



Mobile Applications: a Backdoor into Internet of Things?

Axelle Apvrille - FortiGuard Labs, Fortinet

October 2016

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

That's your new task



How are you going to reverse it?

1/5 - Browse the web for documentation

The image shows two side-by-side screenshots. On the left is a screenshot of the Sony Developer World website for the SmartWatch 2. It features a large image of the watch, a 'Get Started' button, and a 'Quick facts' section with the note 'Operating system: Micrium uC/OS-II'. On the right is a screenshot of an XDA Developers product review for the Sony SmartWatch 2. The review includes a headline 'Win an Honor 8!', a progress bar showing 'By 2020, 72% of Canadian...', and a detailed technical specification table.

	Technology
NETWORK	No cellular connectivity
LAUNCH	Announced 2013, June Status Available. Released 2013, October
BODY	Dimensions 42 x 41 x 9 mm (1.65 x 1.61 x 0.35 in) Weight 122.5 g (4.34 oz) Build Aluminum SIM No <ul style="list-style-type: none">- IP57 certified - dust and water resistant- Water resistant up to 1 meter and 30 minutes- Compatible with standard 24mm straps
DISPLAY	Type Capacitive touchscreen Size 1.6 inches (~46.8% screen-to-body ratio) Resolution 220 x 176 pixels (~176 ppi pixel density) Multitouch Yes <ul style="list-style-type: none">- Always-on display
PLATFORM	OS Android OS compatible
MEMORY	Card slot No
CAMERA	No
SOUND	Alert types Vibration; MP3, WAV ringtones Loudspeaker Yes 3.5mm jack No
COMMS	WLAN No Bluetooth v3.0 GPS No

2/5 - Hardware teardown

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

```
$ lsusb  
... no smart watch :( ...
```

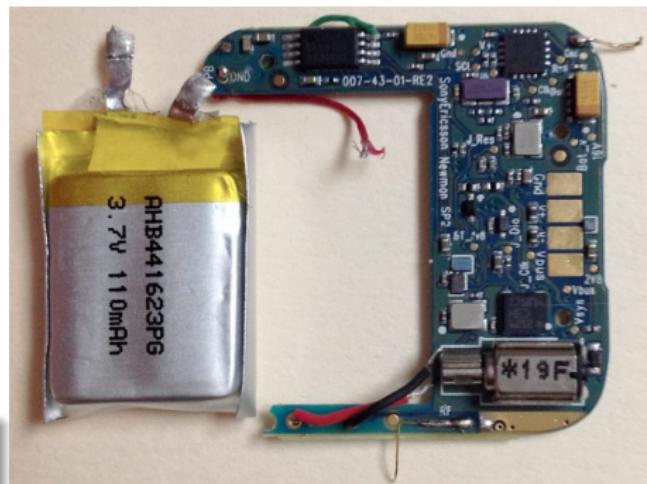


Photo credit: [engadget](#)

3/5 - Social engineering

When I asked around me for another solution:

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

;-)

As a last resort,



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

4/5 - Sniff network traffic

Good idea!

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

- ▶ No Wifi
- ▶ Bluetooth traffic!

4/5 - Sniff network traffic

Good idea!

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

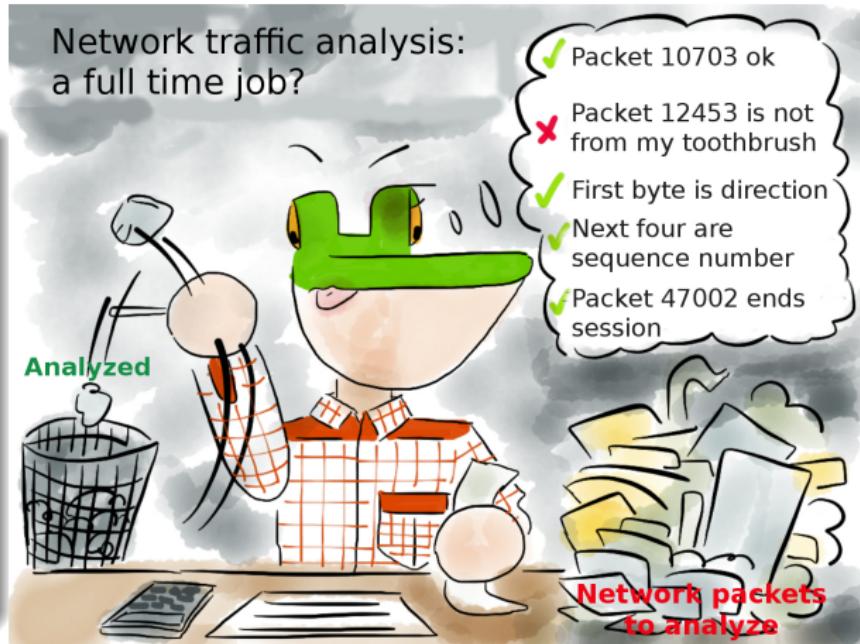
- ▶ No Wifi
- ▶ Bluetooth traffic!
- ▶ ... encrypted! Use [Ubertooth](#)?

4/5 - Sniff network traffic

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.



adapted from [Pico le Croco](#)

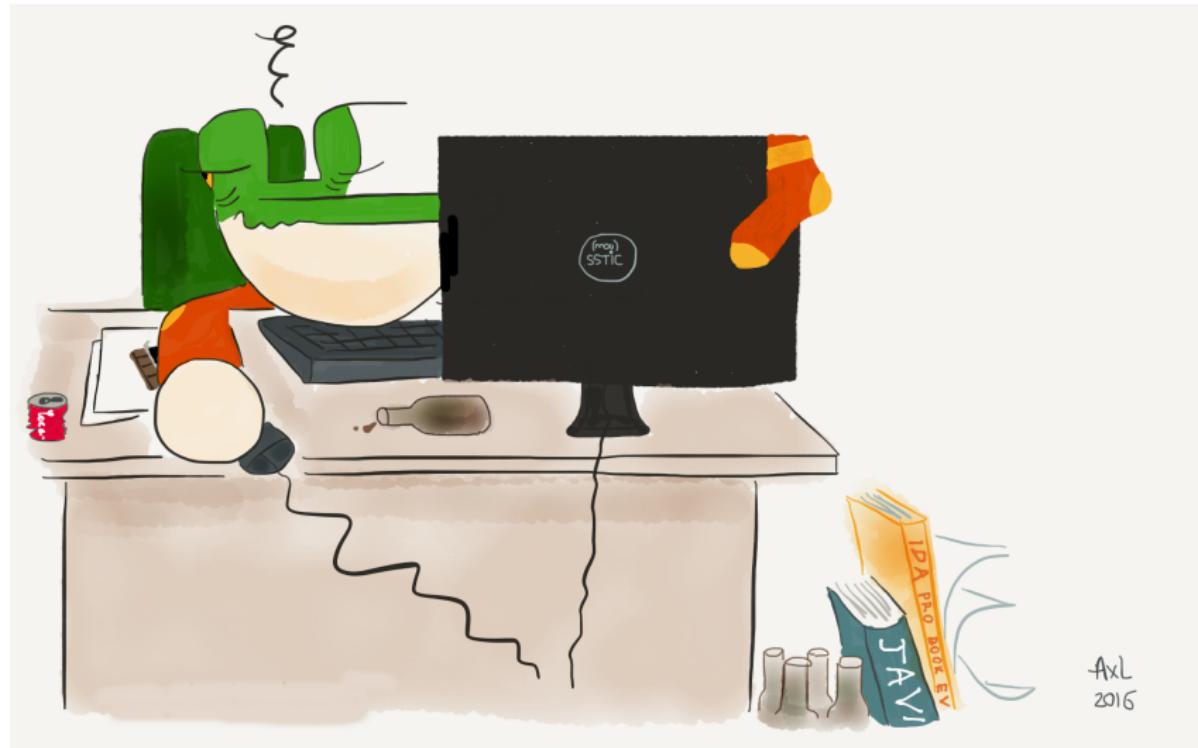
5/5 - Develop a smart app for tests



```
SmartWatchSms.java
115
116     @Override
117     public void onActiveLowPowerModeChange(boolean lowPowerModeOn) {
118         mIsInActiveLowPower = lowPowerModeOn;
119         Log.d(SmartWatchExtensionService.LOG_TAG, "onActiveLowPowerModeChange: lowPower=" +
120               + mIsInActiveLowPower
121               + " powerButton=" + mPowerButtonPressed
122         );
123         sendText(R.id.tv_explanation, "Touch screen");
124         if (mIsInActiveLowPower) {
125             sendText(R.id.tv_title, "Press to leave Active Low Power Mode");
126         } else {
127             sendText(R.id.tv_title, "Press to enter Active Low Power Mode");
128         }
129         mPowerButtonPressed = false;
130     }
131
132     private void setupClickables(Context context) {
133         LayoutInflater inflater = (LayoutInflater) context.getSystemService(
134             Context.LAYOUT_INFLATER_SERVICE);
135         View layout = inflater.inflate(R.layout.sample_main, null);
136         mLayout = parseLayout(layout);
137         if (mLayout != null) {
138             ControlView mode = mLayout.findViewById(R.id.mode);
139             mode.setOnClickListener(new OnClickListener() {
140                 @Override
141                 public void onClick() {
142                     Log.d(SmartWatchExtensionService.LOG_TAG, "User requested to switch to Active Low Power mode");
143                     mPowerButtonPressed = true;
144                     if (!mIsInActiveLowPower) {
145                         Log.d(SmartWatchExtensionService.LOG_TAG, "Requesting to switch to Active Low Power mode");
146                     }
147                 }
148             });
149         }
150     }
151
152     @Override
153     public void onReceive(Context context, Intent intent) {
154         String action = intent.getAction();
155         if (action.equals("com.sony.x1000.intent.action.POWER_BUTTON_PRESSED")) {
156             if (mPowerButtonPressed) {
157                 sendText(R.id.tv_explanation, "Power button pressed");
158             }
159         }
160     }
161
162     private void sendText(int id, String text) {
163         TextView tv = (TextView) findViewById(id);
164         tv.setText(text);
165     }
166
167     private void parseLayout(View layout) {
168         if (layout instanceof ViewGroup) {
169             ViewGroup vg = (ViewGroup) layout;
170             for (int i = 0; i < vg.getChildCount(); i++) {
171                 View v = vg.getChildAt(i);
172                 if (v instanceof ViewGroup) {
173                     parseLayout(v);
174                 }
175             }
176         }
177     }
178 }
```

emacs@alligator:~/Desktop/SmartWatchSms\$ SmartWatchSms.java 62% L127 Git-master (java/l Abbrev) 10:28AM 0.32

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 :(

Hardware teardown

None so far. To be done ;)



Network traffic

No wifi

No Bluetooth.

There is *Bluetooth Low Energy* (\neq Bluetooth)

App development

No possibility to develop an app

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

Is there an easier way to reverse?



→ Yes: reverse engineer the mobile app

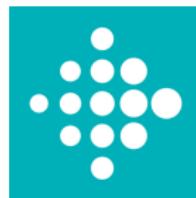
Adapted from <http://picolecroco.free.fr/images/dessins/2013/pico-59-soude.jpg>

Most IoT come with their connected app

IoT



Mobile app



Outline

How would YOU reverse engineer IoT?

A solution for AV analysts & software security researchers

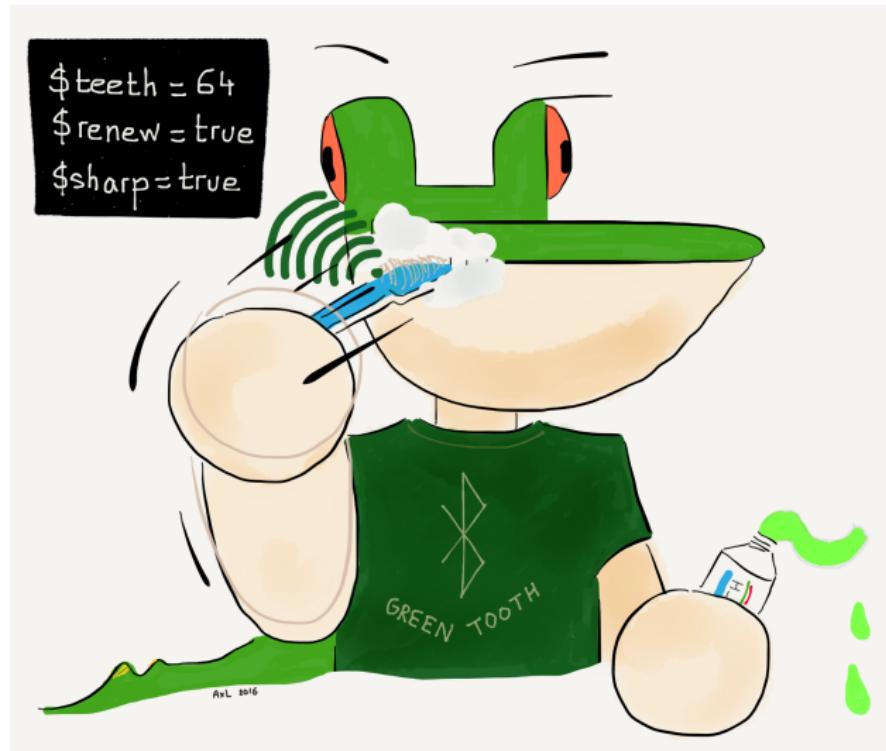
Example 1: Connected toothbrush

Example 2: Sony Smart Watch 2

Example 3: House alarm

Conclusion

Beam toothbrush

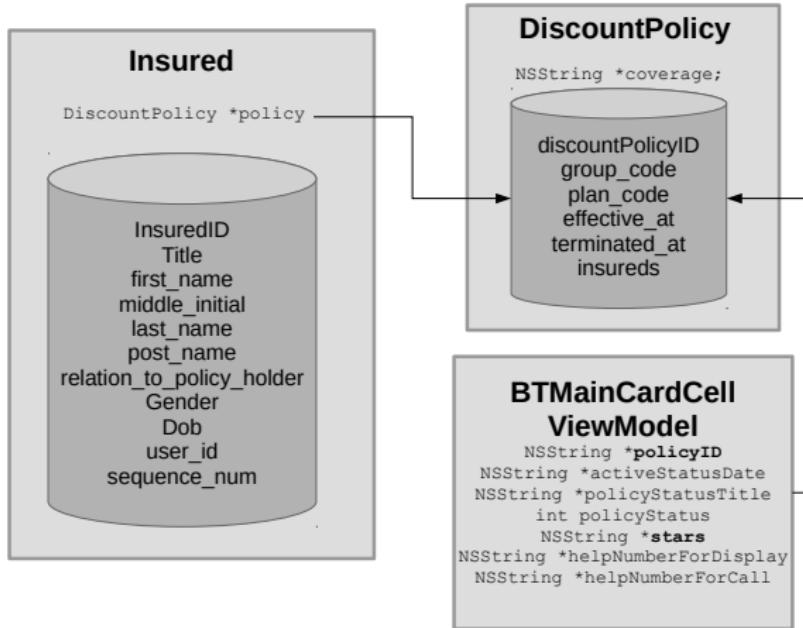


SQL tables - reversing iOS app

Function name	Segment
+[BrushEvent primaryKey]	_text
+[ClientDevice primaryKey]	_text
+[ClientSession primaryKey]	_text
+[ClientSoftware primaryKey]	_text
+[Device primaryKey]	_text
+[DiscountPolicy primaryKey]	_text
+[Insured primaryKey]	_text
+[KeyStore primaryKey]	_text
+[NSManagedObject(Mappings) primaryKey]	_text
+[RollingEvent primaryKey]	_text
+[User primaryKey]	_text
+[UserChallenge primaryKey]	_text
+[UserShare primaryKey]	_text
+[UserSummary primaryKey]	_text
-[BTManagedObject primaryKeyValue]	_text
-[NSRelationshipDescription(BTRelationshipDescriptio...]	_text

- ▶ Tip: search for **primaryKey**
- ▶ Contents of each table:
mappings func

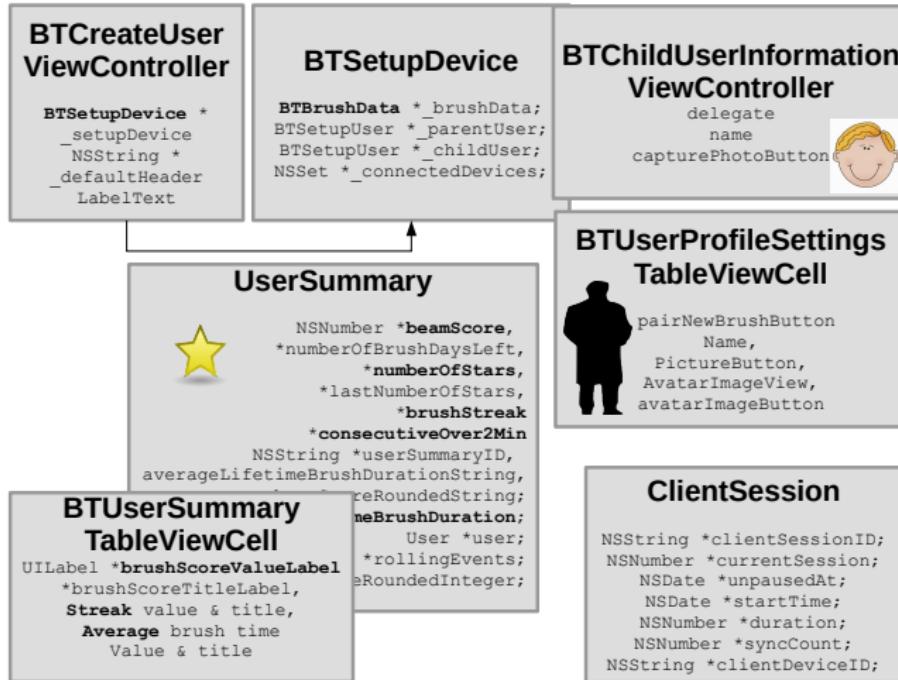
SQL tables: what we work out



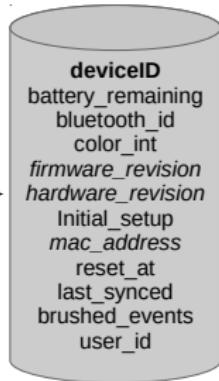
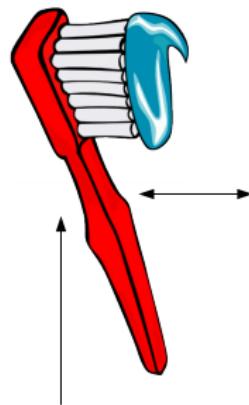
Reconstructing implementation design

```
_OBJC_INSTANCE_METHODS_UserSummary __objc2_meth_list <0xC, 4>
    ; DATA XREF: __objc_const:UserSummary_SclassData↓o
    __objc2_meth <aBeamscorerou_2, a1804, \ ; UserSummary - (int)beamScoreRoundedInteger
        __UserSummary_beamScoreRoundedInteger_+1>
    __objc2_meth <sel_beamScoreRoundedString, a804_0, \ ; UserSummary - (id)beamScoreRoundedString
        __UserSummary_beamScoreRoundedString_+1>
    __objc2_meth <sel_sortedRollingEventsArray, a804_0, \ ; UserSummary - (id)sortedRollingEventsArray
        __UserSummary_sortedRollingEventsArray_+1>
    __objc2_meth <sel_propertiesDictionaryExclusionList, a804_0, \ ; UserSummary - (id)UserSummary_propertiesDictionaryExclusionList
        __UserSummary_propertiesDictionaryExclusionList_+1>
UserSummary_$properties __objc2_prop_list <8, 0xD>
    ; DATA XREF: __objc_const:UserSummary_SclassData↓o
    __objc2_prop <aBeamscore, 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 <aLastnumberoftst, aTNsnumberRDN> ; @property (readonly, retain, @dynamic,
        __objc2_prop <aBrushstreak, aTNsnumberRDN> ; @property (readonly, retain, @dynamic,
        __objc2_prop <aConsecutiveove, aTNsnumberRDN> ; @property (readonly, retain, @dynamic,
        __objc2_prop <aAveragelife_2, aTNsstringRDN> ; @property (readonly, retain, @dynamic,
        __objc2_prop <aAveragelifetim, aTfRDN> ; @property (readonly, @dynamic, nonatomic,
        __objc2_prop <aUser, aTUserRDN> ; @property (readonly, retain, @dynamic, nonatomic,
        __objc2_prop <aRollingevents, aTNssetRDN> ; @property (readonly, retain, @dynamic,
        __objc2_prop <aBeamscoreround, aTiRN> ; @property (readonly, nonatomic) int beamScoreRoundedInteger
        __objc2_prop <aBeamscorerou_3, aTNsstringRDN> ; @property (readonly, nonatomic) NS
```

Classes, methods, fields: what we work out



Classes, methods, fields: what we work out



BTBrushEvent

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

BTBrushData

```
NSUUID *uuid;
Vector3 *accelerometerValues,
NSString *deviceName, *lastAccelerometerValues;
*macAddress, *gyroscopeValues;
*appearance, NSDate
*manufacturer, *lastBrushDetected;
*partialMacAddress, char buttonDown,
*model, *serialNumber IsBrushing,
*firmwareRevision, motorState
*hardwareRevision, double brushingDuration;
notify NSDate
float batteryLevel; *lastTimeFromBrush;
motorIntensity; NSMutableArray
char *brushEvents;
autoOffTimerEnabled, int brushColor,
quadrantTimerEnabled; eventWriteIndex;
char activeConnection; float proximity;
```

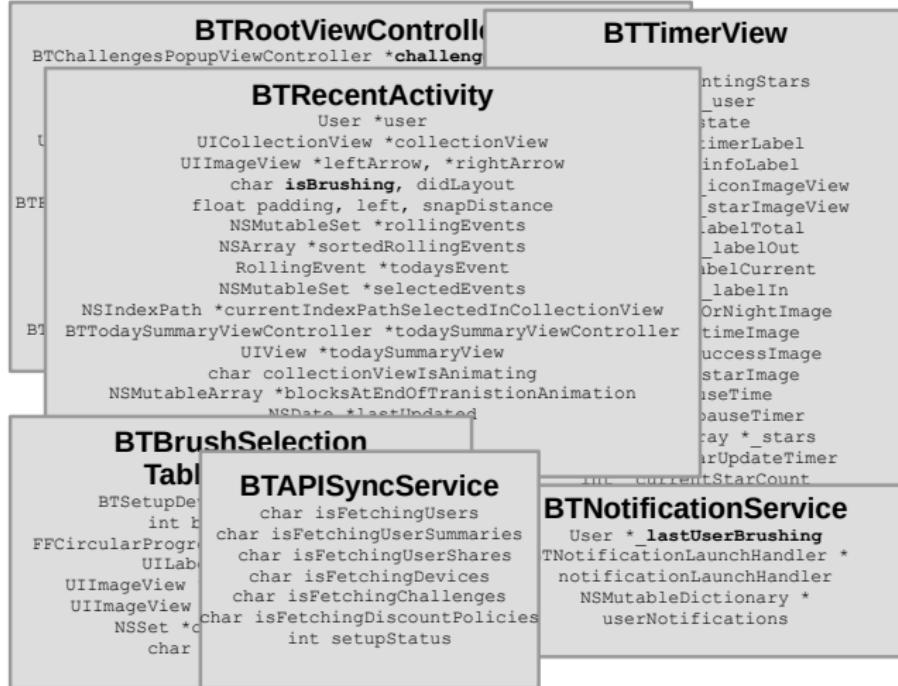
BTFirmwareUpdater

```
char *firmware;
unsigned int totalLength;
unsigned int written;
unsigned int toWrite;
unsigned int loopCount;
int state;
CBService *otaService;
CBCharacteristic
*otaControlPoint, *otaDataPoint;
```

BTFirmwareUpdate ViewController

```
label *labelSV;
TextView *textW;
new in
```

Classes, methods, fields: what we work out



UUID of Bluetooth Low Energy characteristics

```
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: $((1 - \frac{x}{100}) * 139) + 69$
- ▶ Writing to toothbrush: BLE characteristic (833d...) found from RE

Demo: reading toothbrush battery level

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

```
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
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

Outline

How would YOU reverse engineer IoT?

A solution for AV analysts & software security researchers

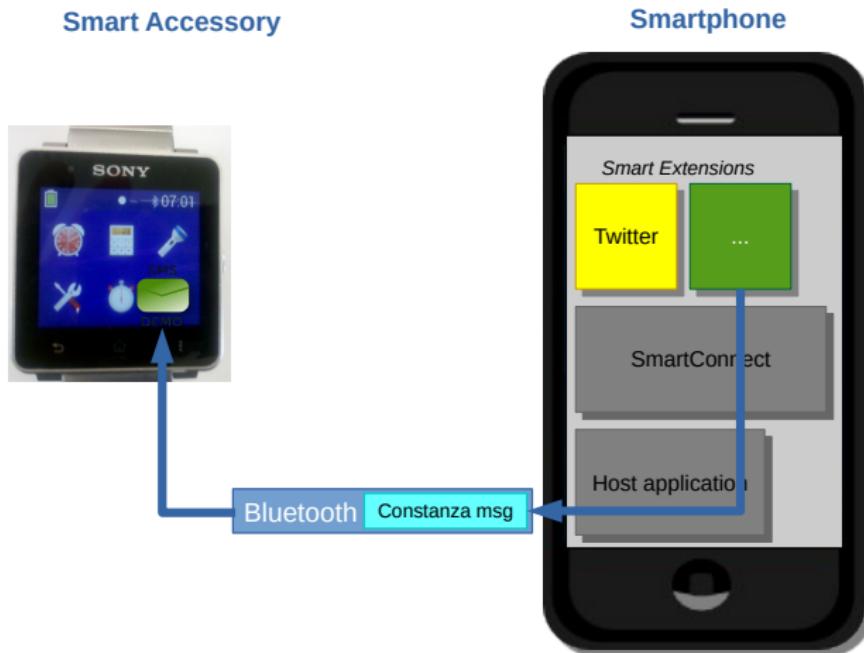
Example 1: Connected toothbrush

Example 2: Sony Smart Watch 2

Example 3: House alarm

Conclusion

Architecture



Reversing host app protocol

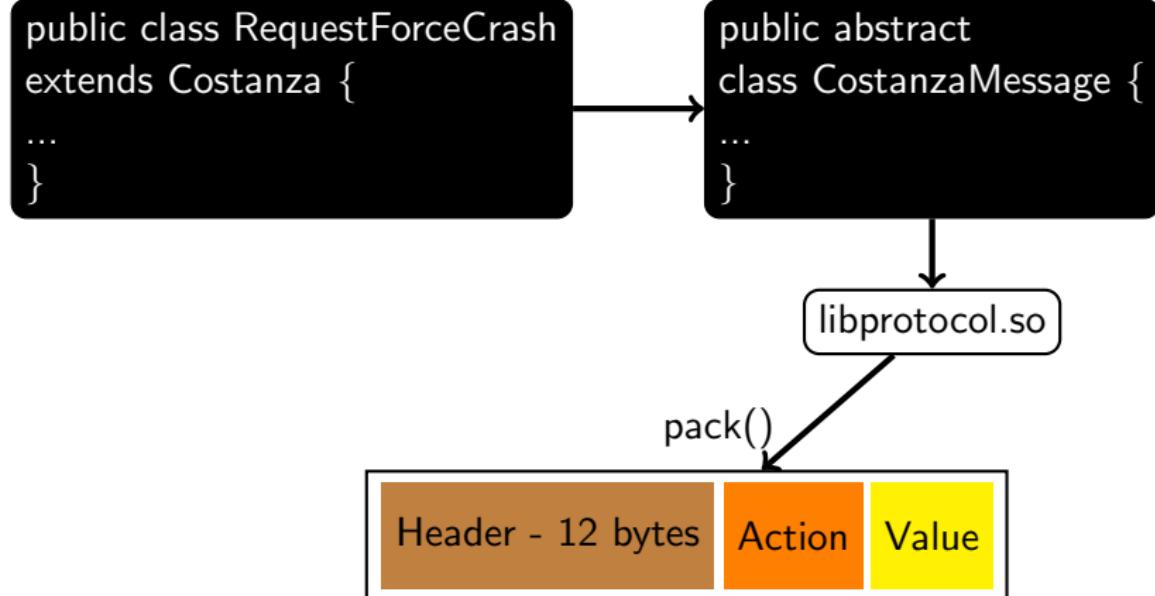
```
public class RequestForceCrash extends CostanzaMessage
    public static final int FORCE_CRASH_REQUEST_MAGIC
        = 0xC057A72A;
    private int mMagic;

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

666 → Number of the Beast

C057A72A → Costanza

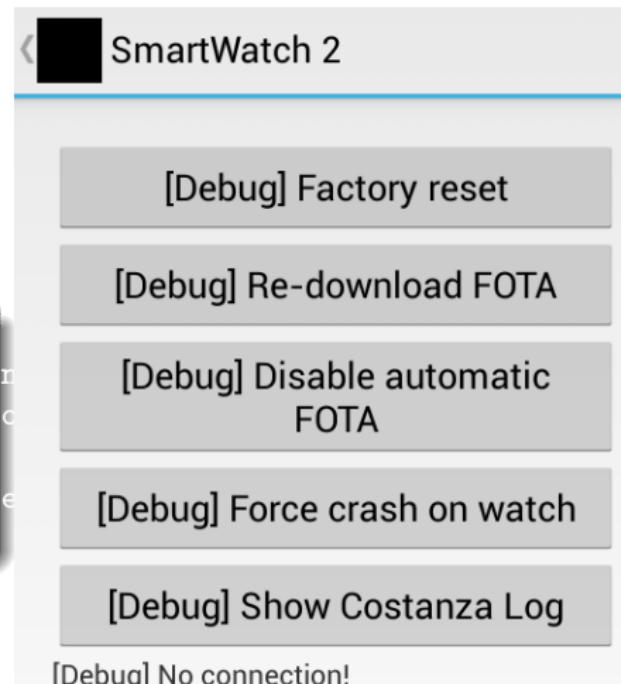
Sending Costanza messages



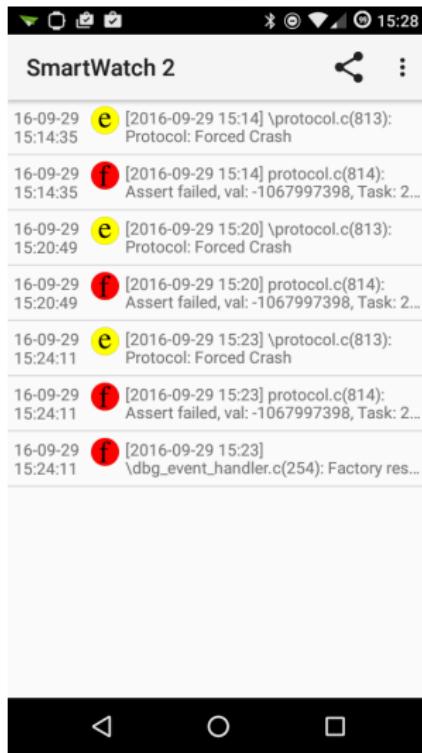
Hidden screen

RequestForceCrash packets are sent by a hidden activity!

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



Debug command work



Debug console

```
$ 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!

```
$ java DecryptParam ../reversing/ [REDACTED]
== Meian parametres.txt decryptor PoC ==
Filename: ../reversing/[REDACTED]
Reading ../reversing/[REDACTED] as bytes:
[-1, [REDACTED], 117, [REDACTED], 72, [REDACTED], 0, [REDACTED], 0, [REDACTED], 4, [REDACTED], 06, [REDACTED], 78, [REDACTED], 66, [REDACTED], 61, 0, 61, 0]
De-obfuscated [REDACTED] algorithm name: [REDACTED]
Decrypting
Phone Number      : 0120304050
Alarm Passcode    : 1234
Auto-control delay: 0
Emergency phone   : 0201030400
```

Weak protection for password: we can recover alarm's phone number, password, delay, emergency phone...

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

Without the app, 1 security issue.

With the app, 2 security issues !!!

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

Thanks for your attention!

Thanks

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](#)