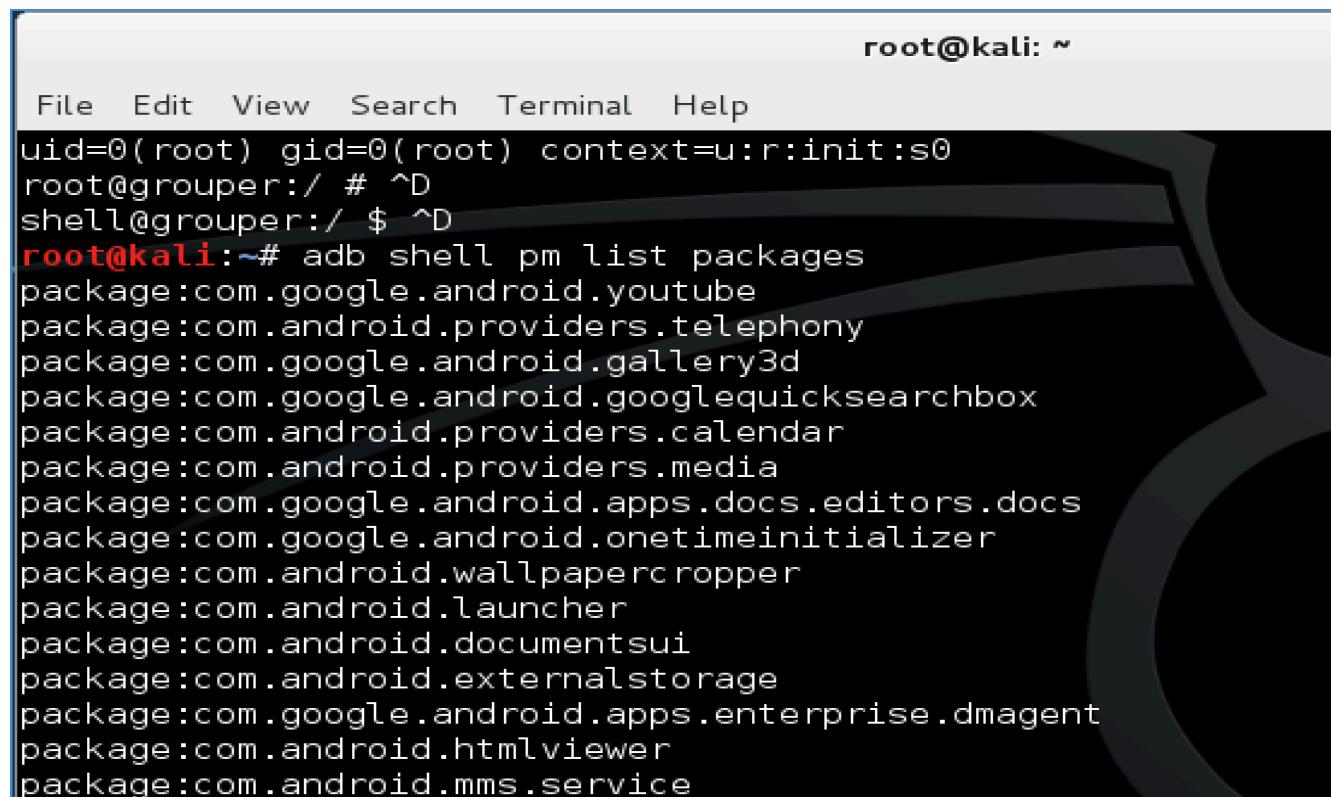
# Android application testing

- Once USB debugging is activated, and the phone is rooted, interesting things may happen...
- Passwords and gestures are stored in /data/system

```
usagestats
users
shell@grouper:/data/system $ ls -l password.key
-rw----- system system 72 2015-05-09 19:41 password.key
shell@grouper:/data/system $
```

# Android application testing

- List installed packages: adb shell pm list packages



```
root@kali: ~
File Edit View Search Terminal Help
uid=0(root) gid=0(root) context=u:r:init:s0
root@grouper:/ # ^D
shell@grouper:/ $ ^D
root@kali:~# adb shell pm list packages
package:com.google.android.youtube
package:com.android.providers.telephony
package:com.google.android.gallery3d
package:com.google.android.googlequicksearchbox
package:com.android.providers.calendar
package:com.android.providers.media
package:com.google.android.apps.docs.editors.docs
package:com.google.android.onetimeinitializer
package:com.android.wallpapercropper
package:com.android.launcher
package:com.android.documentsui
package:com.android.externalstorage
package:com.google.android.apps.enterprise.dmagent
package:com.android.htmlviewer
package:com.android.mms.service
```

# Android application testing

- Look at the logfiles
  - adb logcat -d -f /mnt/sdcard/logcats.log

```
shell@grouper:/ $ ^D
root@kali:~# adb logcat -d -f /mnt/sdcard/logcats.log
root@kali:~# adb shell
shell@grouper:/ $ cd /mnt/sdcard
shell@grouper:/mnt/sdcard $ ls -l
drwxrwx--- root      sdcard_r          2015-05-09 17:21 Alarms
drwxrwx--x root      sdcard_r          2015-06-13 17:16 Android
drwxrwx--- root      sdcard_r          2015-05-09 19:18 DCIM
drwxrwx--- root      sdcard_r          2015-06-13 16:26 Download
drwxrwx--- root      sdcard_r          2015-05-10 17:23 MifareClassicTool
drwxrwx--- root      sdcard_r          2015-05-09 17:21 Movies
drwxrwx--- root      sdcard_r          2015-05-09 17:21 Music
drwxrwx--- root      sdcard_r          2015-05-09 19:06 Notifications
drwxrwx--- root      sdcard_r          2015-05-09 19:14 Pictures
drwxrwx--- root      sdcard_r          2015-05-09 17:21 Podcasts
drwxrwx--- root      sdcard_r          2015-05-09 17:21 Ringtones
-rw-rw---- root      sdcard_r 487863 2015-06-13 18:14 logcats.log
shell@grouper:/mnt/sdcard $
```

# Android application testing

- Modifying the app to be able to test it, but you run into some obstacles:
  - Runtime signature verification
  - Runtime rooting detection
  - Runtime debuggable detection
  - Certificate pinning

# Android application testing

- To intercept HTTP(S) traffic, use burpsuite, similar to iOS...
- However...
  - On Android it is not as easy to set a system wide proxy, better use the transparent proxy, Access Point and redirection technique.

# Android application testing

- Another great tool you need is apktool
  - This tool is used to manage Android install packages (apk files)
  - Current version on Kali is 1.5.2-kali or 2.0.0
  - New 2.0 version is available at <https://ibotpeaches.github.io/Apktool/>.
  - This version also support Android 5 (lollipop) and higher

# Android application testing

```
root@kali:~# cp Downloads/apktool_2.0.0.jar apktool2.jar
root@kali:~# java -jar apktool2.jar
Apktool v2.0.0 - a tool for reengineering Android apk files
with smali v2.0.5 and baksmali v2.0.5
Copyright 2014 Ryszard Wiśniewski <brut.alll@gmail.com>
Updated by Connor Tumbleson <connor.tumbleson@gmail.com>

usage: apktool
  -advance,--advanced      prints advance information.
  -version,--version       prints the version then exits
usage: apktool if|install-framework [options] <framework.apk>
  -p,--frame-path <dir>   Stores framework files into <dir>.
  -t,--tag <tag>          Tag frameworks using <tag>.
usage: apktool d[ecode] [options] <file_apk>
  -f,--force               Force delete destination directory.
  -o,--output <dir>        The name of folder that gets written. Default is apk.out
  -p,--frame-path <dir>    Uses framework files located in <dir>.
  -r,--no-res              Do not decode resources.
  -s,--no-src              Do not decode sources.
  -t,--frame-tag <tag>    Uses framework files tagged by <tag>.
usage: apktool b[uild] [options] <app_path>
  -f,--force-all           Skip changes detection and build all files.
  -o,--output <dir>         The name of apk that gets written. Default is dist/name.apk
  -p,--frame-path <dir>    Uses framework files located in <dir>.

For additional info, see: http://ibotpeaches.github.io/Apktool/
For smali/baksmali info, see: http://code.google.com/p/smali/
root@kali:~#
```

# Android application testing

- Handles apk files, unzips them and converts dex files to smali

```
root@kali:~/Desktop/Android# java -jar apktool2.jar d hacmeandroid_v1.0.apk
I: Using Apktool 2.0.0 on hacmeandroid_v1.0.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /root/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

# Android application testing

- Expanded directory looks like this

```
root@kali:~/Desktop/Android# mv hacmeandroid_saved hacmeandroid
root@kali:~/Desktop/Android# cd hacmeandroid_v1.0
root@kali:~/Desktop/Android/hacmeandroid_v1.0# ls -l
total 20
-rw-r--r-- 1 root root 1365 Jun 13 18:29 AndroidManifest.xml
-rw-r--r-- 1 root root 268 Jun 13 18:29 apktool.yml
drwxr-xr-x 3 root root 4096 Jun 13 18:29 original
drwxr-xr-x 6 root root 4096 Jun 13 18:29 res
root@kali:~/Desktop/Android/hacmeandroid_v1.0# █
```

# Android application testing

- AndroidManifest.xml

```
drwxr-xr-x 8 root root 4096 Jun 13 18:29 res
root@kali:~/Desktop/Android/hacmeandroid_v1.0# cat AndroidManifest.xml
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.and">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:label="@string/app_name" android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:label="Funds Transfer" android:name=".FundsTransferActivity"/>
        <activity android:label="Transaction History" android:name=".TransactionHistoryactivity"
        "/>
        <activity android:label="Profile update" android:name=".profileupdateactivity"/>
        <activity android:label="Account Summary" android:name=".accountsummaryactivity"/>
        <activity android:label="login screen1" android:name=".loginactivity"/>
        <activity android:label="Remember me" android:name=".loginactivity2"/>
        <activity android:label="Home" android:name=".homeActivity"/>
        <activity android:label="Proxy Setting" android:name=".Proxysetting"/>
    </application>
</manifest>
root@kali:~/Desktop/Android/hacmeandroid_v1.0#
```

# Android application testing

- Smali files

```
root@kali: ~/Desktop/Android/hacmeandroid_v1.0/smali/com
File Edit View Search Terminal Help
-rw-r--r-- 1 root root 2081 Jun 13 18:29 AccountSummaryResponse.smali
-rw-r--r-- 1 root root 869 Jun 13 18:29 FundsTransferResponse.smali
-rw-r--r-- 1 root root 1919 Jun 13 18:29 Loginresponse.smali
-rw-r--r-- 1 root root 3013 Jun 13 18:29 ProfileupdateiResponse.smali
-rw-r--r-- 1 root root 869 Jun 13 18:29 ProfileupdateResponse.smali
-rw-r--r-- 1 root root 1731 Jun 13 18:29 TransactionHistoryResponse.smali
-rw-r--r-- 1 root root 2496 Jun 13 18:29 TransactionModified.smali

./and/service:
total 60
-rw-r--r-- 1 root root 5636 Jun 13 18:29 AccountSummaryService.smali
-rw-r--r-- 1 root root 6663 Jun 13 18:29 FundsTransferService.smali
-rw-r--r-- 1 root root 12530 Jun 13 18:29 Loginservice.smali
-rw-r--r-- 1 root root 5808 Jun 13 18:29 Profileupdatelservice.smali
-rw-r--r-- 1 root root 8508 Jun 13 18:29 Profileupdateservice.smali
-rw-r--r-- 1 root root 6369 Jun 13 18:29 TransactionHistoryService.smali

./and/util:
total 36
-rw-r--r-- 1 root root 19616 Jun 13 18:29 Base64Coder.smali
-rw-r--r-- 1 root root 3005 Jun 13 18:29 FileUtils.smali
-rw-r--r-- 1 root root 6837 Jun 13 18:29 GetResponseBody.smali
-rw-r--r-- 1 root root 2338 Jun 13 18:29 Globals.smali

./ssl:
total 4
drwxr-xr-x 2 root root 4096 Jun 13 18:29 bypass

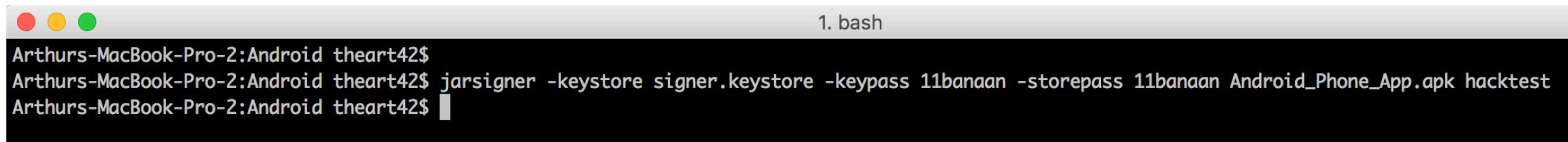
./ssl/bypass:      "the quieter you become, the more you are able to h
total 24
-rw-r--r-- 1 root root 2901 Jun 13 18:29 BypassedHttpClient.smali
-rw-r--r-- 1 root root 8058 Jun 13 18:29 EasySSLSocketFactory.smali
-rw-r--r-- 1 root root 4270 Jun 13 18:29 EasyX509TrustManager.smali
-rw-r--r-- 1 root root 3700 Jun 13 18:29 MyHttpClient.smali
root@kali:~/Desktop/Android/hacmeandroid_v1.0/smali/com#
```

# Android application testing

- Smali files can easily be changes, recompiled and repackaged using apktool
  - This creates a new .apk (modified) and this can be installed and used
- 32 bit vs 64 bit problems will be encountered in your Java environment!!

# Android application testing

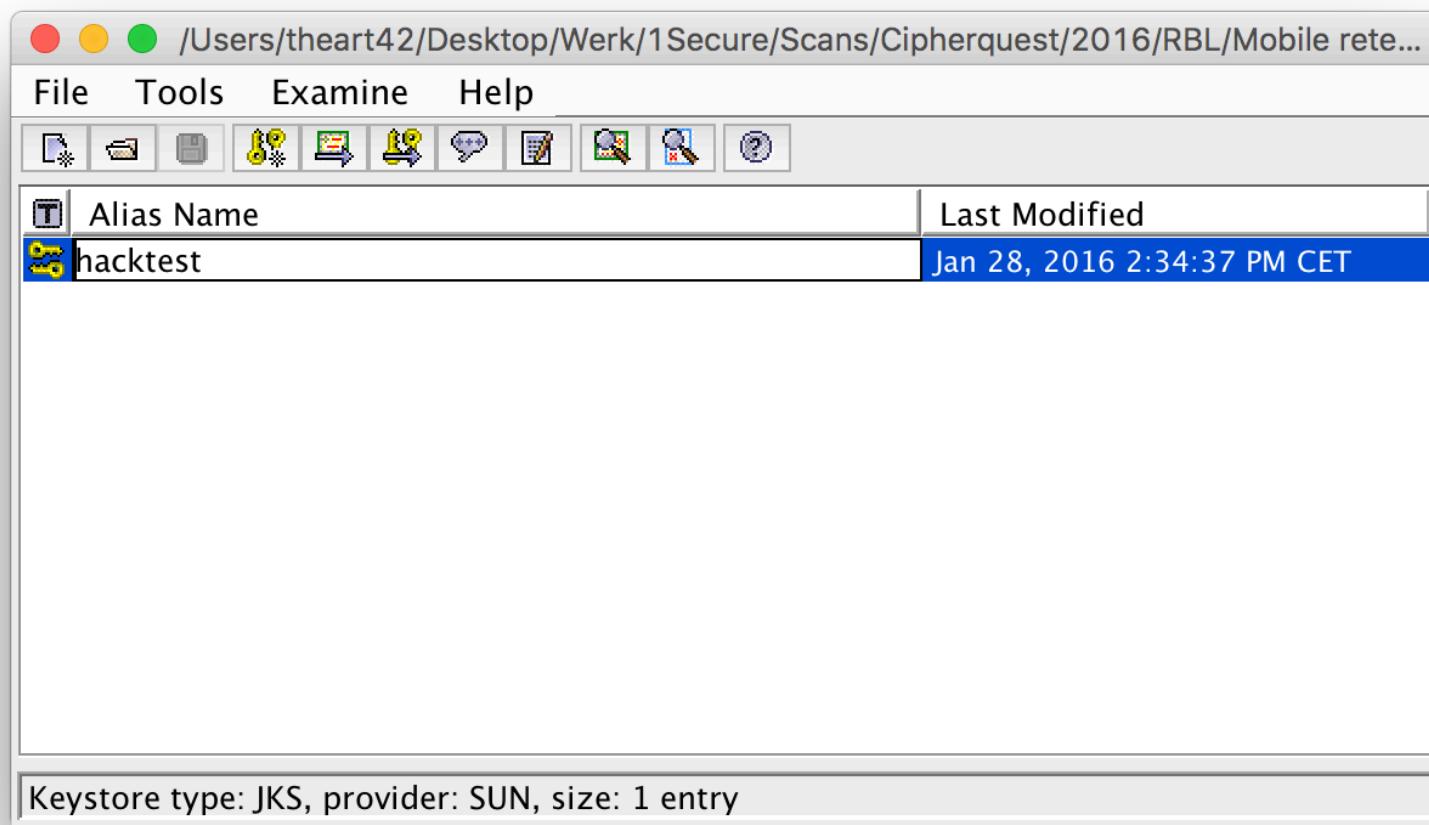
- Application signing
  - Any application developer can create his or her own (self-signed) certificate to sign a package file.
  - Keep this certificate as you need the same certificate for all your applications.



```
1. bash
Arthurs-MacBook-Pro-2:Android theart42$ jarsigner -keystore signer.keystore -keypass 11banaan -storepass 11banaan Android_App.apk hacktest
Arthurs-MacBook-Pro-2:Android theart42$
```

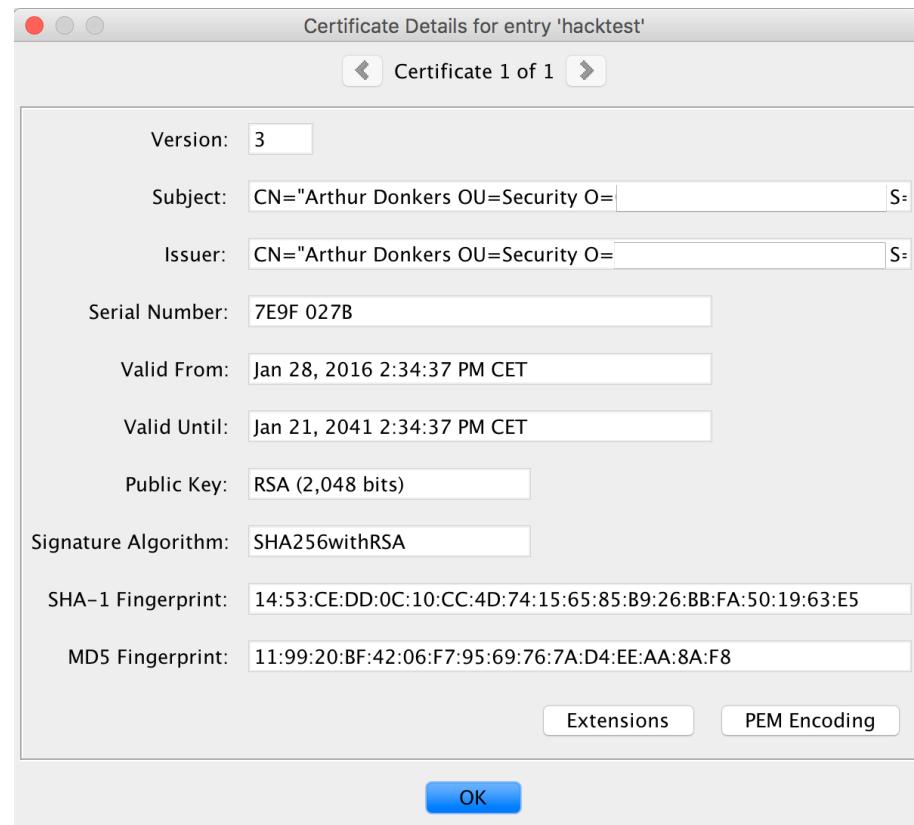
# Android application testing

- You can create keystores with different programs (e.g. Portecle)



# Android application testing

- You can create keystores with different programs



# Android application testing

- Application signing
  - If you want to check the certificate of an installed apk:

```
$ unzip testing.apk  
$ cd META-INF  
$ openssl pkcs7 -in CERT.RSA -print_certs -inform DER -out out.cer  
$ cat out.cer
```

# Android application testing

```
.method public constructor <init>()V
    .locals 1

    invoke-direct {p0}, Landroid/app/Activity;-><init>()V

    const/4 v0, 0x1

    input-boolean v0, p0, Lsrc/com/rbl/controller/SplashActivity;->j:Z

    const-string v0, "SHA-512"

    input-object v0, p0, Lsrc/com/rbl/controller/SplashActivity;->s:Ljava/lang/String;

    sget-object v0, Lcom/infosys/core/a;->p:Ljava/lang/String;

    input-object v0, p0, Lsrc/com/rbl/controller/SplashActivity;->h:Ljava/lang/String;

    const-string v0, "5b424061393532376332387f2decaf80ec8dab5e1bbf50ded7fe1060ba280e0511c825c77df7448a0e1d0ce853738fa187d22b37f482b9db8206646ef79e863225c101fa
1314a236d361aa"

    input-object v0, p0, Lsrc/com/rbl/controller/SplashActivity;->t:Ljava/lang/String;

    return-void
.end method
```

# Android application testing

```
invoke-virtual {v0, v1, v2}, Landroid/content/pm/PackageManager;->getPackageInfo(Ljava/lang/String;I)Landroid/content/pm/PackageInfo;
move-result-object v0

iget-object v0, v0, Landroid/content/pm/PackageInfo;->signatures:[Landroid/content/pm/Signature;
array-length v1, v0
if-eqz v1, :cond_1
const/4 v1, 0x0
aget-object v0, v0, v1
const-string v1, "SHA-512"
invoke-static {v1}, Ljava/security/MessageDigest;->getInstance(Ljava/lang/String;)Ljava/security/MessageDigest;
move-result-object v1

invoke-virtual {v0}, Landroid/content/pm/Signature;->toByteArray()[B
move-result-object v0

invoke-virtual {v1, v0}, Ljava/security/MessageDigest;->update([B)V
invoke-virtual {v1}, Ljava/security/MessageDigest;->digest()[B
move-result-object v0
const/4 v1, 0x0
invoke-static {v0, v1}, Landroid/util/Base64;->encodeToString([BI)Ljava/lang/String;
move-result-object v0

iget-object v1, p0, Lsrc/com/rbl/controller/SplashActivity;->t:Ljava/lang/String;
invoke-static {v0, v1}, Lsrc/com/rbl/controller/SplashActivity;->c(Ljava/lang/String;Ljava/lang/String;)Z
move-result v0
#    if-nez v0, :cond_1
```

# Android application testing

- Detection of debug flag

```
.method public a(Ljava/lang/String;Ljava/util/List;)Ljava/util/List;
.locals 11

    igure-object v0, p0, Lsrc/com/rbl/uicontroller/ek;->a:Ljava/lang/String;

    const-string v1, "here0"

    invoke-static {v0, v1}, Landroid/util/Log;->d(Ljava/lang/String;Ljava/lang/String;)I

    igure-object v0, p0, Lsrc/com/rbl/uicontroller/ek;->a:Ljava/lang/String;

    invoke-static {v0, p1}, Landroid/util/Log;->d(Ljava/lang/String;Ljava/lang/String;)I

    sget-object v0, Lcom/infosys/core/a;->ch:Landroid/app/Activity;

    invoke-virtual {v0}, Landroid/app/Activity;->getApplicationInfo()Landroid/content/pm/ApplicationInfo;

    move-result-object v0

    igure v0, v0, Landroid/content/pm/ApplicationInfo;->flags:I

    and-int/lit8 v0, v0, 0x2

#   if-eqz v0, :cond_0
    goto :cond_0
```

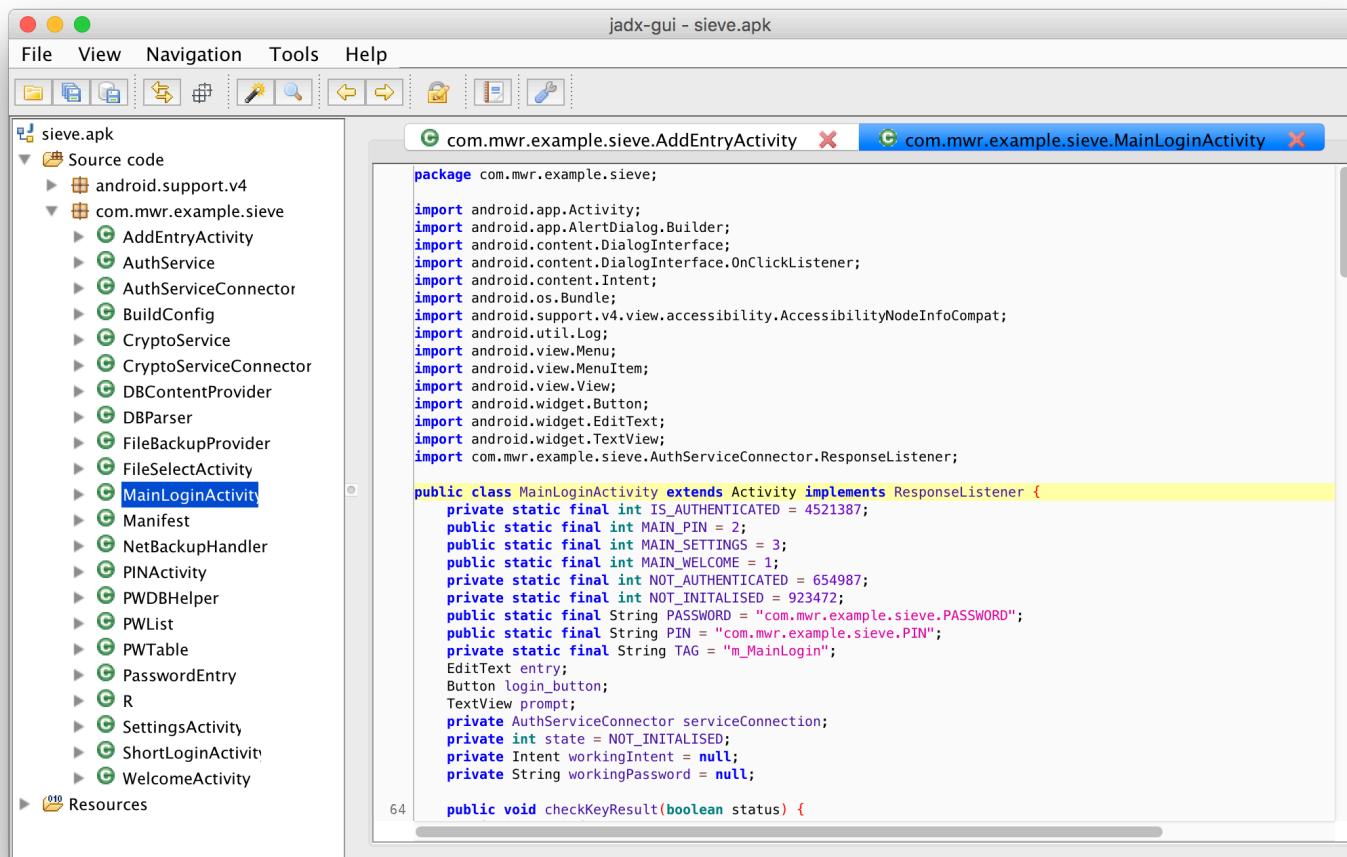
# Android application testing

- Another option is dex2jar and jd-gui to decompile to Java code

```
root@kali:~/Desktop/Android# dex2jar classes.dex
this cmd is deprecated, use the d2j-dex2jar if possible
dex2jar version: translator-0.0.9.15
dex2jar classes.dex -> classes_dex2jar.jar
Done.
root@kali:~/Desktop/Android# ls -l
total 36156
-rw-r--r-- 1 root root 6344307 Jun 13 18:24 apktool2.jar
-rw-r--r-- 1 root root 62492 Mar 29 2012 classes.dex
-rw-r--r-- 1 root root 68085 Jun 13 19:00 classes_dex2jar.jar
drwxr-xr-x 6 root root 4096 Jun 13 18:51 hacmeandroid_v1.0
-rw-r--r-- 1 root root 255117 Jun 13 18:50 hacmeandroid_v1.0.apk
drwxr-xr-x 6 root root 4096 Apr 5 2012 HacmeBank - Android v1.0
-rw-r--r-- 1 root root 30271149 Jun 13 18:27 hacmebank_android.zip
drwxr-xr-x 2 root root 4096 Jun 13 18:41 save
root@kali:~/Desktop/Android#
```

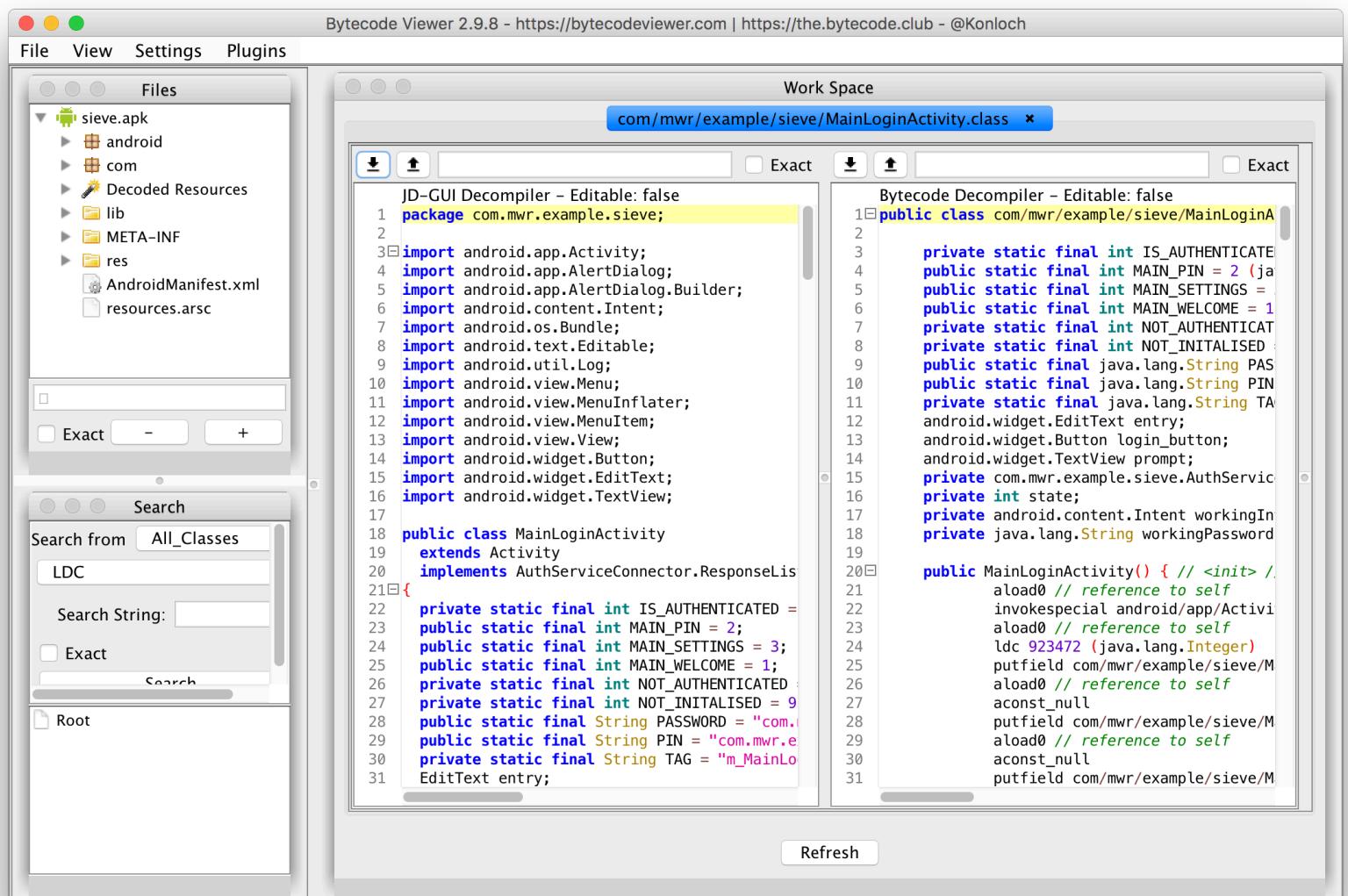
# Android application testing

- JADX is an integrated tool environment



# Android application testing

- Or ByteCodeViewer



# Android application testing

- Runtime rooting detection
  - This almost exclusively is done by looking for the su binary in a number of places on the device
  - Look for the string "su" to see where the existence of this file is checked
  - Sometimes the developer checks for other files as well
  - You have to disable all of these checks, may take a while

# Android application testing

- Once the code has been decompiled, look in the AndroidManifest.xml file for content providers. They will share application data!
- grep -R 'content://' AndroidManifest.xml.
- Each content provider is a potential dataleak point.
- adb shell content query --uri <url> will provide information on content.

# Android application testing

```
$ grep -R 'content://" .  
./NotePad$Media.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/media"  
./NotePad$Media.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/media_api_id"  
./NotePad$Media.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/media_silent_delete"  
./NotePad$Media.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/parent_note_of_media"  
./NotePad$Media.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/media_with_owner"  
./NotePad$Media.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/images_for_note"  
./NotePad$MediaNotes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/media_notes"  
./NotePad$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/topnotes"  
./NotePad$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/notes"  
./NotePad$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/notes_show_deleted"  
./NotePad$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/notes_silent"  
./NotePad$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/notes_silent_delete"  
./NotePad$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/notes_nodeid"  
./NotePad$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/notes_nodeid_silent"  
./NotePad$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/notes_nodeid_silent_delete"  
./NotePad$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/notes_with_images"  
./NotePad$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/add_media_for_note"  
./NotePad$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/add_media_for_note_silent"  
.NotePad.smali:.field public static final SCHEME:Ljava/lang/String; = "content://"  
.NotePad.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePad/wipe"  
.NotePadPending$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePadPending/notes"  
.NotePadPending$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePadPending/notes_nodeid"  
.NotePadPending$Notes.smali:    const-string v0, "content://com.threebanana.notes.provider.NotePadPending/notes_nodeid_parent"
```

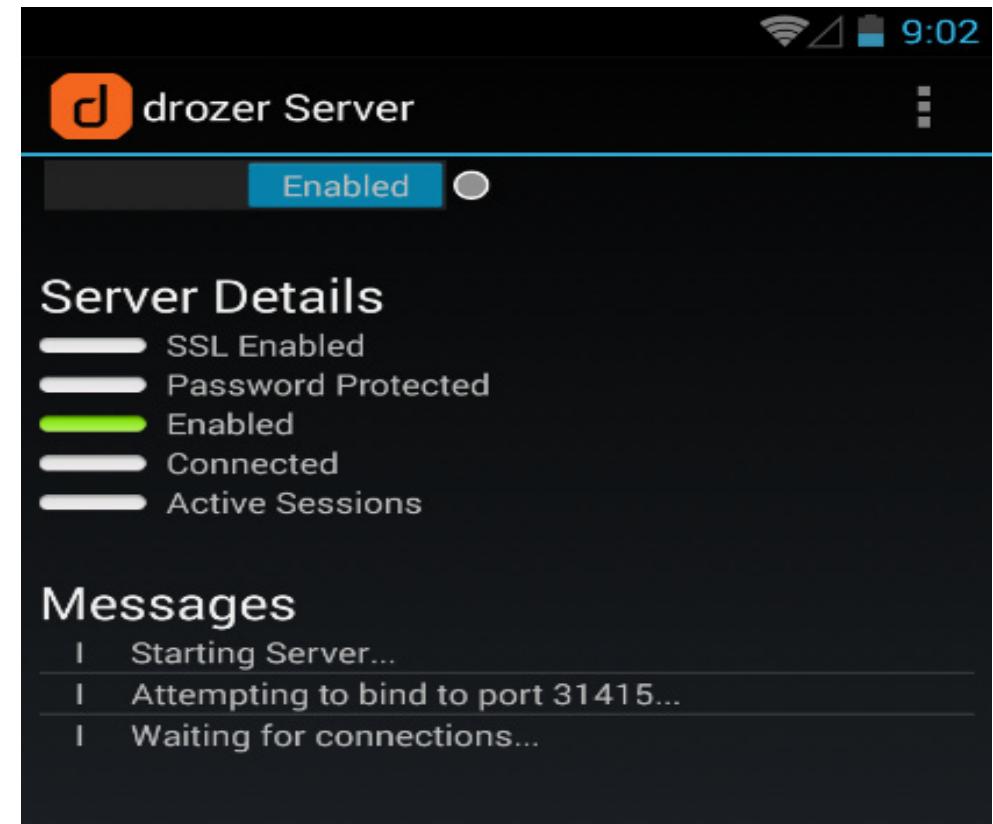


# Android application testing

- We can do this manually, or use Drozer:
  - <https://www.mwrinfosecurity.com/products/drozer/>
  - <https://github.com/mwrlabs/drozer>
- Drozer is a client server product that automates a lot of the testing.
- Uses an agent on the mobile device:
  - adb install agent.apk

# Android application testing

- Use adb to forward port 31415:
  - adb forward tcp:31415 tcp:31415
- Start application/agent on device
- Start drozer on system:
  - drozer console connect



# Android application testing

- Find content URI's

```
dz> run app.provider.finduri com.threebanana.notes
Scanning com.threebanana.notes...
content://com.threebanana.notes.provider.NotePad/notes
content://com.threebanana.notes.provider.NotePadPending/notes/
content://com.threebanana.notes.provider.NotePad/media
content://com.threebanana.notes.provider.NotePad/topnotes/
content://com.threebanana.notes.provider.NotePad/media_with_owner/
content://com.threebanana.notes.provider.NotePad/add_media_for_note
content://com.threebanana.notes.provider.NotePad/notes_show_deleted
content://com.threebanana.notes.provider.NotePad/notes_with_images/
```

# Android application testing

- Query content URI's

```
dz> run app.provider.query content://com.threebanana.notes.provider.NotePad/notes --vertical
      reminder 0
      hashcodes
      child_count 0
      altitude 0
      sharekey 0
      source_url null
      nodeid -1
      timestamp 1386319989407
      _id 1
      created 1386319989407
      longitude 0
      parent_nodeid -1
      api_pending_op 1
      parent_id -1
accuracy_altitude 0
      publicURL
      text This is a secret note
```

# Android application testing

- Other application issues:
  - Insecure file storage (wrong permissions)
  - File inclusion attacks
  - Client side injection attacks
- See OWASP top 10 mobile security

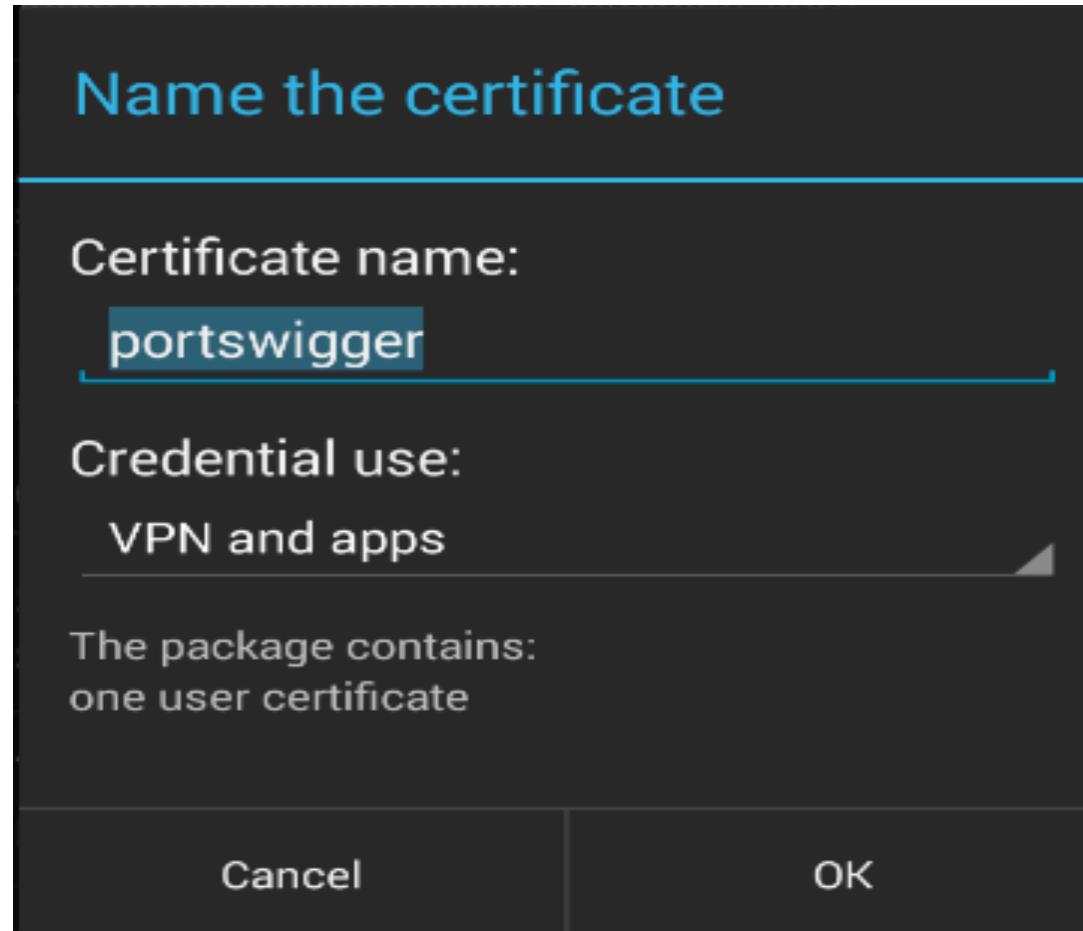
# Android application testing

- Traffic interception
  - Passive, using tcpdump-arm
  - [http://omappedia.org/wiki/USB\\_Sniffing\\_with\\_tcpdump](http://omappedia.org/wiki/USB_Sniffing_with_tcpdump)

# Android application testing

- Traffic interception
  - Active, using a malicious AP, Linux interception etc...
  - Intercepting HTTPS traffic requires installation of proxy CA.
  - adb push portswiggerca.crt /mnt/sdcard/portswiggerca.crt.
  - On device go to Settings -> Personal -> Security and install certificate from sdcard.

# Android application testing



# Android application testing

- Alternative ways to install certificates:
  - Pull the cacerts.bks file from /system/etc/security
  - Add certificate (using Bouncy Castle for instance)
  - Remount /system partition read/write
  - Restore cacerts.bks file

# Android application testing

- What about certificate pinning on Android?
  - The most common way involves a number of steps:
  - Read the server certificate from a file
  - Store it in a keystore object in your app
  - Initialize a TrustManager with this keystore
  - With this TrustManager you create an SSLContext and SSLSocketFactory
  - Use this SSLSocketFactory to make the connection to your server
  - The TrustManager verifies the server certificate against the keystore

# Android application testing

```
InputStream in = resources.openRawResource(certificateRawResource);

keyStore = KeyStore.getInstance("BKS");
keyStore.load(resourceStream, password);
```

# Android application testing

```
HttpParams httpParams = new BasicHttpParams();

SchemeRegistry schemeRegistry = new SchemeRegistry();
schemeRegistry.register(new Scheme("https", new SSLSocketFactory(keyStore), 443));

ThreadSafeClientConnManager clientMan = new ThreadSafeClientConnManager(httpParams, schemeRegistry);

httpClient = new DefaultHttpClient(clientMan, httpParams);
```

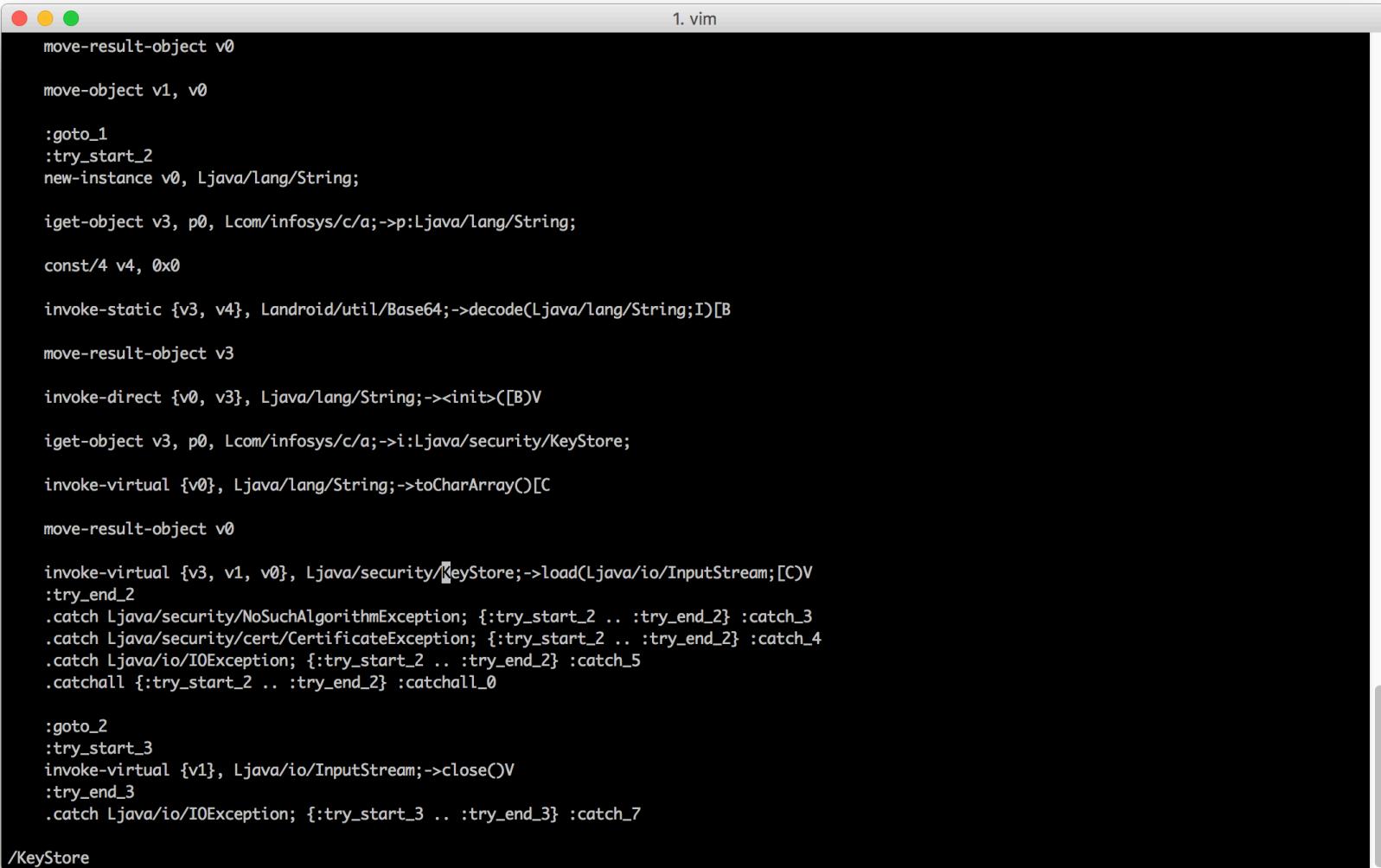
# Android application testing

- How to bypass?
  - Sometimes the code itself verifies the certificates, this check can be disabled in the smali file(s)
  - Often you need to replace the proper BKS keystore with one that contains your man-in-the-middle certificate
  - Challenge is to find the password for opening/modifying the keystore
    - Sometimes this password is stored in plain text in the code (DUMB!)
    - More often this password is encrypted/obfuscated
    - Use the Java Debugger to obtain the password

# Android application testing

- To get the password
  - Look through the code to find the initialization of the KeyStore object
  - Look for the KeyStore.load method and determine the parameters
  - This indicates which parameter contains the password
  - Make a note of the module, method and the parameter for later

# Android application testing



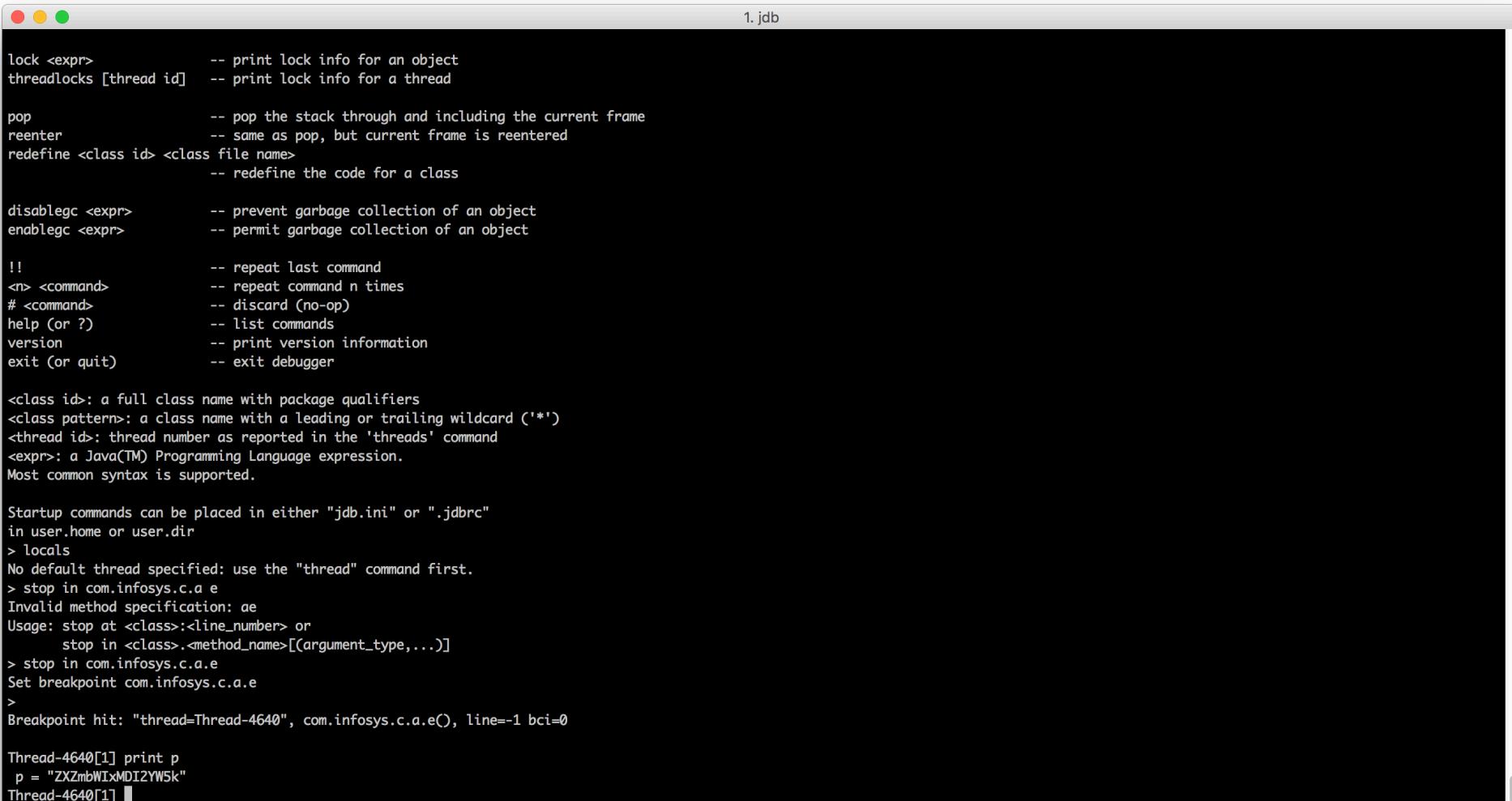
The screenshot shows a Vim window with the title "1. vim". The buffer contains Java bytecode disassembly. The code starts with a move operation on register v0, followed by a move-object operation. It then branches to a try\_start\_2 point, creates a new String object, and performs an iGet-object operation to get a string from a Base64 encoded input. This is followed by a const/4 operation and an invoke-static call to decode the string. The next move-result-object operation is on register v3. An invoke-direct call is made to the constructor of the String class. Following this, an iGet-object operation retrieves a KeyStore object from a static field. An invoke-virtual operation is then performed on the string object to convert it to a character array. Another move-result-object operation is on register v0. An invoke-virtual operation is then performed on the KeyStore object to load a certificate from an InputStream. This is followed by a series of catch clauses for NoSuchAlgorithmException, CertificateException, IOException, and a catchall clause. Finally, the code branches to try\_start\_3, performs an invoke-virtual operation on the input stream to close it, and handles an IOException in the catch\_start\_3 range.

```
move-result-object v0
move-object v1, v0
:goto_1
:try_start_2
new-instance v0, Ljava/lang/String;
idget-object v3, p0, Lcom/infosys/c/a;->p:Ljava/lang/String;
const/4 v4, 0x0
invoke-static {v3, v4}, Landroid/util/Base64;->decode(Ljava/lang/String;I)[B
move-result-object v3
invoke-direct {v0, v3}, Ljava/lang/String;-><init>([B)V
idget-object v3, p0, Lcom/infosys/c/a;->i:Ljava/security/KeyStore;
invoke-virtual {v0}, Ljava/lang/String;->toCharArray()[C
move-result-object v0
invoke-virtual {v3, v1, v0}, Ljava/security/KeyStore;->load(Ljava/io/InputStream;[C)V
:try_end_2
.catch Ljava/security/NoSuchAlgorithmException; {:try_start_2 .. :try_end_2} :catch_3
.catch Ljava/security/cert/CertificateException; {:try_start_2 .. :try_end_2} :catch_4
.catch Ljava/io/IOException; {:try_start_2 .. :try_end_2} :catch_5
.catchall {:try_start_2 .. :try_end_2} :catchall_0
:goto_2
:try_start_3
invoke-virtual {v1}, Ljava/io/InputStream;->close()V
:try_end_3
.catch Ljava/io/IOException; {:try_start_3 .. :try_end_3} :catch_7
/KeyStore
```

# Android application testing

- To get the password
  - Modify the AndroidManifest.XML file to enable debugging
  - Rebuild the APK, sign and install
  - Run the application
  - Use adb jdwp to find the portnumber where the app is listening for JDB
  - Forward this port: adb forward tcp:8000 tcp:<jdwp port>
  - JDB –attach localhost:8000
  - Set a breakpoint on the method in the module
  - Once it breaks, print the contents of the variable

# Android application testing



The screenshot shows a terminal window titled "1. jdb". The window displays the Java Debug (jdb) command help, which includes various commands and their descriptions. Below the help text, there is a series of command-line inputs and outputs related to setting breakpoints and printing variable values.

```
lock <expr>          -- print lock info for an object
threadlocks [thread id] -- print lock info for a thread

pop                  -- pop the stack through and including the current frame
reenter              -- same as pop, but current frame is reentered
 redefine <class id> <class file name>
                     -- redefine the code for a class

 disablegc <expr>    -- prevent garbage collection of an object
 enablegc <expr>     -- permit garbage collection of an object

 !!
<n> <command>        -- repeat command n times
# <command>           -- discard (no-op)
help (or ?)           -- list commands
version               -- print version information
exit (or quit)         -- exit debugger

<class id>; a full class name with package qualifiers
<class pattern>; a class name with a leading or trailing wildcard ('*')
<thread id>; thread number as reported in the 'threads' command
<expr>; a Java(TM) Programming Language expression.
Most common syntax is supported.

Startup commands can be placed in either "jdb.ini" or ".jdbrc"
in user.home or user.dir
> locals
No default thread specified: use the "thread" command first.
> stop in com.infosys.c.a e
Invalid method specification: ae
Usage: stop at <class>:<line_number> or
      stop in <class>.<method_name>[(argument_type,...)]
> stop in com.infosys.c.a.e
Set breakpoint com.infosys.c.a.e
>
Breakpoint hit: "thread=Thread-4640", com.infosys.c.a.e(), line=-1 bci=0

Thread-4640[1] print p
p = "ZXZmbWIxMDI2YW5k"
Thread-4640[1]
```

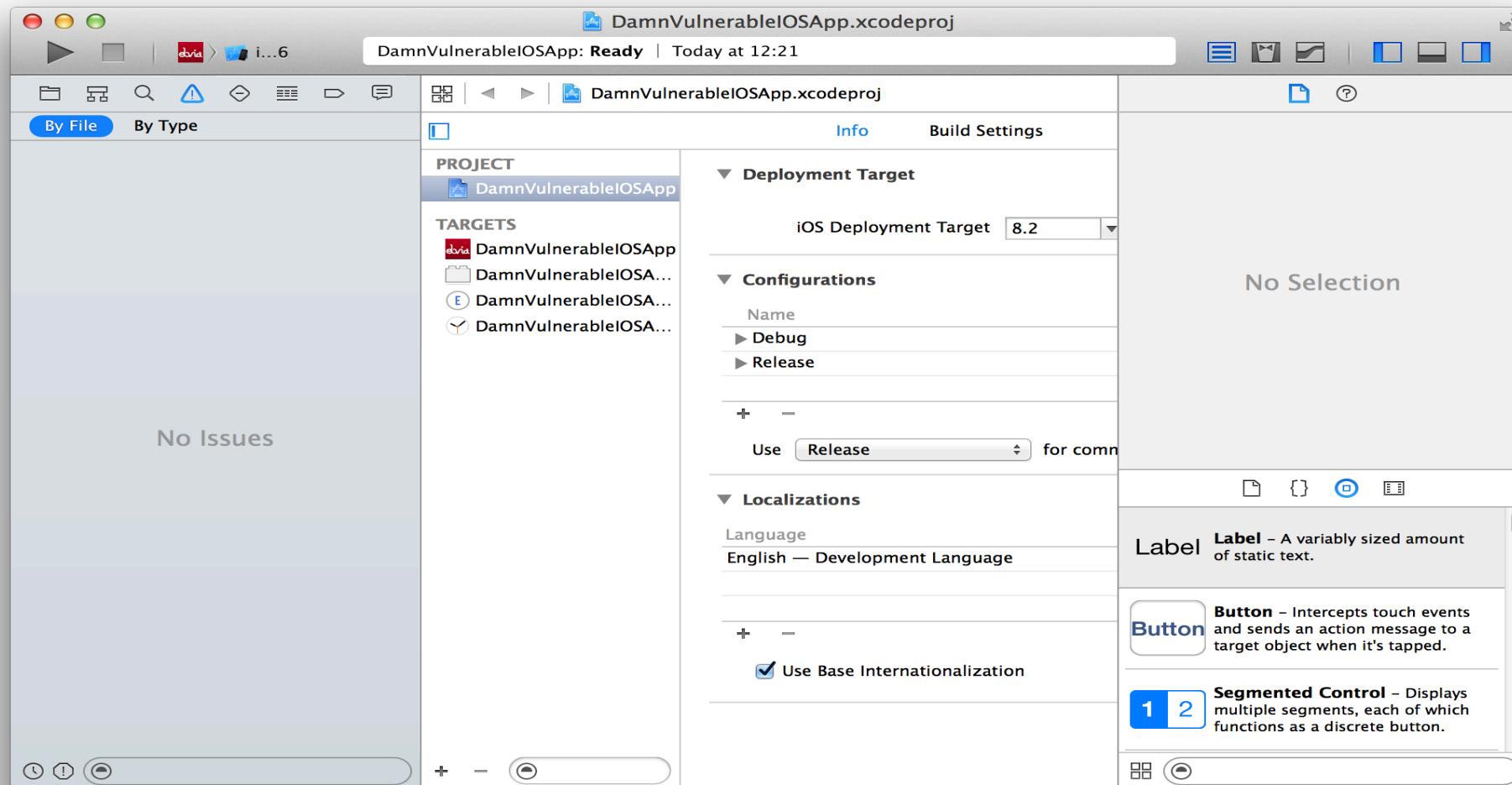
# iOS application testing



# iOS application testing

- Background information
- iOS has its roots in OS X, based on the Darwin OS, sources are available!
- Apps are written in Objective-C or Swift
- Xcode is the development environment, runs on OS X and provides a simulator
- <http://damnvulnerableiosapp.com/> Damn Vulnerable iOS application

# iOS application testing



# iOS application testing

- Objective-C is object oriented environment
- Applications are built on a collection of objects
- Objects exchange messages (as their way of method calling)
- Libraries are often called frameworks

# iOS application testing

- As objects exchange messages, all objects are active once an application has started
- If you can send a message to an object, you can bypass the regular flow

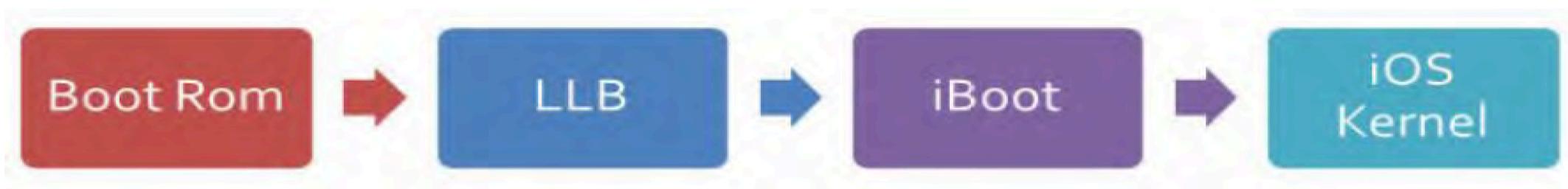
```
@implementation Classname
@synthesize blorg; // generates set/get methods
@synthesize gurgle;

Instance *myInstance = [[Instance alloc] init];

[myInstance setGurgle:@"foo"]; // infix notation
myInstance.gurgle = @"foo"; // dot notation
```

# iOS application testing

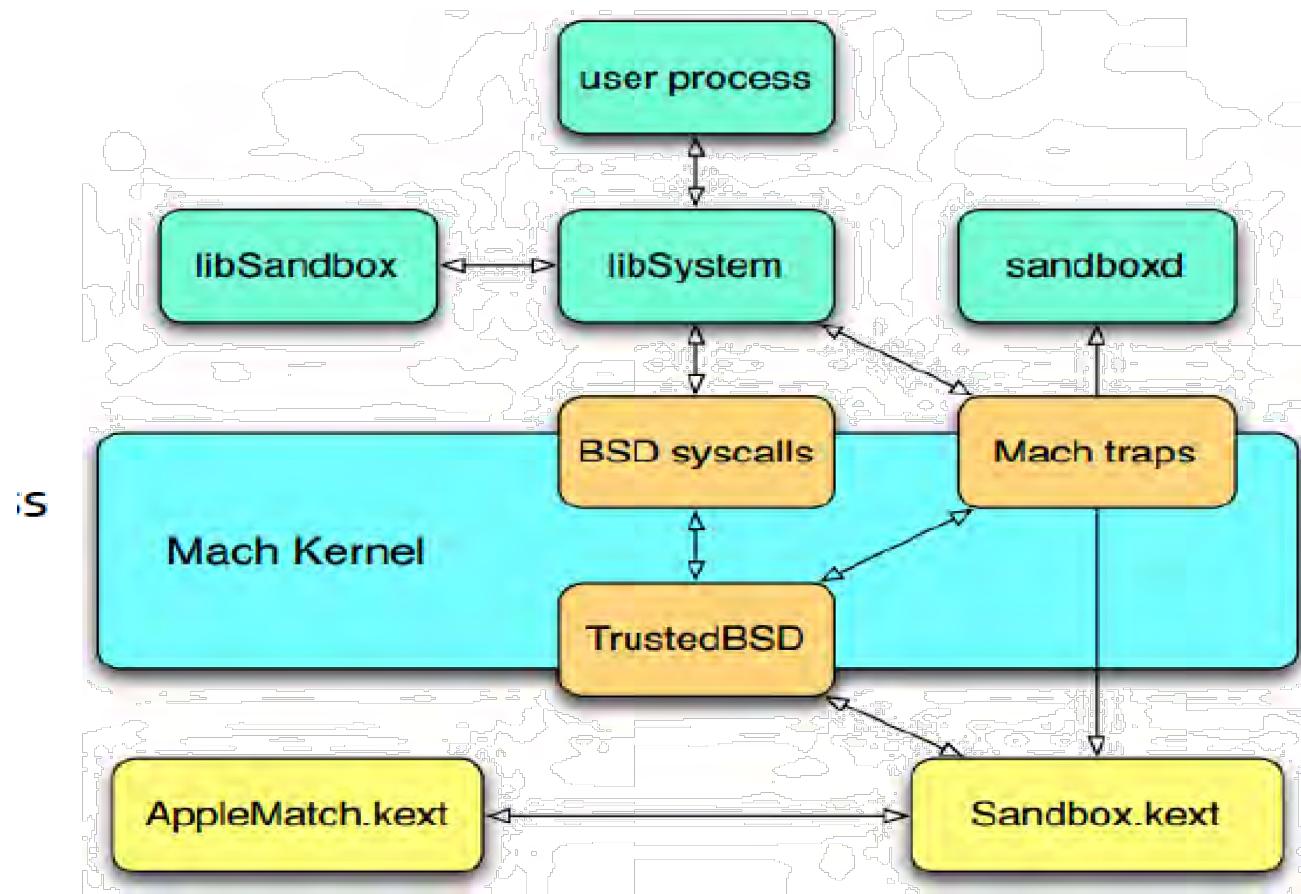
- First security control is the secure boot
- Each step ensures that the proper software is running:



# iOS application testing

- Once system is running, application sandbox (seatbelt) takes over
- Entitlements control access to user information and system-wide resources
- All apps run under 1 user: mobile
- Apps are digitally signed by Apple, an Enterprise or Developer certificate
- Signature is checked before starting an application

# iOS application testing



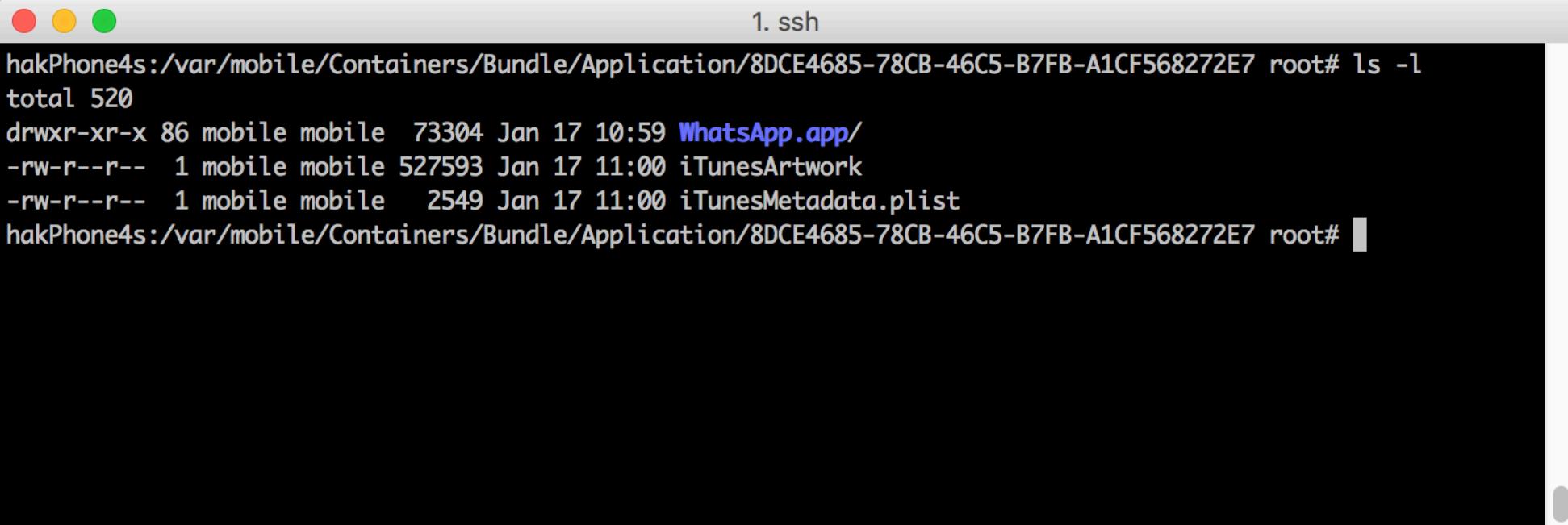
# iOS application testing

- Device has full disk encryption (FDE) by default, protects against hardware attacks and allows for quick erase
- Each file is encrypted with its own key, to implement access control
- Data and apps are not protected against:
  - Jailbreaking
  - Backup attacks
  - Software vulnerabilities

# iOS application testing

- Each app has a unique UUID and is stored in:
  - /var/mobile/Containers/Bundle (code)
  - /var/mobile/Containers/Data (data)
- Distributed as a (signed) .ipa file (ZIP file)
- Native apps are stored in /Applications
- On older (< 9) iOS:
  - /private/var/mobile/Library/Caches/com.apple.mobile.installation.plist contains list of all installed applications

# iOS application testing



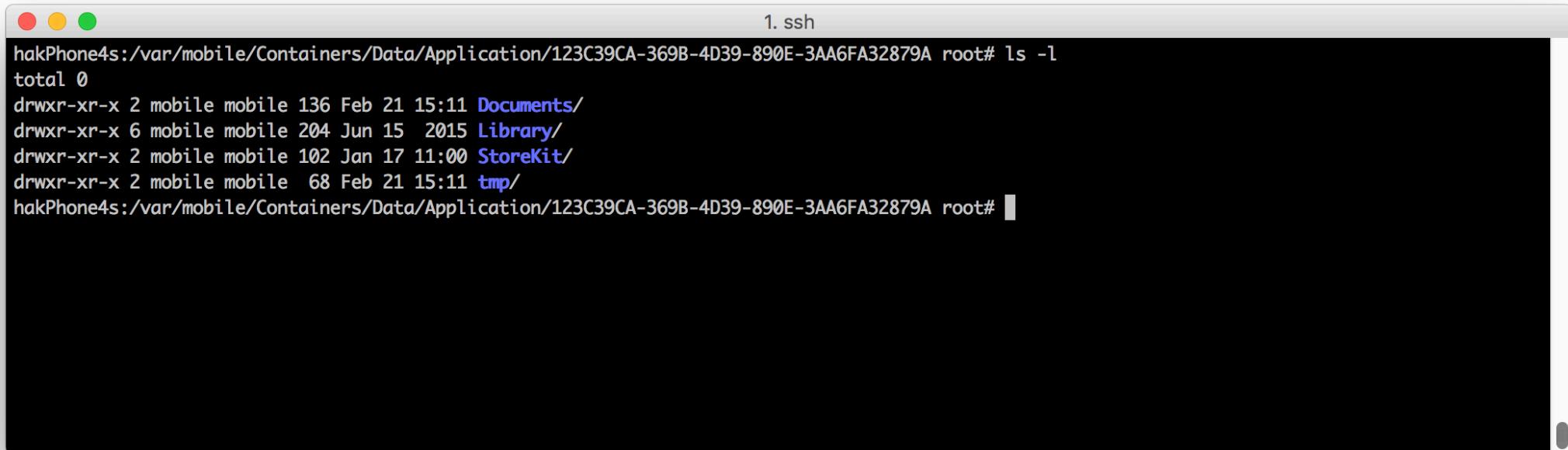
A screenshot of a terminal window titled "1. ssh". The window shows a file listing from a root shell on an iPhone 4s. The command "ls -l" is run in the directory "/var/mobile/Containers/Bundle/Application/8DCE4685-78CB-46C5-B7FB-A1CF568272E7". The output shows the "WhatsApp.app" folder and its contents: "iTunesArtwork" and "iTunesMetadata.plist". The "WhatsApp.app" folder is highlighted in blue.

```
hakPhone4s:/var/mobile/Containers/Bundle/Application/8DCE4685-78CB-46C5-B7FB-A1CF568272E7 root# ls -l
total 520
drwxr-xr-x 86 mobile mobile 73304 Jan 17 10:59 WhatsApp.app/
-rw-r--r-- 1 mobile mobile 527593 Jan 17 11:00 iTunesArtwork
-rw-r--r-- 1 mobile mobile 2549 Jan 17 11:00 iTunesMetadata.plist
hakPhone4s:/var/mobile/Containers/Bundle/Application/8DCE4685-78CB-46C5-B7FB-A1CF568272E7 root#
```

# iOS application testing

```
1. ssh
drwxr-xr-x 2 mobile mobile    170 Jan 17 10:59 WAStorageWarningViewController~iphone.nib/
drwxr-xr-x 4 mobile mobile    170 Jan 17 10:59 WAVerificationCloudReminder.storyboardc/
drwxr-xr-x 2 mobile mobile    136 Jan 17 10:59 WAVideoPreviewViewController.nib/
drwxr-xr-x 2 mobile mobile    136 Jan 17 10:59 WAVoiceCallViewController.nib/
drwxr-xr-x 2 mobile mobile    136 Jan 17 10:59 WAWallpaperPickerController.nib/
-rw-r--r-- 1 mobile mobile   3936 Dec 30 22:13 WAWebClientSessionTableViewCell.nib
drwxr-xr-x 2 mobile mobile    136 Jan 17 10:59 WAWebLinkTableViewCell.nib/
-rw-r--r-- 1 mobile mobile   2089 Dec 30 22:13 WS-12.png
-rw-r--r-- 1 mobile mobile   2725 Dec 30 22:13 WS-16.png
-rw-r--r-- 1 mobile mobile   3267 Dec 30 22:13 WS-24.png
-rw-r--r-- 1 mobile mobile   5209 Dec 30 22:13 WS-32.png
drwxr-xr-x 2 mobile mobile    1394 Jan 17 10:59 Wallpaper/
-rwrxr-xr-x 1 mobile mobile 19735504 Jan  5 23:46 WhatsApp*
drwxr-xr-x 2 mobile mobile    986 Jan 17 10:59 WhatsAppChat.momd/
-rw-r--r-- 1 mobile mobile   931 Dec 30 22:13 WhitePixel.png
-rw-r--r-- 1 mobile mobile   2527 Dec 30 22:13 XE-12.png
-rw-r--r-- 1 mobile mobile   3291 Dec 30 22:13 XE-16.png
-rw-r--r-- 1 mobile mobile   4050 Dec 30 22:13 XE-24.png
-rw-r--r-- 1 mobile mobile   5906 Dec 30 22:13 XE-32.png
-rw-r--r-- 1 mobile mobile  374585 Dec 30 22:13 Xylophone.caf
-rw-r--r-- 1 mobile mobile   2032 Dec 30 22:13 YE-12.png
-rw-r--r-- 1 mobile mobile   2699 Dec 30 22:13 YE-16.png
-rw-r--r-- 1 mobile mobile   3184 Dec 30 22:13 YE-24.png
-rw-r--r-- 1 mobile mobile   4670 Dec 30 22:13 YE-32.png
-rw-r--r-- 1 mobile mobile   2319 Dec 30 22:13 ZA-12.png
-rw-r--r-- 1 mobile mobile   3211 Dec 30 22:13 ZA-16.png
-rw-r--r-- 1 mobile mobile   3893 Dec 30 22:13 ZA-24.png
-rw-r--r-- 1 mobile mobile   6272 Dec 30 22:13 ZA-32.png
-rw-r--r-- 1 mobile mobile   1778 Dec 30 22:13 ZM-12.png
-rw-r--r-- 1 mobile mobile   2260 Dec 30 22:13 ZM-16.png
-rw-r--r-- 1 mobile mobile   2664 Dec 30 22:13 ZM-24.png
-rw-r--r-- 1 mobile mobile   4338 Dec 30 22:13 ZM-32.png
```

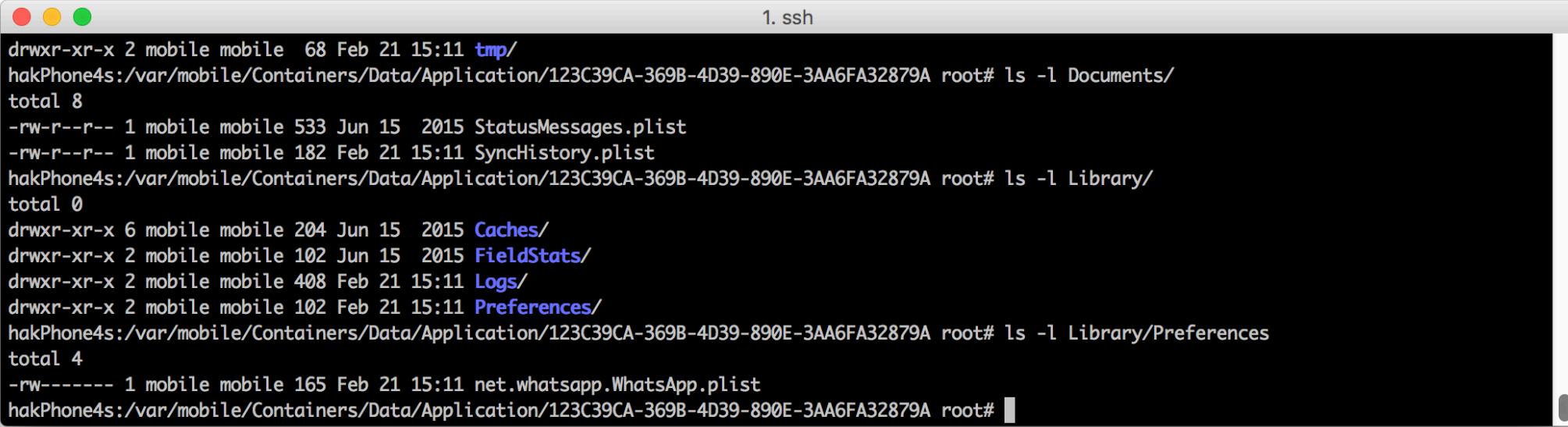
# iOS application testing



The terminal window shows the command `ls -l` being run in the root directory of an iOS application's container. The output lists several directories with their permissions, last modified dates, and sizes. The directories listed are `Documents`, `Library`, `StoreKit`, and `tmp`.

```
hakPhone4s:/var/mobile/Containers/Data/Application/123C39CA-369B-4D39-890E-3AA6FA32879A root# ls -l
total 0
drwxr-xr-x 2 mobile mobile 136 Feb 21 15:11 Documents/
drwxr-xr-x 6 mobile mobile 204 Jun 15 2015 Library/
drwxr-xr-x 2 mobile mobile 102 Jan 17 11:00 StoreKit/
drwxr-xr-x 2 mobile mobile 68 Feb 21 15:11 tmp/
hakPhone4s:/var/mobile/Containers/Data/Application/123C39CA-369B-4D39-890E-3AA6FA32879A root#
```

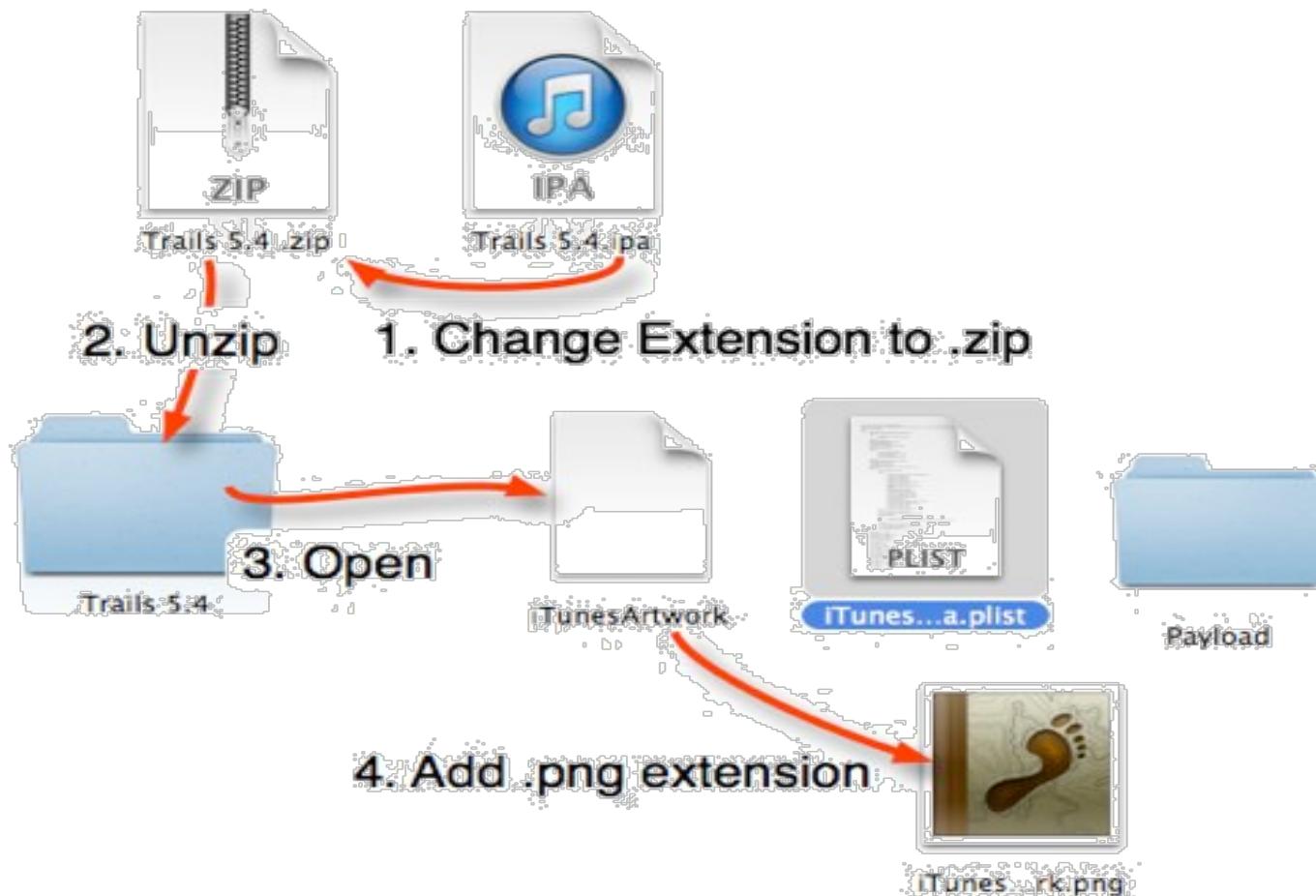
# iOS application testing



A terminal window titled "1. ssh" showing the file structure of an iOS application's container. The command "ls -l" is used to list files in various directories.

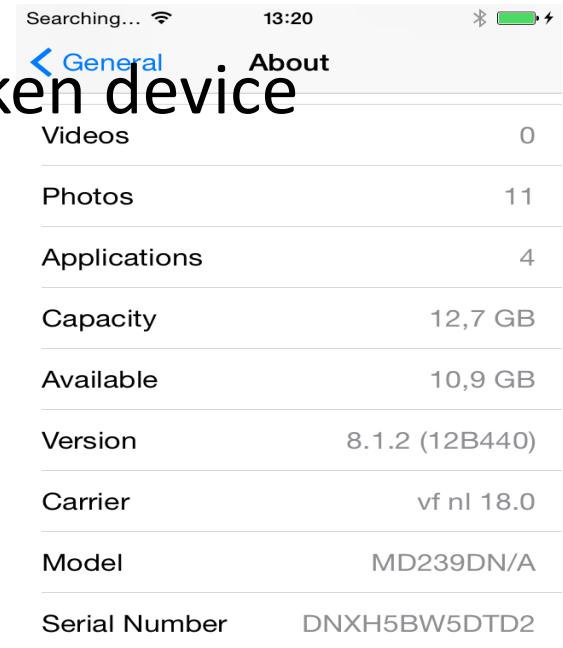
```
drwxr-xr-x 2 mobile mobile 68 Feb 21 15:11 tmp/
hakPhone4s:/var/mobile/Containers/Data/Application/123C39CA-369B-4D39-890E-3AA6FA32879A root# ls -l Documents/
total 8
-rw-r--r-- 1 mobile mobile 533 Jun 15 2015 StatusMessages.plist
-rw-r--r-- 1 mobile mobile 182 Feb 21 15:11 SyncHistory.plist
hakPhone4s:/var/mobile/Containers/Data/Application/123C39CA-369B-4D39-890E-3AA6FA32879A root# ls -l Library/
total 0
drwxr-xr-x 6 mobile mobile 204 Jun 15 2015 Caches/
drwxr-xr-x 2 mobile mobile 102 Jun 15 2015 FieldStats/
drwxr-xr-x 2 mobile mobile 408 Feb 21 15:11 Logs/
drwxr-xr-x 2 mobile mobile 102 Feb 21 15:11 Preferences/
hakPhone4s:/var/mobile/Containers/Data/Application/123C39CA-369B-4D39-890E-3AA6FA32879A root# ls -l Library/Preferences
total 4
-rw----- 1 mobile mobile 165 Feb 21 15:11 net.whatsapp.WhatsApp.plist
hakPhone4s:/var/mobile/Containers/Data/Application/123C39CA-369B-4D39-890E-3AA6FA32879A root#
```

# iOS application testing



# iOS application testing

- To be able to REALLY test an app, you need a Jailbroken device
- Jailbreaking allows you to:
  - Run any app;
  - Install your own tools;
  - Alternative app stores;
  - Unsigned binaries;
  - Disable sandbox;
  - SSH access to device.
- Jailbreaks become increasingly rare



A screenshot of an iPhone's General settings screen. At the top, there is a search bar labeled "Searching..." and a status bar showing "13:20" and battery level. Below the search bar, there are two tabs: "General" (which is selected) and "About". The "About" tab shows the following information:

Videos	0
Photos	11
Applications	4
Capacity	12,7 GB
Available	10,9 GB
Version	8.1.2 (12B440)
Carrier	vf nl 18.0
Model	MD239DN/A
Serial Number	DNXH5BW5DTD2

# iOS application testing

- After jailbreak, install Cydia app store:

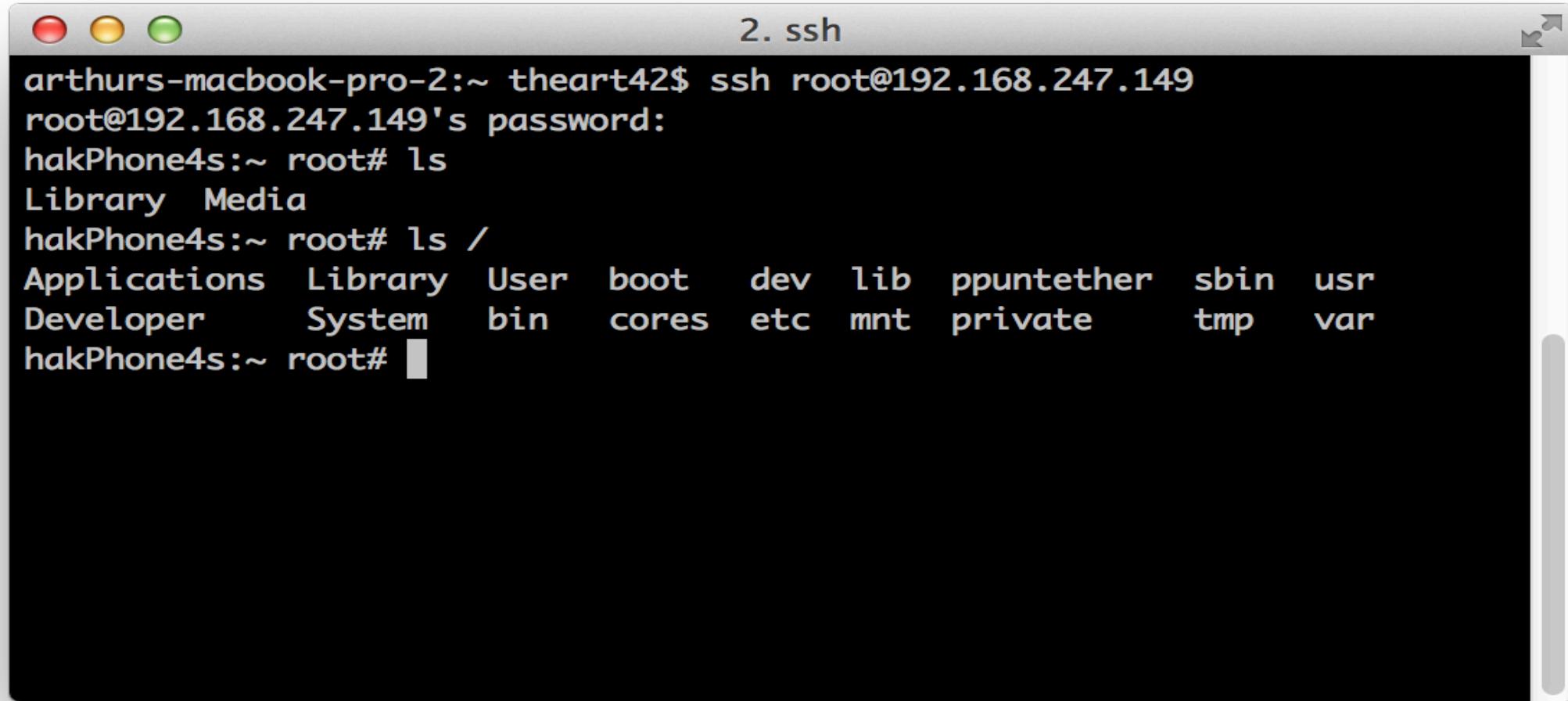
- Repositories with additional tools
- Packaged as .deb files (Linux)
- Install APT packages
- Install OpenSSH



# iOS application testing

- Jailbreak security issues:
  - Default password for root and mobile user is alpine.
  - Change this after jailbreaking and installing OpenSSH!!

# iOS application testing

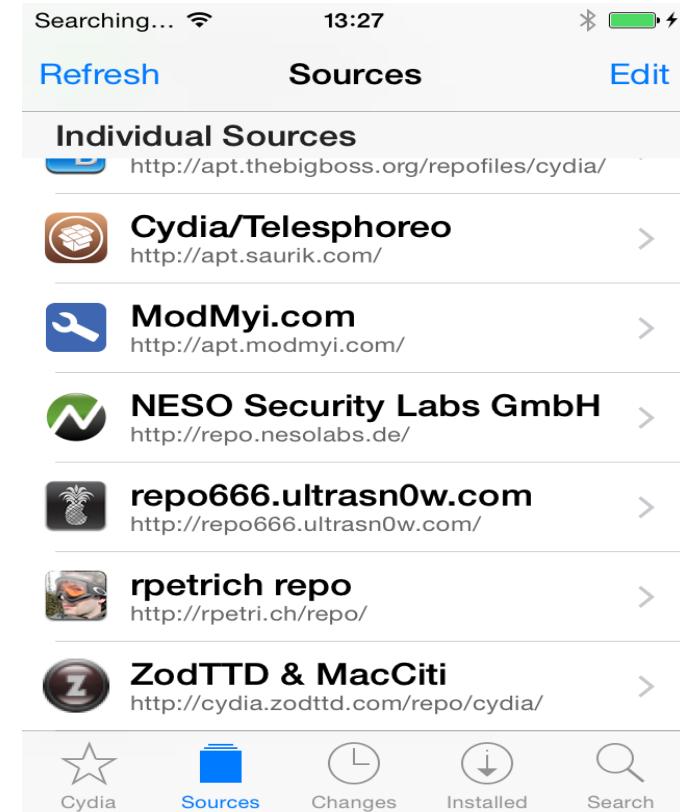


The terminal window is titled "2. ssh". The session shows a root shell on a Mac OS X system (arthur's-macbook-pro-2). The user logs in via SSH and runs several "ls" commands to list directory contents.

```
arthurs-macbook-pro-2:~ theart42$ ssh root@192.168.247.149
root@192.168.247.149's password:
hakPhone4s:~ root# ls
Library Media
hakPhone4s:~ root# ls /
Applications Library User boot dev lib ppuntether sbin usr
Developer System bin cores etc mnt private tmp var
hakPhone4s:~ root#
```

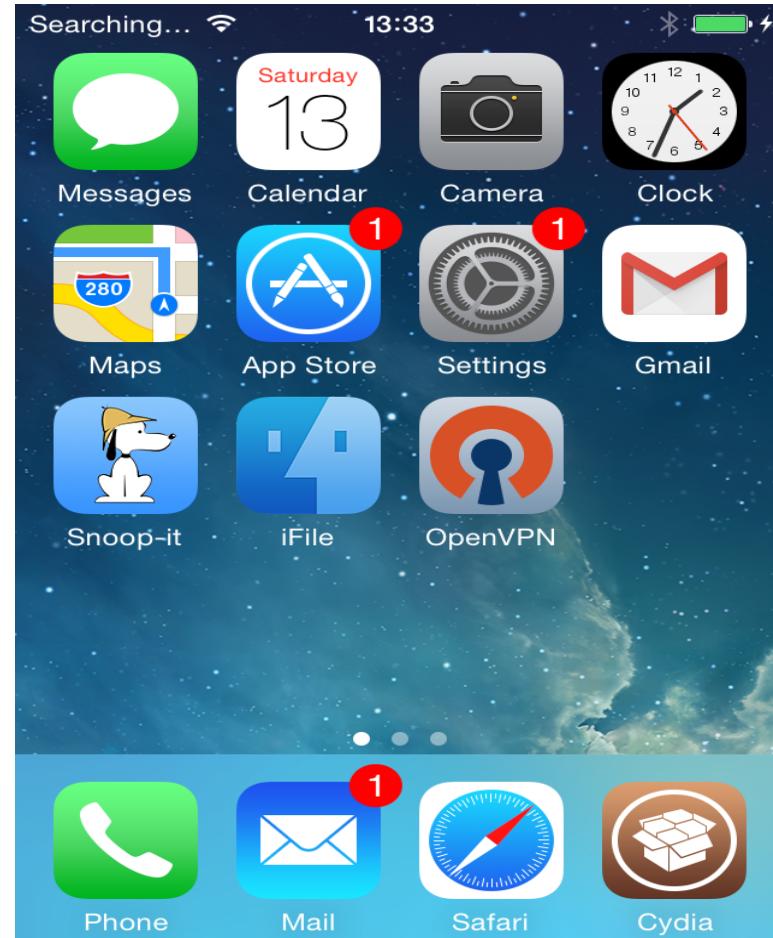
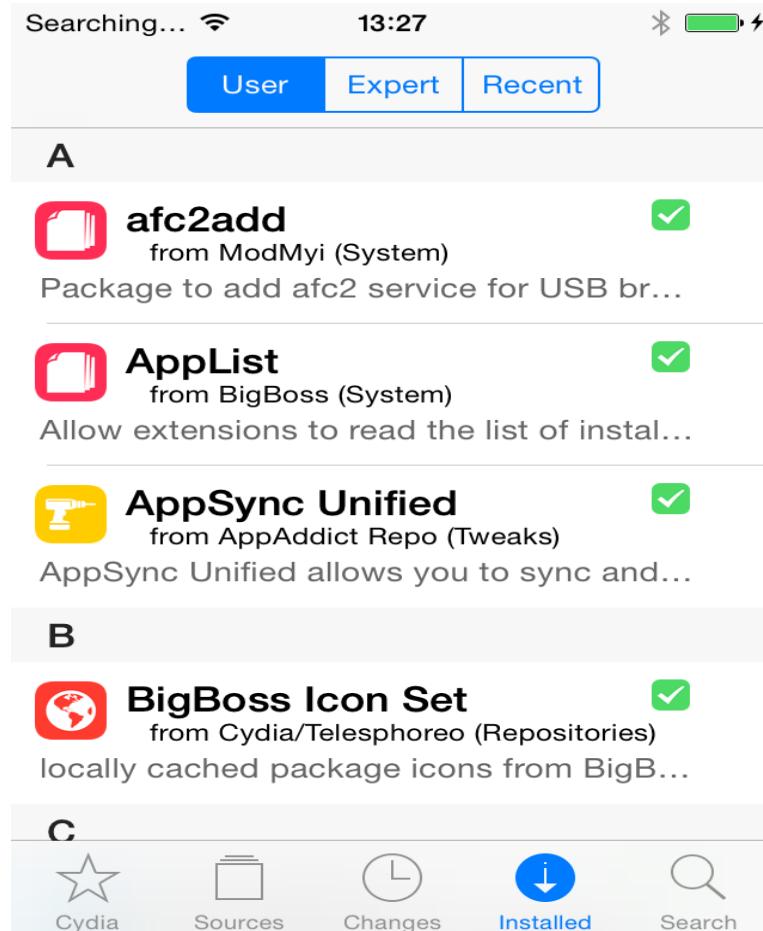
# iOS application testing

- Cydia repositories



# iOS application testing

- Packages



# iOS application testing

- Packages install most useful tools in one go:

```
iPhone:~ root# apt-get install adv-cmds com.sull.clutchpatched  
curl cycript odcctools developer-cmds dpkg  
com.ericasadun.utilities file file-cmds findutils gawk git grep  
inetutils com.autopear.installipa ldid less lsif mobilesubstrate  
com.saurik.substrate.safemode mobileterminal-applesdk nano netcat  
network-cmds python sed shell-cmds sqlite3 syslogd system-cmds  
tcpdump top uikittools unrar unzip vim wget whois zip
```

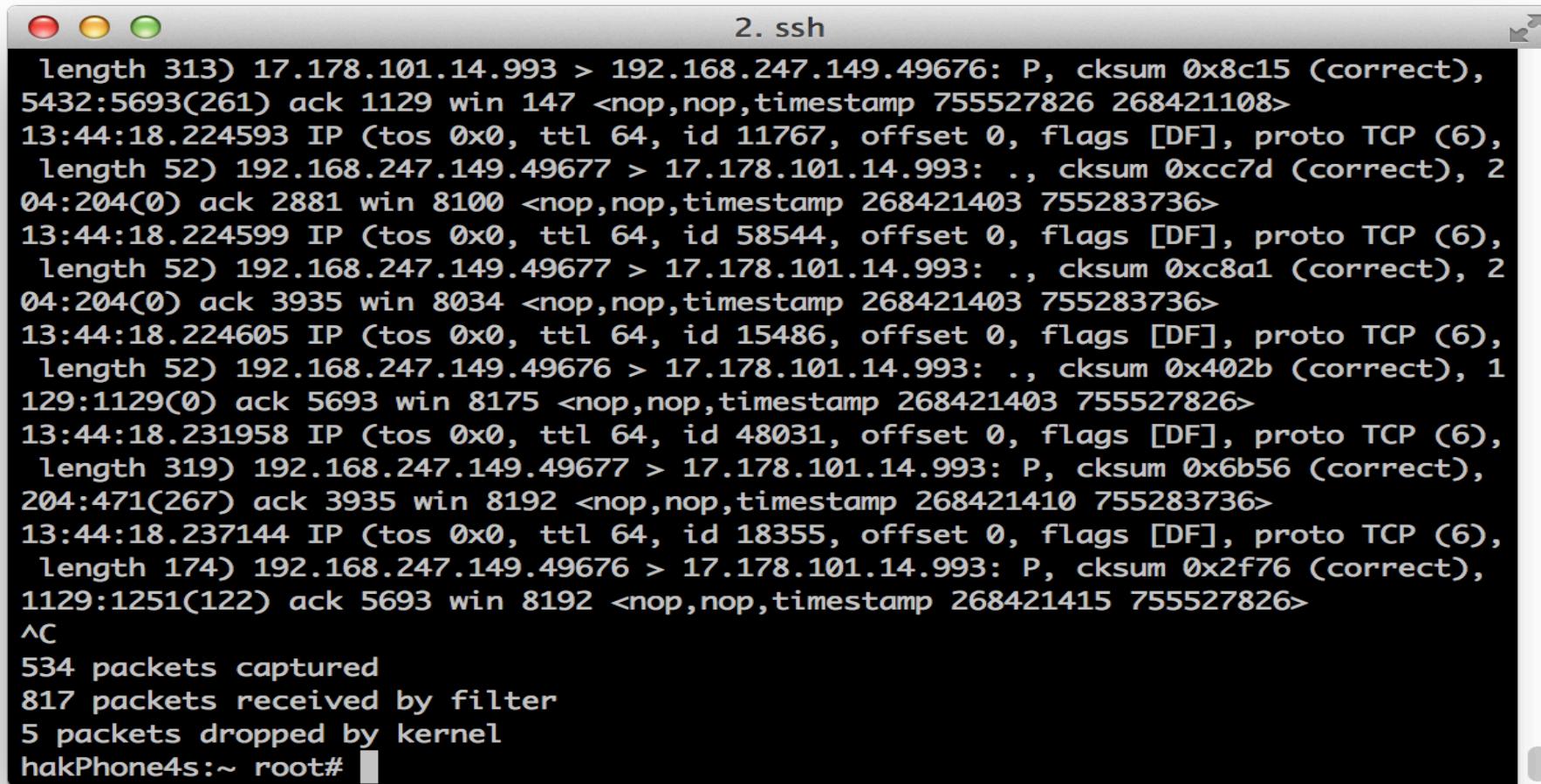
# iOS application testing

- In Xcode you can:
  - Disable code signing.
  - Build an application.
  - Make a distribution file (.ipa).
  - Copy the package to the device (scp).
  - Deploy using IPA Installer Console or use iPhone configuration utility.

# iOS black box testing

- Traffic analysis:
  - Install tcpdump on device and ‘sniff’ traffic
    - `tcpdump -i en0 -n -vv -s 0 -U not port 22`

# iOS black box testing



A screenshot of a terminal window titled "2. ssh". The window displays a list of captured network packets. The traffic shows several TCP connections between an iPhone (17.178.101.14.993) and a host (192.168.247.149.49676). The logs include details like sequence numbers, ACK numbers, window sizes, and checksums. The terminal session ends with a Ctrl-C interrupt and summary statistics: 534 packets captured, 817 received by filter, and 5 dropped by kernel. The prompt "hakPhone4s:~ root#" is visible at the bottom.

```
length 313) 17.178.101.14.993 > 192.168.247.149.49676: P, cksum 0x8c15 (correct),  
5432:5693(261) ack 1129 win 147 <nop,nop,timestamp 755527826 268421108>  
13:44:18.224593 IP (tos 0x0, ttl 64, id 11767, offset 0, flags [DF], proto TCP (6),  
length 52) 192.168.247.149.49677 > 17.178.101.14.993: ., cksum 0xcc7d (correct), 2  
04:204(0) ack 2881 win 8100 <nop,nop,timestamp 268421403 755283736>  
13:44:18.224599 IP (tos 0x0, ttl 64, id 58544, offset 0, flags [DF], proto TCP (6),  
length 52) 192.168.247.149.49677 > 17.178.101.14.993: ., cksum 0xc8a1 (correct), 2  
04:204(0) ack 3935 win 8034 <nop,nop,timestamp 268421403 755283736>  
13:44:18.224605 IP (tos 0x0, ttl 64, id 15486, offset 0, flags [DF], proto TCP (6),  
length 52) 192.168.247.149.49676 > 17.178.101.14.993: ., cksum 0x402b (correct), 1  
129:1129(0) ack 5693 win 8175 <nop,nop,timestamp 268421403 755527826>  
13:44:18.231958 IP (tos 0x0, ttl 64, id 48031, offset 0, flags [DF], proto TCP (6),  
length 319) 192.168.247.149.49677 > 17.178.101.14.993: P, cksum 0x6b56 (correct),  
204:471(267) ack 3935 win 8192 <nop,nop,timestamp 268421410 755283736>  
13:44:18.237144 IP (tos 0x0, ttl 64, id 18355, offset 0, flags [DF], proto TCP (6),  
length 174) 192.168.247.149.49676 > 17.178.101.14.993: P, cksum 0x2f76 (correct),  
1129:1251(122) ack 5693 win 8192 <nop,nop,timestamp 268421415 755527826>  
^C  
534 packets captured  
817 packets received by filter  
5 packets dropped by kernel  
hakPhone4s:~ root#
```

# iOS black box testing

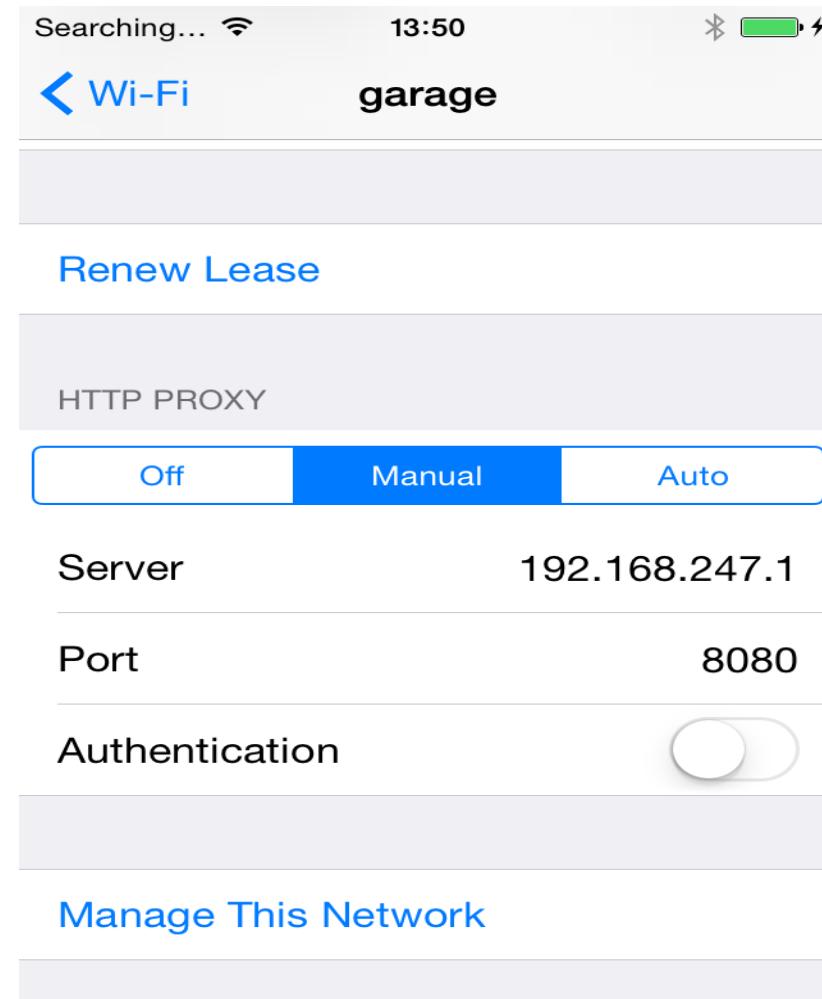
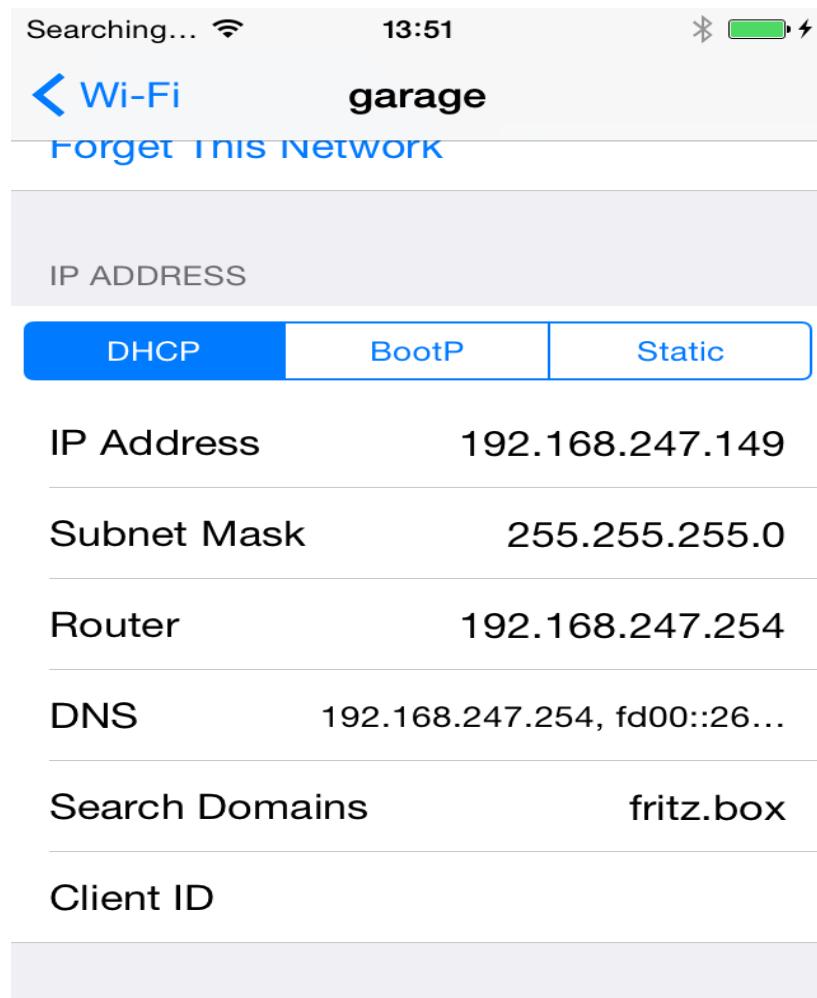


- Or:
  - Build an Access Point with Kali or Pineapple
  - Analyse traffic with Wireshark or tcpdump
  - Don't forget to disable firewall on your access point
  - And enable ip\_forwarding
- Much more powerful and more tools at your disposal (especially if you are using Kali)
- Use a wifi dongle that has monitoring capabilities!

# iOS black box testing

- Use proxy software on Kali to intercept Web traffic
  - Burpsuite
  - ZAP
- Set proxy in wifi profile on device,
- or use iPhone Configuration Utility,
- or use transparent redirection on Kali

# iOS black box testing

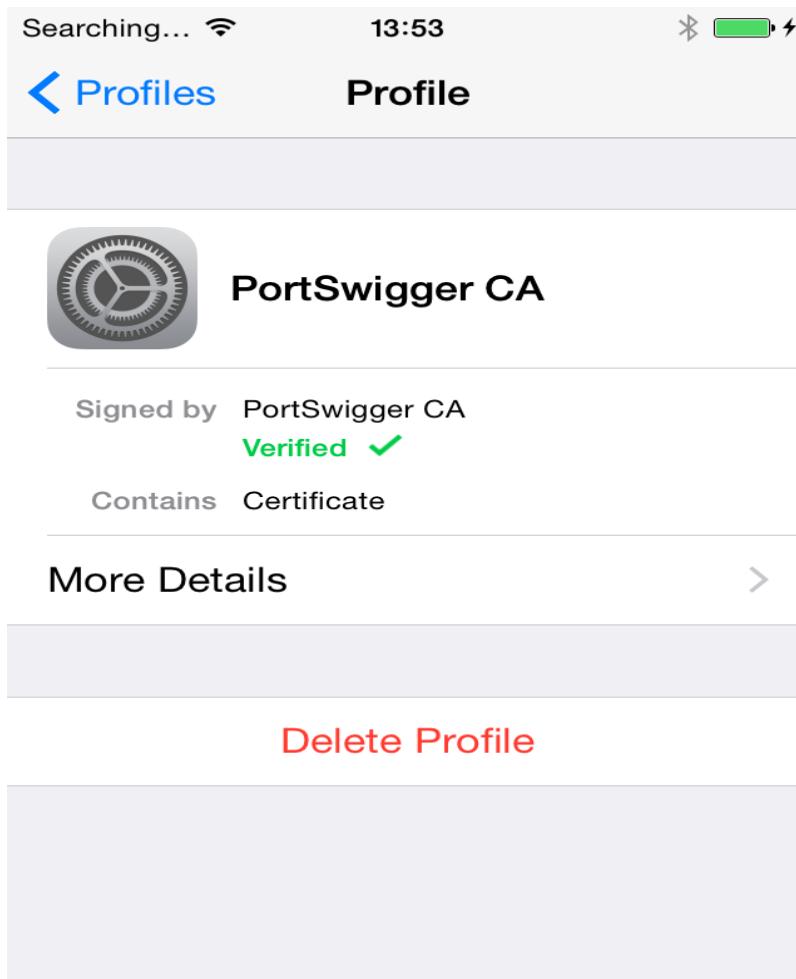


# iOS black box testing

- Install the burpsuite CA:
  - Send .crt file as mail attachment and install
  - Use the iPhone Configuration Utility
- Settings -> General -> Profiles



# iOS black box testing



SEARCHING...	13:54	BATTERY
<a href="#">Back</a>	PortSwigger CA	
SUBJECT NAME		
Country	PortSwigger	
State/Province	PortSwigger	
Locality	PortSwigger	
Organization	PortSwigger	
Organizational Unit	PortSwigger CA	
Common Name	PortSwigger CA	
ISSUER NAME		
Country	PortSwigger	

# iOS black box testing

- If an application does not honor the HTTP proxy settings, what can you do?
  - Intercept DNS traffic and return the IP address of your test machine as the target IP. Don't forget to forward the traffic to the actual destination
  - Perform transparent redirection on your Linux machine using iptables.
- In both cases, your proxy must support invisible/transparent proxying

# iOS black box testing

- An example:

```
iptables -t nat -F PREROUTING
iptables -t nat -F POSTROUTING
iptables -t nat -A PREROUTING -i eth2 -s 172.16.42.0/24 -d 1.2.3.4 -p tcp --dport 443 -j DNAT --to
172.16.42.42:443
iptables -t nat -A POSTROUTING -o eth0 -d 1.2.3.4 -j SNAT --to 172.16.98.222
iptables -t nat -A POSTROUTING -s 172.16.42.0/24 -d 0.0.0.0/0 -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward
```

# iOS black box testing

- For other traffic (non-HTTP) you can use Mallory as a generic proxy or write your own scripts.... Both are complicated.
- Use iptables for redirection.
- Luckily most (> 90%) of client / server traffic is web based:
  - Screen scraping
  - Web services

# iOS black box testing

- What about certificate pinning?
  - Hack the application (replace certificate files with your own).
  - Use ios-ssl-killswitch or trustme.
- They overrule or patch the SSL framework to always accept Certificate.
- If done properly, pinning is hard to bypass (actual verifications in application code).

# iOS black box testing

- In client/server communication, check for:
  - Proper certificate handling (does the app accept self-signed certificates)
  - Is sensitive information cached somewhere?
  - Where are the (persistent) cookies?
  - Any Javascript in use?
  - XML injection
  - Be careful with Unique Device ID  
(can be spoofed)



# iOS black box testing

- In client / server communication, to protect your customers, check for:
  - Access to address book, photos
  - Geolocation
  - Access to phone etc...

# iOS black box testing

- Local data storage:
  - User preferences
  - Media files
  - Attachments
  - Config settings

# iOS black box testing

- Local data stored in:
  - plain files
  - .plist files
  - SQLite database
- You can often find credentials, urls, password, tokens, cookies etc.

# iOS black box testing

- You can use the SSH access to browse the file system
- Or use iExplorer to browse device
- Don't forget that these files are stored in your backup as well, so force encrypted backups
- Use the data protection classes to properly protect data on the device

```
[ [NSFileManager defaultManager] createFileAtPath: [self filePath]
    contents: @{@"file content" dataUsingEncoding:NSUTF8StringEncoding]
    attributes: [NSDictionary dictionaryWithObject: NSFileProtectionComplete
    forKey: NSFileProtectionKey] ];
```

# iOS black box testing

## Backups

The screenshot shows the 'Backups' settings screen. It has two main sections: 'Automatically Back Up' and 'Manually Back Up and Res'. Under 'Automatically Back Up', there are two options: 'iCloud' (unchecked) and 'This computer' (checked). Below 'This computer', there is a checked checkbox for 'Encrypt local backup' with a note explaining it allows account passwords, Health, and HomeKit data to be backed up. A 'Change Password...' button is at the bottom. To the right, under 'Manually Back Up and Res', it says 'Manually back up your iPad' and 'Backup stored on this computer'. A 'Back Up Now' button is present. At the bottom, it shows 'Latest Backup: 27/03/16 18:11 to this computer'.

**Automatically Back Up**

iCloud  
Back up the most important data on your iPad to iCloud.

This computer  
A full backup of your iPad will be stored on this computer.

Encrypt local backup  
This will allow account passwords, Health, and HomeKit data to be backed up.

[Change Password...](#)

**Manually Back Up and Res**

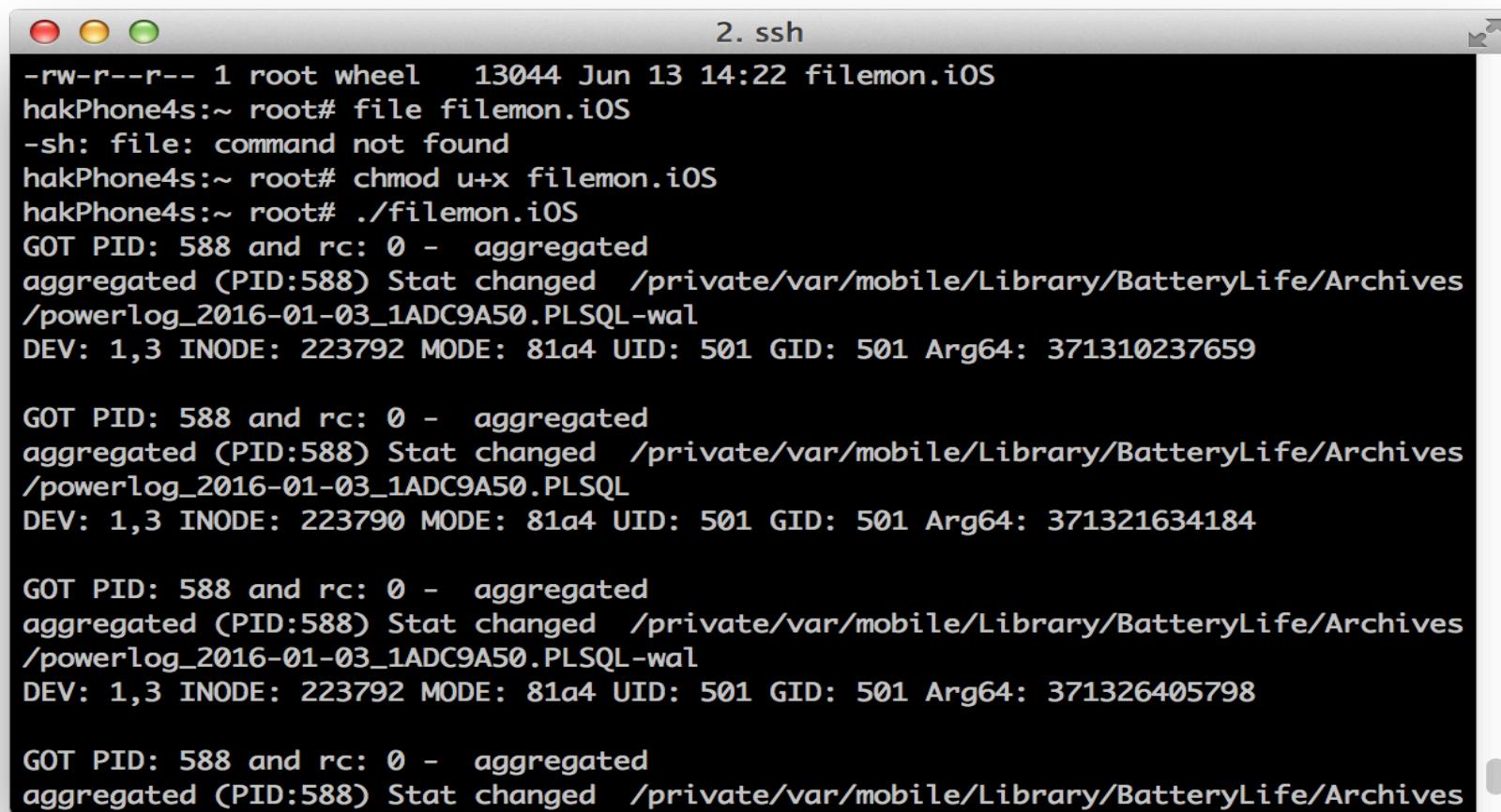
Manually back up your iPad  
Backup stored on this computer

[Back Up Now](#)

**Latest Backup:**  
27/03/16 18:11 to this computer

# iOS black box testing

- You can use filemon.iOS to monitor file access in realtime



The screenshot shows a terminal window titled "2. ssh" running on a Mac OS X system. The window contains the following text output from the filemon.iOS application:

```
-rw-r--r-- 1 root wheel 13044 Jun 13 14:22 filemon.iOS
hakPhone4s:~ root# file filemon.iOS
-sh: file: command not found
hakPhone4s:~ root# chmod u+x filemon.iOS
hakPhone4s:~ root# ./filemon.iOS
GOT PID: 588 and rc: 0 - aggregated
aggregated (PID:588) Stat changed /private/var/mobile/Library/BatteryLife/Archives
/powerlog_2016-01-03_1ADC9A50.PLSQL-wal
DEV: 1,3 INODE: 223792 MODE: 81a4 UID: 501 GID: 501 Arg64: 371310237659

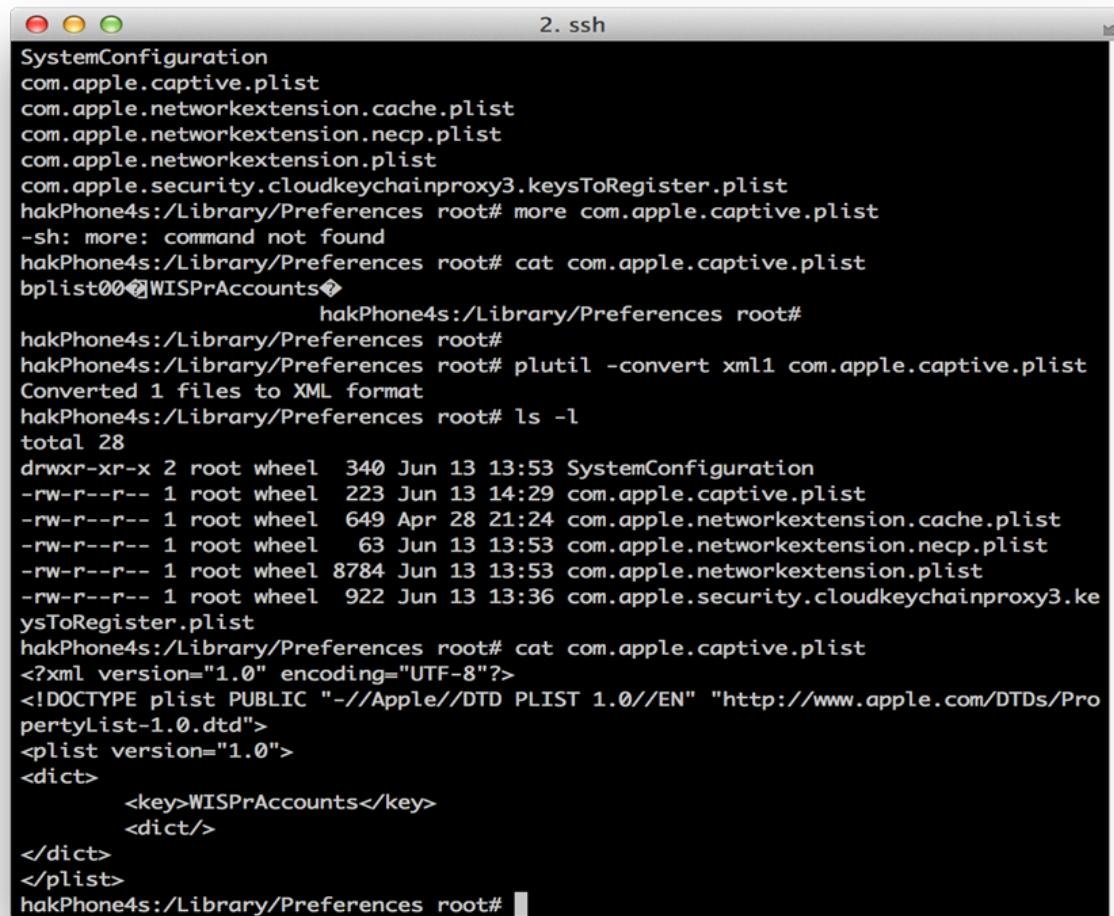
GOT PID: 588 and rc: 0 - aggregated
aggregated (PID:588) Stat changed /private/var/mobile/Library/BatteryLife/Archives
/powerlog_2016-01-03_1ADC9A50.PLSQL
DEV: 1,3 INODE: 223790 MODE: 81a4 UID: 501 GID: 501 Arg64: 371321634184

GOT PID: 588 and rc: 0 - aggregated
aggregated (PID:588) Stat changed /private/var/mobile/Library/BatteryLife/Archives
/powerlog_2016-01-03_1ADC9A50.PLSQL-wal
DEV: 1,3 INODE: 223792 MODE: 81a4 UID: 501 GID: 501 Arg64: 371326405798

GOT PID: 588 and rc: 0 - aggregated
aggregated (PID:588) Stat changed /private/var/mobile/Library/BatteryLife/Archives
```

# iOS black box testing

- Use plutil to examine the encoded plist files

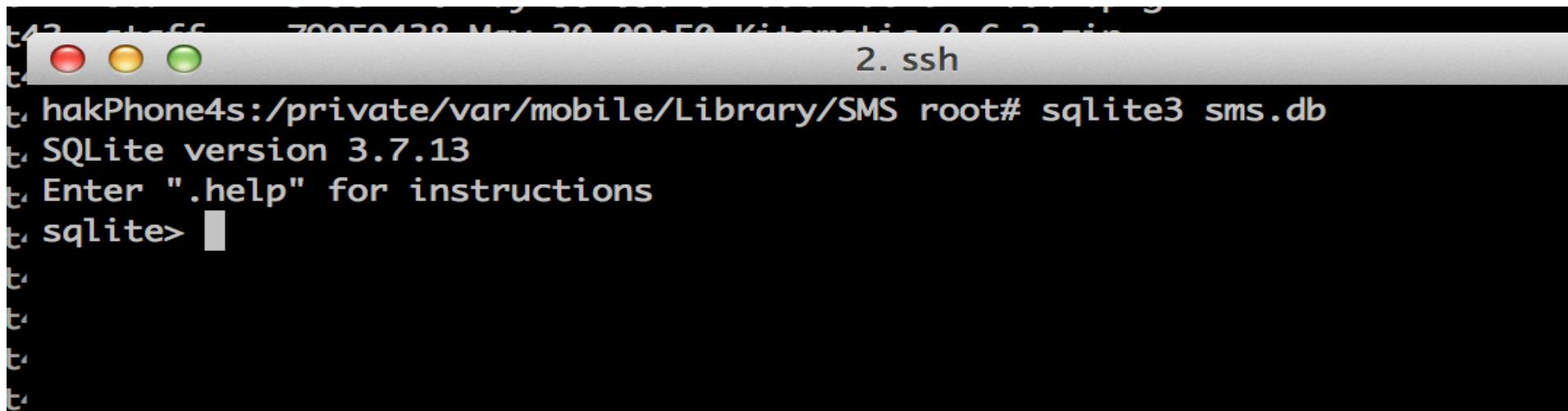


The terminal window shows the following command sequence:

```
SystemConfiguration  
com.apple.captive.plist  
com.apple.networkextension.cache.plist  
com.apple.networkextension.necp.plist  
com.apple.networkextension.plist  
com.apple.security.cloudkeychainproxy3.keysToRegister.plist  
hakPhone4s:/Library/Preferences root# more com.apple.captive.plist  
-sh: more: command not found  
hakPhone4s:/Library/Preferences root# cat com.apple.captive.plist  
bplist00@WISPrAccounts◆  
hakPhone4s:/Library/Preferences root#  
hakPhone4s:/Library/Preferences root# hakPhone4s:/Library/Preferences root# plutil -convert xml1 com.apple.captive.plist  
Converted 1 files to XML format  
hakPhone4s:/Library/Preferences root# ls -l  
total 28  
drwxr-xr-x 2 root wheel 340 Jun 13 13:53 SystemConfiguration  
-rw-r--r-- 1 root wheel 223 Jun 13 14:29 com.apple.captive.plist  
-rw-r--r-- 1 root wheel 649 Apr 28 21:24 com.apple.networkextension.cache.plist  
-rw-r--r-- 1 root wheel 63 Jun 13 13:53 com.apple.networkextension.necp.plist  
-rw-r--r-- 1 root wheel 8784 Jun 13 13:53 com.apple.networkextension.plist  
-rw-r--r-- 1 root wheel 922 Jun 13 13:36 com.apple.security.cloudkeychainproxy3.keysToRegister.plist  
hakPhone4s:/Library/Preferences root# cat com.apple.captive.plist  
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">  
<plist version="1.0">  
<dict>  
<key>WISPrAccounts</key>  
<dict/>  
</dict>  
</plist>  
hakPhone4s:/Library/Preferences root#
```

# iOS black box testing

- SQLite databases
- Use the sqlite3 command to read data



A screenshot of an iPhone's terminal application window titled "2. ssh". The window shows a root shell session on an iPhone 4S. The user has run the command "sqlite3 sms.db" to open the SQLite database located at "/private/var/mobile/Library/SMS". The SQLite version is 3.7.13, and the prompt "sqlite>" is visible, indicating the user can now enter SQL commands.

```
t43 staff 79959438 May 29 09:59:59 KitKat 3.6.3 (iPhone)
```

```
2. ssh
```

```
t hakPhone4s:/private/var/mobile/Library/SMS root# sqlite3 sms.db
t SQLite version 3.7.13
t Enter ".help" for instructions
t sqlite> |
```

# iOS black box testing

- Read SMS messages

```
sqlite> select * from message;  
1|76694598-F845-523B-E758-D5643FD039D9|Uw tegoed voor deze periode is bijgeschreven  
. Uw internetbundel is weer aangevuld.|0||1||1  
streamtyped|||||NSMutableAttributedString  
String|10|0|SMS|e:|2BABFF44-9950-4348-A3FA-4D7773C6A3C8|0|449052586|449065311|0|1|1  
|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|1|1|0|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|  
2|6382363B-EB34-4A00-A085-B4AC1114AC89|0|1|2|1|1|1|0|0|iMessage|p:+31637422847|667CC0  
B3-D401-4C99-80C1-3AC6AA0F6738|0|449064641|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|1  
|0|0|0|0|0|1|3|1|0|0|0|0|0|0|0|  
3|874E9F55-9540-4A9D-9ADC-3248CE154027|Testje|0||1|3|||  
streamtyped|||||NSMutableAttributedString  
ributedString|10|0|iMessage|p:+31637422847|667CC0B3-D401-4C99-80C1-3AC6AA0F6738|0|4  
49064576|449065312|0|1|1|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|1|1|0|0|0|0|0|0|1|0|0|0|0|1|0|1|0|  
0  
4|98F6AA3C-DBC7-F62B-4C2A-9E42BF3B32C8|Your allocated phone no. is 310820152. Thank  
you for joining the REDteam BTS. Call the Yatebts IVR at(32843)|0||1|4|||  
streamtyped  
|||||NSMutableAttributedString|10|0|SMS|e:|2BABFF44-9950-4348-A3FA-4D7773C6A3C8|0  
|449141542|449141545|0|1|1|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|1|1|0|0|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|1|0|0|  
0|0|
```

# iOS black box testing

- Keychain
  - Securely stores password, certificates, keys
  - Is encrypted with device and keybag specific keys
  - iOS has a single keychain (is a sqlite database) that stores all items for all applications
  - Application only has access to its own keys
  - `/private/var/Keychains/keychain-2.db`
  - Uses dataprotection classes as well (interesting: ThisDeviceOnly)

# iOS black box testing

```
hakPhone4s:/private/var/Keychains root# ls -l
total 1328
-rw----- 1 _securityd wheel 16384 Apr 10 10:54 TrustStore.sqlite3
-rw----- 1 _securityd wheel      47 Jan  1 1970 accountStatus.plist
-rw----- 1 _securityd wheel      0 Feb  1 12:20 caissuercache.sqlite3
-rw----- 1 _securityd wheel    512 Jan  1 1970 caissuercache.sqlite3-journal
-rw----- 1 _securityd wheel 315392 Jun 13 13:41 keychain-2.db
-rw----- 1 _securityd wheel 32768 Jun 13 14:04 keychain-2.db-shm
-rw----- 1 _securityd wheel 416152 Jun 13 13:51 keychain-2.db-wal
-r----- 1 root          wheel 141368 Jun 13 13:51 keychain-ota-backup.plist
-rw----- 1 _securityd wheel 45056 Jun 13 13:19 ocspcache.sqlite3
-rw----- 1 _securityd wheel 32768 Jun 13 14:07 ocspcache.sqlite3-shm
-rw----- 1 _securityd wheel 346112 Jun 13 13:53 ocspcache.sqlite3-wal
hakPhone4s:/private/var/Keychains root#
```

# iOS black box testing

Keychain table	Description
genp	Generic passwords – kSecClassGenericPassword
inet	Internet passwords – kSecClassInternetPassword
cert and keys	Certificates, keys and digital identity (cert+key) items – kSecClassCertificates and kSecClassIdentity

# iOS black box testing

- Keychain

```
: Generic Password
-----
: Service: CommCenter
: Account: SIM_PIN
: Entitlement Group: apple
: Label: (null)
: Generic Field: <38393331 30373131 31323030 38323031 353232>
: Keychain Data: 0123
```

# iOS black box testing

- Log files
  - /var/log directory

```
total 0
drwxr-xr-x 3 root wheel 102 Feb  1 12:26 CoreCapture
drwxr-xr-x 2 root wheel 102 Mar  7 10:02 apt
drwxr-xr-x 2 root wheel  68 Feb  1 12:09 asl
drwx----- 2 root wheel 102 Feb  1 12:41 com.apple.revisiond
drwxr-xr-x 2 root wheel  68 Feb  1 12:09 com.apple.xpc.launchd
drwxr-xr-x 2 root wheel  68 Feb  1 12:09 ppp
drwxr-xr-x 2 root wheel  68 Feb  1 12:09 sa
hakPhone4s:~ root#
```

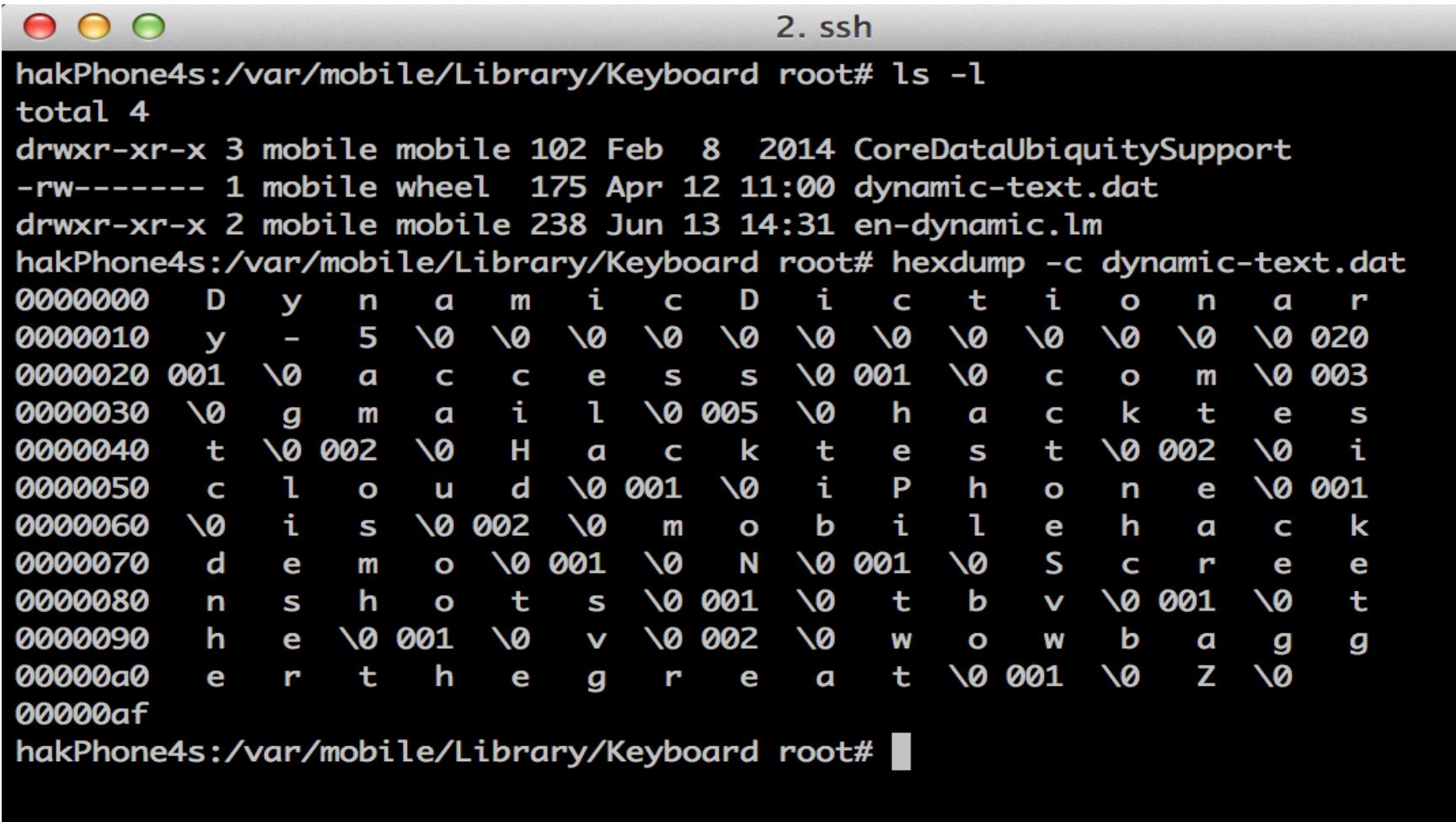
# iOS black box testing

- Log files
  - May contain interesting system information

# iOS black box testing

- Cache
  - Snapshots are taken in the Cache directory of the Application root
  - Can contain sensitive information from screenshot
  - Keyboard cache in /var/mobile/Library/Keyboard/dynamic-text.dat

# iOS black box testing



2. ssh

```
hakPhone4s:/var/mobile/Library/Keyboard root# ls -l
total 4
drwxr-xr-x 3 mobile mobile 102 Feb  8 2014 CoreDataUbiquitySupport
-rw----- 1 mobile wheel 175 Apr 12 11:00 dynamic-text.dat
drwxr-xr-x 2 mobile mobile 238 Jun 13 14:31 en-dynamic.lm

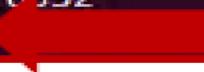
hakPhone4s:/var/mobile/Library/Keyboard root# hexdump -c dynamic-text.dat
00000000 D y n a m i c D i c t i o n a r y
00000010 y - 5 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 020
00000020 001 \0 a c c e s s \0 001 \0 c o m \0 003
00000030 \0 g m a i l \0 005 \0 h a c k t e s
00000040 t \0 002 \0 H a c k t e s t \0 002 \0 i
00000050 c l o u d \0 001 \0 i P h o n e \0 001
00000060 \0 i s \0 002 \0 m o b i l e h a c k
00000070 d e m o \0 001 \0 N \0 001 \0 S c r e e
00000080 n s h o t s \0 001 \0 t b v \0 001 \0 t
00000090 h e \0 001 \0 v \0 002 \0 w o w b a g g
000000a0 e r t h e g r e a t \0 001 \0 Z \0
00000af

hakPhone4s:/var/mobile/Library/Keyboard root#
```

# iOS black box testing

- Applications are encrypted using FairPlay DRM
- To be able to test an application, you need to decrypt it
- Check with the otool

```
Tests-iPhone:/var/mobile/Applications/84593008-5B13-4CFC-BCEB-727DDC29DD50/Twitter.app root# otool -l Twitter
  cmd LC_ENCRYPTION_INFO
  cmdsize 20
cryptoff 8192
cryptsize 5476352
cryptid 1
```



Encrypted

# iOS black box testing

- Use Clutchpatched or dumpdecrypted for decryption

```
DYLD_INSERT_LIBRARIES=dumpdecrypted.dylib /var/mobile/Applications/xxxxxxxx-xxxx-  
xxxx-xxxx-xxxxxxxxxxxx/Scan.app/Scan
```

# iOS black box testing

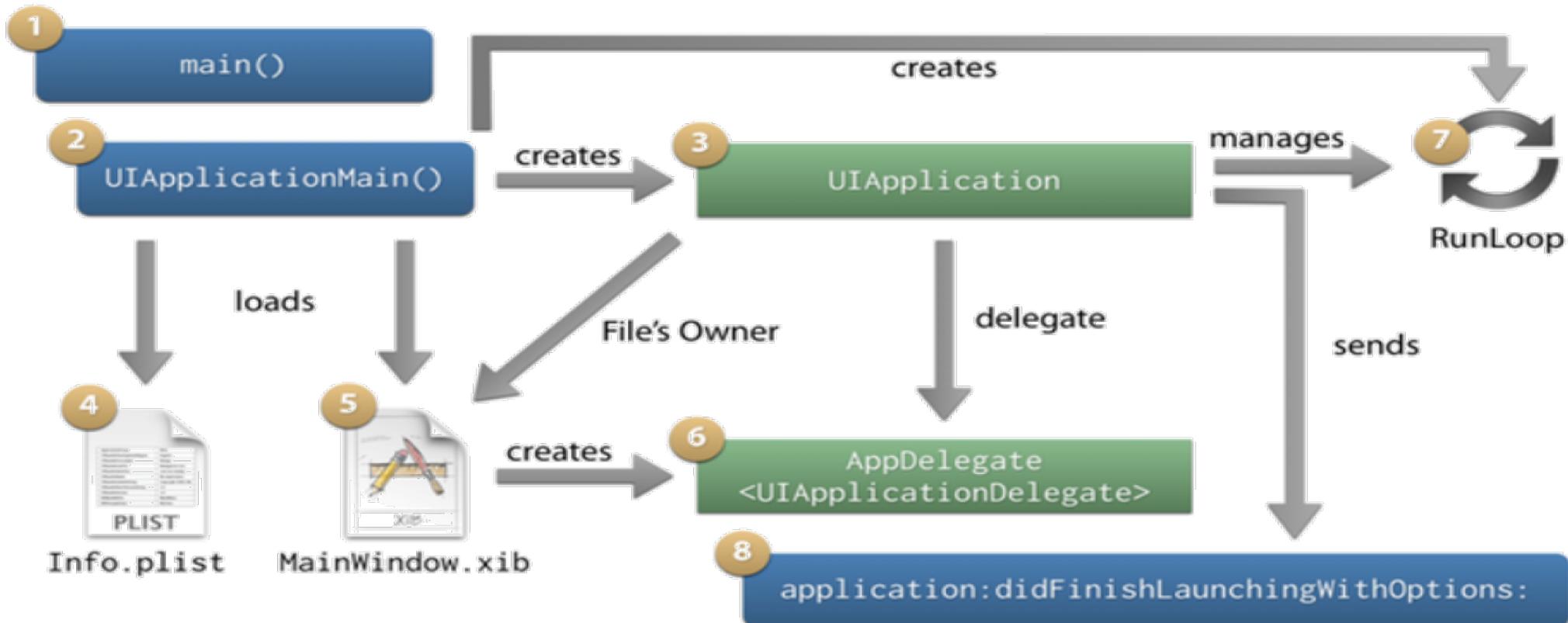
- Runtime manipulation
  - Cycript

```
cy# function printMethods(className) {  
cy>     var count = new new Type("I");  
cy>     var methods = class_copyMethodList(objc_getClass(className), count);  
cy>     var methodsArray = [];  
cy>     for(var i = 0; i < *count; i++) {  
cy>         var method = methods[i];  
cy>         methodsArray.push({selector:method_getName(method), implementation:method_getImplementation(method)});  
cy>     }  
cy>     free(methods);  
cy>     free(count);  
cy>     return methodsArray;  
cy> }  
cy#
```

# iOS black box testing

- Cycript
  - Combination of Interactive Objective-C interpreter and Javascript
  - App runtime behaviour can be manipulated
  - Can connect to an active process
  - Gives access to all classes and variables of an app
  - Used for runtime analysis to:
    - Bypass security locks
    - Access sensitive information in memory
    - Bypass authentication
    - Access restrictive parts of the application

# iOS black box testing

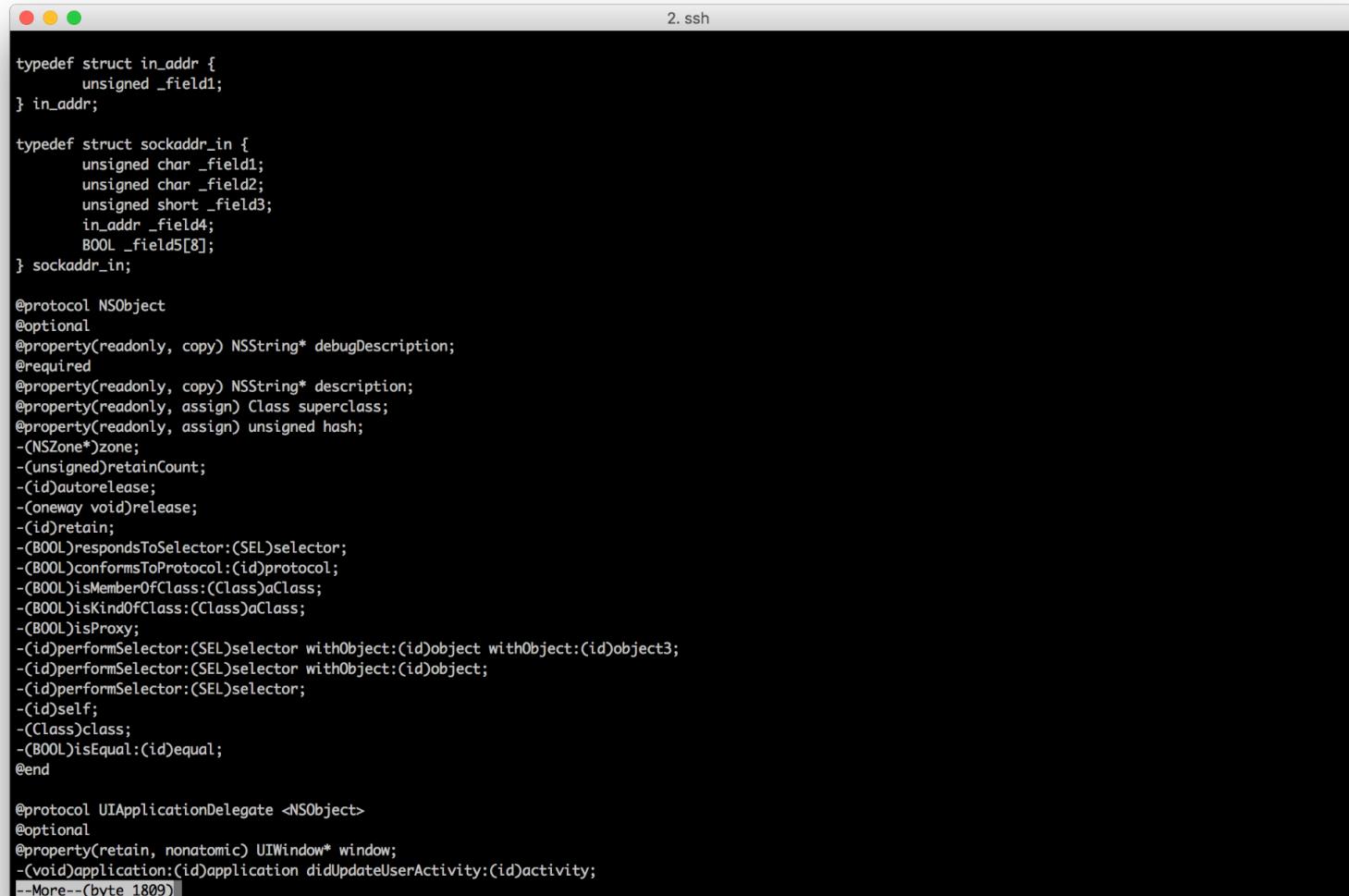


# iOS black box testing

- To be able to determine the object structure of the app you can dump it with class-dump or class-dump-z

# iOS black box testing

- class-dump-z



A terminal window titled "2. ssh" displaying Objective-C code. The code defines the `NSObject` protocol and its implementation. It includes declarations for `in_addr` and `sockaddr_in` structures, and various methods such as `-zone`, `-retainCount`, and `-release`. The `NSObject` protocol is also shown conforming to `UIApplicationDelegate`.

```
typedef struct in_addr {
    unsigned _field1;
} in_addr;

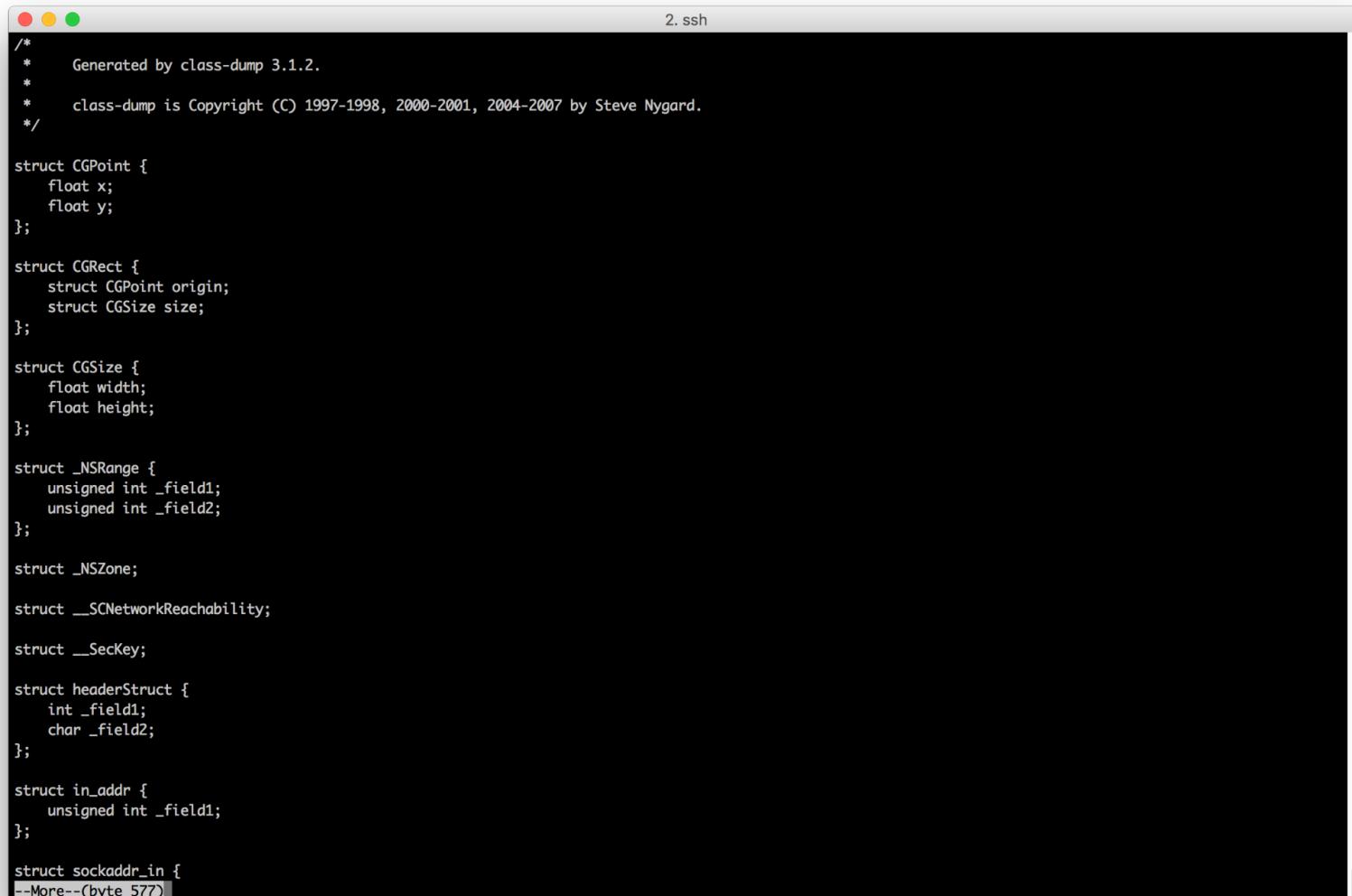
typedef struct sockaddr_in {
    unsigned char _field1;
    unsigned char _field2;
    unsigned short _field3;
    in_addr _field4;
    BOOL _field5[8];
} sockaddr_in;

@protocol NSObject
@optional
@property(nonatomic, copy) NSString* debugDescription;
@required
@property(nonatomic, copy) NSString* description;
@property(nonatomic, assign) Class superclass;
@property(nonatomic, assign) unsigned hash;
-(NSZone*)zone;
-(unsigned)retainCount;
-(id)autorelease;
-(oneway void)release;
-(id)retain;
-(BOOL)respondsToSelector:(SEL)selector;
-(BOOL)conformsToProtocol:(id)protocol;
-(BOOL)isMemberOfClass:(Class)aClass;
-(BOOL)isKindOfClass:(Class)aClass;
-(BOOL)isProxy;
-(id)performSelector:(SEL)selector withObject:(id)object withObject:(id)object3;
-(id)performSelector:(SEL)selector withObject:(id)object;
-(id)performSelector:(SEL)selector;
-(id)self;
-((Class)class;
-(BOOL)isEqual:(id)equal;
@end

@protocol UIApplicationDelegate <NSObject>
@optional
@property(nonatomic, nonatomic) UIWindow* window;
-(void)application:(id)application didUpdateUserActivity:(id)activity;
--More--(byte 1809)
```

# iOS black box testing

- class-dump



The screenshot shows a terminal window titled "2. ssh" displaying the output of the class-dump command. The output is a C-style code dump of Objective-C structures and unions from an iPhone application. The code includes definitions for CGPoint, CGRect, CGSize, NSRange, NSZone, SCNetworkReachability, SecKey, headerStruct, in\_addr, and sockaddr\_in. A callout box highlights the sockaddr\_in definition at the bottom of the screen.

```
/*
 * Generated by class-dump 3.1.2.
 *
 * class-dump is Copyright (C) 1997-1998, 2000-2001, 2004-2007 by Steve Nygard.
 */

struct CGPoint {
    float x;
    float y;
};

struct CGRect {
    struct CGPoint origin;
    struct CGSize size;
};

struct CGSize {
    float width;
    float height;
};

struct _NSRange {
    unsigned int _field1;
    unsigned int _field2;
};

struct _NSZone;

struct __SCNetworkReachability;

struct __SecKey;

struct headerStruct {
    int _field1;
    char _field2;
};

struct in_addr {
    unsigned int _field1;
};

struct sockaddr_in {
    --More--(byte 577)
```

# Challenges?

- No jailbreak available
- Extensive rooting detections
- Runtime detection by applications

# Solution?

# Frida

<http://www.frida.re/>

```
Quick-start Instructions

~ $ sudo pip install frida
~ $ frida-trace -i "recv*" Twitter
recvfrom: Auto-generated handler: .../recvfrom.js
Started tracing 21 functions.
1442 ms    recvfrom()
# Live-edit recvfrom.js and watch the magic!
5374 ms    recvfrom(socket=67, buffer=0x252a618,
length=65536, flags=0, address=0xb0420bd8,
address_len=16)
```

# Frida

Provides a runtime environment that allows you to hook functions in an application and manipulate the flow, read data etc.

# Frida

## WORKSHOP!!!