How we tear into that little green man
Who are you?!

Mathew Rowley (@wuntee)
Senior security consultant at Matasano
Agenda

- Techniques
  - MITM - SSL
  - Static analysis -> Skype secret menu
  - Modifying an app -> Injecting Java to APK
  - Manual debugging without source
  - Automated debugging (NEW TOOLS: JavaTap, CryptoTap)
technique one

MITM - SSL
We need to trick the android device to think that the malicious user is "http://somesite.com"

That is not good enough though

Android HTTPS communication will fail

Tools to help with SSL MITM:

- Mallory
- Burp
SSL

- By default SSL certificate checking is performed by the Android OS
- Optionally, developers can add or override those checks with their own
- This presents two problems when attempting to MITM an SSL communication stream
  - How do we trick the OS to trust endpoint
  - How can we get around custom developer checks
Circumventing OS checks

- Install the CA.
- There is no interface for doing this.
- Demos...
Before ICS

```bash
$ cd $JAVA_HOME/lib/ext
$ sudo wget 'http://www.bouncycastle.org/download/bcprov-jdk15on-147.jar'
$ cd -
$ adb pull /system/etc/security/cacerts.bks
$ keytool -keystore cacerts.bks -storetype BKS -provider org.bouncycastle.jce.provider.BouncyCastleProvider -storepass changeit -importcert -trustcacerts -alias idontcare -file ca.cer
$ adb shell mount | grep system
$ adb shell mount -o remount,rw /dev/block/mtdblock0 /system
$ adb shell chmod 777 /system/etc/security/cacerts.bks
$ adb push cacerts.bks /system/etc/security/cacerts.bks
$ adb shell chmod 644 /system/etc/security/cacerts.bks
```
ICS+

$ adb shell mount -o remount,rw /dev/block/mtdblock0 /system

$ openssl x509 -in [cert] -text -noout > cert.txt

$ adb push cert.txt /system/etc/security/cacerts/
Caveats

• Emulator
  - Does not persist changes to the .img file. Bug reported, was supposed to be fixed in r17 (still not fixed - android 4.1.2)
  - Bug or feature?
  - Can be done with unyaffs/mkyaffs2image

• Ice Cream Sandwitch
  - Switched from BouncyCastle to a directory of x509 text certificates
Persistent /system workaround

- The current /system image is:
  - /tmp/android-[user]/emulator-[rand]
- Make changes
- Before powering off, copy that somewhere
- emulator -system [copy] @[avd]
Preventative Measures

- Custom certificate checks
  - This is done by extending X509TrustManager
    - (void) checkServerTrusted(...)
- Can we get around this? Yes...
  - Baksmali
  - Modify smal source
  - re-smali
  - re-package
  - re-sign
$ java -jar apktool.jar d X509ModifiedSSL.apk
$ grep -ri x509 * | grep -i implements
-- Edit file to have 'checkServerTrusted' return void
$ java -jar apktool.jar b X509ModifiedSSL X509ModifiedSSL-mod.apk
$ keytool -genkey -v -keystore keystore.ks -alias idontcare -keyalg RSA -keysize 2048 -validity 10000
$ jarsigner -keystore keystore.ks X509ModifiedSSL-mod.apk idontcare
$ adb install X509ModifiedSSL-mod.apk
I have SSL MITM, now what?

- This will typically be used to attack backend infrastructure, not the device
- Backend servers typically expect sanitized input, and don't expect non-device interaction.
- What I have seen:
  - 101 read/write of arbitrary files through SOAP interface
  - XML entity inclusion to gain access to private ssh key
  - Ability to arbitrarily lock/wipe another person's device
How to prevent MITM?

- You can’t... Remember the device is out of your trust zone. ALWAYS treat user input as untrusted

- But, we can make it really hard
  - Real obfuscation - means obfuscating strings
  - Custom certificate checks
  - Authenticate client app - The application has access to itself (APK), send a custom hash of it for each request
technique two
Static analysis
It’s a skill
Project Setup

- jar - dex2jar(*.apk)
- src - jdgui(*.jar)
- unapk - baksimali(*.apk)
- unzip - unzip(*.apk)
#!/bin/bash

# Scripts
apktool="/Applications/hacking/apktool-install-macosx-r04-brut1/apktool"
dex2jar="/Applications/hacking/dex2jar-0.0.9.7/dex2jar.sh"
jdgui="/Applications/JD-GUI.app/Contents/MacOS/jd-gui"
apk=$1
mkdir jar src unapk unzip
# Unzip
pushd unzip
unzip ../$apk
popd
# Unapk
pushd unapk
$apktool d -f ../$apk .
popd
# Get jar
$dex2jar $apk
mv *.jar ./jar
# Decompile
$jdgui jar/*.jar
Example
Skypes hidden menu

- Logcat... What is skype.properties?

```
12:01:16:807  info  ActivityManager  61  Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] }
12:01:17:09  info  ActivityManager  61  Start proc com.skype.raider for activity com.skype.raider/.Main: pid=347
12:01:18:621  info  com.skype.raider.MainApp  347  onCreate +
12:01:18:621  info  com.skype.raider.MainApp  347  DEFAULTING TO PRODUCTION CONFIGURATION SETTINGS
12:01:18:664  info  com.skype.raider.MainApp  347  onCreate -
12:01:18:704  verbose  com.skype.ni  347  /data/data/com.skype.raider/lib/libvheadhost_skype.so
12:01:18:730  warn  com.skype.mt  347  No config file name: /mnt/sdcard/skype.properties
12:01:18:762  warn  com.skype.mt  347  No config file name: /mnt/sdcard/skype.properties
12:01:18:788  warn  com.skype.mt  347  No config file name: /mnt/sdcard/skype.properties
12:01:18:833  warn  com.skype.mt  347  No config file name: /mnt/sdcard/skype.properties
12:01:18:861  warn  com.skype.qp  347  data class: com.skype.LiveData
12:01:19:119  debug  dalvikvm  347  GC_EXTERNAL_ALLOC freed 331K, 48% free 3255K/6215K, external 1688K
12:01:19:281  debug  dalvikvm  347  GC_EXTERNAL_ALLOC freed 20K, 48% free 3278K/6215K, external 32K
12:01:19:395  debug  FlurryAgent  347  Starting new session
12:01:19:412  warn  Settings  347  Setting android_id has moved from android.provider.Settings.System (foreground) not calling setShow) because view stack is empty
12:01:19:453  warn  com.skype.DefaultUINavig...  347  CONNECTIVITY_ACTION connected:true
12:01:19:482  verbose  com.skype.kit.dy  347  ...
```
Locate string “skype.properties”

- grep is your friend
- Notice how it is only located in .smali files
- What about .java?

```
[16:06:29] wuntee:@/matasano/tmp$ ls -l
total 20128
-rw-r--r-- 1 wuntee staff 10297777 Apr 18 16:05 com.skype.raider-1.apk
-rwxr-xr-x 1 wuntee staff 424 Apr 18 16:05 decompileAPK.sh
drwxr-xr-x 3 wuntee staff 102 Apr 18 16:05 jar
drwxr-xr-x 5 wuntee staff 170 Apr 18 16:06 src
drwxr-xr-x 8 wuntee staff 272 Apr 18 16:05 unapk
drwxr-xr-x 9 wuntee staff 306 Apr 18 16:05 unzip

[16:06:30] wuntee:@/matasano/tmp$ grep -ri skype.properties *
unapk/smalicom/skype/mt.smali: const-string v2, "skype.properties"
unapk/smalicom/skype/mt.smali: const-string v1, "/sdcard/skype.properties"
unapk/smalicom/skype/mt.smali: const-string v2, "/mnt/sdcard/skype.properties"
unapk/smalicom/skype/mt.smali: const-string v2, "skype.properties"
Binary file unzip/classes.dex matches
```
move-result-object v1
const-string v2, "skype.properties"
invoke-virtual {v1, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
move-result-object v1
invoke-virtual {v1}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;
move-result-object v1
aput-object v1, v0, v7
const-string v1, "/sdcard/skype.properties"
aput-object v1, v0, v8
const/4 v1, 0x2
const-string v2, "/mnt/sdcard/skype.properties"
aput-object v2, v0, v1

String str4 = I;
StringBuilder localStringBuilder33 = localStringBuilder32.append(str4);
StringBuilder localStringBuilder34 = localStringBuilder1.append(" kitLogging: ");
boolean bool11 = o;
StringBuilder localStringBuilder35 = localStringBuilder34.append(bool11);
return localStringBuilder1.toString();

// ERROR //
public static final void a(android.app.Application paramApplication)
{
// Byte code:
// 0: ldc 2
// 2: invokevirtual 224 java/lang/Class:getName ()Ljava/lang/String;
// 5: invokevirtual 229 com/skype/tj:a (Ljava/lang/String;)Z
// 8: ifeq +44 == 52
// 11: ldc 2
// 13: invokevirtual 224 java/lang/Class:getName ()Ljava/lang/String;
// 16: astore_1
// 17: new 153 java/lang/StringBuilder
// 20: dup
// 21: invokevirtual 230 java/lang/StringBuilder:<init> ()V
// 24: ldc 232
// 26: invokevirtual 164 java/lang/StringBuilder:append (Ljava/lang/String;)Ljava/lang/StringBuilder;
// 29: astore_2
// 33: astore_3
// 34: aload_2
// 35: aload_1

skype.properties processing

```java
:try_start_ce
const-string v4, "login"
invoke-virtual {v5, v4}, Ljava/util/Properties;->containsKey(Ljava/lang/Object;)
move-result v4
if-eqz v4, :cond_e7
const-string v4, "login"
invoke-virtual {v5, v4}, Ljava/util/Properties;->getProperty(Ljava/lang/String;)
move-result v4
call v4, v8
:goto_e5
sput-boolean v4, Lcom/skype/mt;->z:
:cond_e7
const-string v4, "daemon"
invoke-virtual {v5, v4}, Ljava/util/Properties;->containsKey(Ljava/lang/Object;)
move-result v4
if-eqz v4, :cond_100
const-string v4, "daemon"
invoke-virtual {v5, v4}, Ljava/util/Properties;->getProperty(Ljava/lang/String;)
move-result v4
call v4, v8
:goto_e5
sput-boolean v4, Lcom/skype/mt;->z:
```

```java
login=Boolean
daemon=Boolean
update=Boolean
answer=Boolean
callVoiceMailDelay=Integer
videoQualityLow=Integer
userWantsVideo=Boolean
checkSharedXML=Boolean
kit.logging=Boolean
debugMenu=Boolean
test.monkey.enabled=Boolean
test.acs.enabled=Boolean
test.video=Boolean
test.skypename=String
test.password=String
videoInfo=String
```
The test...

- Create skype.properties with "debugMenu=1" and place it on SD card
Logging...
Preventing Static Analysis/Decompilation

- You can’t...
- But, we can make it really hard
  - Real obfuscation - means obfuscating strings
  - Disable logging
  - Cause decompilers to crash
    - Blackhat 2012 - Practicing safe DEX [1]
    - APKfuscator [2]

HackersChallenge...

- Challenge for Matasano coworkers - nobody has completed it
- Performs an HTTPS request to get a "secret" of a character from the movie Hackers, then displays it
- Goal is to figure out everyones secret
- Preventative measures
  - Tamper proof - custom hash of APK
  - Custom X509TrustManager
technique three

Modifying an application
field static staticString:Ljava/lang/String;
.method static constructor <clinit>()V
    .registers 1
    .prologue
    .line 8
    const-string v0, "static string"
    sput-object v0, Lcom/wuntee/Downloader;->staticString:Ljava/lang/String;
    .line 7
    return-void
.end method
.method public constructor <init>()V
    .registers 1
    .prologue
    .line 7
    invoke-direct {p0}, Landroid/app/Activity;-><init>()V
    return-void
.end method
# virtual methods
.method public onCreate(Landroid/os/Bundle;)V
    .registers 5
    .parameter "savedInstanceState"
    .prologue
    .line 11
    invoke-super {p0, p1}, Landroid/app/Activity;->onCreate(Landroid/os/Bundle;)V
    .line 12
    const/high16 v1, 0x7f03
    invoke-virtual {p0, v1}, Lcom/wuntee/Downloader;->setContentView(I)V
    .line 13
    const-string v0, "tesstest2"
    .line 14
    .local v0, anotherString:Ljava/lang/String;
    sget-object v1, Lcom/wuntee/Downloader;->staticString:Ljava/lang/String;
    invoke-virtual {v0, v1}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z
    move-result v1
    if-eqz v1, :cond_19
    .line 15
    sget-object v1, Lcom/wuntee/Downloader;->staticString:Ljava/lang/String;
    sget-object v2, Lcom/wuntee/Downloader;->staticString:Ljava/lang/String;
    invoke-static {v1, v2}, Landroid/util/Log;->d(Ljava/lang/String;Ljava/lang/String;)I
    .line 17
    :cond_19
    return-void
.end method

GET IN MY BELLY!!

KAMIKAZE LUNCHTIME!!
Normal compilation path:

```
.java      .class     .dex      .apk
  |         |            |           |
  javac    dx          zip          
```

Simple modification path:

```
.apk     .dex    .smali    .dex    .apk
  |     unzip    baksmali  samli  zip   
```

Logic

Thursday, October 25, 12
Why cant I just...
You can. However...

- Nuances
  - Smali calling convention
  - Namespaces

- Demo (HackersChallenge - print hash algorithm output)...

Thursday, October 25, 12
So what?

- Simple example of injecting arbitrary Java into an Android application
- Malware?
- Debugging obfuscation?
- Control circumvention aka “cracks”? 
- You can inject any Java source into ANY application!
technique four

Debugging applications without source
Debugging

- System.out.println(debug_info) or attach to a debugger?
- Android applications are debugged via the typical Java Platform Debug Architecture (JPDA)
- Interacting with an Android application
  - Eclipse - GUI
  - JDB - command line
  - Java JPDA API
  - Ruby/JRuby with jdi-hook
- Caveats
  - You need to know “where” you must breakpoint
- Benefits
  - See method arguments and return values
  - HackersChallenge?
- Demo (HackersChallenge - obtain hash value without modification)...
Ok... And?

- Debugging is a key reversing concept
- Relate this to debugging using GDB or ImmunityDebugger/Ollydbg
- Not only can you break on local methods, but you can on core Android methods as well. Ever want to see all IPC?
  - Break: android.content.Intent.<init>
JavaTap/CryptoTap

- Originally part of OterTool
- Simplify debugging of an application
- CryptoTap
  - Original problem - pull AES keys from obfuscated application
  - Attach or launch a java process, and it will spit out all crypto info
- JavaTap
  - Config file defines entry/exit methods
  - Prints out passed in values and return values
Live demo...
[21:24:35] wuntee:$
Force Application to Wait for debugger

adb shell am set-debug-app -w [app]
Prevention

- You can not
- AndroidManifest.xml - debuggable=false
  - What does this even do? I can still use 'am' to attach to it...
- If for some reason you can not attach, apktool can pull out the manifest, you can modify and re-packages
Live File Browser
sqlite/text/hex
Questions?

mathew rowley (@wuntee)
mathew@matasano.com

otertool - https://github.com/wuntee/

LOOKING FOR A JOB IN SECURITY CONSULTING? (NYC, Chicago, Mountain View)
Send me your resume