Android Analysis Tools

Yuan Tian
Malware are more creative: XcodeGhost

• More than 300 apps are infected, including WeChat and Netease
• Collect device ID, Apple ID and password
Even more attacks on Android

• Package Repacking
• Abuse of Telephony Services
• Root Exploitation
• Sensitive Information Exposure
• Update attack
WHY ARE THESE ATTACKS POSSIBLE?
Android security framework

• Linux process sandbox

• Permission based control for accessing information

• Applications need to be signed

• APP scanning-Bouncer
Linux process sandbox

- Applications collude with each other to steal information
- Application exploit bugs to get root access to the device
Permission

- Too coarse-grained
- Users ignore or misunderstand the permissions
App signature

• Applications are self-signed; no CA required

• Signature define persistence
  ➢ Detect if the application has changed
  ➢ Application update

• Signatures define authorship
  ➢ Establish trust between applications
  ➢ Run in same Linux ID

• Vulnerabilities
  ➢ Repackaged Apps
Bouncer

• Attackers can bypass the Google servers
HOW CAN WE ANALYSIS THESE ATTACKS?
Tools summary

• Static analysis tools

• Dynamic analysis tools
STATIC ANALYSIS TOOLS
Smali/baksmali

- http://code.google.com/p/smali/
- smali/baksmali is an assembler/disassembler for the dex format used by Dalvik, Android’s Java VM implementation.
- The syntax is loosely based on Jasmin’s/dedexer’s syntax, and supports the full functionality of the dex format (annotations, debug info, line info, etc).
Dex2jar

- [http://code.google.com/p/dex2jar](http://code.google.com/p/dex2jar)
- It can convert Android’s .dex format to Java’s .class format, just one binary format to another binary format, not to source.
public class Dialog extends Activity
{
    private Button cancel = null;
    private TextView message = null;
    private Button ok = null;
    private String type = null;

    public void init()
    {
        Button localButton1 = this.ok;
        Dialog local1 = new Dialog(this);
        localButton1.setOnClickListener(local1);
        Button localButton2 = this.cancel;
        Dialog local2 = new Dialog2(this);
        localButton2.setOnClickListener(local2);
    }

    public void onCreate(Bundle paramBundle)
    {
        super.onCreate(paramBundle);
        LinearLayout localLinearLayout = new LinearLayout(this);
        localLinearLayout.setOrientation(0);
        LinearLayout.LayoutParams localLayoutParams = new LinearLayout.LayoutParams(-1, -1);
        localLinearLayout.setLayoutParams(localLayoutParams);
        localLinearLayout.setGravity(17);
        TextView localTextView1 = new TextView(this);
        this.message = localTextView1;
        this.message.setTextSize(26.0F);
        TextView localTextView2 = this.message;
    }
Apktool


• Features:
  – Decoding resources to nearly original form and rebuilding them
  – Smali debugging
  – Helping with some repetitive tasks
Androguard


- Features:
  - Access to the static analysis of your code (basic blocks, instructions, permissions.)
  - Malware database
  - Diffing of android applications
  - Visualize your application into gephi (gexf format), cytoscape (xgmml format), or PNG/DOT output,
APKInspector Overview

- Integrate the previous static analysis tools and provides graphic features which bring convenience to the malware analysis.
APKInspector Features

• Features:
  - CFG
  - Call Graph
  - Static Instrumentation
  - Permission Analysis
  - Dalvik codes
  - Smali codes
  - Java codes
  - APK Information
APKInspector Components

- Permission Analysis
  - Extract permissions from APK
  - Is the permission set in the APK?
    - Yes: Static Analysis
      - Is it really malicious?
        - Yes: Delete rules
        - No: End
      - Is it really harmless?
        - Yes: Add rules
        - No: End
    - No: Static Analysis

- Static Analysis
  - Flexible to different Versions of Android
  - Smali/backsmali
Usage of APKInspector

• Installation with Shell Script
• Analysis of APK
Usage of APKInspector

• Filter of Malicious behavior by permission analysis
Usage of APKInspector

- Smali code
Usage of APKInspector

• Static Code Instrumentation
Usage of APKInspector

• Dalvik Bytecode
Usage of APKInspector

- Control Flow Graph
Usage of APKInspector

- Java
Usage of APKInspector

- Navigation
  Back & Forward
  Current Method displayed
Usage of APKInspector

• Call Graph
DYNAMIC ANALYSIS TOOLS
Tcpdump

- When writing Android applications that heavily rely on networking it can sometimes be useful to inspect the network traffic going out and coming into your device. Especially when writing applications that implement networking protocols (like ftp, smtp, ssh, xmpp,..) the ability to inspect packets at TCP-level is invaluable.
Tcpdump

• 1  ./adb push /home/....../tcpdump-arm /data/local/
• 2  tcpdump-arm -s 0 -w out.txt
• 3  ./adb pull /data/local/out.txt /home/....../out.txt
• 4  wireshark

ROOT!!!

It also can be used in the emulator, but need the root privilege.
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>131</td>
<td>54.795916</td>
<td>RealtekU 12:34:56</td>
<td>RealtekU 12:35:03</td>
<td>ARP</td>
<td>Who has 10.0.2.3? Tell 10.0.2.15</td>
</tr>
<tr>
<td>132</td>
<td>54.796185</td>
<td>RealtekU 12:35:03</td>
<td>RealtekU 12:34:56</td>
<td>ARP</td>
<td>10.0.2.3 is at 52:54:00:12:35:03</td>
</tr>
<tr>
<td>110</td>
<td>49.800692</td>
<td>10.0.2.15</td>
<td>10.0.2.3</td>
<td>DNS</td>
<td>Standard query A <a href="http://www.google.com">www.google.com</a></td>
</tr>
<tr>
<td>111</td>
<td>49.810474</td>
<td>10.0.2.3</td>
<td>10.0.2.15</td>
<td>DNS</td>
<td>Standard query response CNAME <a href="http://www.l.google.com">www.l.google.com</a> A 74.125.71.13</td>
</tr>
<tr>
<td>133</td>
<td>56.887906</td>
<td>10.0.2.15</td>
<td>10.0.2.3</td>
<td>DNS</td>
<td>Standard query A <a href="http://www.google.com">www.google.com</a></td>
</tr>
<tr>
<td>134</td>
<td>56.893863</td>
<td>10.0.2.3</td>
<td>10.0.2.15</td>
<td>DNS</td>
<td>Standard query response CNAME <a href="http://www.l.google.com">www.l.google.com</a> A 74.125.71.13</td>
</tr>
<tr>
<td>222</td>
<td>68.659604</td>
<td>10.0.2.15</td>
<td>10.0.2.3</td>
<td>DNS</td>
<td>Standard query A ssl.gstatic.com</td>
</tr>
<tr>
<td>223</td>
<td>68.666020</td>
<td>10.0.2.3</td>
<td>10.0.2.15</td>
<td>DNS</td>
<td>Standard query response A 74.125.71.120</td>
</tr>
<tr>
<td>115</td>
<td>50.207497</td>
<td>10.0.2.15</td>
<td>74.125.71.103</td>
<td>HTTP</td>
<td>GET /complete/search?hl=en&amp;gl=us&amp;json=true&amp;q=h HTTP/1.1</td>
</tr>
<tr>
<td>138</td>
<td>57.328727</td>
<td>10.0.2.15</td>
<td>74.125.71.105</td>
<td>HTTP</td>
<td>GET /m?hl=en&amp;gl=us&amp;source=android-launcher-widget&amp;q=hao HTTP/1.1</td>
</tr>
<tr>
<td>149</td>
<td>57.696780</td>
<td>74.125.71.105</td>
<td>10.0.2.15</td>
<td>HTTP</td>
<td>[TCP Previous segment lost] Continuation or non-HTTP traffic</td>
</tr>
<tr>
<td>152</td>
<td>57.785801</td>
<td>74.125.71.105</td>
<td>10.0.2.15</td>
<td>HTTP</td>
<td>Continuation or non-HTTP traffic</td>
</tr>
<tr>
<td>145</td>
<td>57.786212</td>
<td>74.125.71.105</td>
<td>10.0.2.15</td>
<td>HTTP</td>
<td>Continuation or non-HTTP traffic</td>
</tr>
<tr>
<td>156</td>
<td>57.786395</td>
<td>74.125.71.105</td>
<td>10.0.2.15</td>
<td>HTTP</td>
<td>Continuation or non-HTTP traffic</td>
</tr>
<tr>
<td>150</td>
<td>57.786500</td>
<td>74.125.71.105</td>
<td>10.0.2.15</td>
<td>HTTP</td>
<td>Continuation or non-HTTP traffic</td>
</tr>
<tr>
<td>159</td>
<td>57.787654</td>
<td>74.125.71.105</td>
<td>10.0.2.15</td>
<td>HTTP</td>
<td>[TCP Previous segment lost] Continuation or non-HTTP traffic</td>
</tr>
<tr>
<td>161</td>
<td>57.787814</td>
<td>74.125.71.105</td>
<td>10.0.2.15</td>
<td>HTTP</td>
<td>Continuation or non-HTTP traffic</td>
</tr>
<tr>
<td>163</td>
<td>57.873890</td>
<td>74.125.71.105</td>
<td>10.0.2.15</td>
<td>HTTP</td>
<td>[TCP Previous segment lost] Continuation or non-HTTP traffic</td>
</tr>
<tr>
<td>165</td>
<td>57.874266</td>
<td>74.125.71.105</td>
<td>10.0.2.15</td>
<td>HTTP</td>
<td>Continuation or non-HTTP traffic</td>
</tr>
<tr>
<td>167</td>
<td>57.874393</td>
<td>74.125.71.105</td>
<td>10.0.2.15</td>
<td>HTTP</td>
<td>Continuation or non-HTTP traffic</td>
</tr>
<tr>
<td>169</td>
<td>57.874504</td>
<td>74.125.71.105</td>
<td>10.0.2.15</td>
<td>HTTP</td>
<td>Continuation or non-HTTP traffic</td>
</tr>
</tbody>
</table>
DroidBox

- [http://honeynet.org/gsoc/slot5](http://honeynet.org/gsoc/slot5)
- Hashes for the analyzed package
- Incoming/outgoing network data
- File read and write operations
- Started services and loaded classes through DexClassLoader
- Information leaks via the network, file and SMS
- Circumvented permissions
- Cryptography operations performed using Android API
- Listing broadcast receivers
- Sent SMS and phone calls
DroidBox

• Geinimi

W/dalvikvm (369): TaintLog: Decrypted data[IMEI] with DES
W/dalvikvm (369): TaintLog: Decrypted data[IMSI] with DES
W/dalvikvm (369): TaintLog: Decrypted data[CFID] with DES
W/dalvikvm (369): TaintLog: Decrypted data[_value0] with DES
W/dalvikvm (369): TaintLog: Decrypted data[_value0] with DES
W/dalvikvm (369): TaintLog: Decrypted data[DID] with DES
W/dalvikvm (369): TaintLog: Decrypted data[_value0] with DES
W/dalvikvm (369): TaintLog: Decrypted data[autosdkver] with DES
W/dalvikvm (369): TaintLog: Decrypted data[latitude] with DES
W/dalvikvm (369): TaintLog: Decrypted data[longitude] with DES
W/dalvikvm (369): TaintLog: Decrypted data[debug_internal] with DES

W/dalvikvm (369): TaintLog: Encryption: KEY = { 1, 2, 3, 4, 5, 6, 7, 8 } with algorithm: DES
W/dalvikvm (369): TaintLog: Decrypted data[cmd] with DES

W/dalvikvm (369): TaintLog: OSNetworkSystem.sendStream(localhost:5432)
sending data=[hi, are you online?]

W/dalvikvm (369): TaintLog: OSNetworkSystem.receiveStream()
Response=[hi, are you online?????????????....] from null:0 ID: 30

W/dalvikvm (369): TaintLog: OSNetworkSystem.sendStream(unknown:0)
sending data=[yes, I’m online!]

W/dalvikvm (369): TaintLog: OSNetworkSystem.receiveStream()
Response=[yes, I’m online!?????....] from localhost:5432 ID: 30
TaintDroid

- [http://appanalysis.org/](http://appanalysis.org/)

Figure 2: TaintDroid architecture within Android.
TaintDroid

• Limitations:
  – It can only identify that privacy sensitive information has left the phone, and not if the event is a privacy violation.
  – It can only tracks explicit flows. Therefore, a malicious developer can use implicit flows within an application to “scrub” taint markings from variables.
Thanks!
Questions?