Analysis in Practice: Compositional Analysis of Android inter-app Security Vulnerabilities

FQ 2016
IN4MATX 221

Alireza Sadeghi
Outline

Motivation

- Mobile Security Threats
- Android Overview
- Inter-app Vulnerability

COVERT

- Static Analysis
- Formal Verification
- Challenges
- Evaluation
- Demo
Motivation

- Mobile Security Threats
  - Android Overview
  - Inter-app Vulnerability

COVERT

- Static Analysis
- Formal Verification
- Challenges
- Evaluation
- Demo
The Rise of Mobile

Internet Users (Millions)


Source: comSCORE
The Rise of Mobile

3.65 Billion
unique global mobile users

2:54 hours/day
Time spent on mobile device by Americans

$400 Billion
U.S. Mobile Revenue

55%
Share of Mobile devices for U.S Internet usage

Sources: TechCrunch, CNN Money, Statista, eMarketer
The Rise of Mobile Security Threats

Total Mobile Malware

Source: McAfee
The Rise of Mobile Security Threats

Stagefright: It Only Takes One Text To Hack 950 Million Android Phones

Six critical vulnerabilities have left 95 per cent of Google Android phones open to an attack, a security expert warned today. In some cases, where phones parse the attack code prior to the message, the user would have little chance of defending their data. The vulnerabilities are said to be the worst Android bugs ever recorded.

Joshua Drake, from Zimperium zLabs, who reported the bugs in April this year, said whilst Google and most manufacturers have not made fixes available to protect their customers. “All devices should be fixed as a matter of priority,” he told FORBES. He believes as many as 950 million Android phones are vulnerable.

Stagefright: It Only Takes One Text To Hack 950 Million Android Phones

Our Android smartphone could be attacked with a text message

stickers could take over Android smartphones with nothing more than mobile number, according mobile security experts at cybersecurity firm

Stagefright
Android is the Primary Target

- Android: 96.54%
- Symbian: 3.45%
- iOS: 0.00001%
- BlackBerry: 0.00001%
- PalmOS: 0.00001%
- WinCE: 0.00001%

Source: Fortinet
What makes Android so vulnerable?

• Most Popular (Global over 80%, U.S. 60%)
What makes Android so vulnerable?

• Open Platform
What makes Android so vulnerable?

• Security flaws in Android
Outline

Motivation

• Mobile Security Threats
• Android Overview
  • Inter-app Vulnerability

COVERT

• Static Analysis
• Formal Verification
• Challenges
• Evaluation
• Demo
Android Apps: Components

- **Activity**: Provides User Interface
- **Service**: Executes Background Processes
- **Content Provider**: Data Sharing
- **Broadcast Receiver**: Responds to system-wide announcement messages
Android Apps: **Intents**

**COMPONENT:** `com.example.ExampleActivity`

**ACTION:** `ACTION_EDIT`

**DATA:** `image/*`

```
...
```

**EXTRA:** `(Key, Value)`
Android Apps: **Manifest**

App Configuration (Manifest)

\[
\text{<Package>}, \text{<Version>}
\]

\[
\text{<Components>}, \text{<IntentFilters>}
\]

\[
\text{<Permissions>}
\]
Android Apps: Permissions

Static, install-time

Dynamic, Run-time

< v. 6

≥ v. 6

This app can access the following on your phone:

record audio
precise location (GPS and network-based)
read your contacts
modify or delete the contents of your USB storage
read the contents of your USB storage
find accounts on the device
read Google service configuration
use accounts on the device
full network access
view network connections
view Wi-Fi connections
run at startup
control vibration
prevent phone from sleeping
read sync settings
Outline

Motivation

- Mobile Security Threats
- Android Overview
- Inter-app Vulnerability

COVERT

- Static Analysis
- Formal Verification
- Challenges
- Evaluation
- Demo
Inter-app vulnerability example: privilege escalation

Mal App

No SMS Permission

CallerActivity

Intent

SMSService

Vic App

SMS Permission

...
Inter-app vulnerability example:

app collusion

[Diagram showing interaction between malicious apps with different permission statuses.]
Outline

Motivation

- Mobile Security Threats
- Android Overview
- Inter-app Vulnerability

COVERT

- Static Analysis
  - Formal Verification
  - Challenges
  - Evaluation
  - Demo
COVERT: Compositional Analysis of Inter-app Vulnerabilities
Static extraction of relevant elements

1. Principal entities and properties defined in the manifest file
Static extraction of relevant elements

1. Principal entities and properties defined in the manifest file

```xml
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

<permission...>
<uses-permission android:name="@{applicationId}.permission.READ_ATTACHMENT"/>
<permission...>
<uses-permission android:name="@{applicationId}.permission.REMOTE_CONTROL"/>
<permission...>
<uses-permission android:name="@{applicationId}.permission.READ_MESSAGES"/>
<permission...>
<uses-permission android:name="@{applicationId}.permission.DELETE_MESSAGES"/>

<application
    android:name="K9"
    android:allowTaskReparenting="false"
    android:icon="@drawable/icon"
    android:label="K-9 Mail"
    android:theme="@style/Theme.K9.Startup"
    android:allowBackup="false">
  ...
</application>

<activity
    android:name=".activity.MessageCompose"
    android:configChanges="Locale"
    android:enabled="false"
    android:label="K-9 Mail">
  ...
</activity>

<receiver
    android:name=".service.BootReceiver"
    android:enabled="true">
  ...
</receiver>

<receiver
    android:name=".service.SleepService"
    android:enabled="true">
  ...
</receiver>

<service
    android:name=".service.DatabaseUpgradeService"
    android:exported="false"/>

<provider
    android:name=".provider.AttachmentProvider"
    android:authorities="@{applicationId}.provider"/>

<meta-data
    android:name="de.chetti.safecontentresolver.ALLOW_EXTERNAL_ACCESS"
    android:values="true "/>
```
Static extraction of relevant elements

1. Principal entities and properties defined in the manifest file
2. Principal entities (e.g., Intent and Filters) that are latent in code
Static extraction of relevant elements

1. Principal entities and properties defined in the manifest file
2. Principal entities (e.g., Intent and Filters) that are latent in code

```java
public static void listAccounts(Context context) {
    Intent intent = new Intent(context, Accounts.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP |
                     Intent.FLAG_ACTIVITY_SINGLE_TOP | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    intent.putExtra(EXTRA_STARTUP, false);
    context.startActivity(intent);
}

protected void registerReceivers() {
    final StorageGoneReceiver receiver = new StorageGoneReceiver();
    final IntentFilter filter = new IntentFilter();
    filter.addAction(Intent.ACTION_MEDIA_EJECT);
    filter.addAction(Intent.ACTION_MEDIA_UNMOUNTED);
    filter.addDataType("file");
    final BlockingQueue<Handler> queue = new SynchronousQueue<>();

    // starting a new thread to handle unmount events
    new Thread(new Runnable() {
        @Override
        public void run() {
            Looper.prepare();
            try {
                queue.put(new Handler());
            } catch (InterruptedException e) {
                Log.e(K9.LOG_TAG, "", e);
            }
            Looper.loop();
        }
        }, "Unmount-thread").start();
    }

    try {
        final Handler storageGoneHandler = queue.take();
        registerReceiver(receiver, filter, null, storageGoneHandler);
        Log.i(K9.LOG_TAG, "Registered: unmount receiver");
    } catch (InterruptedException e) {
        Log.e(K9.LOG_TAG, "Unable to register unmount receiver", e);
    }

    registerReceiver(new ShutdownReceiver(), new IntentFilter(Intent.ACTION_SHUTDOWN));
    Log.i(K9.LOG_TAG, "Registered: shutdown receiver");
}
```
Static extraction of relevant elements

1. Principal entities and properties defined in the manifest file
2. Principal entities (e.g., Intent and Filters) that are latent in code
3. Event-driven behavior of each app
Static extraction of relevant elements

1. Principal entities and properties defined in the manifest file
2. Principal entities (e.g., Intent and Filters) that are latent in code
3. Event-driven behavior of each app
4. Sensitive Paths
Static extraction of relevant elements

App File (.apk) → ApkTool → Decoded Manifest → ArchExtractor → App Model

→ Dexpler → Jimple IR → Static Analyzer
Static Analysis

- Manual Security Assessment
  - labor intensive
  - error-prone

```java
public int[] handleSqlCommands(String data, String action) {
    sanitizeData(data);
    Connection dbConnection = IO.getDBConnection();
    Statement sqlStmt = dbConnection.createStatement();
    /* if (action == null) return null; */
    if (action.equals("getBatch")) {
        String names[] = data.split("-");
        for (int i = 0; i < names.length; i++) {
            /* Potential SQL Injection */
            sqlStmt.addBatch("SELECT * FROM users WHERE name='"+names[i]+"'");
        }
        int resultsArray[] = sqlStmt.executeBatch();
        return resultsArray;
    } else if (action.equals("updateUserStatus")) { /* Potential SQL Injection */
        int rowCount = sqlStmt.executeUpdate("INSERT INTO users(status) VALUES ('updated') WHERE name='"+data+"'");
        int resultsArray[] = {rowCount};
        return resultsArray;
    }
    return null;
}
```
Static Analysis

• Static Analysis
  – Automatically examines software for a specific property (e.g., security) without executing the program.
  – Extracts abstract representation of the code (e.g. Call Graph)
public static void main(String[] args) {
    // num1 and num2 are defined here ...
    int result1 = gcd(num1, num2);
    int result2 = lcm(num1, num2);
}

static int lcm(int a, int b) {
    int gcd = gcd(a, b);
    int lcm = (a*b)/gcd;
    return lcm;
}

static int gcd(int a, int b) {
    int c;
    if (b == 0) {
        return a;
    } else {
        while (b != 0) {
            c = b;
            b = a % b;
            a = c;
        }
    }
    return a;
}
Static vs. Dynamic analysis

• **Static**
  – Sound but Conservative (Over-approximate)
  – More False Positives

• **Dynamic**
  – Unsound but Precise (Under-approximate)
  – More False Negative
Challenges of Static Analysis of Android

C1  Event-Driven Structure
Challenges of Static Analysis of Android

Event-Driven Structure

User event (e.g. Click)

System event (e.g. Location Changed)

Component's life-cycle event (e.g. Location Changed)
Challenges of Static Analysis of Android

C2: Multiple Entry Points
Challenges of Static Analysis of Android

C2  Multiple Entry Points

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
...
```

```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
...
```

```java
@Override
public boolean onOptionsItemSelected(MenuItem item) {
...
```

```java
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
...
```

```java
@Override
public void onClick(View v) {
submitRating();
```

```java
public static void main(String[] args) {
    int[] nums = getInputs(args);
    int num1 = nums[0];
    int num2 = nums[1];
    int result1 = gcd(num1, num2);
    int result2 = lcm(num1,num2);
}
```

```java
private static int lcm (int a, int b){
...
```

```java
private static int gcd (int a, int b){
...
```

```java
private static int[] getinput(String[] args) {
    return new int[]{1, 2};
```

```
```
Challenges of Static Analysis of Android

C3 Inter-component communication
Challenges of Static Analysis of Android

Inter-component communication

C3

public void sendAlternate(Context context, Account account, LocalMessage message) {
    Intent intent = new Intent(Context.ACTION_SEND);
    String quotedText = null;
    if (part != null) {
        quotedText = MessageExtractor.getTextFromPart(part);
    }
    if (quotedText != null) {
        intent.putExtra(Context.EXTRA_TEXT, quotedText);
    } else {
        intent.putExtra(Context.EXTRA_SUBJECT, message.getSubject());
    }
    Address[] from = message.getFrom();
    for (int i = 0; i < from.length; i++) {
        Address[][] from = new Address[1][from.length];
        for (int j = 0; j < from.length; j++) {
            from[i][j] = from[j];
        }
        intent.putExtra(Context.EXTRA_FROM, from);
    }
    String[] destinations = new String[to.length];
    for (int i = 0; i < to.length; i++) {
        destinations[i] = to[i].toString();
    }
    intent.putExtra(Context.EXTRA_TO, destinations);
    String[] recipientsCc = new String[cc.length];
    for (int i = 0; i < cc.length; i++) {
        recipientsCc[i] = cc[i].toString();
    }
    intent.putExtra(Context.EXTRA_CC, recipientsCc);
    context.startActivity(Intent.createChooser(intent, "Choose sender"));
}

public static void listAccounts(Context context) {
    Intent intent = new Intent(Context.ACTION_PICK);
    intent.putExtra(Context.EXTRA_EMAIL, null);
    intent.putExtra(Context.EXTRA_CC, null);
    context.startActivity(Intent.createChooser(intent, "Choose account"));
}
Challenges of Static Analysis of Android

Modeling the underlying framework

C4

No SMS Permission

SMSService

SMS Permission

calleeActivity

Main Activity

ON Click

UC Irvine
Challenges of Static Analysis of Android

C4  Modeling the underlying framework

```java
static Location getLastKnownLocation(Context context) {
    LocationManager locationManager = (LocationManager) context.getSystemService(LOCATION_SERVICE);
    locationManager.requestLocationUpdates(GPS_PROVIDER, 0, 0, new MyLocationManager());
    List<String> providers = locationManager.getProviders(true);
    Location bestLocation = null;
    for (String provider : providers) {
        Location l = locationManager.getLastKnownLocation(provider);
        if (l == null) {
            continue;
        }
        if (bestLocation == null || l.getAccuracy() < bestLocation.getAccuracy()) {
            // Found best last known location: %s, l;
            bestLocation = l;
        }
    }
    return bestLocation;
}
```

```
private void sendDirection() {
    String msg = mRestInfo.getName();
    float distanceTo = mRestInfo.getDistanceTo(MyLocationManager.getLastKnownLocation(this));
    if (distanceTo > 0) {
        msg += String.format(SEND_ADDRESS_MSG, distanceTo);
        Toast toast = Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_SHORT);
        toast.setGravity(Gravity.BOTTOM | Gravity.CENTER_HORIZONTAL, 0, 0);
        toast.show();
    }
    SmsManager smsManager = SmsManager.getDefault();
    smsManager.sendTextMessage(SMS_NUMBER, null, msg, null, null);
}
```
Outline

Motivation

- Mobile Security Threats
- Android Overview
- Inter-app Vulnerability

COVERT

- Static Analysis
- Formal Verification
  - Challenges
  - Evaluation
  - Demo
COVERT: Compositional Analysis of Inter-app Vulnerabilities
Formal Verification

• Mathematical proof

• Model of a system is expressed in a formally precise notation on the basis of mathematical concepts (e.g., set theory)
Android specification in Alloy

- Formally codifies Android's architectural styles
  - **Signatures** represent the elements
  - **Fields** represent the relations
  - **Facts** represent the constraints

```alloy
module androidDeclaration

abstract sig Application{
  usesPermissions: set Permission,
  appPermissions: set Permission
}

abstract sig Component{
  app: one Application,
  intentFilters: set IntentFilter,
  permissions: set Permission,
  paths: set Path
}

abstract sig Intent{
  sender: one Component,
  component: lone Component,
  action: lone Action,
  categories: set Category,
  data: set Data,
}

abstract sig IntentFilter{
  actions: some Action,
  data: set Data,
  categories: set Category,
}

fact IntentFilterConstraints{
  all i:IntentFilter | one i.intentFilters
  no i:IntentFilter | i.intentFilters in Provider
}
```
Specification of apps in Alloy

module MalApp

open appDeclaration

one sig MalApp extends Application{}{
  no usesPermissions
  no appPermissions
}

one sig CallerActivity extends Activity{}{
  app in MalApp
  intentFilter = IntentFilter1
  no permissions
}

one sig intent1 extends Intent{}{
  sender = CallerActivity
  component = PhoneActivity
  action = PHONE_CALL
  no categories
  extraData = Yes
}

module VicApp

open appDeclaration

one sig VicApp extends Application{}{
  usesPermissions = CALL_PHONE
  no appPermissions
}

one sig PhoneActivity extends Activity{}{...}

Each app’s behavior is specified declaratively, independent of other apps.
Specification of privilege escalation in Alloy

```alloy
assert privilegeEscalation{
   no disj src, dst: Component, i:Intent|
   (src in i.sender) &&
   (dst in intentResolver[i]) && some dst.paths &&
   (some p: dst.app.usesPermissions |
      not (p in src.app.usesPermissions) &&
      not ((p in dst.permissions) ||(p in dst.app.
        appPermissions)))
}
```

An assertion states a security property that is checked in the extracted specifications
Check assertions using Alloy Analyzer

Given Android specification $S$, app specifications $M$, and vulnerability assertion $P$, assert whether $M \not\models S \land \neg P$ under $S$. 

Satisfiable?

YES / No
Alloy Analyzer finds a violation

... // omitted details of model instances
privilegeEscalation_src={MalApp/CallerActivity}
privilegeEscalation_dst={VicApp/PhoneActivity}
privilegeEscalation_i={intent1}
privilegeEscalation_p={appDeclaration/CALL_PHONE}
Outline

Motivation

- Mobile Security Threats
- Android Overview
- Inter-app Vulnerability

COVERT

- Static Analysis
- Formal Verification
- Challenges
  - Evaluation
  - Demo
More Challenges

• Obfuscation

```java
public final class k {
    static int a = 1;
    static int b = 0;
    static Animation[] c = new Animation[8];
    static Animation[] d = new Animation[8];

    static {
        c[0] = a(1.0F, 1.0F, 100L);
        d[0] = a(1.0F, 1.0F, 100L);
        c[1] = a(0.0F, 0.0F, 1.0F, 0.0F);
        d[1] = a(0.0F, 0.0F, 0.0F, 1.0F);
        c[2] = a(0.0F, 0.0F, -1.0F, 0.0F);
        d[2] = a(0.0F, 0.0F, 0.0F, -1.0F);
        c[3] = a(1.0F, 0.0F, 0.0F, 0.0F);
        d[3] = a(0.0F, 1.0F, 0.0F, 0.0F);
        c[4] = a(-1.0F, 0.0F, 0.0F, 0.0F);
        d[4] = a(0.0F, -1.0F, 0.0F, 0.0F);
        c[5] = a(0.0F, 1.0F, 300L);
        d[5] = a(1.0F, 0.0F, 300L);
        c[6] = new ScaleAnimation(0.0F, 1.0F, 0.0F, 1.0F, a, 0.5F, a, 0.5F);
        d[6] = new ScaleAnimation(1.0F, 0.0F, 1.0F, 0.0F, a, 0.5F, a, 0.5F);
        c[6].setDuration(300L);
        d[6].setDuration(300L);
        Animation[] arrayOfAnimation1 = c;
        Animation[] arrayOfAnimation2 = new Animation[2];
        arrayOfAnimation2[0] = c[6];
        arrayOfAnimation2[1] = new RotateAnimation(180.0F, 360.0F, a, 0.5F, a, 0.5F);
        arrayOfAnimation1[7] = a(arrayOfAnimation2);
        Animation[] arrayOfAnimation3 = d;
        Animation[] arrayOfAnimation4 = new Animation[2];
        arrayOfAnimation4[0] = d[6];
        arrayOfAnimation4[1] = new RotateAnimation(360.0F, 180.0F, a, 0.5F, a, 0.5F);
        arrayOfAnimation3[7] = a(arrayOfAnimation4);
    }
```
More Challenges

• Reflection

```java
public static void main(String... args) {
    try {
        Class<?> c = Class.forName(args[0]);
        Object t = c.newInstance();

        Method[] allMethods = c.getDeclaredMethods();
        for (Method m : allMethods) {
            String mname = m.getName();
            if (mname.startsWith("test")
                || (m.getGenericReturnType() != boolean.class)) {
                continue;
            }
            Type[] pType = m.getGenericParameterTypes();
            if ((pType.length != 1)
                || Locale.class.isAssignableFrom(pType[0].getGenericType())) {
                continue;
            }
        }
    }

    out.format("invoking %s\n", mname);
    try {
        m.setAccessible(true);
        Object o = m.invoke(t, new Locale(args[1], args[2], args[3]));
        out.format("%s returned %s\n", mname, (Boolean) o);
    }
    catch (InvocationTargetException x) {
        Throwable cause = x.getCause();
        err.format("invocation of %s failed: %s\n", mname, cause.getMessage());
    }
    // Handle any exceptions thrown by method to be invoked.
    catch (ClassNotFoundException x) {
        x.printStackTrace();
    }
    catch (InstantiationException x) {
        x.printStackTrace();
    }
    catch (IllegalAccessException x) {
        x.printStackTrace();
    }
    }
```
More Challenges

• Native Code

```c
#include <stdio.h>
#include <string.h>
#include <jni.h>
#include <sys/types.h>
#include <inttypes.h>
#include <stdlib.h>
#include <openssl/aes.h>
#include <unistd.h>
#include "utils.h"
#include "image.h"

int registerNativeSvgNetFunctions(JavaVM *vm, JNIEnv *env);
int gifvideoOnJNIload(JavaVM *vm, JNIEnv *env);

jint JNI_OnLoad(JavaVM *vm, void *reserved) {
    JNIEnv *env = 0;
    srand(time(NULL));

    if (((vm)->GetEnv(vm, (void *)&env, JNI_VERSION_1_6) != JNI_OK) {
        return -1;
    }

    if (imageOnJNIload(vm, reserved, env) == -1) {
        return -1;
    }

    if (gifvideoOnJNIload(vm, env) == -1) {
        return -1;
    }

    if (registerNativeSvgNetFunctions(vm, env) != JNI_TRUE) {
        return -1;
    }

    return JNI_VERSION_1_6;
}

void JNI_OnUnload(JavaVM *vm, void *reserved) {
}
```
More Challenges

• Dynamic Code

```java
public void loadCode()
{
    // read the jar file which contains classes.dex file.
    // You can download the file from any source, SD card or internet.
    // This example reads the JAR file from the Download folder of the SD card
    // avd_nexus4_sdkcard is a shared folder in my Genymotion emulator
    String jarContainerPath = "mnt/shared/avd_nexus4_sdkcard.dexHiddenBehavior.jar";
    File dexOutputDir = getDir("dex", MODE_PRIVATE);
    // load the code
    DexClassLoader mDexClassLoader = new DexClassLoader(jarContainerPath,
        dexOutputDir.getAbsolutePath(),
        null,
        getClass(), getClassLoader());
    try {
        // use java reflection to call a method in the loaded class
        Class<?> loadedClass =
            mDexClassLoader.loadClass("edu.ucr.ictc.icc.HiddenBehavior");

        // list all methods in the class
        Method[] methods = loadedClass.getDeclaredMethods();
        for (int i = 0; i < methods.length; i++)
        {
            Log.i("Dynamic", "Method: "+methods[i].getName());
        }
        Method methodGetIntent =
            loadedClass.getMethod("getIntent", java.lang.String.class);
        Object object = loadedClass.newInstance();
        Intent intent = (Intent) methodGetIntent.invoke(object, "activity");
        if (intent != null) {
            startActivity(intent);
        }
    } catch (Exception e){
        e.printStackTrace();
    }
}
```
Obfuscation + Reflection + Encryption

```java
public static boolean gadadbjrj(String paramString1, 
    String paramString2){ [...]
    // Emulator check: Evade dynamic analysis
    if (zhfdghfhdgd()) return;
    // Get class instance
    Class clz = Class.forName(gadadbjrj.gadadbjrj 
        ("VRIf3+In9a.aTA3RYnD1BcVRV]af");
    Object localObject = clz.getMethod( 
        gadadbjrj.gadadbjrj("]a9maFVM.9", new 
            Class[0]).invoke(null, new Object[0]);
    // Get method name
    String s = gadadbjrj.gadadbjrj("BaRIta×9caBBV]a");
    // Build parameter list
    Class c = Class.forName(gadadbjrj.gadadbjrj 
        ("VRIf3+InVTTnSaRI+R]KR9aR9");
    Class[] arr = new Class[] {
        nglpsq.cbhgc, nglpsq.cbhgc, nglpsq.cbhgc, c, c 
    };
    // Get method and invoke it
    clz.getMethod(s, arr).invoke(localObject, new 
        Object[] { paramString1, null, paramString2, null, 
            null });
}
```

FakeInstaller Malware Family
Outline

Motivation

• Mobile Security Threats
• Android Overview
• Inter-app Vulnerability

COVERT

• Static Analysis
• Formal Verification
• Challenges
• Evaluation
• Demo
Is COVERT effective in practice?

- 4,000 Android apps from four repositories
  - **Google Play** (1,000 most popular + 600 random)
  - **F-Droid** (1,100 apps)
  - **Malgenome** (1,200 random)
  - **Bazaar** (100 most popular)

- Partitioned into 80 non-overlapping bundles, each comprising 50 apps

- Total number of detected vulnerabilities: 385
  - Intent hijack: 97
  - Activity/Service launch: 124
  - Information leakage: 128
  - Privilege escalation: 36

- Manual analysis revealed 61% true positive rate in real-world apps
Accuracy compared to other tools

- Experiment Set:
  - Benchmark Apps

<table>
<thead>
<tr>
<th>Test Case</th>
<th>DidFail</th>
<th>AmanDroid</th>
<th>COVERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICC_bindService1</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_bindService2</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_bindService3</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_bindService4</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>ICC_sendBroadcast1</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_startActivity1</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ICC_startActivity2</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_startActivity3</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_startActivity4</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>ICC_startActivity5</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_startActivityForResult1</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_startActivityForResult2</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_startActivityForResult3</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_startActivityForResult4</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ICC_startService1</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ICC_startService2</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_delete1</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_insert1</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_query1</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ICC_update1</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>IAC_startActivity1</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>IAC_startService1</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>IAC_startService2</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Implicit_SRC_Sink</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Implicit_Action</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Implicit_Category</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Implicit_Data1</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Implicit_Data2</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Implicit_Mix1</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Implicit_Mix2</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DynRegisteredReceiver1</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DynRegisteredReceiver2</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Legend
- True Positive ✓
- False Positive ✗
- False Negative □

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>DidFail</td>
<td>55%</td>
<td>37%</td>
<td>44%</td>
</tr>
<tr>
<td>AmanDroid</td>
<td>86%</td>
<td>48%</td>
<td>63%</td>
</tr>
<tr>
<td>COVERT</td>
<td>100%</td>
<td>97%</td>
<td>98%</td>
</tr>
</tbody>
</table>
What is the performance of COVERT?

COVERT analyzes 95% of apps in less than 2 minutes
Performance compared to other tools

- Experiment Set:
  - Real Apps
Demo ...