Mobile Applications: a Backdoor into Internet of Things?

Axelle Apvrille - FortiGuard Labs, Fortinet

October 2016
How would YOU reverse engineer IoT?

A solution for AV analysts & software security researchers

Example 1: Connected toothbrush

Example 2: Sony Smart Watch 2

Example 3: House alarm

Conclusion
That’s your new task

How are you going to reverse it?
1/5 - Browse the web for documentation

![Sony SmartWatch 2 Image]

---

**Quick facts**

- Operating system: Micrium uC/OS-II

---

**Network**

<table>
<thead>
<tr>
<th>Technology</th>
<th>No cellular connectivity</th>
</tr>
</thead>
</table>

**Launch**

<table>
<thead>
<tr>
<th>Announced Status</th>
<th>2013, June Available, Released 2013, October</th>
</tr>
</thead>
</table>

**Body**

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>42 x 41 x 9 mm (1.64 x 1.61 x 0.35 in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>122.5 g (4.34 oz)</td>
</tr>
<tr>
<td>Build</td>
<td>Aluminum</td>
</tr>
<tr>
<td>SIM</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>- IP57 certified - dust and water resistant</td>
</tr>
<tr>
<td></td>
<td>- Water resistant up to 1 meter and 30 minutes</td>
</tr>
<tr>
<td></td>
<td>- Compatible with standard 24mm straps</td>
</tr>
</tbody>
</table>

**Display**

<table>
<thead>
<tr>
<th>Type</th>
<th>Capacitive touchscreen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>1.6 inches (~46.8% screen-to-body ratio)</td>
</tr>
<tr>
<td>Resolution</td>
<td>220 x 176 pixels (~176 ppi pixel density)</td>
</tr>
<tr>
<td>Multitouch</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- Always-on display</td>
</tr>
</tbody>
</table>

**Platform**

<table>
<thead>
<tr>
<th>OS</th>
<th>Android OS compatible</th>
</tr>
</thead>
</table>

**Memory**

<table>
<thead>
<tr>
<th>Card slot</th>
<th>No</th>
</tr>
</thead>
</table>

**Camera**

| No |

**Sound**

<table>
<thead>
<tr>
<th>Alert types</th>
<th>Vibration, MP3, WAV ringtones</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loudspeaker</td>
<td>Yes</td>
</tr>
<tr>
<td>3.5mm jack</td>
<td>No</td>
</tr>
</tbody>
</table>

**Communications**

<table>
<thead>
<tr>
<th>WLAN</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluetooth</td>
<td>v3.0</td>
</tr>
<tr>
<td>GPS</td>
<td>No</td>
</tr>
</tbody>
</table>
2/5 - Hardware teardown

- Microscope
- Oscilloscope
- Silicon die analysis
- Firmware
- Interface analysis: JTAG, USB, CAN, Serial...

$ lsusb
... no smart watch :(

Photo credit: engadget
When I asked around me for another solution:

“Kidnap the developer, get access to his/her PC and grab the sources”

;-(

Adapted from Pico le Croco
4/5 - Sniff network traffic

Good idea!

In practice, how well does it work for the smart watch?

- No Wifi
Good idea!

In practice, how well does it work for the smart watch?

- No Wifi
- Bluetooth traffic!
Good idea!

In practice, how well does it work for the smart watch?

- No Wifi
- Bluetooth traffic!
- ... encrypted! Use Ubertooth?
Good idea!

In practice, how well does it work for the smart watch?

- No Wifi
- Bluetooth traffic!
- ... encrypted! Use Ubertooth?
- Flow of bytes. No label.

Network traffic analysis: a full time job?

- Packet 10703 ok
- Packet 12453 is not from my toothbrush
- First byte is direction
- Next four are sequence number
- Packet 47002 ends session

adapted from Pico le Croco
5/5 - Develop a smart app for tests

```java
@Override
public void onActiveLowPowerModeChange(boolean lowPowerModeOn) {
    mIsInActiveLowPower = lowPowerModeOn;
    Log.d(SmartWatchExtensionService.LOG_TAG, "onActiveLowPowerModeChange: lowPower="
            + mIsInActiveLowPower
            + " powerButton=" + mPowerButtonPressed
    );
    sendText(R.id.tv_explanation, "Touch screen."
    if (mIsInActiveLowPower) {
        sendText(R.id.tv_title, "Press to leave Active Low Power Mode."
    } else {
        sendText(R.id.tv_title, "Press to enter Active Low Power Mode."
    }
    mPowerButtonPressed = false;
}

private void setupClickables(Context context)
    LayoutInflator inflator = (LayoutInflator) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    View layout = inflator.inflate(R.layout.sample_activity);
    mLayout = parseLayout(layout);
    if (mLayout != null) {
        ControlView mode = mLayout.findViewById(R.id.address_mode);
        mode.setOnClickListener(new OnClickListener()
            @Override
            public void onClick() {
                Log.d(SmartWatchExtensionService.LOG_TAG, "Requesting to switch to Active Low Power mode.
                mPowerButtonPressed = true;
                if (!mIsInActiveLowPower) {
                    Log.d(SmartWatchExtensionService.LOG_TAG, "Switching to Active Low Power mode.
                }
```
It is feasible but... **good luck**
Now, reverse this one!

No. Your experience with the smart watch won’t help.
How well do our RE techniques work for the toothbrush in practice?

Browse for documentation
No technical info :(  

Hardwear teardown
None so far. To be done ;)

Network traffic
No wifi
No Bluetooth.
There is Bluetooth Low Energy (≠ Bluetooth)

App development
No possibility to develop an app
How would YOU reverse engineer IoT?

A solution for AV analysts & software security researchers

Example 1: Connected toothbrush

Example 2: Sony Smart Watch 2

Example 3: House alarm

Conclusion
Is there an easier way to reverse?

→ Yes: reverse engineer the mobile app

Adapted from http://picolecroco.free.fr/images/dessins/2013/pico-59-soude.jpg
Most IoT come with their connected app
How would YOU reverse engineer IoT?

A solution for AV analysts & software security researchers

Example 1: Connected toothbrush

Example 2: Sony Smart Watch 2

Example 3: House alarm

Conclusion
Beam toothbrush

$\text{teeth} = 64$
$\text{renew} = \text{true}$
$\text{sharp} = \text{true}$
SQL tables - reversing iOS app

Tip: search for `primaryKey`

Contents of each table:
mappings func
SQL tables: what we work out

**Insured**
- InsuredID
- Title
- first_name
- middle_initial
- last_name
- post_name
- relation_to_policy_holder
- Gender
- Dob
- user_id
- sequence_num

**DiscountPolicy**
- NSString *coverage;
- discountPolicyID
- group_code
- plan_code
- effective_at
- terminated_at
- insureds

**BTMainCardCell**
- ViewModel
- NSString *policyID
- NSString *activeStatusDate
- NSString *policyStatusTitle
- int policyStatus
- NSString *stars
- NSString *helpNumberForDisplay
- NSString *helpNumberForCall
Reconstructing implementation design

_OBJC_CLASS_METHODS_UserSummary, 0, 0, 0, 0>, UserSummary
_OBJC_INSTANCE_METHODS_UserSummary ___objc2_meth_list <0xC, 4>
  ; DATA XREF: ___objc_const:UserSummary_$classData↓0
    __objc2_meth <aBeamscorerou_2, aI804, \ ; UserSummary - (int)beamScoreRoundedInteger+
      __UserSummary_beamScoreRoundedInteger_+1>
    __objc2_meth <sel_beamScoreRoundedString, a804_0, \ ; UserSummary - (id)beamScoreRounded+
      __UserSummary_beamScoreRoundedString_+1>
    __objc2_meth <sel_sortedRollingEventsArray, a804_0, \ ; UserSummary - (id)sortedRoll+
      __UserSummary_sortedRollingEventsArray_+1>
    __objc2_meth <sel_propertiesDictionaryExclusionList, a804_0, \ ; UserSummary - (id)+
      __UserSummary_propertiesDictionaryExclusionList_+1>
UserSummary_$properties __objc2_prop_list <8, 0xD>
  ; DATA XREF: __objc_const:UserSummary_$classData↓0
    __objc2_prop <aBeanscore, aTNsnumberRDN> ; @property (readonly, retain,@dynamic,
    __objc2_prop <aNumberofbrushd, aTNsnumberRDN> ; @property (readonly, retain,@dynamic,
    __objc2_prop <aUsersummaryid, aTNsstringRDN> ; @property (readonly, retain,@dynamic,
    __objc2_prop <aNumberofstars, aTNsnumberRDN> ; @property (readonly, retain,@dynamic,
    __objc2_prop <aLastnumberofst, aTNsnumberRDN> ; @property (readonly, retain,@dynamic,
    __objc2_prop <aBrushstreak, aTNsnumberRDN> ; @property (readonly, retain,@dynamic,
    __objc2_prop <aConsecutiveove, aTNsnumberRDN> ; @property (readonly, retain,@dynamic,
    __objc2_prop <aAveragelifet_2, aTNsstringRDN> ; @property (readonly, retain,@dynamic,
    __objc2_prop <aAveragelifetim, aTFrdn> ; @property (readonly, @dynamic, nonatomic)
    __objc2_prop <aUser, aUSerRDN> ; @property (readonly, retain, @dynamic, nonatomic)
    __objc2_prop <aRollingevents, aTNssetRDN> ; @property (readonly, retain, @dynamic
    __objc2_prop <aBeamscoreround, aTiRN> ; @property (readonly, nonatomic) int beamS
    __objc2_prop <aBeamscorerou_3, aTNsstringRN> ; @property (readonly, nonatomic) NS
Classes, methods, fields: what we work out

**BTCreateUserController**
- `BTSetupDevice *setupDevice`
- `NSString *defaultHeader`
- `LabelText`

**BTSetupDevice**
- `BTBrushData *_brushData;`
- `BTSetupUser *parentUser;`
- `BTSetupUser *childUser;`
- `NSSet *connectedDevices;`

**BTChildUserInfoViewController**
- `delegate`
- `name`
- `capturePhotoButton`

**UserSummary**
- `NSNumber *beamScore,`
- `*numberOfBrushDaysLeft,`
- `*numberOfStars,`
- `*lastNumberOfStars,`
- `*brushStreak`
- `*consecutiveOver2Min`
- `NSString *userSummaryID,`
- `averageLifetimeBrushDurationString,`
- `beamScoreRoundedString;`  
- `averageLifetimeBrushDuration`  
- `User *user;`
- `*rollingEvents;`
- `*brushScoreValueLabel`
- `*brushScoreTitleLabel,`
- `Streak value & title,`
- `Average brush time Value & title`

**BTUserProfileSettingsTableViewCell**
- `pairNewBrushButton`
- `Name,`
- `PictureButton,`
- `AvatarImageView,`
- `avatarImageButton`

**ClientSession**
- `NSString *clientSessionID;`
- `NSNumber *currentSession;`
- `NSDate *unpausedAt;`
- `NSDate *startTime;`
- `NSNumber *duration;`
- `NSNumber *syncCount;`
- `NSString *clientDeviceID;`
Classes, methods, fields: what we work out

**BTBrushEvent**

- int eventType;
- NSDate *date;
- float duration;
- NSString *macAddress;
- int eventIndex;
- NSString *rawData;

**BTBrushData**

- NSUUID *uuid;
- NSString *deviceName;
- *macAddress;
- *appearance;
- *manufacturer,
- *partialMacAddress,
- *model, *serialNumber
- *firmwareRevision,
- *hardwareRevision,
- notify
- float batterylevel;
- motorIntensity;
- char
- autoOffTimerEnabled,
- quadrantTimerEnabled;
- *lastTimeFromBrush;
- *lastAccelerometerValues;
- *accelerometerValues,
- *lastBrushDetected;
- char buttonDown,
- IsBrushing,
- motorState
- double brushingDuration;
- NSDate
- *lastBrushDetected;
- char buttonDown,
- IsBrushing,
- motorState
- double brushingDuration;
- NSDate
- *lastTimeFromBrush;
- *lastAccelerometerValues;
- *accelerometerValues,
- *lastBrushDetected;
- char buttonDown,
- IsBrushing,
- motorState
- double brushingDuration;
- NSDate
- *lastBrushDetected;
- char buttonDown,
- IsBrushing,
- motorState
- double brushingDuration;
- NSDate
- *lastTimeFromBrush;
- *lastAccelerometerValues;
- *accelerometerValues,
- *lastBrushDetected;
- char buttonDown,
- IsBrushing,
- motorState
- double brushingDuration;
- NSDate
- *lastBrushDetected;
- char buttonDown,
- IsBrushing,
Classes, methods, fields: what we work out
UUID of Bluetooth Low Energy characteristics

```java
public void writeQuadrantBuzz(BLEDevice device, boolean arg6, boolean arg7) {
    BluetoothGatt gatt = this.getBluetoothGatt(device);
    if (gatt != null) {
        this.send2charac(gatt, "04234f8e-75b0-4525-9a32-193d9c899d30", "19dc94fa-7bb3-4248-9b2d-1a0cc6437af5",
                        ByteSerialize.boolean2byte(arg6, arg7));
    }
}

public void setMotorSpeed(BLEDevice arg5, float arg6) {
    BluetoothGatt v0 = this.getBluetoothGatt(arg5);
    if (v0 != null) {
        this.send2charac(v0, "04234f8e-75b0-4525-9a32-193d9c899d30", "833da694-51c5-4418-b4a9-3482de840aa8",
                        ByteSerialize.float2byte(arg6));
    }
}
```
Demo: changing toothbrush motor speed

- Percentage to byte conversion: $((1 - \frac{x}{100}) \times 139) + 69$
- Writing to toothbrush: BLE characteristic (833d...) found from RE
Demo: reading toothbrush battery level

- Byte to battery level formula: \( 100 \times \frac{0.001221x-1.1}{1.5-1.1} \)
- 5 V for 12 bits = \( \frac{5}{2^{12}} \)
- 1.1 min voltage, 1.5 max voltage?

```bash
axelle@labtop ~/git-cuckoo/beam-brush/prog $ sudo python read-battery.py -v
======== Beam Brush Battery Level Utility ========
>read_battery(): handle=0x2f verbose=1
Connecting...
Connected
    GATT response: 704b
    Little Endian: 1207
    Returning: 93.4 percent
Disconnected
< read_battery()
Battery percentage 93.4
```

```bash
axelle@labtop ~/git-cuckoo/beam-brush/prog $ ```
Sidenote: why should we care?

Who cares changing toothbrush motor speed?!
Sidenote: why should we care?

Who cares changing toothbrush motor speed?!

Two scenarios:

1. **Ransomware**. Attacker drains your batteries if you don’t pay.
2. **Propagating virus**. Infected bytes? infected firmware?

Even harmless IoT need to be secured
How would YOU reverse engineer IoT?

A solution for AV analysts & software security researchers

Example 1: Connected toothbrush

Example 2: Sony Smart Watch 2

Example 3: House alarm

Conclusion
Architecture

Smart Accessory

- Bluetooth
- Constanza msg

Smartphone

- Host application
- SmartConnect
- Twitter
- Smart Extensions
Reversing host app protocol

```java
public class RequestForceCrash extends CostanzaMessage {
    public static final int FORCE_CRASH_REQUEST_MAGIC
        = 0xC057A72A;
    private int mMagic;

    public RequestForceCrash(int newMessageId) {
        super(newMessageId);
        this.type = 666;
        this.mMagic = 0xC057A72A;
    }
}
```

666 → Number of the Beast
C057A72A → Costanza
public class RequestForceCrash extends Costanza {
...
}

public abstract class CostanzaMessage {
...
}

libprotocol.so

pack()

Header - 12 bytes  Action  Value

Sending Costanza messages
Hidden screen

RequestForceCrash packets are sent by a hidden activity!

```bash
$ su root
$ am start -n com.sonymobile.smartconnect.smartwatch2/com.sonymobile.smartconnect.hostapp.costanza.StartupActivity
Starting: Intent { cmp=com.sonymobile.smartconnect.hostapp.costanza.StartupActivity }
```

![SmartWatch 2 debug menu]

- [Debug] Factory reset
- [Debug] Re-download FOTA
- [Debug] Disable automatic FOTA
- [Debug] Force crash on watch
- [Debug] Show Costanza Log
- [Debug] No connection!
Debug command work
$ adb forward tcp:58616 tcp:58616
$ telnet localhost 58616
Trying 127.0.0.1...
Connected to localhost.
Escape character is `^]`.
Debug console for Costanza.
Connection will be closed when you leave the log
(hit the "Back" button on your phone.)

Please issue commands:
Outline

How would YOU reverse engineer IoT?

A solution for AV analysts & software security researchers

Example 1: Connected toothbrush

Example 2: Sony Smart Watch 2

Example 3: House alarm

Conclusion
There’s an Android app for the alarm

- Protect your house against burglars
- Controllable by SMS

But it’s not very user friendly...

Comply to a strict SMS formatting

💡 So, they created an **Android app** to assist end-users
Outbox is not secure

In the **outbox**, the SMS contains the **password** and **phone number** of the alarm.

You get it? You control the alarm!

Fake data, of course :D

Let's suppose you are a **wise person** and erase the SMS
You are wise, aren’t you?
With the Android app, it’s **worse**!

With the app, **2 security issues**!!

Your credentials are at risk even if you erased the SMS!

Without the app, **1 security issue**.

With the app, **2 security issues** !!!
How would YOU reverse engineer IoT?

A solution for AV analysts & software security researchers

Example 1: Connected toothbrush

Example 2: Sony Smart Watch 2

Example 3: House alarm

Conclusion
Thanks for your attention!

Thanks to Beam Technologies for providing a free user account for testing purposes.
Aurélien Francillon, Ludovic Apvrille and Ruchna Nigam
Students: Axel Ehrenstrom and Soufiane Joumar

References
- Fortinet’s blog
- FortiGuard Research

Awesome slides? Thanks! That’s LaTeX
Like the crocodile? He’s called Pico