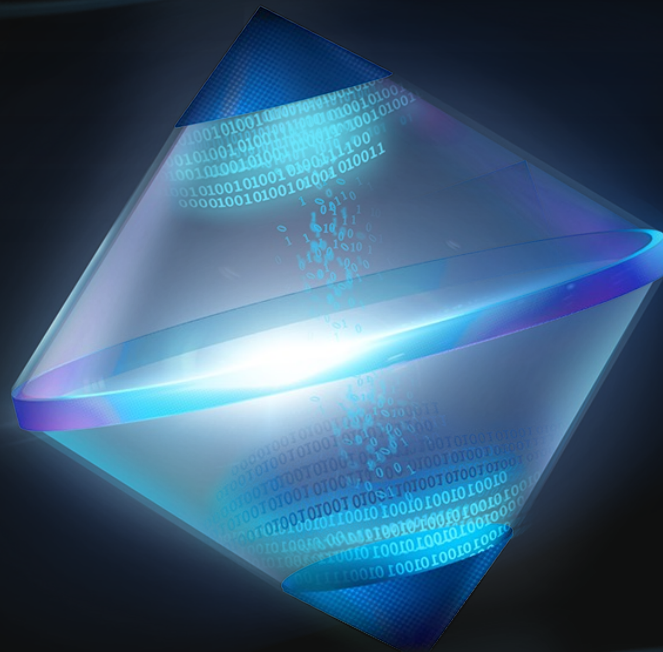


移动人脸识别 安全攻防对抗之旅

演讲人：谭桂涛



自我介绍

谭桂涛

中国工商银行安全攻防实验室核心成员
日常主要负责移动安全攻防技术的研究、应用安全测试及漏洞挖掘；跟随队伍参加攻防演练及安全相关比赛；黑灰产攻防对抗研究等

2022年“金融密码杯”全国密码应用和技术创新大赛挑战赛及正式赛双冠军；
“2023年度首期移动互联网APP产品安全漏洞技术沙龙”演讲嘉宾；
多次参加国家安全演练，成功入选公安部优秀技战法，得到国家网络与信息安全通报中心的通报表扬。



为什么来？归源，智变！

目录 / CONTENTS

人脸识别的前
世今生

攻防对抗之旅

常见的防护策
略和思路

总结与展望

◆ 人脸识别的前世今生

人脸识别技术最早是源自科幻小说和电影中的构想。

伍德罗·威尔逊·布莱索 (Woodrow Wilson Bledsoe) 是世界公认的人脸识别之父。早在20世纪60年代，布莱索发明了一个系统，可以使用平板电脑手工整理人脸照片。人们可以使用这个系统来手动记录面部特征的坐标区域，比如眼睛、鼻子、嘴巴和发际线等。

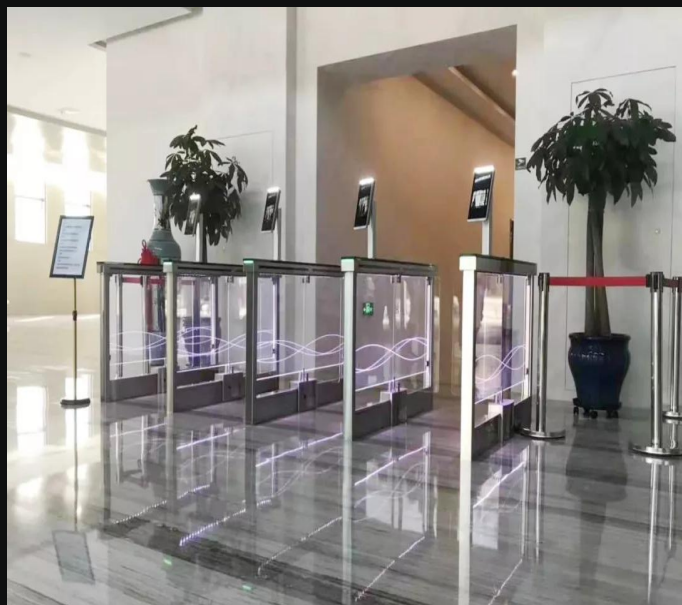
1988年，Sirovich和Kirby开始使用线性代数来解决人脸识别问题。

1993年至2000年期间，DARPA（美国国防部高级研究计划局）和NIST（美国国家标准与技术研究所）发布了FERET计划，以鼓励商业人脸识别市场。

2002年，执法人员将人脸识别应用于关键技术测试。

国内人脸识别最早是应用在安防领域。2001年公安部门开始利用人脸识别技术防范和打击重大刑事犯罪，伴随着2008年北京奥运会和2010年上海世博会的成功举办，人脸识别技术得到了更加广泛的关注，国内各大公司也开始争相加入该技术的研究，人脸识别技术在国内开始了大规模的应用和发展。

◆ 人脸识别的前世今生



人脸识别技术应用领域：

- 安防领域；
- 金融领域；
- 教育领域；
- 医疗领域；
- 商业领域等。

人脸识别技术发展的阻碍和威胁：

- 隐私泄露；
- 误判率高；
- 歧视性强；
- 安全风险。



◆ 攻防对抗之旅

1

业务流程

梳理移动端人脸识别刷脸流程，全面排查风险

攻防介质

灵活选择不同的攻防介质，拓展攻防场景

2

3

攻防设备

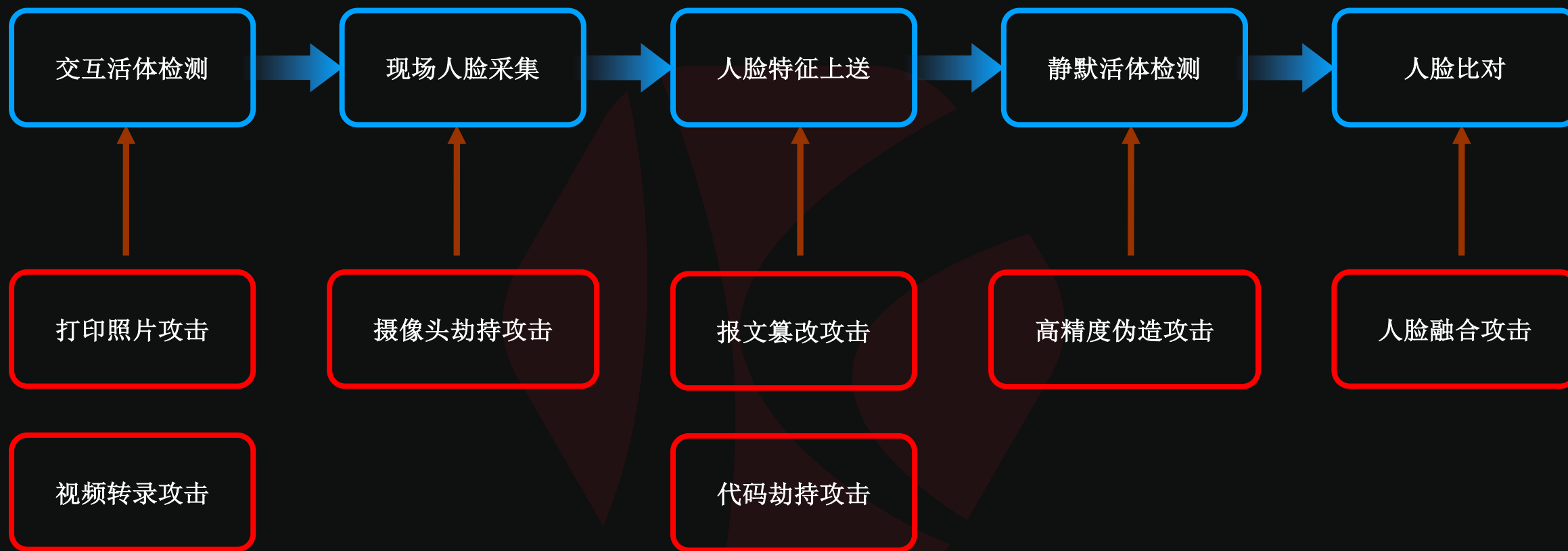
跟进黑产作案设备更新，同步提升防护

技术手段

持续深入技术研究，动态的技术博弈

4

◆ 攻防对抗之旅

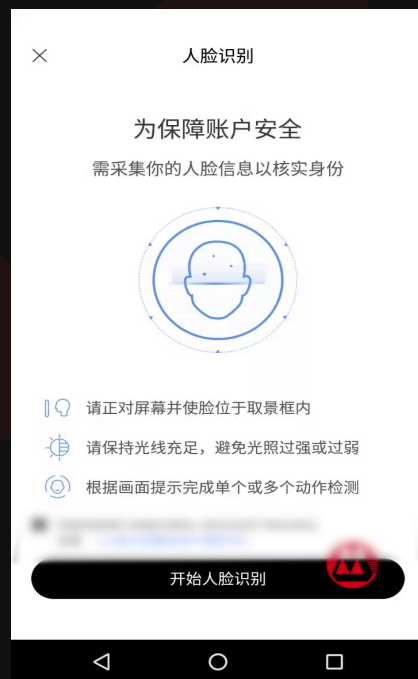
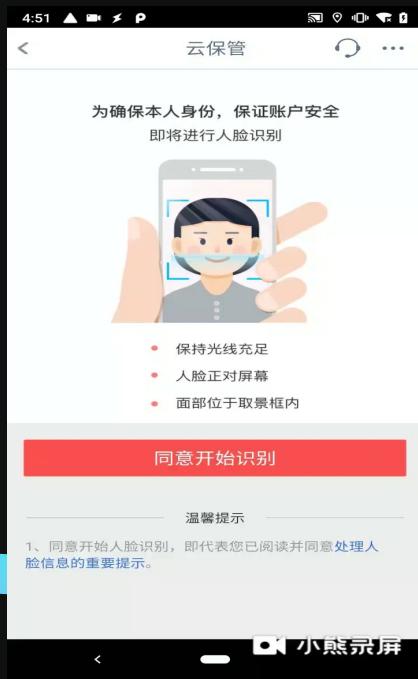


人脸识别全流程均存在一定的安全风险!!!

◆ 攻防对抗之旅

人脸识别安全攻防介质指的是照片或视频，黑灰产对应的概念叫做“物料”，攻击者拿到受害人的照片或者视频等介质，就可以利用一些技术手段实现人脸识别的破解。

- 1.无介质：无需照片或视频，完成人脸识别的破解
- 2.单照片：仅需要一张受害人的照片，即可突破人脸识别
- 3.多照片：需要受害人的基准照、动作照等一组照片，从而突破人脸识别
- 4.单视频：获取到受害人人脸的一段视频，即可突破人脸
- 5.多视频：将受害人视频切割成不同的动作视频，突破人脸识别



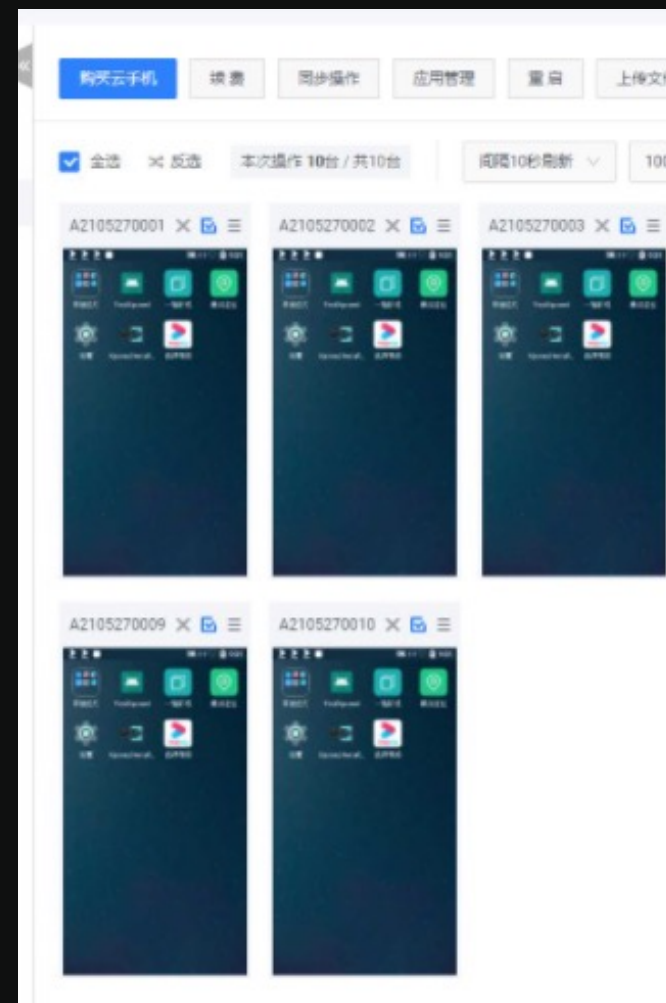
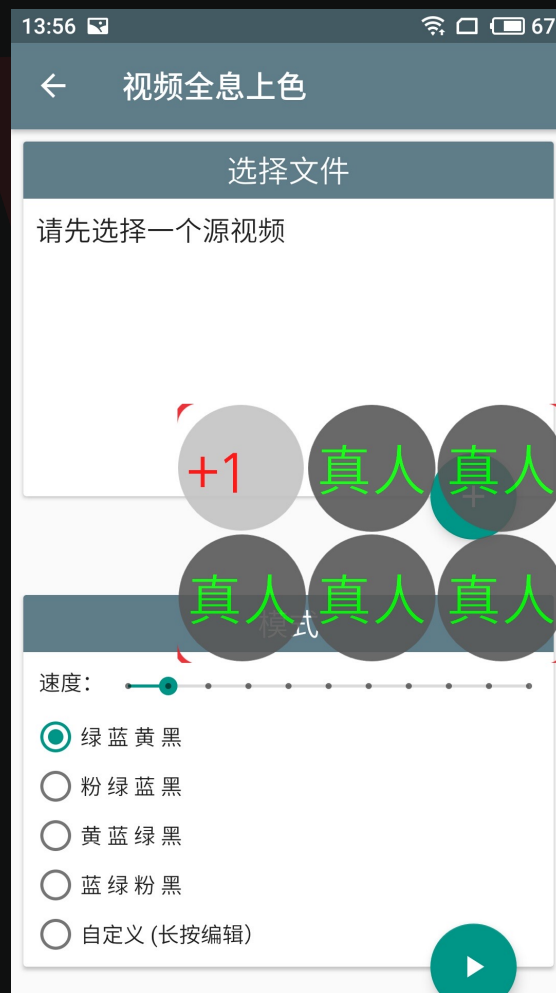
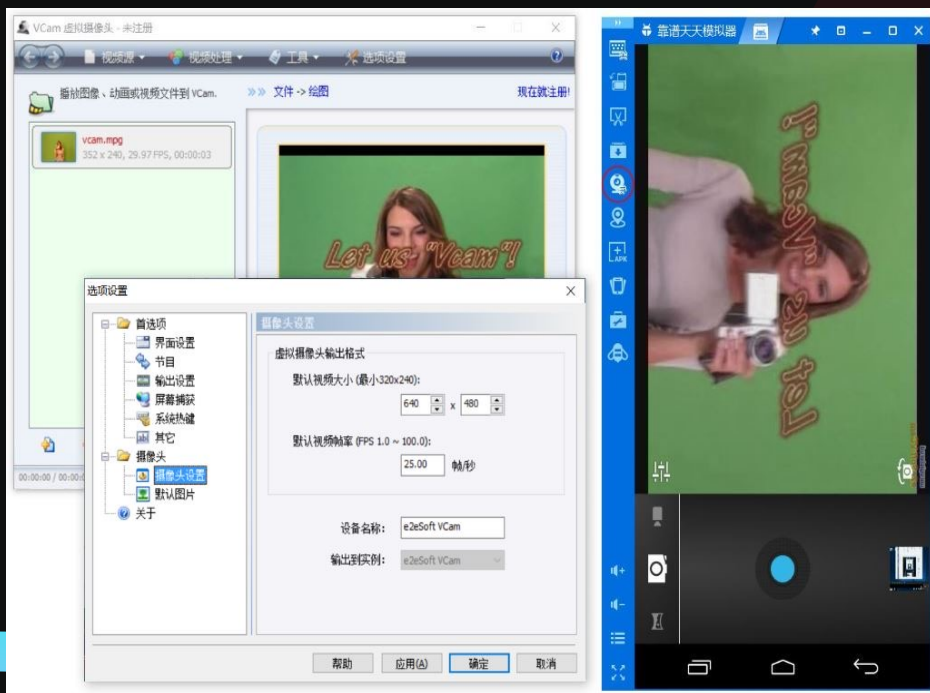
◆ 攻防对抗之旅

人脸识别破解是黑灰产常见的需求之一，常见的作案设备：

真机：定制ROM

模拟器：雷电模拟器、夜神模拟器、逍遥模拟器等

云手机：河马云手机、多多云手机、红手指云手机等



攻防对抗之旅

云相机

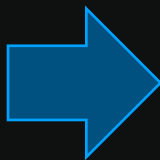


请按照以下顺序操作

- 1、电脑浏览器登录云手机系统，到云机预览界面，鼠标移至云机截图，点击[云机识别码]
- 2、点击页面底部[扫云机识别码]按钮，扫描并成功绑定云机后进入功能选择界面
- 3、在云机内打开所需应用，再点击本机功能界面上的[开启摄像头]、[开启录音]或[本地图片]按钮，完成同步操作

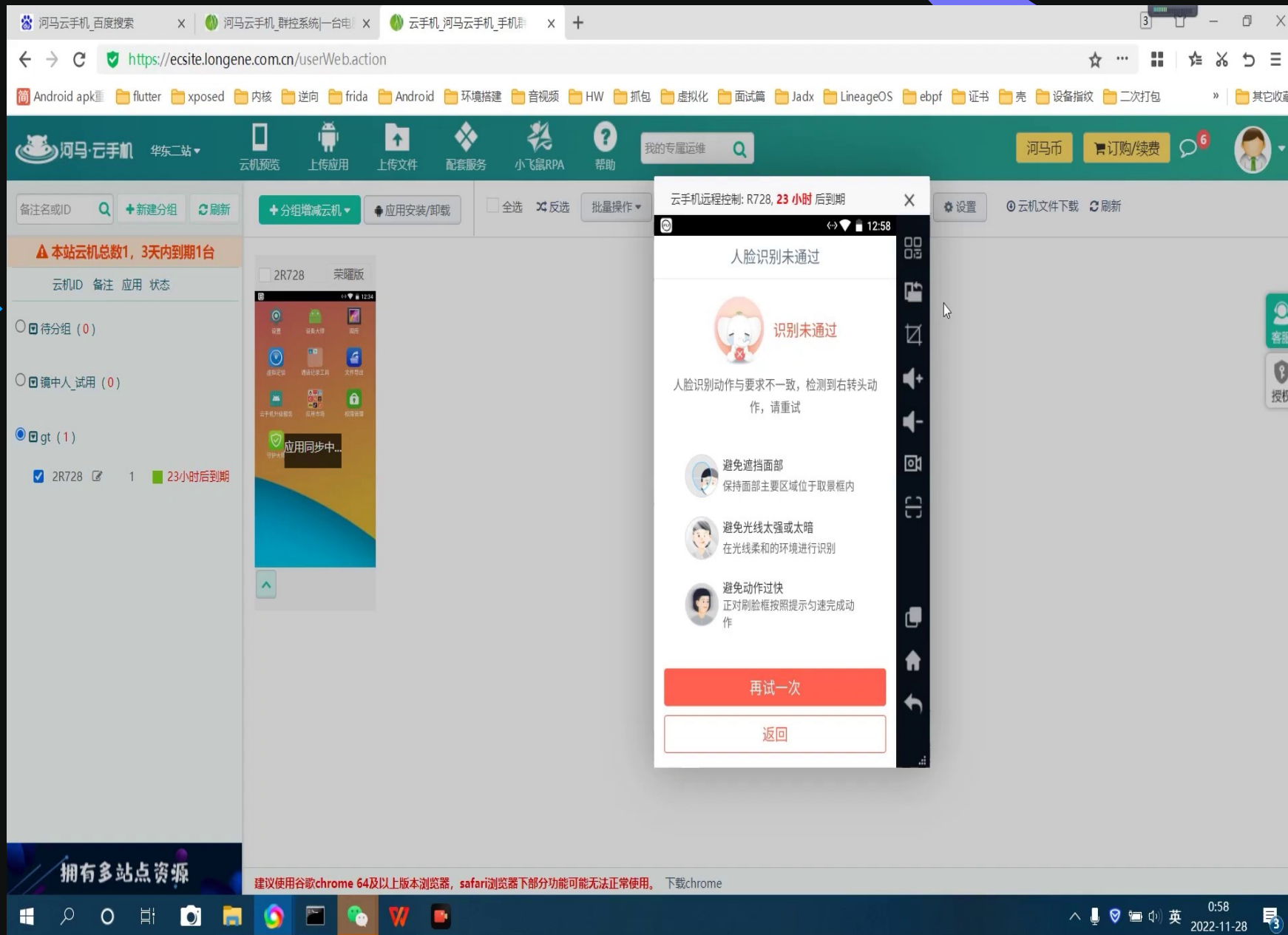
扫云机识别码

小熊录屏



云手机人脸识别测试流程：

1. 下载并安装云相机APP
2. 云手机显示云机识别二维码
3. 测试手机使用云相机扫描，完成绑定
4. 两端配合完成人脸识别



◆ 攻防对抗之旅

```
com.aliyun.identity.face.ToygerPresenter x com.aliyun.identity.face.p190ui.IdentityFaceActivity x

    final /* synthetic */ C44074 this$1;
    {
        JniLib.m15444cV(this, r5, 214);
    }
    @Override // com.aliyun.identity.face.p190ui.IdentityScanFrameView.AnimationListener
    public void onAnimationFinish() {
        this.this$1.this$0.onFaceVerify();
    }
    });
    });
}
}

/* access modifiers changed from: private */
/* access modifiers changed from: public */
191 private void onFaceVerify() {
192     RecordService.getInstance().recordEvent(RecordLevel.LOG_INFO, "faceScan", "status", "face collect completed");
197     IdentityCenter.getInstance().setFaceBitmap(ToygerPresenter.getInstance().getHighQualityFaceImage());
200     startActivity(new Intent(this, FinalVerifyActivity.class));
201     finish();
}

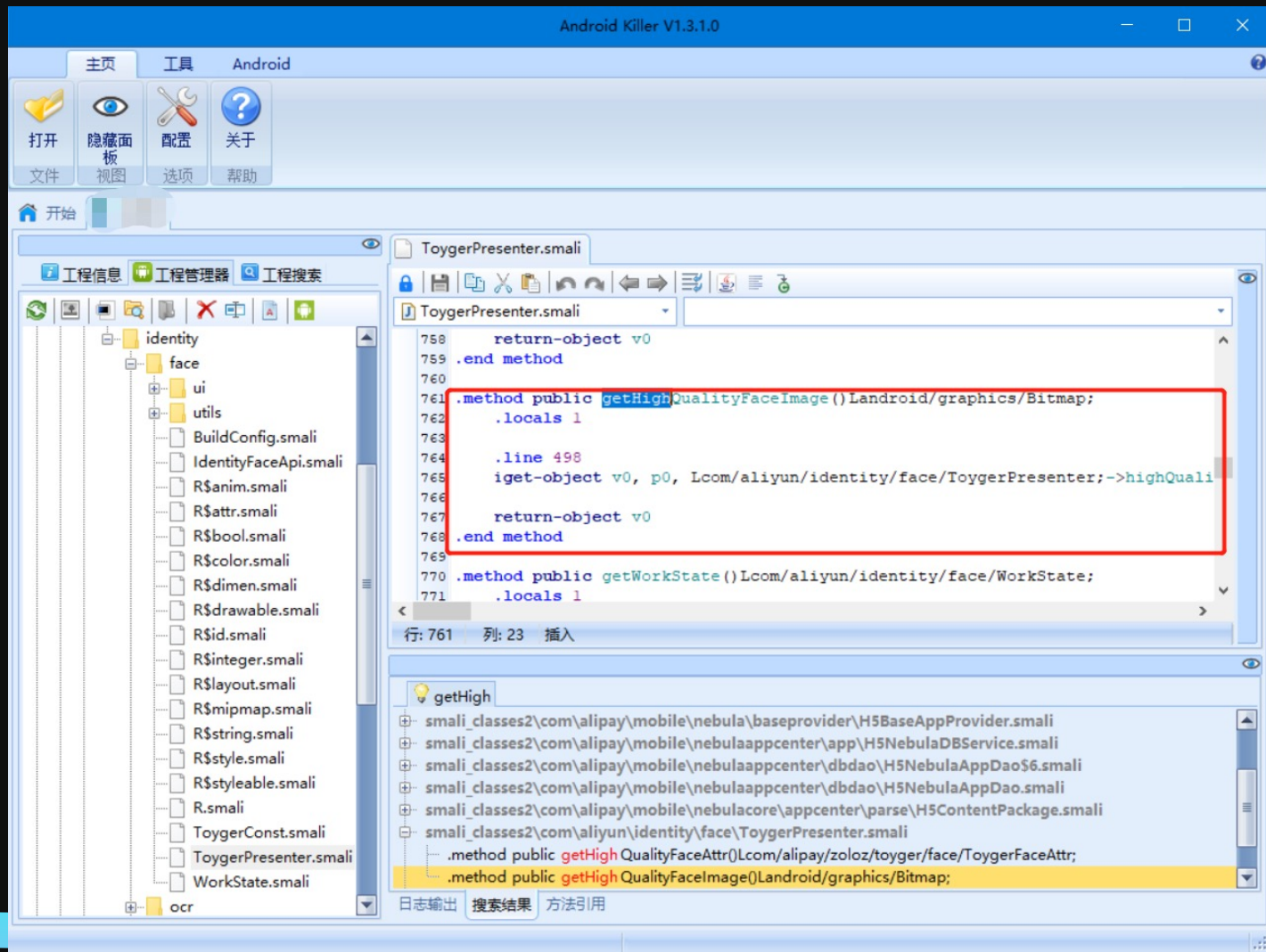
/* access modifiers changed from: private */
/* access modifiers changed from: public */
209 private void sendErrorCode(String str) {
210     Message obtain = Message.obtain();
211     obtain.what = 903;
212     obtain.obj = str;
214     this.uiHandler.sendMessage(obtain);
}

/* access modifiers changed from: private */
/* access modifiers changed from: public */
```



逆向分析某APP，可明显看到其刷脸对应的Activity中定义了一个 **onFaceVerify** 函数，继续阅读代码可看到最终会获取到一张照片传递给下一个Activity进行人脸对比。所以，我们只要能够 Hook **getHighQualityFaceImage** 函数替换照片，即可成功破解目标APP的人脸识别。

◆ 攻防对抗之旅



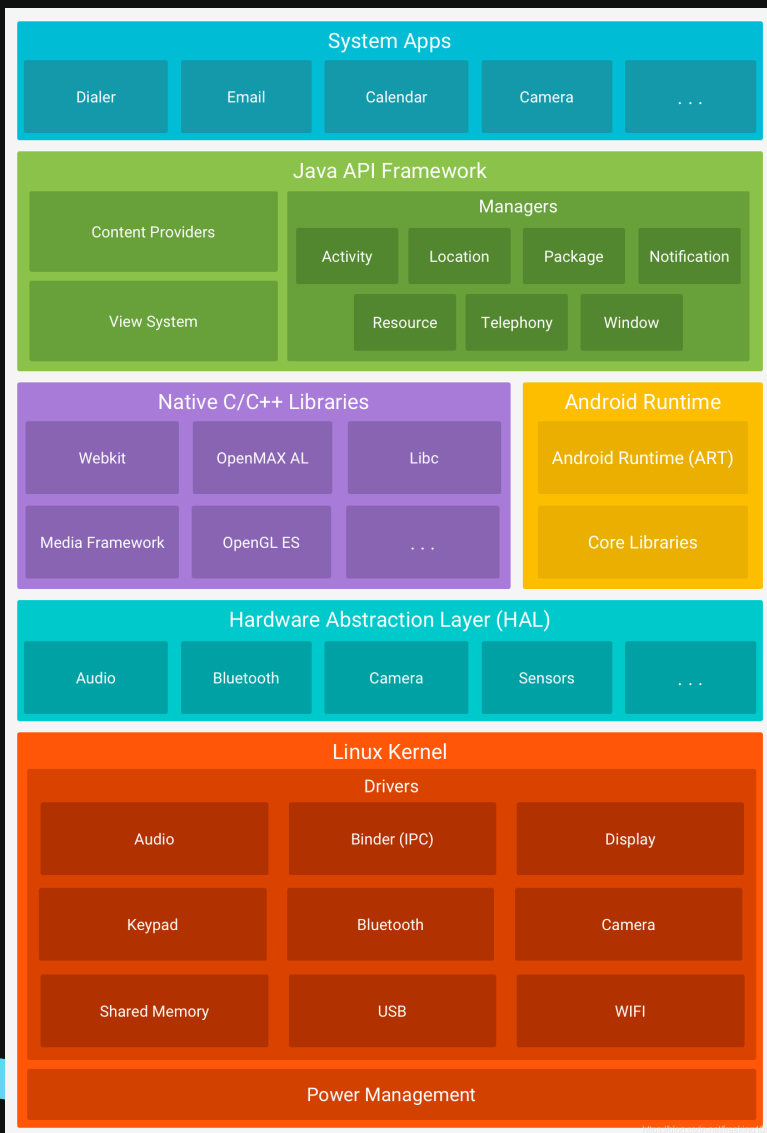
使用AndroidKiller将目标APP反编译成Smali代码，定位关键函数 `getQualityFaceImage` 如左图所示，阅读Smali代码可看到该函数逻辑非常简单，就是将Bitmap类型的变量 `highQualityFace` 返回。

所以我们可以编写代码将预置在手机本地的一张受害人照片转变成Bitmap，然后返回即可。

最后，使用工具提供的二次打包功能及签名功能生成一个新的APP后安装测试。



◆ 攻防对抗之旅



Android系统和APP的关系?

Android是一个基于Linux的操作系统，左侧为安卓系统架构图，可以看出该系统是一个分层系统，自上至下分别是应用层、应用框架层、系统运行库层、硬件抽象层和Linux内核层。而我们所谓的Android APP就是系统中的一个应用，应用层既包括“电话”、“短信”、“相机”等系统自带的APP，也包含“微信”、“支付宝”等APP。简单说，Android APP可以分为两类：系统APP和第三方APP。

APP的运行需要依赖系统底层的支持，比如应用框架层通过接口的形式将一些基本功能提供给上层调用。

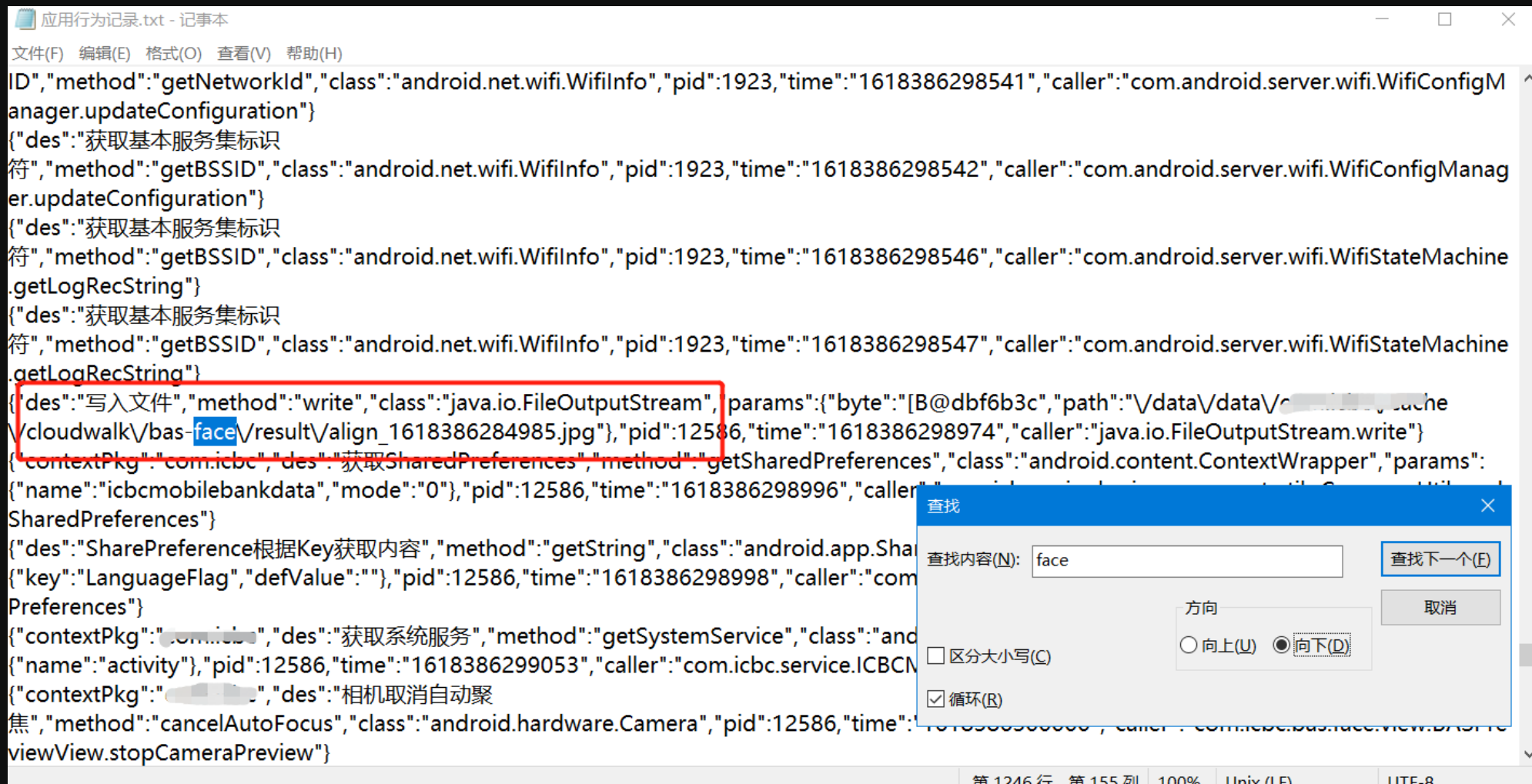
由于Android系统开源的特性，我们可以基于AOSP对系统的任意一层进行修改，从而实现对目标APP的运行进行全流程、全阶段、无感知的监听、劫持甚至篡改。

◆ 攻防对抗之旅

基于AOSP打造一款应用行为分析沙箱，对目标APP运行期间的函数调用、数据交互等行为进行记录。

监听目标：

- 文件操作相关API
- 网络通信相关API
- 资源管理相关API
- 组件交互相关API
- 数据结构相关API
- 密码算法相关API



```
应用行为记录.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
ID,"method":"getNetworkId","class":"android.net.wifi.WifiInfo","pid":1923,"time":"1618386298541","caller":"com.android.server.wifi.WifiConfigM
anager.updateConfiguration"}
{"des":"获取基本服务集标识
符","method":"getBSSID","class":"android.net.wifi.WifiInfo","pid":1923,"time":"1618386298542","caller":"com.android.server.wifi.WifiConfigManag
er.updateConfiguration"}
{"des":"获取基本服务集标识
符","method":"getBSSID","class":"android.net.wifi.WifiInfo","pid":1923,"time":"1618386298546","caller":"com.android.server.wifi.WifiStateMachine
.getLogRecString"}
{"des":"获取基本服务集标识
符","method":"getBSSID","class":"android.net.wifi.WifiInfo","pid":1923,"time":"1618386298547","caller":"com.android.server.wifi.WifiStateMachine
.getLogRecString"}
{"des":"写入文件","method":"write","class":"java.io.FileOutputStream", "params":{"byte":["B@dbf6b3c","path":"\\data\\data\\com.icbc.cache
\\cloudwalk\\bas-face\\result\\align_1618386284985.jpg"],"pid":12586,"time":"1618386298974","caller":"java.io.FileOutputStream.write"}
{"contextPkg":"com.icbc","des":"获取SharedPreferences","method":"getSharedPreferences","class":"android.content.ContextWrapper","params":
{"name":"icbcmobilebankdata","mode":"0"},"pid":12586,"time":"1618386298996","caller":"com.icbc.service.ICBCN
SharedPreferences"}
{"des":"SharePreference根据Key获取内容","method":"getString","class":"android.app.Share
{"key":"LanguageFlag","defValue":""},"pid":12586,"time":"1618386298998","caller":"com
Preferences"}
{"contextPkg":"com.icbc","des":"获取系统服务","method":"getSystemService","class":"and
{"name":"activity"},"pid":12586,"time":"1618386299053","caller":"com.icbc.service.ICBCN
{"contextPkg":"com.icbc","des":"相机取消自动聚
焦","method":"cancelAutoFocus","class":"android.hardware.Camera","pid":12586,"time":"1618386299053","caller":"com.icbc.service.ICBCN
viewView.stopCameraPreview"}
第1246行 第155列 100% Unix (LF) UTF-8
```

◆ 攻防对抗之旅

案例一：

在分析某款APP人脸识别的应用行为记录时发现存在WebView相关函数的调用，可大致猜测该APP的人脸识别过程中会涉及到同H5的交互，进一步打印Webview.loadUrl函数的参数及调用堆栈。

根据参数和调用堆栈可大致猜测该APP刷脸相关界面由H5实现，刷脸完成后会调用loadUrl函数触发回调。

于是尝试对参数内容进行修改，等待超时或手动退出刷脸即可实现破解。

```
*****start_web_log*****
loadUrl
javascript:newFaceRecognitionCallBackFunc({"TranErrorCode":"0402","retFlagCS":"1","OAASKey":"","TranErrorDisplayMsg":"内部出错","TranErrorDisplayMsgEN":""})
android.webkit.WebView.loadUrl(WebView.java:972)
component.webview.CustomWebView.loadUrl(SourceFile:3)
unionlogin.UnionLoginNativeWebViewProxy$10$a$a.a(SourceFile:11)
services.ICBCCComponentLoginService$x0.a(SourceFile:5)
test.a.b(SourceFile:40)
com.icbc.commonrequest.a.onPostExecute(SourceFile:1)
android.os.AsyncTask.finish(AsyncTask.java:695)
android.os.AsyncTask.-wrap1(Unknown Source:0)
android.os.AsyncTask$InternalHandler.handleMessage(AsyncTask.java:712)
android.os.Handler.dispatchMessage(Handler.java:108)
android.os.Looper.loop(Looper.java:164)
android.app.ActivityThread.main(ActivityThread.java:6541)
java.lang.reflect.Method.invoke(Native Method)
com.android.internal.os.Zygote$MethodAndArgsCaller.run(Zygote.java:240)
com.android.internal.os.ZygoteInit.main(ZygoteInit.java:767)
*****end_web_log*****
```



```
*****start_web_log*****
loadUrl
javascript:newFaceRecognitionCallBackFunc({"TranErrorCode":"","retFlagCS":"0","OAASKey":"","TranErrorDisplayMsg":"","TranErrorDisplayMsgEN":""})
android.webkit.WebView.loadUrl(WebView.java:972)
component.webview.CustomWebView.loadUrl(SourceFile:3)
unionlogin.UnionLoginNativeWebViewProxy$10$a$a.a(SourceFile:11)
```


◆ 攻防对抗之旅

案例二：

抓取到网络请求报文后，初步分析发现上传多张照片并经过加密和重排序处理，于是重点观察HashMap、ArrayList等相关API的调用。

```
*****start_arraylist_add*****
size:27453
current:0
ja... ArrayList.add(ArrayList.java:513)
cc... face.base.BASFaceBaseActivity.onMotionCompleted(BASFaceBaseActivity.java:7)
cc... ni.Lib.cV(Native Method)
cc... face.activity.BASFaceActivity.onMotionCompleted(Unknown Source:21)
cc... face.base.BASFaceBaseActivity.doFrontFaceLivess(BASFaceBaseActivity.java:11)
cc... face.base.BASFaceBaseActivity.access$1100(BASFaceBaseActivity.java:1)
cc... face.base.BASFaceBaseActivity$MainHandler.handleMessage(BASFaceBaseActivity.java:28)
an... ndler.dispatchMessage(Handler.java:108)
android.os.Looper.loop(Looper.java:164)
android.app.ActivityThread.main(ActivityThread.java:6541)
java.lang.reflect.Method.invoke(Native Method)
com.android.internal.os.Zygote$MethodAndArgsCaller.run(Zygote.java:240)
com.android.internal.os.ZygoteInit.main(ZygoteInit.java:767)

*****end_arraylist_add*****
*****start_arraylist_add*****
size:27098
current:1
ja... ArrayList.add(ArrayList.java:513)
cc... face.base.BASFaceBaseActivity.onMotionCompleted(BASFaceBaseActivity.java:8)
cc... ni.Lib.cV(Native Method)
cc... face.activity.BASFaceActivity.onMotionCompleted(Unknown Source:21)
cc... face.base.BASFaceBaseActivity.doFrontFaceLivess(BASFaceBaseActivity.java:11)
cc... face.base.BASFaceBaseActivity.access$1100(BASFaceBaseActivity.java:1)
cc... face.base.BASFaceBaseActivity$MainHandler.handleMessage(BASFaceBaseActivity.java:28)
an... ndler.dispatchMessage(Handler.java:108)
android.os.Looper.loop(Looper.java:164)
android.app.ActivityThread.main(ActivityThread.java:6541)
java.lang.reflect.Method.invoke(Native Method)
com.android.internal.os.Zygote$MethodAndArgsCaller.run(Zygote.java:240)
com.android.internal.os.ZygoteInit.main(ZygoteInit.java:767)

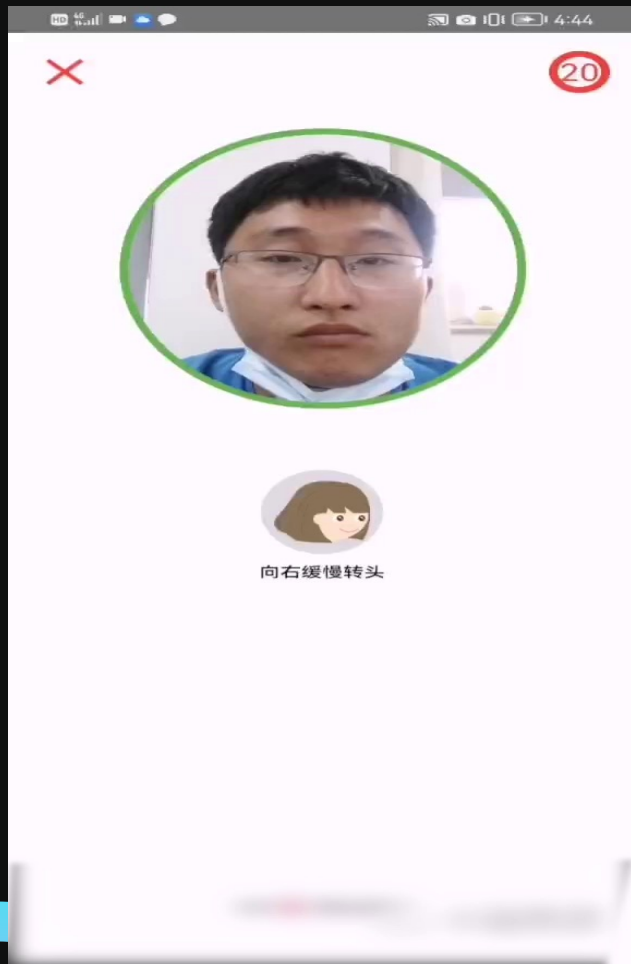
*****end_arraylist_add*****
*****start_arraylist_add*****
size:27917
current:0
ja... ArrayList.add(ArrayList.java:513)
cc... face.base.BASFaceBaseActivity.onMotionCompleted(BASFaceBaseActivity.java:9)
cc... ni.Lib.cV(Native Method)
cc... face.activity.BASFaceActivity.onMotionCompleted(Unknown Source:21)
cc... face.base.BASFaceBaseActivity.doFrontFaceLivess(BASFaceBaseActivity.java:11)
cc... face.base.BASFaceBaseActivity.access$1100(BASFaceBaseActivity.java:1)
cc... face.base.BASFaceBaseActivity$MainHandler.handleMessage(BASFaceBaseActivity.java:28)
an... ndler.dispatchMessage(Handler.java:108)
```



◆ 攻防对抗之旅

案例三:

仔细观察这个视频~



Activity跳转:

- 1.startActivity(Intent intent);
- 2.startActivityForResult(Intent intent,int requestCode);

```
/**
 * Add extended data to the intent. The name must include a package
 * prefix, for example the app com.android.contacts would use names
 * like "com.android.contacts.ShowAll".
 *
 * @param name The name of the extra data, with package prefix.
 * @param value The String data value.
 *
 * @return Returns the same Intent object, for chaining multiple calls
 * into a single statement.
 *
 * @see #putExtras
 * @see #removeExtra
 * @see #getStringExtra(String)
 */
public @NonNull Intent putExtra(String name, String value) {
    if (mExtras == null) {
        mExtras = new Bundle();
    }
    mExtras.putString(name, value);
    return this;
}
```

```
/**
 * Add extended data to the intent. The name must include a package
 * prefix, for example the app com.android.contacts would use names
 * like "com.android.contacts.ShowAll".
 *
 * @param name The name of the extra data, with package prefix.
 * @param value The CharSequence data value.
 *
 * @return Returns the same Intent object, for chaining multiple calls
 * into a single statement.
 *
 * @see #putExtras
 * @see #removeExtra
 * @see #getCharSequenceExtra(String)
 */
public @NonNull Intent putExtra(String name, CharSequence value) {
    if (mExtras == null) {
        mExtras = new Bundle();
    }
    mExtras.putCharSequence(name, value);
}
```

```
public double getDoubleExtra(String name, double defaultValue) {
    return mExtras == null ? defaultValue :
        mExtras.getDouble(name, defaultValue);
}

/**
 * Retrieve extended data from the intent.
 *
 * @param name The name of the desired item.
 *
 * @return the value of an item previously added with putExtra(),
 * or null if no String value was found.
 *
 * @see #putExtra(String, String)
 */
public String getStringExtra(String name) {
    return mExtras == null ? null : mExtras.getString(name);
}

/**
 * Retrieve extended data from the intent.
 *
 * @param name The name of the desired item.
 *
 * @return the value of an item previously added with putExtra(),
 * or null if no CharSequence value was found.
 *
 * @see #putExtra(String, CharSequence)
 */
public CharSequence getCharSequenceExtra(String name) {
    return mExtras == null ? null : mExtras.getCharSequence(name);
}
```

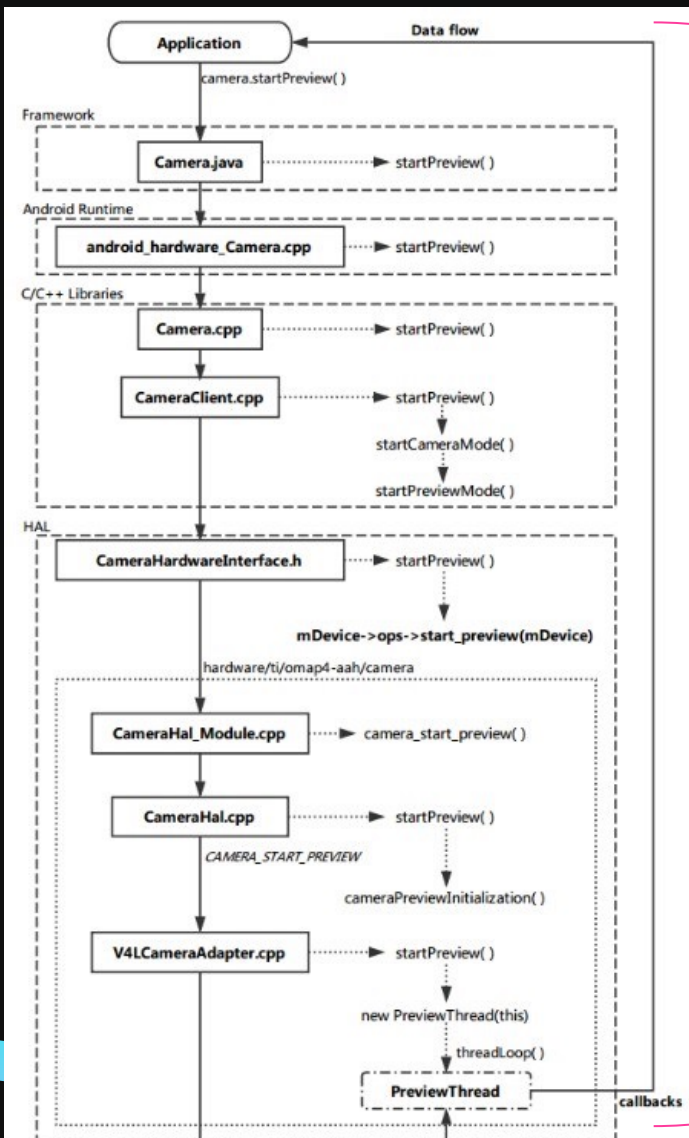
◆ 攻防对抗之旅

The image shows two windows from an Android Studio interface. The left window is the Logcat view, displaying log messages for the 'ICBC_LOG' filter. A red box highlights the log entry: `result_image_path:/storage/emulated/0/Android/data/com.as.face.dev/cache/ce.dev/cache/ce.dev/face/result/align_1666090690950.jpg`. The right window is the Device File Explorer, showing the file system of the emulator. The 'cache' directory is expanded, and a file named 'replace.jpg' is highlighted in yellow.

破解步骤:

1. 将受害人照片存储在手机本地
2. 替换result_image_path为预置照片路径
3. 由任意一人完成人脸识别

攻防对抗之旅



xref: /frameworks/base/core/java/android/hardware/Camera.java

android-8.0.0_r36 | History | Annotate | Line# | Scopes# | Navigate# | Raw | Download

```
782 public final void setPreviewCallback(PreviewCallback cb) {
783     mPreviewCallback = cb;
784     mOneShot = false;
785     mWithBuffer = false;
786     if (cb != null) {
787         mUsingPreviewAllocation = false;
788     }
789     // Always use one-shot mode. We fake camera preview mode by
790     // doing one-shot preview continuously.
791     setHasPreviewCallback(cb != null, false);
792 }
793
```

调用Camera的startPreview函数后，会从应用层经过应用框架层逐层向下调用，直至访问到硬件部分。而数据的返回则是从底层向上，直到应用层，所以我们只需要关注左图中右边这条线，理论上可以在系统的任意一层进行修改从而实现相机数据的劫持和替换。当然，越深入系统底层，对抗成本越高、攻击效果也会更好，

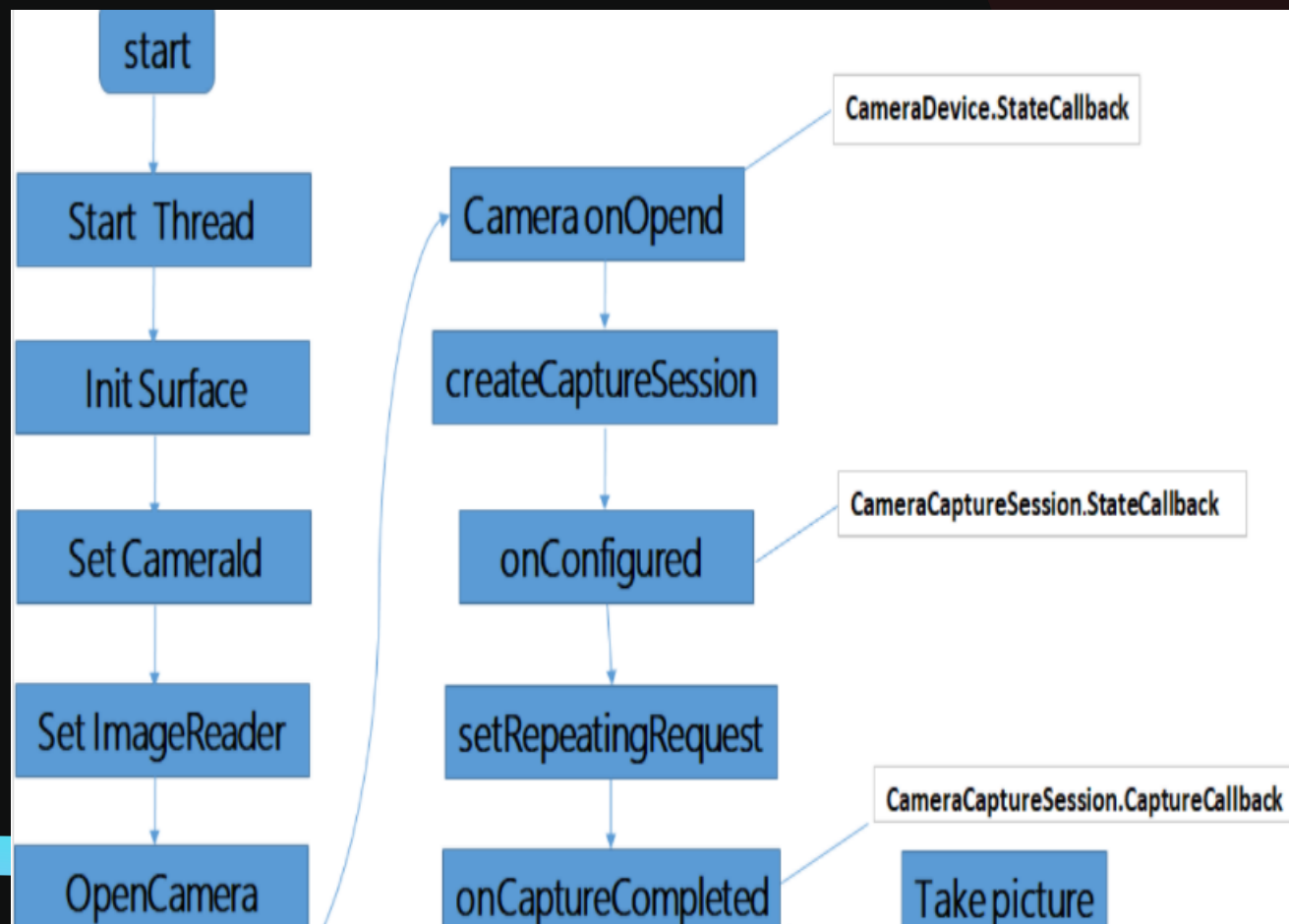
xref: /frameworks/base/core/java/android/hardware/Camera.java

android-8.0.0_r36 | History | Annotate | Line# | Scopes# | Navigate# | Raw | Download

```
0 public void handleMessage(Message msg) {
1     switch(msg.what) {
2     case CAMERA_MSG_SHUTTER:
3         if (mShutterCallback != null) {
4             mShutterCallback.onShutter();
5         }
6         return;
7     case CAMERA_MSG_RAW_IMAGE:
8         if (mRawImageCallback != null) {
9             mRawImageCallback.onPictureTaken((byte[])msg.obj, mCamera);
10        }
11        return;
12    case CAMERA_MSG_COMPRESSED_IMAGE:
13        if (mJpegCallback != null) {
14            mJpegCallback.onPictureTaken((byte[])msg.obj, mCamera);
15        }
16        return;
17    case CAMERA_MSG_PREVIEW_FRAME:
18        PreviewCallback pCb = mPreviewCallback;
19        if (pCb != null) {
20            if (mOneShot) {
21                // Clear the callback variable before the callback
22                // in case the app calls setPreviewCallback from
23                // the callback function
24                mPreviewCallback = null;
25            } else if (!mWithBuffer) {
26                // We're faking the camera preview mode to prevent
27                // the app from being flooded with preview frames.
28                // Set to oneshot mode again.
29                setHasPreviewCallback(true, false);
30            }
31            pCb.onPreviewFrame((byte[])msg.obj, mCamera);
32        }
33    }
34    return;
35}
```

◆ 攻防对抗之旅

安卓系统自带的相机Camera API有两套，Camera2同Camera相比更加复杂，简单使用如下图所示：



与Camera1不同，根据Google的官方文档说明，Camera2取消了实时回调。也就是说，类似Camera1的onPreviewCallback在Camera2中是不存在的，而Camera2提供了ImageReader实现图像的获取。

```
public Image acquireNextImage() {  
    // Initialize with reader format, but can be overwritten by native if the image  
    // format is different from the reader format.  
    SurfaceImage si = new SurfaceImage(mFormat);  
    int status = acquireNextSurfaceImage(si);  
  
    switch (status) {  
        case ACQUIRE_SUCCESS:  
            return si;  
        case ACQUIRE_NO_BUFS:  
            return null;  
        case ACQUIRE_MAX_IMAGES:  
            throw new IllegalStateException(  
                String.format(  
                    "maxImages (%d) has already been acquired, " +  
                    "call #close before acquiring more.", mMaxImages));  
        default:  
            throw new AssertionError("Unknown nativeImageSetup return code " + status);  
    }  
}
```

The screenshot shows a code editor displaying the `acquireNextImage()` method in `ImageReader.java`. The code includes comments and a switch statement that handles different acquisition statuses: `ACQUIRE_SUCCESS`, `ACQUIRE_NO_BUFS`, `ACQUIRE_MAX_IMAGES`, and a default case that throws an `AssertionError`.

◆ 攻防对抗之旅

通过劫持系统返回给目标APP的相机数据，替换每一帧图像。这样，只要可以获取到受害人的相关视频即可实现一种通用性比较高的人脸识别绕过方案，而视频的来源可以是以下几种：

- 1.真实视频
- 2.人脸活化
- 3.人脸合成
- 4.AI换脸

关键攻击步骤如下：

- 1.物料准备，将相关视频导入手机设备，并存储至指定目录
- 2.选择攻击目标，并开始人脸识别的绕过攻击
- 3.启动目标APP，等待攻击环境的初始化
- 4.环境初始化，将视频解析成图像数据，使用数组缓存
- 5.正常运行目标APP，进入到人脸识别流程
- 6.实时切换不同视频



◆ 攻防对抗之旅



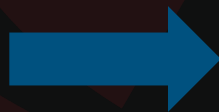
照片“活化”深受黑灰产的欢迎，利用技术手段将人脸照片“活化”，从而生成眨眼、张嘴、摇头、点头等动作视频，而这些视频则可能被黑客用来突破人脸识别。



◆ 攻防对抗之旅

偷拍这一可耻的行为在生活中时常发生，不法分子在受害人不知情的情况下，偷偷录下一些私密的镜头，在一些色情网站上常常会有一些“密室偷拍”的剧情，偷拍是非常可怕的。

回归到今天的主题，偷拍也可能导致人脸信息的泄露，比如在拥挤的地铁上，当我们困意来袭，环顾四周找寻座位的时候，如果这几个瞬间被不法分子偷拍到，那么他们就可以通过一些合成技术生成突破人脸识别所需要的眨眼、张嘴、摇头等一系列的动作视频。



◆ 攻防对抗之旅

AI换脸，科技术语，是指通过AI人工智能技术，把别人的脸换成自己的脸。2023年5月24日，中国互联网协会发文提示“AI换脸”新骗局，利用“AI换脸”、“AI拟声”等虚假音视频，进行诈骗、诽谤的违法行为屡见不鲜。

在右侧视频中，我们利用“AI”换脸技术将同事A的脸换成同事B的脸，通过屏幕转录尝试攻击，并使用同事B的身份证骗过客服人员的判断，并且可以成功绕过后续的人脸识别。

如何应用到我们的通杀方案中？

基于RTMP推拉流的“直播式”人脸识别破解方案：

- 1.通过RTMP推流实时上传“AI换脸”后的图像数据到RTMP服务端
- 2.手机端通过RTMP拉流获取图像数据后实时的进行替换

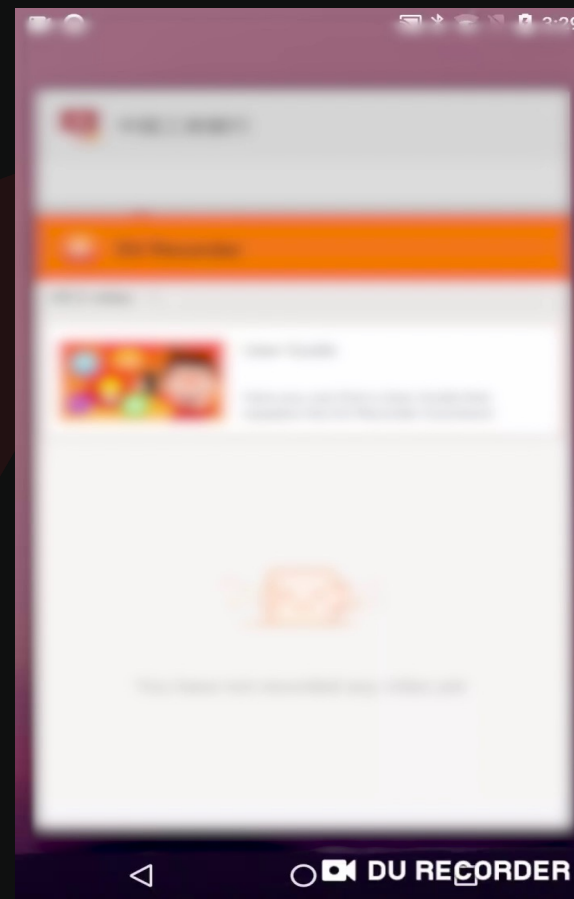


◆ 攻防对抗之旅

关于移动人脸识别安全攻防，我们不能仅关注人脸识别破解本身，也应该关注一些可以提升破解效率、降低攻击成本的内容，比如自动化绕过人脸识别、人脸参数篡改等等。



重点监控JSONObject、Gson、FastJson等函数的调用，定位动作相关参数，可发现该APP动作对应参数名为“ActionType”，使用1-4分别对应四个动作，可多动作。



◆ 常见防护策略

以攻促防，攻防相长！前面重点介绍了许多人脸识别破解的思路和方案，移动人脸识别安全攻防的本质也是移动APP的安全攻防，因此移动端人脸识别的安全性同样依赖主流的客户端防护策略，比如：

- 1.加固
- 2.APP运行环境检测
- 3.防抓包

“加固+环境检测+防抓包”的体系化防护策略对移动端人脸识别的安全性起到一定的保护作用，除此之外部分安全防护水平较高的厂商还会从提升数据篡改难度（防篡改）、关键函数保护等角度采取有效性和针对性更强的防护措施。

数据防篡改：

- 1.照片加水印
- 2.人脸位置偏移
- 3.三色光
- 4.相机参数变化

关键函数保护：

- 1.函数调用堆栈
- 2.检测关键函数是否被Hook或调试

◆ 总结与展望

四个维度：业务流程、攻防介质、攻防设备、技术手段

- 1.人脸识别全流程均存在一定的安全风险
- 2.无介质、照片和视频均可能实现人脸识别的破解
- 3.模拟器、定制ROM、云手机等黑灰产常用的“过人脸”设备方案可直接用于攻防对抗
- 4.移动人脸识别安全攻防对抗的本质也是移动APP的安全攻防对抗
- 5.定制系统可实现对APP的降维打击，全面排查人脸识别的安全风险

未来的潜在威胁？

- 1.技术+欺诈的高度结合
- 2.人工智能带来的安全挑战
- 3.来自内核级别的降维打击
- 4.跨平台的人脸识别安全攻防
- 5.跨时空的人脸识别安全攻防



Android系统

iOS系统

Windows Phone系统

◆ 总结与展望

跨平台的人脸识别安全攻防：

一种攻击思路，很可能在其他的平台上复现成功，未来要做到多平台防护水平的同步也是不小的挑战。

```
; NSMutableArray *__cdecl -[LiveFaceViewController imageDataArr](LiveFaceViewController *self, SEL)
__LiveFaceViewController_imageDataArr_
; __unwind {
ADRP    X8, #_OBJC_IVAR_$_LiveFaceViewController._imageDataArr@PAGE ; NSMutableArray *imageDataArr;
LDRSW  X8, [X8,#_OBJC_IVAR_$_LiveFaceViewController._imageDataArr@PAGEOFF] ; NSMutableArray *imageDataArr;
LDR    X0, [X0,X8]
RET
; } // starts at A774
; End of function -[LiveFaceViewController imageDataArr]
```



```
ret---(
  {length = 30236, bytes = 0xffd8ffe0 00104a46 49460001 010000d8 ... f000a876
    633ffffd9 },
  {length = 27976, bytes = 0xffd8ffe0 00104a46 49460001 010000d8 ... 7a0a1b29
    2d4ffffd9 },
  {length = 26016, bytes = 0xffd8ffe0 00104a46 49460001 010000d8 ... 2493d00a
    7611ffffd9 },
  {length = 26016, bytes = 0xffd8ffe0 00104a46 49460001 010000d8 ... 2493d00a
    7611ffffd9 }
)
```

跨时空的人脸识别安全攻防：

- 1.逻辑切换开关
- 2.旧流程遗留代码
- 3.历史留存客户端



感谢您的观看！

THANK YOU FOR YOUR WATCHING

