Playing with skype

4knahs sl~~acking @~~work

For educational purposes only!
I do not support the use of any of the mentioned techniques for illegal activities..

Monkey taken from : http://www.websimians.com/

lame Disclaimer

Your viewing of these slides signifies your acceptance of the following Terms of Usage/Disclaimer. If you do not accept these terms, you should discontinue viewing this site.

You may use this knowledge for background, informational purposes only. You agree to use the information provided solely for your own noncommercial use and benefit, and only when in compliance with the terms of use of the application itself.

These slides are educational only. The creator of the slides has no intention to disrupt any skype service or incentivate the use of the presented techniques. Therefore any information here should be regarded as educational and **not to be practiced or distributed**.

The choice of application is merely due to its popularity and the challenge it presents. Although the writer of these slides only makes available a small subset of the altered code which is not enough to reverse engineer the behavior of the application or to distribute it as a new app, **you should not use it to replicate its work**.

For any concerns regarding the availability of these slides please feel free to contact me and I shall take them out.

Used tools

- **apktool** - for decompiling/compiling apk
- **jarsigner** - for signing apk
- **xposed** - for intercepting runtime methods

Dalvik opcodes : http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html

Quick tutorial

Altering smali and recompile:

- java -jar apktool.jar d -f -r Skype2.apk
- java -jar apktool.jar b -f Skype2 satan.apk
- jarsigner -verbose -sigalg MD5withRSA -digestalg SHA1 -keystore ~/.android/debug.keystore -storepass android -signedjar satan-sign.apk satan.apk androiddebugkey

Obfuscation and Tampering

Note that not all the smali files contain the original class and method names, this is often because apps use the android ProGuard (http://developer.android.com/tools/help/proguard.html) .

Fortunately it only removes unused stuff and attributes semantically obscure names.

Other techniques can be used though to detect interception of methods (probably expensive to verify on runtime), tampering of files (comparing hash of stuff), etc. One commercial example is http://www.saikoa.com/comparison_proguard_dexguard

Skype smali packages

After decompiling the .dex files, the smali folder should contain:

```
aknahs@hackatron:~/apktool/Skype2/smali$ ls
android  com  javax  net  roboguice
```

Lets investigate each of them in detail :)

smali/roboguice

Framework for easing android development, we can ignore it :)

“RoboGuice 2 smoothes out some of the wrinkles in your Android development experience and makes things simple and fun. Do you always forget to check for null when you getIntent().getExtras()? RoboGuice 2 will help you. Think casting findViewById() to a TextView shouldn’t be necessary? RoboGuice 2 is on it. (...)” - https://github.com/roboguice/roboguice

smali/android

Contains only a subfolder named support (v4 and v7). Most probably the **android support library**:

“The Android Support Library package is a set of code libraries that provide backward-compatible versions of Android framework APIs as well as features that are only available through the library APIs. Each Support Library is backward-compatible to a specific Android API level. This design means that your applications can use the libraries' features and still be compatible with devices running Android 1.6 (API level 4) and up.” - http://developer.android.com/tools/support-library/index.html

Ignore! :D

smali/net

Contains hockeyapp:

“HockeyApp is the best way to collect live crash reports, get feedback from your users, distribute your betas, and analyze your test coverage.” - http://hockeyapp.net/features/

I love ignore! <3

smali/javax

Contains javax.inject:

“This package specifies a means for obtaining objects in such a way as to maximize reusability, testability and maintainability compared to traditional approaches such as constructors, factories, and service locators (e.g., JNDI). This process, known as dependency injection, is beneficial to most nontrivial applications.” - http://docs.oracle.com/javaee/6/api/javax/inject/package-summary.html

Wonder what.. ignore..

smali/com

This folder contains all the cool folders:

```
aknahs@hackatron:~/Development/trapmaster/SmaliParser/Skype2/smali/com$ ls
a  b  c  flurry  google  jess  microsoft  qik  skype
```

- flurry - ads & analytics - http://www.flurry.com/
- jess - rule engine for java - http://herzberg.ca.sandia.gov/
- qik - mobile video sharing and capturing - http://qik.com/info/overview
- google:
  - inject - again inject, is used by guice (http://code.google.com/p/google-guice/)
  - android - contains the GCM code (http://developer.android.com/google/gcm/index.html)
- microsoft - microsoft advertising
- skype - most of the application logic

Interpreting smali

A cool way to infer the application behavior is to actually run it and observe its execution flow. Lets try it for skype ads :)

adb shell "su -c 'cat data/data/de.robv.android.xposed.installer/log/debug.log'" | grep 'com.skype.android.ads'

```
aknahs@hackatron:~/Development/trapmaster$ adb shell "su -c 'cat data/data/de.robv.android.xposed.installer/log/debug.log'" | grep LG_LVL_EXE | grep 'com.skype.android.ads'
Thu Nov 07 11:49:30 CET 2013| LG_LVL_EXE |com.skype.raider | com.skype.android.ads.SkypeAdManager()[thread:Thread-189] [calling obj=55fff0e2-5f7b-458d-b8a9-fde29eef046f]
Thu Nov 07 11:49:35 CET 2013| LG_LVL_EXE |com.skype.raider | com.skype.android.ads.SkypeAdManager.a(com.skype.android.app.SkypeApplication:nonPrimitiveOrJavaLang,java.lang.S
tring:call4knahs,java.lang.Integer:0,java.lang.Integer:0,java.lang.Long:21627597,java.lang.String:20703/4.4.0.31835,java.lang.String:20703/4.4.0.31835)[thread:events] [calli
ng obj=55fff0e2-5f7b-458d-b8a9-fde29eef046f]
Thu Nov 07 11:49:35 CET 2013| LG_LVL_EXE |com.skype.raider | com.skype.android.ads.SkypeAdTrackingManager.a()[thread:events] [calling obj=86bb40f4-28d6-40fb-ae24-ce403216234
8]
Thu Nov 07 11:49:35 CET 2013| LG_LVL_EXE |com.skype.raider | com.skype.android.ads.SkypeAdTrackingManager.b()[thread:events] [calling obj=3baeb22e-6840-471e-a55a-b18732d4e50
b]
Thu Nov 07 11:49:35 CET 2013| LG_LVL_EXE |com.skype.raider | com.skype.android.ads.SkypeAdManager.a(android.content.Intent:nonPrimitiveOrJavaLang)[thread:events] [calling ob
j=55fff0e2-5f7b-458d-b8a9-fde29eef046f]
Thu Nov 07 11:49:35 CET 2013| LG_LVL_EXE |com.skype.raider | com.skype.android.ads.SkypeAdManager.b(java.lang.Boolean:false)[thread:events] [calling obj=55fff0e2-5f7b-458d-b
8a9-fde29eef046f]
```

Xposed can intercept methods during runtime!

Interpreting smali

Execution within com.skype.android.ads goes something like this (not necessarily this order):

- SkypeAdManager()
- SkypeAdTrackingManager - interacts with com.microsoft.advertising
- SkypeAdPlacer - probably ad placing methods. interacts with scroll

Both Ad Tracking and Placer are invoked by AdManager :D So lets look at AdManager!

SkypeAdManager

There are actually 3 SkypeAdManager classes in com/skype/android/ads. A quick look at their superclasses tells:

- SkypeAdManager.smali - class that implements AdManager
- SkypeAdManager$1.smali - Broadcast receiver (innerclass)
- SkypeAdManager$2.smali - Broadcast receiver (innerclass)

```
.class final Lcom/skype/android/ads/SkypeAdManager$2;
.super Landroid/content/BroadcastReceiver;


# annotations
.annotation system Ldalvik/annotation/EnclosingClass;
    value = Lcom/skype/android/ads/SkypeAdManager;
.end annotation

.annotation system Ldalvik/annotation/InnerClass;
    accessFlags = 0x0
    name = null
.end annotation
```

SkypeAdManager

So what intents are these broadcast receivers intercepting?

Clearly this one is receiving information on the connectivity state!

What would happen if we tampered with the connectivity? :P

```
# virtual methods
.method public final onReceive(Landroid/content/Context;Landroid/content/Intent;)V
    .locals 3
    .parameter "context"
    .parameter "intent"

    .prologue
    invoke-virtual {p2}, Landroid/content/Intent;->getAction()Ljava/lang/String;

    move-result-object v1

    const-string v2, "android.net.conn.CONNECTIVITY_CHANGE"

    invoke-virtual {v1, v2}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z

    move-result v1

    if-eqz v1, :cond_0

    const-string v1, "noConnectivity"

    const/4 v2, 0x0

    invoke-virtual {p2, v1, v2}, Landroid/content/Intent;->getBooleanExtra(Ljava/lang/String;Z)Z

    move-result v0

    .local v0, networkOffline:Z
    iget-object v1, p0, Lcom/skype/android/ads/SkypeAdManager$2;->a:Lcom/skype/android/ads/SkypeAdManager;

    invoke-static {v1}, Lcom/skype/android/ads/SkypeAdManager;->a(Lcom/skype/android/ads/SkypeAdManager;)Z

    move-result v1

    if-eq v1, v0, :cond_0
```

end of function

SkypeAdManager

So what intents are these broadcast receivers intercepting?

```
# virtual methods
.method public final onReceive(Landroid/content/Context;Landroid/content/Intent;)V
    .locals 1
    .parameter "context"
    .parameter "intent"

    .prologue
    iget-object v0, p0, Lcom/skype/android/ads/SkypeAdManager$1;->a:Lcom/skype/android/ads/SkypeAdManager;

    invoke-static {v0, p2}, Lcom/skype/android/ads/SkypeAdManager;->a(Lcom/skype/android/ads/SkypeAdManager;Landroid/content/Intent;)V

    return-void
.end method
```

Print intent type during runtime..

This one calls a(SkypeAdManager,Intent) on SkypeAdManager

How to kill adds?

We can do lots of tricks we could perform in order to do this:

- Alter the SkypeAdManager files to NOOP:
  - Using Xposed module to detect method and cancel.
  - Alter the smali files in order to avoid the display of ads.
    - Less elegant way would to say its always offline.
    - Pretend to have premium account

- Discover who is instantiating SkypeAdManager and avoid it from the source :D
  - comment some smali line -> needs recompiling and signing
  - intercept method on Xposed -> harder but cleaner

Who creates it?

Looking at the execution flow, the AdManager is only created once:

```
Thu Nov 07 11:49:30 CET 2013| LG_LVL_EXE |com.skype.raider | com.google.inject.y$a.get(com.google.inject.internal.o:nonPrimitiveOrJavaLang)[
thread:Thread-189] [calling obj=03a4bc69-6805-4c87-b5a7-60b5a796103d]
Thu Nov 07 11:49:30 CET 2013| LG_LVL_EXE |com.skype.raider | com.skype.android.ads.SkypeAdManager()[thread:Thread-189] [calling obj=55fff0e2-
5f7b-458d-b8a9-fde29eef046f]
Thu Nov 07 11:49:30 CET 2013| LG_LVL_EXE |com.skype.raider | com.google.inject.y$a.get(com.google.inject.internal.o:nonPrimitiveOrJavaLang)[
thread:Thread-189] [calling obj=bf3fdaad-8e5a-4752-ae62-c1cc1f8dd3f0]
```

And grepping the smali files for “SkypeAdManager” gives only one class outside of android ads folder:

```
smali/com/skype/android/SkypeModule.smali:    const-string v4, "com.skype.android.ads.SkypeAdManager"
```

Why SkypeModule? Remember the package guice? That thing to ease the use of factories in java… well, a guice module is generally the class that binds the specific classes.

SkypeModule

Skype binds all Ad classes within a nice try catch.

Lets try to comment all ad related stuff and run to see how it goes.

In Xposed, we would have to replicate the method without the try part or by throwing this exception

```
:try_start_0
const-class v3, Lcom/skype/android/ads/AdPlacer;

invoke-virtual {p0, v3}, Lcom/skype/android/SkypeModule;->bind(Ljava/lang/Class;)Lcom/google/inject/a/a;

move-result-object v3

const-string v4, "com.skype.android.ads.SkypeAdPlacer"

invoke-static {v4}, Ljava/lang/Class;->forName(Ljava/lang/String;)Ljava/lang/Class;

move-result-object v4

const-class v5, Lcom/skype/android/ads/AdPlacer;

invoke-virtual {v4, v5}, Ljava/lang/Class;->asSubclass(Ljava/lang/Class;)Ljava/lang/Class;

move-result-object v4

invoke-interface {v3, v4}, Lcom/google/inject/a/a;->to(Ljava/lang/Class;)Lcom/google/inject/a/f;

move-result-object v3

const-class v4, Lcom/google/inject/Singleton;

invoke-interface {v3, v4}, Lcom/google/inject/a/f;->in(Ljava/lang/Class;)V

.line 104
const-class v3, Lcom/skype/android/ads/AdManager;

invoke-virtual {p0, v3}, Lcom/skype/android/SkypeModule;->bind(Ljava/lang/Class;)Lcom/google/inject/a/a;

move-result-object v3
```

```
:try_end_0
.catch Ljava/lang/ClassNotFoundException; {:try_start_0 .. :try_end_0} :catch_0
```

First experiment

By commenting everything within the try and recompiling, signing and installing this what happens:

```
E/AndroidRuntime( 4183): FATAL EXCEPTION: Thread-197
E/AndroidRuntime( 4183): com.google.inject.as: Guice provision errors:
E/AndroidRuntime( 4183):
E/AndroidRuntime( 4183): 1) No implementation for com.skype.android.ads.AdManager was bound.
E/AndroidRuntime( 4183):   while locating com.skype.android.ads.AdManager
E/AndroidRuntime( 4183):     for field at com.skype.android.app.signin.AccountAgent.adManager(Unknown Source)
E/AndroidRuntime( 4183):   while locating com.skype.android.app.signin.AccountAgent
E/AndroidRuntime( 4183):   while locating com.skype.android.inject.AgentBootstrapProvider
E/AndroidRuntime( 4183):   while locating com.skype.android.app.AgentBootstrap
E/AndroidRuntime( 4183):
E/AndroidRuntime( 4183): 1 error
E/AndroidRuntime( 4183):        at com.google.inject.ac$4.get(InjectorImpl.java:767)
E/AndroidRuntime( 4183):        at com.google.inject.ac.getInstance(InjectorImpl.java:793)
E/AndroidRuntime( 4183):        at roboguice.inject.ContextScopedRoboInjector.getInstance(ContextScopedRoboInjector.java:140)
E/AndroidRuntime( 4183):        at com.skype.android.app.main.SplashActivity.doStuffInBackground(SplashActivity.java:99)
E/AndroidRuntime( 4183):        at roboguice.activity.RoboSplashActivity$1.run(RoboSplashActivity.java:40)
E/AndroidRuntime( 4183):        at java.lang.Thread.run(Thread.java:856)
```

Lets clean this

Unfortunately, it is not resilient, as expected, the bound classes are used in multiple places.

Incremental cleanup

Lets start (smartly) commenting all com.skype.android.ads entries in AccountAgent (without breaking functionality) :)

Methods with commented code:

- initializeAdComponent(Lcom/skype/Account;)V
- handleSubscriptionChange()V
- handleSkypeoutBalanceChange()V
- onLogout()V
- handleSkypeoutBalanceChange()V
- handleSubscriptionChange()V

```
.method private handleSkypeoutBalanceChange()V
    .locals 2

    .prologue
    .line 222
    #iget-object v1, p0, Lcom/skype/android/app/signin/AccountAgent;->adManager:Lcom/skype/android/ads/AdManager;
    #invoke-virtual {p0}, Lcom/skype/android/app/signin/AccountAgent;->getAccount()Lcom/skype/Account;

    #move-result-object v0

    #invoke-virtual {v0}, Lcom/skype/Account;->getSkypeoutBalanceProp()I

    #move-result v0

    #const/4 v0, 0x1

    #if-lez v0, :cond_0

    #const/4 v0, 0x1

    :goto_0
    #invoke-interface {v1, v0}, Lcom/skype/android/ads/AdManager;->b(Z)V

    .line 223
    return-void

    .line 222
    :cond_0
    const/4 v0, 0x0

    goto :goto_0
.end method
```

Useless methods...

```
.line 218
:cond_2
#iget-object v3, p0, Lcom/skype/android/app/signin/AccountAgent;->adManager:Lcom/skype/android/ads/AdManager;

#invoke-interface {v3, v0}, Lcom/skype/android/ads/AdManager;->a(Z)V
```

If we would either fake connection or change to premium there would be no ads

Interesting findings

With what data are the ads initialized?

```
.method private initializeAdComponent(Lcom/skype/Account;)V
```

```
.end local v2           #username:Ljava/lang/String;
.end local v3           #birthday:I
.end local v4           #gender:I
.end local v7           #lastVersion:Ljava/lang/String;
.end local v8           #currentVersion:Ljava/lang/String;
```

```
invoke-interface/range {v0 .. v8}, Lcom/skype/android/ads/AdManager;->a(
Landroid/content/Context;Ljava/lang/String;IIJLjava/lang/String;Ljava/lang/String;)V
```

A bit more research and we would find the rest of the arguments :)

loop!

Greping for com/skype/android/ads would be faster but is nice to see skype crash looping :D

```
1) No implementation for com.skype.android.ads.AdManager was bound.
  while locating com.skype.android.ads.AdManager
    for field at com.skype.android.app.calling.CallAgent.adManager(Unknown Source)
```

Methods changed: onEvent(Lcom/skype/android/gen/ConversationListener$OnPropertyChange;)V

```
1) No implementation for com.skype.android.ads.AdPlacer was bound.
  while locating com.skype.android.ads.AdPlacer
    for field at com.skype.android.app.main.HubActivity.adPlacer(Unknown Source)
```

Methods changed: onPageScrolled(IFI)V ; onPause()V ; OnResume()V

```
1) No implementation for com.skype.android.ads.AdPlacer was bound.
  while locating com.skype.android.ads.AdPlacer
    for field at com.skype.android.app.recents.RecentListFragment.adPlacer(Unknown Source)
```

Methods changed: done(Lcom/skype/async/AsyncResult;)V ; onViewCreated(Landroid/view/View;Landroid/os/Bundle;)V

```
1) No implementation for com.skype.android.ads.AdPlacer was bound.
  while locating com.skype.android.ads.AdPlacer
    for field at com.skype.android.app.contacts.ContactListFragment.adPlacer(Unknown Source)
```

Methods changed: onViewCreated(Landroid/view/View;Landroid/os/Bundle;)V

It works!!...???

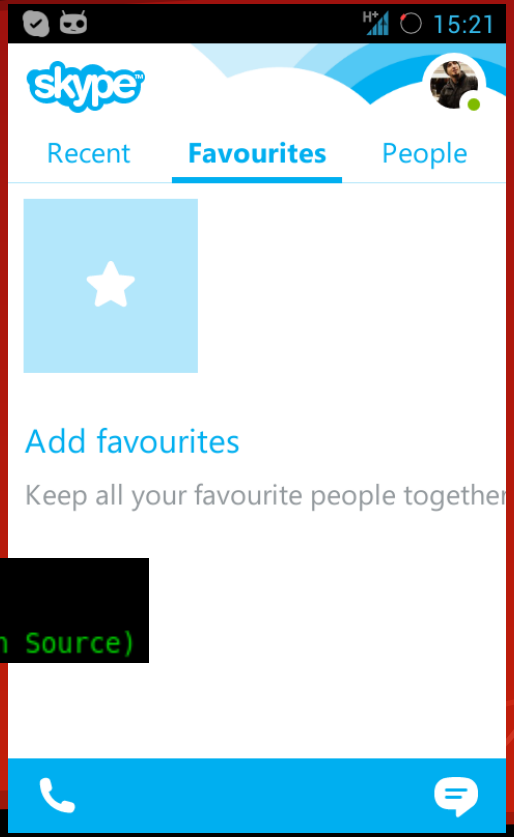
Now Skype works! :D

...but we get again a crash during phone calls...

```
1) No implementation for com.skype.android.ads.AdPlacer was bound.
  while locating com.skype.android.ads.AdPlacer
    for field at com.skype.android.app.calling.InCallFragment.adPlacer(Unknown Source)
```

Methods changed: handleVideoOrientationChange(Landroid/content/res/Configuration;)V

onResume()V



Lets clear all!

So before reattempting to run, lets check if we cleaned all:

```
Skype2/smali/com/skype/android/app/main/HubSection.smali:.field private adPlacement:Lcom/skype/android/ads/AdPlacement;
Skype2/smali/com/skype/android/app/main/HubSection.smali:    sget-object v5, Lcom/skype/android/ads/AdPlacement;->
a:Lcom/skype/android/ads/AdPlacement;
Skype2/smali/com/skype/android/app/main/HubSection.smali:    invoke-direct/range {v0 .. v5}, Lcom/skype/android/app/main/HubSection;-><init>(
Ljava/lang/String;IILcom/skype/android/analytics/AnalyticsEvent;Lcom/skype/android/ads/AdPlacement;)V
Skype2/smali/com/skype/android/app/main/HubSection.smali:    sget-object v8, Lcom/skype/android/ads/AdPlacement;->
b:Lcom/skype/android/ads/AdPlacement;
Skype2/smali/com/skype/android/app/main/HubSection.smali:    invoke-direct/range {v3 .. v8}, Lcom/skype/android/app/main/HubSection;-><init>(
Ljava/lang/String;IILcom/skype/android/analytics/AnalyticsEvent;Lcom/skype/android/ads/AdPlacement;)V
Skype2/smali/com/skype/android/app/main/HubSection.smali:    sget-object v8, Lcom/skype/android/ads/AdPlacement;->
c:Lcom/skype/android/ads/AdPlacement;
Skype2/smali/com/skype/android/app/main/HubSection.smali:    invoke-direct/range {v3 .. v8}, Lcom/skype/android/app/main/HubSection;-><init>(
Ljava/lang/String;IILcom/skype/android/analytics/AnalyticsEvent;Lcom/skype/android/ads/AdPlacement;)V
Skype2/smali/com/skype/android/app/main/HubSection.smali:.method private constructor <init>(
Ljava/lang/String;IILcom/skype/android/analytics/AnalyticsEvent;Lcom/skype/android/ads/AdPlacement;)V
Skype2/smali/com/skype/android/app/main/HubSection.smali:            "Lcom/skype/android/ads/AdPlacement;",
Skype2/smali/com/skype/android/app/main/HubSection.smali:    iput-object p5, p0, Lcom/skype/android/app/main/HubSection;->
adPlacement:Lcom/skype/android/ads/AdPlacement;
Skype2/smali/com/skype/android/app/main/HubSection.smali:.method public final getAdPlacement()Lcom/skype/android/ads/AdPlacement;
Skype2/smali/com/skype/android/app/main/HubSection.smali:    iget-object v0, p0, Lcom/skype/android/app/main/HubSection;->
adPlacement:Lcom/skype/android/ads/AdPlacement;
```

Just missing HubSection smali and we should be done!

Best for last

Ofc the last is always the toughest. It is some spaghetti logic to circumvent. Since we don't want to change the method signatures lets just replace all the ads arguments with null and treat them correctly in the respective constructors:

```
#sget-object v8, Lcom/skype/android/ads/AdPlacement;->b:Lcom/skype/android/ads/AdPlacement;
#added null in v8
const v8, 0

move v5, v9

move v6, v9

invoke-direct/range {v3 .. v8}, Lcom/skype/android/app/main/HubSection;-><init>(
Ljava/lang/String;IILcom/skype/android/analytics/AnalyticsEvent;Lcom/skype/android/ads/AdPlacement;)V
```

---> Methods altered : <clinit>()V,

<init>(Ljava/lang/String;IILcom/skype/android/analytics/AnalyticsEvent;Lcom/skype/android/ads/AdPlacement;)V

Done!

Congrats! We have a fully functional (not to be distributed or used) **skype app without ads :)**

Now go on and delete it! :P

Smali files here :

http://tinyurl.com/nl3hn67

(pass: 4knahs - this link will autodestruct in 30 days)