Dangerous Hare: Hanging Attribute References Hazards Due to Vendor Customization

Speaker: Nan Zhang
System Security Lab, Indiana University
Who are we?

• System Security Lab, Indiana University

• Focus on novel problems in system security

Android Customization

Google

Manufacturer

Carrier
Not just UI, Components too

“Among all the apps pre-installed by the major smartphone vendors (Samsung, HTC, LG, Sony) on their phones, only about 18% come from AOSP.”
Android Customization Process

- Unregulated, Decentralized
- Android Compatibility Program (ACP)
- Fails when proper precautions have not been taken
“Hare” Problem

• Hanging Attribute References

• An attribute is used while the party defining it has been removed

• A type of problem never investigated before
Attributes of Hares

- Package, Activity, Action names
- Content Provider
- Permission
An example
An example

Tablet Version

SMS/MMS Content Provider

VoIP Message Content Provider

Instant Messaging App

New Message
An example
Not All Hare are Exploitable

- Certain apps are aware of the fact that a referenced resource might not exist on the image.
  - Before invoking a certain resource within an app, some apps will first check its signature.
  - Some apps will check the image properties before invoking a non-existing app.

- Not all hares can be easily protected.
```java
public boolean extendAccessToken(Context context,
        ServiceListener serviceListener)
{
    Intent intent = new Intent();
    try{
        PackageInfo pi =
                context.getPackageManager().getPackageInfo
        ("com.facebook.katana",
             PackageManager.GET_SIGNATURES);
        // Compare signature to the legitimate Facebook
        // app Signature
        if (!compareSignatures
                (pi.signatures[0].toByteArray())){
            return false;
        } else{
        intent.setClassName("com.facebook.katana",
                        "com.facebook.katana.platform.
                         TokenRefreshService");
        return context.bindService(intent, new
                        TokenRefreshServiceConnection(context,
                         serviceListener), 0);}catch(PackageManager.NameNotFoundException e){
        return false;
    }
}
```java
private void ViewVideo(Uri uri){
    Intent intent = new Intent("android.intent.action.VIEW", uri);
    if (getPackageManager().hasSystemFeature("com.google.android.tv")){
        intent.setPackage("com.google.android.youtube.gogletv");
    } else{
        intent.setPackage("com.google.android.youtube");
    }
    startActivity(intent);
}
```
Hares are

• Not random, isolated bugs.

• Caused by the fundamental conflict between the under-regulated Android Customization process and the complicated relationship among apps.

• Lack verification between two components.
Trust, but verify.

TRUST, BUT VERIFY

YOU'RE DOING IT WRONG
Exploiting Hares

- Missing Packages, Activities, Actions

  Missing Content Providers

  Missing Permissions
If `com.samsung.android.app.memo` exists

If `com.sec.android.app.smemo` exists
Stealing Voice Notes

If `com.samsung.android.app.memo` exists

If `com.sec.android.app.smemo` exists
Stealing Voice Notes

If `com.samsung.android.app.memo` exists

com.samsung.android.app.memo

If `com.sec.android.app.smemo` exists

S Voice
Demos

- Stealing voice note
- Faking Dropbox on LG
- Replacing official voice recorder
- Hulu on Watch

All demos can be found at https://sites.google.com/site/androidharethunting/
Exploiting Hares

Missing Packages, Activities, Actions

- Missing Content Providers

Missing Permissions
LG Cloud Scam

Query “com.lge.lgaccount.provider” for additional Cloud Services and add an item.
LG Cloud Scam

Query “com.lge.lgaccount.provider” for additional Cloud Services and add an item.
LG Cloud Scam

com.lge.lgaccount.provider
LG Cloud Scam

Handle com.lge.ADD_ACCOUNT
Intent Hijacking

Google Email App
version 6.3-1218562

Account Setting Activity
Within Google Email App

Action: android.intent.action.EDIT
Data: content://ui.email.android.com/settings?account=id

<!-- Account Settings Intent Filters-->
<activity
  android:name=".activity.setup.AccountSettings"
  android:exported="true">
  <intent-filter>
    <action android:name="android.intent.action.EDIT"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:scheme="content"
      android:host="ui.email.android.com"
      android:pathPrefix="/settings"/>
  </intent-filter>
</activity>
Intent Hijacking

Google Email App
version 6.3-1218562

AccountSetting Activity
Within Google Email App
Intent Hijacking

The authority of content provider “ui.email.android.com” does not exist.

How do we leverage this to be the only one to receive the intent?
Intent Hijacking

AMS

- Look for mime type of authority: `ui.email.android.com`

Provider with authority `ui.email.android.com` exists?

- Yes: GetType of Provider
- No: Launch activity

<intent-filter>
  <action name="Action.EDIT" />
  <data scheme="content" host="ui.email.android.com" pathPrefix="/settings"/>
<intent-filter>

Email Account Setting Activity
Intent Hijacking

AMS

Look for mime type of authority: ui.email.android.com

Provider with authority "ui.email.android.com" exists?

Yes → GetType of Provider

No → Launch activity

<intent-filter>
  <action name="Action.EDIT" />
  <data scheme="content" host="ui.email.android.com" pathPrefix="/settings"/>
<intent-filter>

Email Account Setting Activity
Intent Hijacking

AMS

Look for mime type of authority: ui.email.android.com

Provider with authority “ui.email.android.com” exists?

Yes

GetType of Provider

No

Launch activity

<intent-filter>
  <action name="Action.EDIT" />
  <data scheme="content"
       host="ui.email.android.com"
       pathPrefix="/settings"/>
<intent-filter>

Email Account Setting Activity
Intent Hijacking

AMS

Look for mime type of authority: ui.email.android.com

Provider with authority "ui.email.android.com" exists?

Yes

GetType of Provider

Malicious provider with authority "ui.email.android.com"

public String getType(Uri uri) {
    return "ABC";
}

No

Launch activity

<intent-filter>
    <action name="Action.EDIT" />
    <data scheme="content"
        host="ui.email.android.com"
        pathPrefix="/settings"/>
    <intent-filter>

Email Account Setting Activity

<intent-filter>
    <action name="Action.EDIT" />
    <data scheme="content"
        host="ui.email.android.com"
        pathPrefix="/settings/"
        mimeType="ABC"/>
    <intent-filter>

Malicious Setting Activity
Intent Hijacking

Google Email App
version 6.3-1218562

Malicious App
Demos

- Google Email Attack
- LG CloudHub Attack
- Facebook Service Confusion
- Skype Service Confusion

All demos can be found at https://sites.google.com/site/androidharehunting/ and https://sites.google.com/site/perplexedmsg/
Exploiting Hares

- Missing Packages, Activities, Actions
- Missing Content Providers
- Missing Permissions
Apps use multiple push-messaging services, including
• Google Cloud Messaging (GCM)
  com.google.android.c2dm.permission.SEND
• Amazon Device Messaging (ADM)
  com.amazon.device.messaging.permission.SEND
• Nokia Notification
  com.nokia.pushnotifications.permission.SEND
Service Confusion

Google Cloud Messaging

Amazon Device Messaging

GCM Receiver

ADM Receiver

Nokia Receiver

Nokia Notification
Service Confusion

On Nexus

- GCM Receiver
- ADM Receiver
- Nokia Receiver

Google Cloud Messaging

Amazon Device Messaging

Nokia Notification
Service Confusion

On Nexus

GCM Receiver
ADM Receiver
Nokia Receiver

Google Cloud Messaging

I have Nokia permission!

Push Message
Demos

• Google Email Attack
• LG CloudHub Attack
• Facebook Service Confusion
• Skype Service Confusion

All demos can be found at https://sites.google.com/site/androidharehunting/ and https://sites.google.com/site/perplexedmsg/
I have GCM permission!
GCM Setup

1. Application Registration
2. App Registration (RegID)
3. RegID/UserID Binding
Message Stealing

1. Application Registration
2. App Registration (RegID)
3. RegID/UserID Binding

Connection Server

Malicious App

Victim Phone

Push RegID

Any time

Attacker Phone

Service App
Demos

• Google Email Attack
• LG CloudHub Attack
• Facebook Service Confusion
• Skype Message Stealing

All demos can be found at https://sites.google.com/site/androidharethunting/ and https://sites.google.com/site/perplexedmsg/
App Store Submission

• We have submitted some of attack apps to several app stores (e.g. Google, Amazon, Samsung).

• Passed vetting process easily.

• Removed immediately after approval.
• All vulnerabilities we have discovered have submitted to vendors at least 3 months prior to publication.

• Most of them have been acknowledged and fixed.
Automatically Detect Hare

Attribute Reference

- StartService()
- StartActivity
- Query
- readPermission

Attribute Definition

- Package Names
- Actions
- Content Providers
- Permissions
Construct APK

Image → ODEX → APK

Pre-Processor
Construct APK
Generate Likely Hare
Guard Detection

• Taint Analysis
  • Source: guard API
  • Sink: Attribute reference API

• If a relationship is established between the source and the sink, then the Attribute Reference API is guarded.
Generate Likely Hare
## Large Scale Measurement

<table>
<thead>
<tr>
<th>Vendor</th>
<th># of Images</th>
<th># of System Apps</th>
<th>Avg # of System Apps Per Images</th>
<th># of Countries</th>
<th># of Carriers</th>
<th># of OS Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>83</td>
<td>21733</td>
<td>261</td>
<td>36</td>
<td>23</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>1561</td>
<td>223</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>174</td>
<td>174</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>398</td>
<td>99</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>319</td>
<td>159</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>97</strong></td>
<td><strong>24185</strong></td>
<td><strong>183</strong></td>
<td><strong>36</strong></td>
<td><strong>23</strong></td>
<td><strong>10</strong></td>
</tr>
</tbody>
</table>
### Findings

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Hares in Android 4.X</th>
<th>Hares in Android 5.X</th>
<th>Avg Hares per Device</th>
<th>Min Hares per Device</th>
<th>Max Hares per Device</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of Hares</td>
<td># of Vulnerable Apps</td>
<td># of Hares</td>
<td># of Vulnerable Apps</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>19279</td>
<td>3045 (18%)</td>
<td>608</td>
<td>99 (6%)</td>
<td>239</td>
</tr>
<tr>
<td>B</td>
<td>679</td>
<td>121 (13.3%)</td>
<td>425</td>
<td>85 (15.5%)</td>
<td>157</td>
</tr>
<tr>
<td>C</td>
<td>N/A</td>
<td>N/A</td>
<td>248</td>
<td>33 (21.5%)</td>
<td>248</td>
</tr>
<tr>
<td>D</td>
<td>107</td>
<td>31 (12.4%)</td>
<td>8</td>
<td>5 (5%)</td>
<td>29</td>
</tr>
<tr>
<td>E</td>
<td>187</td>
<td>23 (15.6%)</td>
<td>16</td>
<td>8 (12.1%)</td>
<td>101</td>
</tr>
<tr>
<td>Total</td>
<td><strong>20252</strong></td>
<td><strong>3220 (14.3%)</strong></td>
<td><strong>1305</strong></td>
<td><strong>230 (11.7%)</strong></td>
<td><strong>153</strong></td>
</tr>
</tbody>
</table>
Discussion

- In the absence of sufficient guidance and a proper enforcement mechanism, hanging references become inevitable.

- Systematic efforts should be made to eliminate hare flaw.
Eliminating Hare

- Secure each attributes reference.
  - explicit authentication

- Document interdependent relations between components.
Dangerous Hare

• Hare Hunting in the Wild Android: A Study on the Threat of Hanging Attribute References. (CCS '15)

• Perplexed Messengers from the Cloud: Automated Security Analysis of Push-Messaging Integrations. (CCS '15)
  • [Yangyi Chen, Tongxin Li]*, Xiaofeng Wang, Kai Chen, and Xinhui Han. 2015.

* Co-first Author
Thank you!
Any questions?
System Security Lab, Indiana University