



# Remoting Android applications for fun & profit

Damien Cauquil, Pierre Jaury

Hack In Paris  
June 20, 2013

## Introduction

Damien Cauquil

**Company** Sysdream (head of research)

**Twitter** @virtualabs

**Blog** <http://virtualabs.fr>

Pierre Jaury

**Company** Sysdream

**Twitter** @kaiyou\_

**Blog** <http://kaiyou.fr>

Sysdream, IT security services

**Location** Paris, France

**Website** <http://sysdream.com>

# Table Of Contents

- 1 Android remoting 101
- 2 Fino architecture
- 3 Exploring and reversing
- 4 DTMF Fuzzing
- 5 Fun & profit

# Android remoting 101

- 1 Android remoting 101
  - To debug or not to debug
  - Remoting vs. Debugging
  - Root?
- 2 Fino architecture
- 3 Exploring and reversing
- 4 DTMF Fuzzing
- 5 Fun & profit

# To debug or not to debug

- Some ways to debug applications
  - Dalvik Debugging Monitor Server combined with JDWP
  - Android logcat
- Many drawbacks:
  - Very difficult to alter and monitor the target application state
  - Requires Android Debug Bridge (ADB)
  - Bytecode level
- Why debugging?
  - Debugging android apps is very useful for developers . . .
  - But not convenient for reverse engineers (*sic*) !

# Remoting vs. Debugging

App phone home

Debugging is so low-level and hardcore, why not getting a higher level view of an application and its components?

- Many benefits:
  - Abstraction of Android Dalvik VM bytecode
  - Better idea of how the application works
  - Java-like access to core components or the application itself
  - Bypass OOP restrictions
- Objectives:
  - Interact with the target application
  - Automate complex processes through scripting

# Remoting vs. Debugging

## First infection

- How?
  - Through the injection of a service running inside the application context
  - Compatible with Android > 2.0
  - Remotely controlled over the service API
- Limitations:
  - Cannot send your private information to the NSA
  - Can only interact with known and launched activities or services
  - Cannot interact with native applications

# Root?

Where we go, we don't need root.

- Rooting your Android phone ...
  - May void its warranty
  - May alter the behavior of your phone
  - May be detected by an application
- Remoting applications does not require:
  - Root access to the phone
  - USB debugging
  - Any special option set



- 1 Android remoting 101
- 2 **Fino architecture**
  - Application components
  - General overview
- 3 Exploring and reversing
- 4 DTMF Fuzzing
- 5 Fun & profit

# Application components

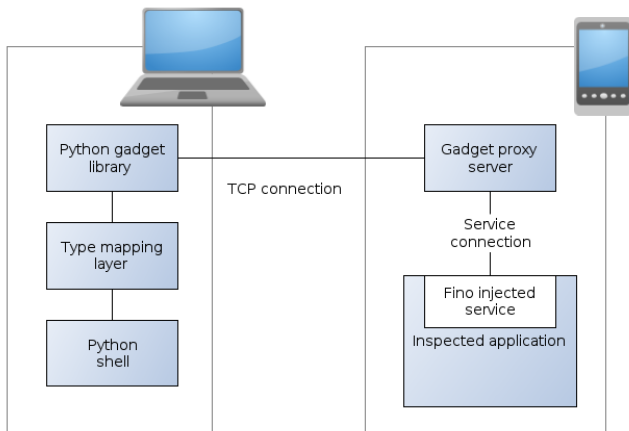
Components... so many components

- Fino**
- Hundreds of bytes of dalvik bytecode
  - Provides a minimal inspection API
  - Listens for **service** connections
  - Dynamic macro loading

- Gadget**
- Listens for **network** connections
  - Forwards calls to the Fino service

- Client**
- Python-driven gadget client
  - Handles modules and uploads macros

## General overview



# Exploring and reversing

- 1 Android remoting 101
- 2 Fino architecture
- 3 Exploring and reversing**
  - Getting the original APK
  - Injecting
  - Installing
- 4 DTMF Fuzzing
- 5 Fun & profit

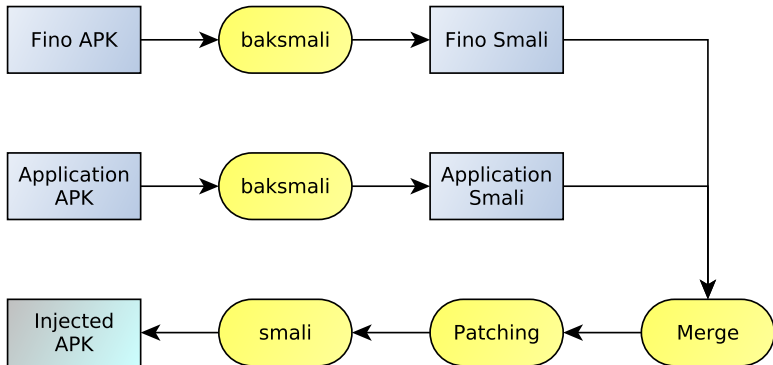
## Getting the original APK

- Some usual adb magic
- Must have USB debug and root access enabled

```
adb shell su -c 'ls /data/app'
```

- Some more adb pull
- Any other APK source is valid!

# Injecting



# Installing

Then have fun!

- Some more adb magic!

```
adb install package.apk
```

- Be careful with certificate inconsistency
- Run the Gadget server

# DTMF Fuzzing

- 1 Android remoting 101
- 2 Fino architecture
- 3 Exploring and reversing
- 4 DTMF Fuzzing**
  - IVR testing
  - Android as an IVR testing platform
  - Creating the system application
  - Dial, send DTMF and record conversation
  - Sign and install
  - Let's fuzz !
- 5 Fun & profit



# IVR testing

## What is DTMF fuzzing about?

- Interactive Voice Response Service
  - Provides a voice service to customers (i.e. answering service)
  - Interaction is DTMF-based
- Fuzzing DTMF tones
  - DTMF: **Dual-Tone Multi-Frequency**
  - Supported by a large number of phones
  - Fuzzing consists in sending a large amount of randomly generated DTMF sequences

# Android as an IVR testing platform

Reveal the power of the Droid

- Usual IVR testing systems
  - Not affordable (Call Master, NuBot, ...)
  - Requires specific hardware
- Android phones
  - Open-source
  - Cheap
  - May be tweaked to allow IVR testing

# Android as an IVR testing platform

Simon says, make a phone call!

## App

- Custom system application
- Provides an interface with the phone
- Fino service already injected


## DTMFFuzz

- Python controller based on Gadget
- Drives the DTMF fuzzing application

## Creating the system application

- 1 Root your Android phone
- 2 Patch ADT to allow access to `com.android.internal.*`<sup>1</sup>
- 3 Build an `android.jar` with Android internal classes
- 4 Hack into the Phone application through `AndroidManifest.xml` and reflection
- 5 Compile, sign and install

---

<sup>1</sup><http://virtualabs.fr/msi/android-core-hacking.pdf> 

## Creating the system application

Knock knock, Neo.

- Import some internals

```
import com.android.internal.telephony.*;
```

- Get a Phone instance with a line of Java

```
Phone phone = PhoneFactory.getDefaultPhone();
```

- Modify the `AndroidManifest.xml` to start the application inside the Phone application process

```
<activity android:process='com.android.phone' />
```

# Dial, send DTMF and record conversation

## PRISM ?

- Use `Phone.dial()` method
- To send DTMF, get a `Call` object and use `sendDtmf()`

```
Call call = phone.getForegroundCall();  
call.getPhone().sendDtmf('1');
```

# Dial, send DTMF and record conversation

PRISM ?

- To record, Android provides `android.media.MediaRecorder`
  - 1 Set audio source (`VOICE_DOWNLINK`)
  - 2 Set output format (`THREE_GPP`)
  - 3 Set audio encoder and output file
  - 4 Start recording (call `start()`)

## Sign and install

Like a boss.

- Signing requires your custom ROM certificates, public and private keys
- Remount root and drop into `/system/app/`

```
# adb remount  
# adb push DTMFuzz.apk /system/app/
```



## Let's fuzz !

1\*#098675#\*\*0875#\*747654765

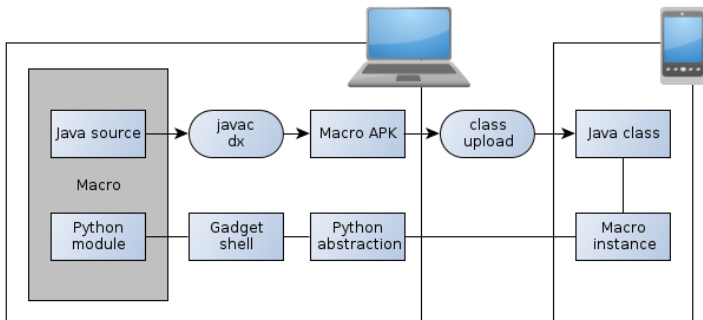
- Connect to the fuzzing system app
- Remote control the system app
- Retrieve the conversation record



- 1 Android remoting 101
- 2 Fino architecture
- 3 Exploring and reversing
- 4 DTMF Fuzzing
- 5 Fun & profit
  - Uploading macros
  - Let's cheat!

## Uploading macros

- Compiled to dex, packed as apk
- Shipped over the network
- Class loaded or replaced dynamically



# Let's cheat!

Sounds like good old times



## Conclusion

**Fino** `github.com/sysdream/fino`

**Gadget** `github.com/sysdream/gadget`

**Client** `github.com/sysdream/gadget-client`

