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

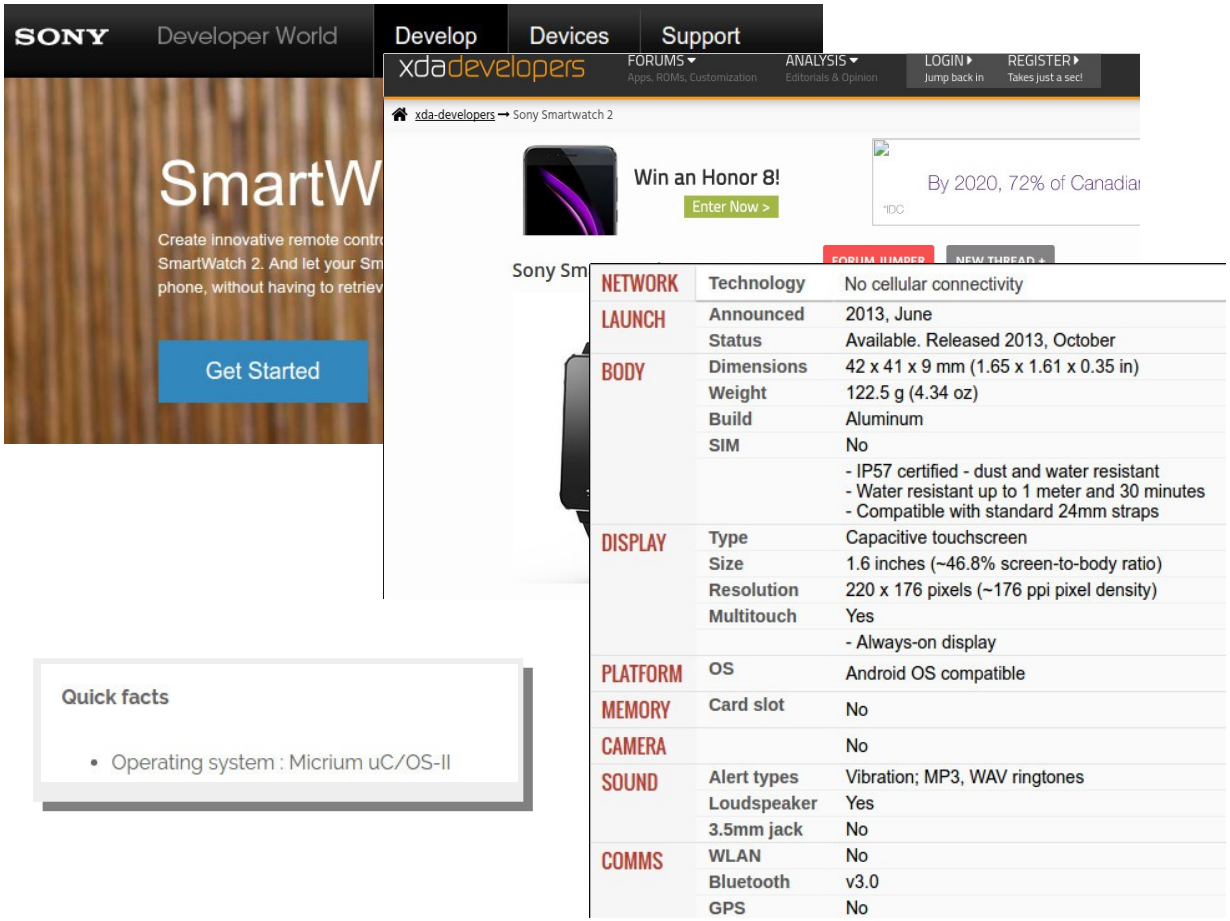
Example 3: House alarm

Conclusion

That's your new task

How are you going to reverse it?

1/5 - Browse the web for documentation



Screenshots of smartwatchforum.com, xda-developers, developer.sony.com

2/5 - Hardware teardown

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

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

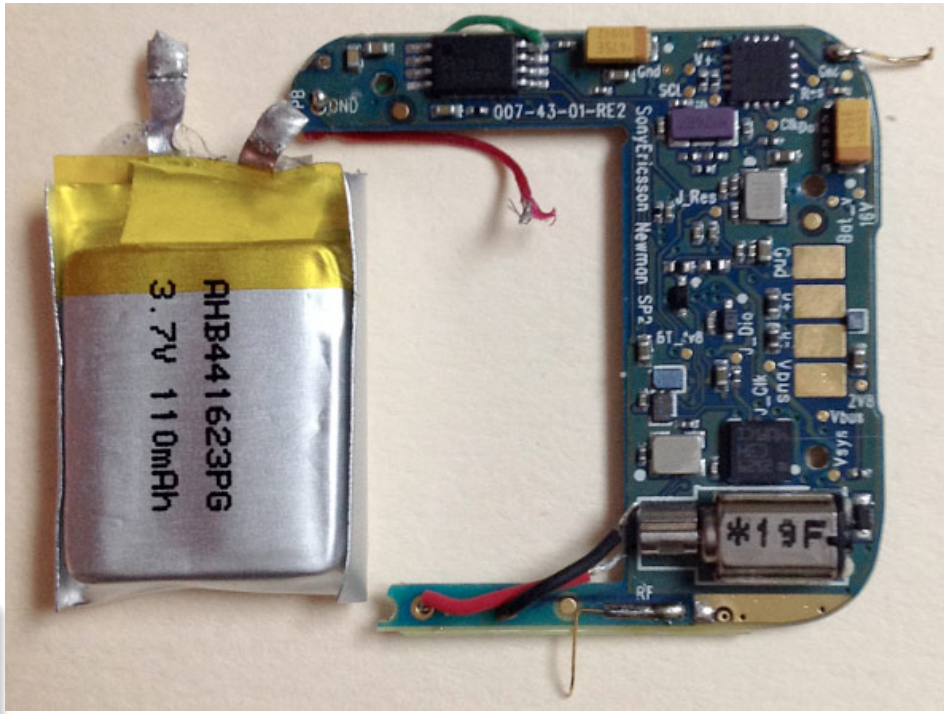
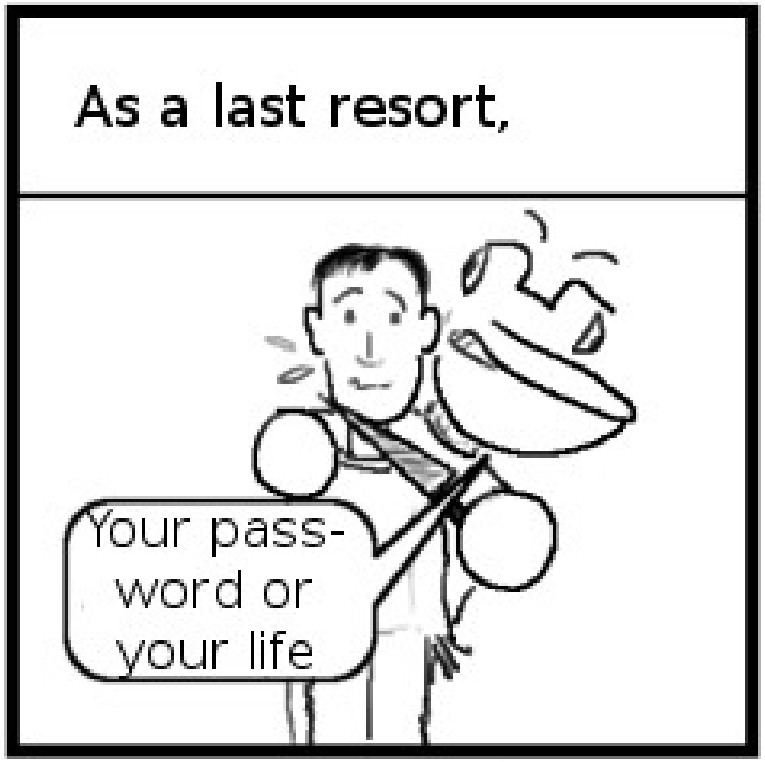


Photo credit: engadget

3/5 - Social engineering

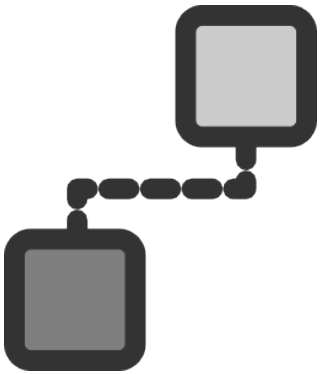
"Kidnap the developer, get access to his/her PC and grab the sources"

LOL ;-)



Adapted from Pico le Croco

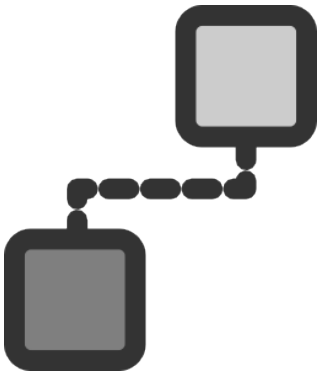
4/5 - Sniff network traffic



In practice for the smart watch

- No Wifi

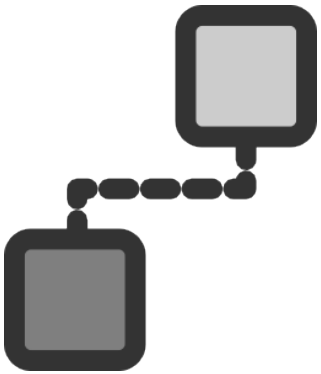
4/5 - Sniff network traffic



In practice for the smart watch

- No Wifi
- Bluetooth traffic!

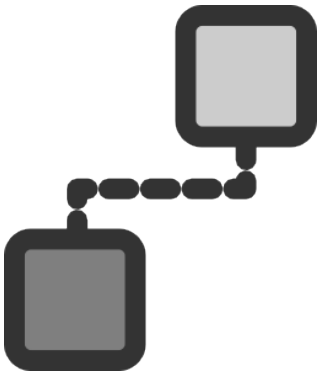
4/5 - Sniff network traffic



In practice for the smart watch

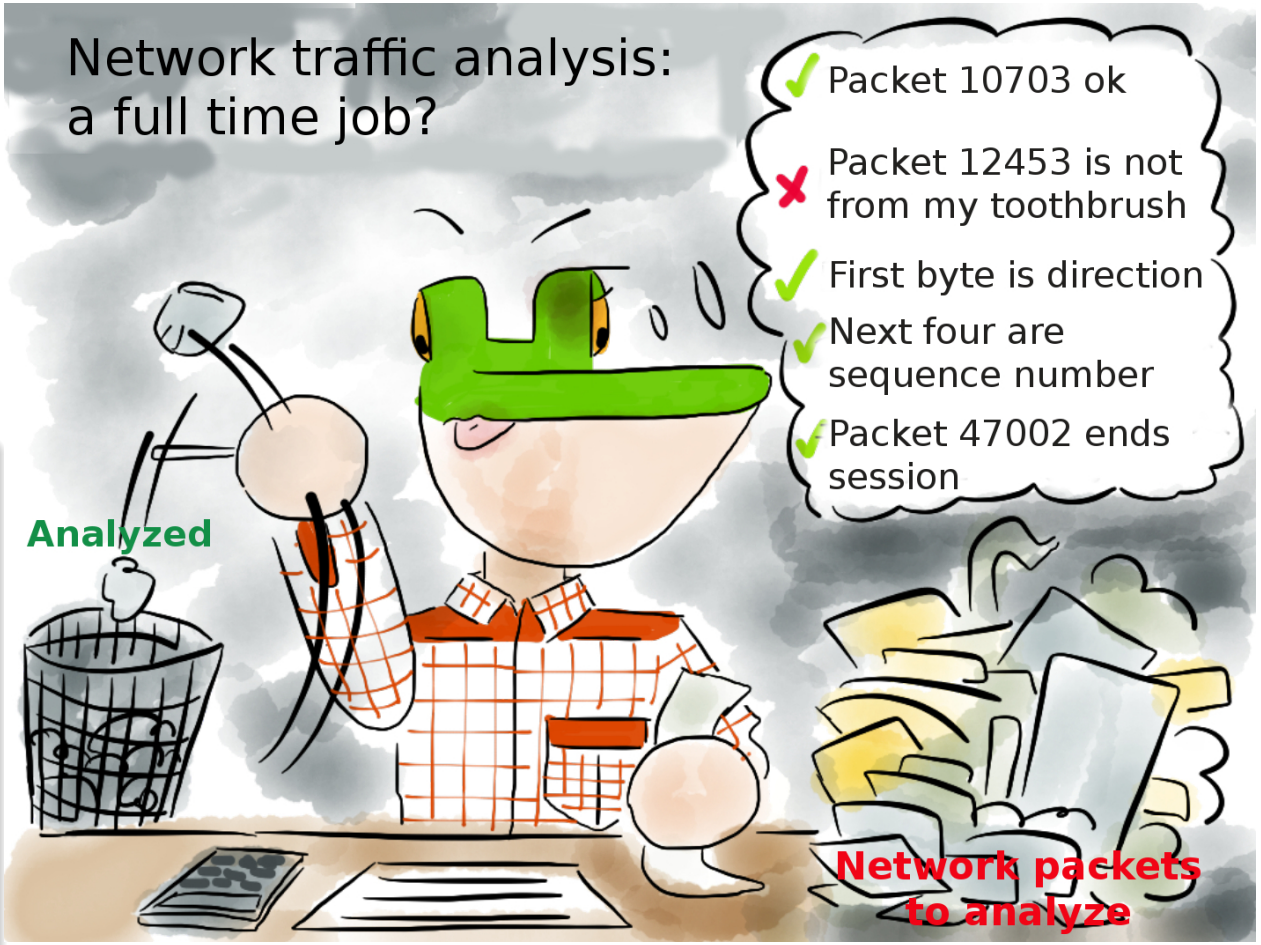
- No Wifi
- Bluetooth traffic!
- ... encrypted! Use Ubertooth?

4/5 - Sniff network traffic



In practice 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 scree
124        if (mIsInActiveLowPower) {
125            sendText(R.id.tv_title, "Press to leave Ac
126        } else {
127            sendText(R.id.tv_title, "Press to enter Ac
128        }
129        mPowerButtonPressed = false;
130    }
131
132    private void setupClickables(Context context)
133        LayoutInflater inflater = (LayoutInflater) co
134                (Context.LAYOUT_INFLATER_SERVICE)
135        View layout = inflater.inflate(R.layout.samp
136        mLayout = parseLayout(layout);
137        if (mLayout != null) {
138            ControlView mode = mLayout.findViewB
139            mode.setOnClickListener(new OnClickLis
140                @Override
141                public void onClick() {
142                    Log.d(SmartWatchExtensionService
143                    mPowerButtonPressed = true;
144                     if (!mIsInActiveLowPower) {
145                         Log.d(SmartWatchExtensionService.LOG_TAG, "Requesting to switch to Active Low Power mo
    de");
```

emacs@alligator

-:--- **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.

Different architecture

Different hardware

Different protocols

You'll be starting from scratch again!

Outline

Is there an easier way to reverse?

Yes: reverse engineer the mobile app

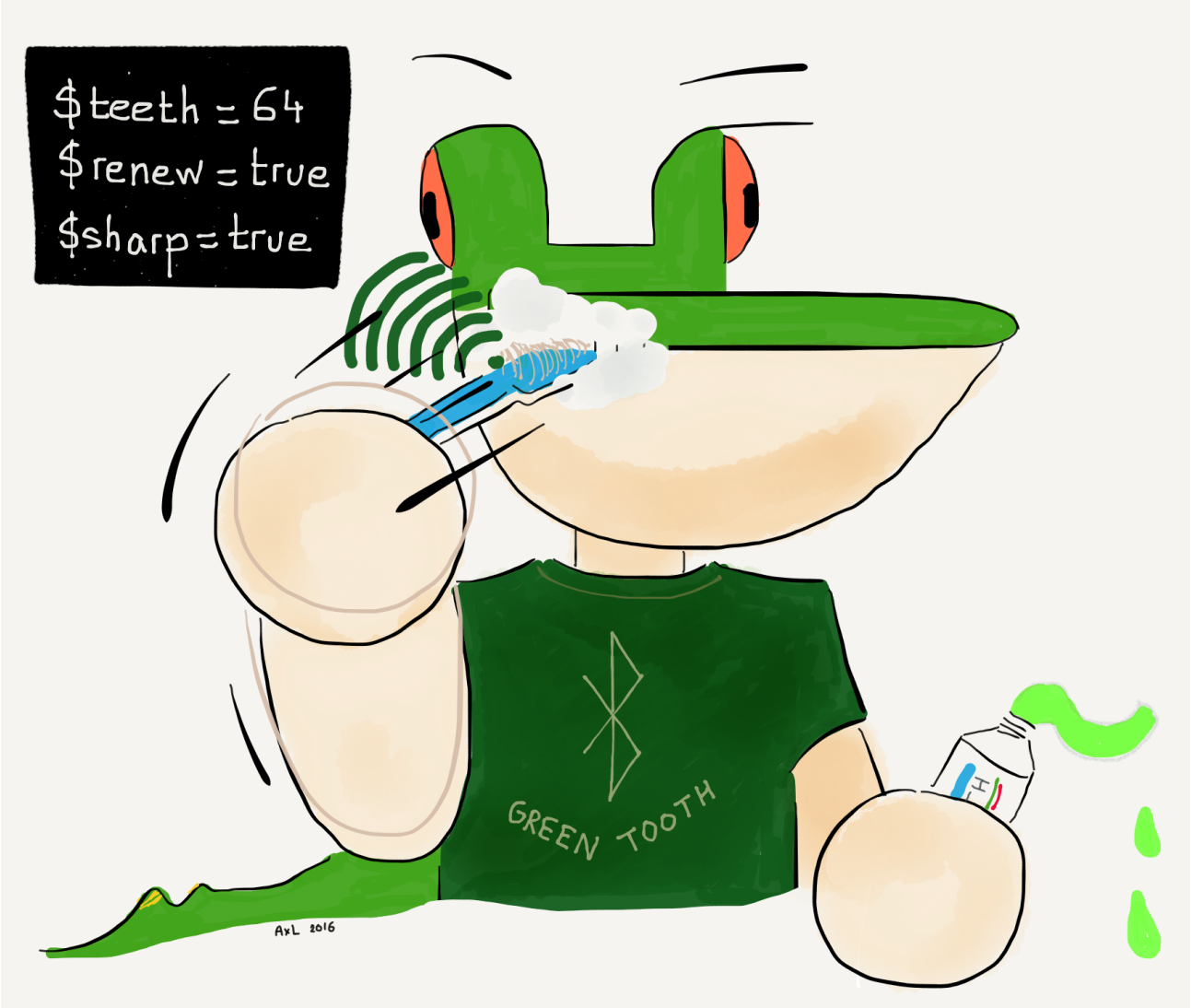
Adapted from http://picolecroco.free.fr/images/dessins/2013/pico-59-soude.jpg

Most IoT come with their connected app

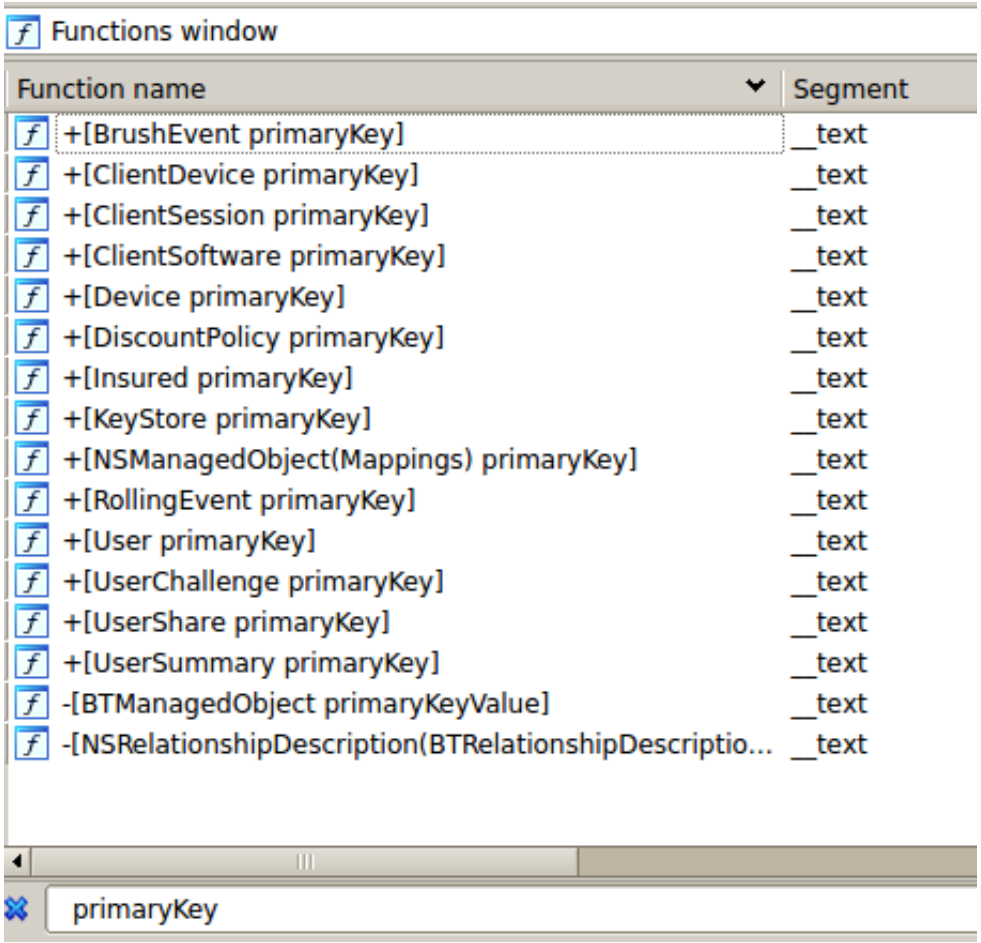
- Sony SmartWatch 2 has a mobile application (to install new extensions)
- Beam Toothbrush has a mobile application to track your brushing experience
- Fitbit Flex has a mobile application to see how fit you are
- Wilson X basketball has a mobile application to see how well you score
- etc

Outline

Beam toothbrush

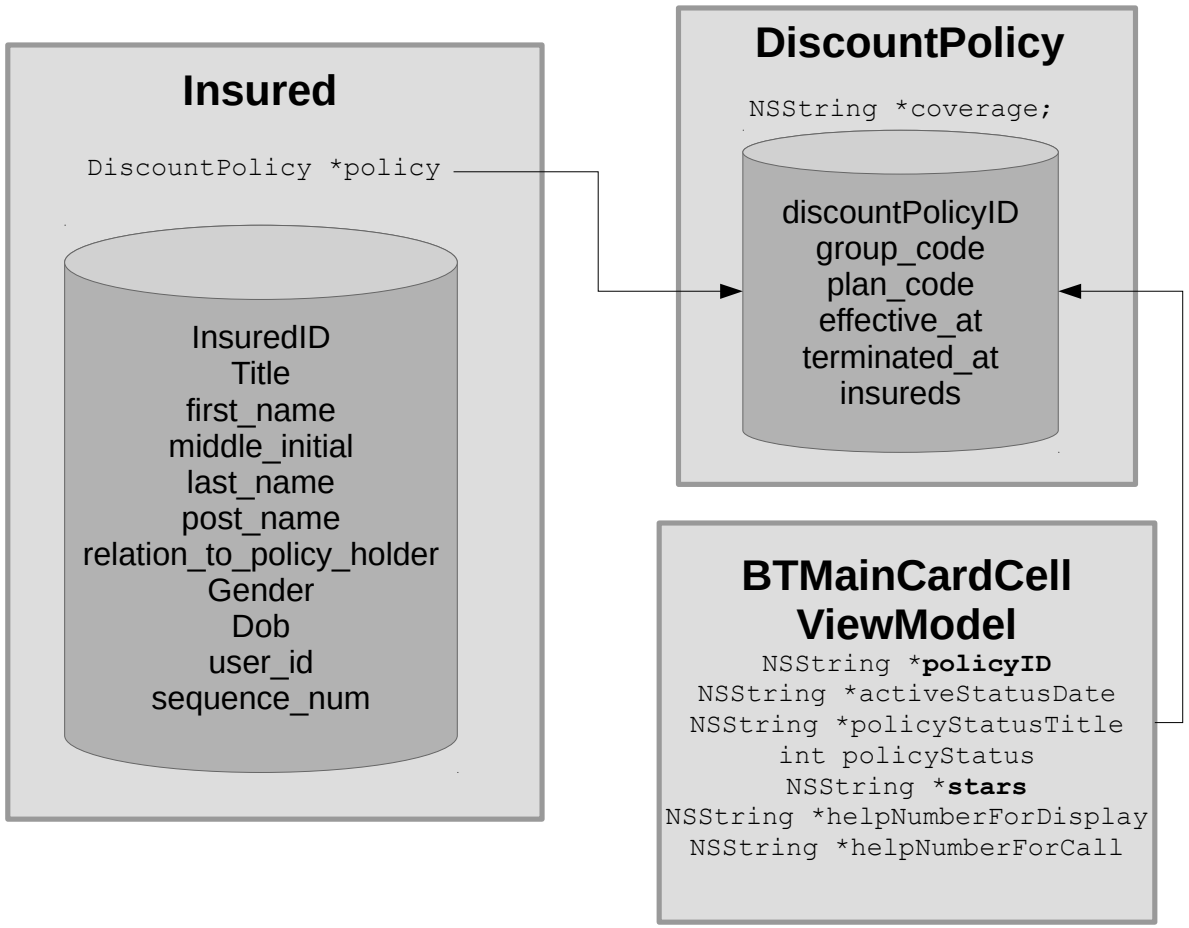


SQL tables - reversing iOS app



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

SQL tables: what we work out



Reconstructing implementation design

Classes, methods, fields: what we work out

BTCreateUser ViewController

BTSetupDevice *
_setupDevice
NSString *
_defaultHeader
LabelText

BTSetupDevice

BTBrushData *_brushData;
BTSetupUser *_parentUser;
BTSetupUser *_childUser;
NSSet *_connectedDevices;

BTChildUserInformation ViewController

delegate
name
capturePhotoButton

BTUserProfileSettings TableViewCell

pairNewBrushButton
Name,
PictureButton,
AvatarImageView,
avatarImageButton

UserSummary

NSNumber ***beamScore**,
*numberOfBrushDaysLeft,
***numberOfStars**,
*lastNumberOfStars,
***brushStreak**
***consecutiveOver2Min**
NSString *userSummaryID,
averageLifetimeBrushDurationString,
reRoundedString;
meBrushDuration;
User *user;
*rollingEvents;
eRoundedInteger;

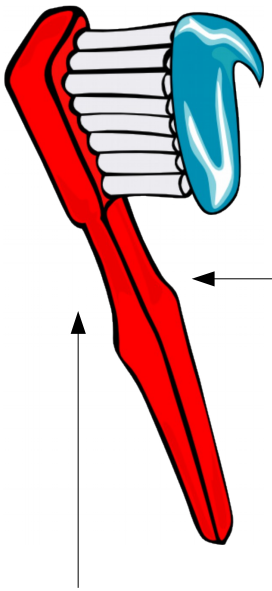
BTUserSummary TableViewCell

UILabel ***brushScoreValueLabel**
*brushScoreTitleLabel,
Streak value & title,
Average brush time
Value & title

ClientSession

NSString *clientSessionID;
NSNumber *currentSession;
NSDate *unpausedAt;
NSDate *startTime;
NSNumber *duration;
NSNumber *syncCount;
NSString *clientDeviceID;

Classes, methods, fields: what we work out



deviceID
battery_remaining
bluetooth_id
color_int
firmware_revision
hardware_revision
Initial_setup
mac_address
reset_at
last_synced
brushed_events
user_id

brushEventID
custom_data
end_time
start_time
device_id

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;
```

```
Vector3 *
accelerometerValues,
*lastAccelerometerValues;
*gyroscopeValues;
NSDate
*lastBrushDetected;
char buttonDown,
IsBrushing,
motorState
double brushingDuration;
NSDate
*lastTimeFromBrush;
NSMutableArray
*brushEvents;
int brushColor,
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

Remotely controlling the toothbrush

Uses Bluetooth Low Energy

Characteristics ≈ entries to read and/or write

How? We get the UUID to access them in the code!

```
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: remote control of 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?!

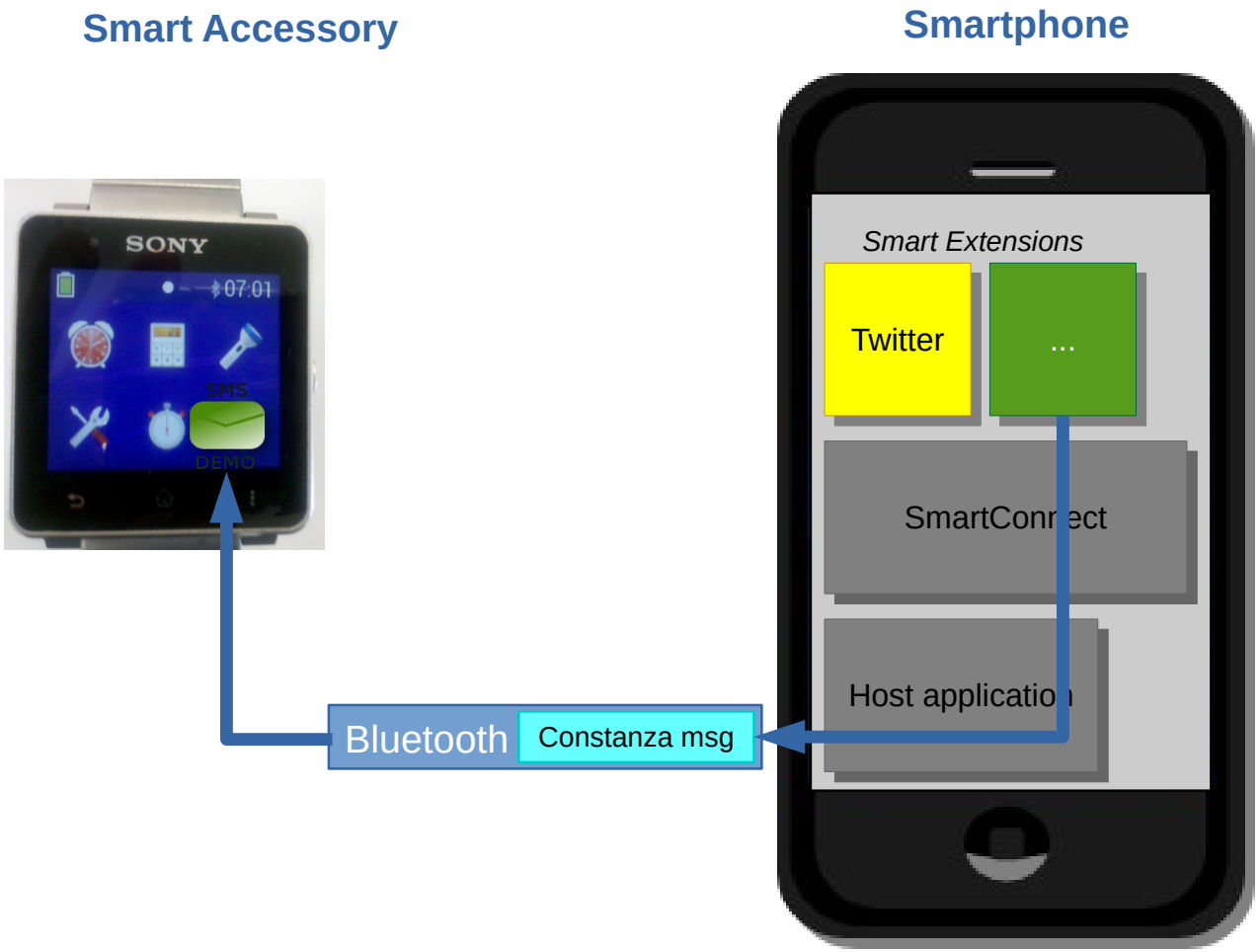
Two scenarios:

1. **DoS or Ransomware**. *"Your pocket money or I tell your mom you don't brush your teeth"*
2. **Propagating virus**. Infected BLE discovery responses? Infected firmware?

Even harmless IoT need to be secured
With *Mirai* IoT botnet, attackers did not care about CCTV cameras!

Outline

Architecture



Reversing host app protocol

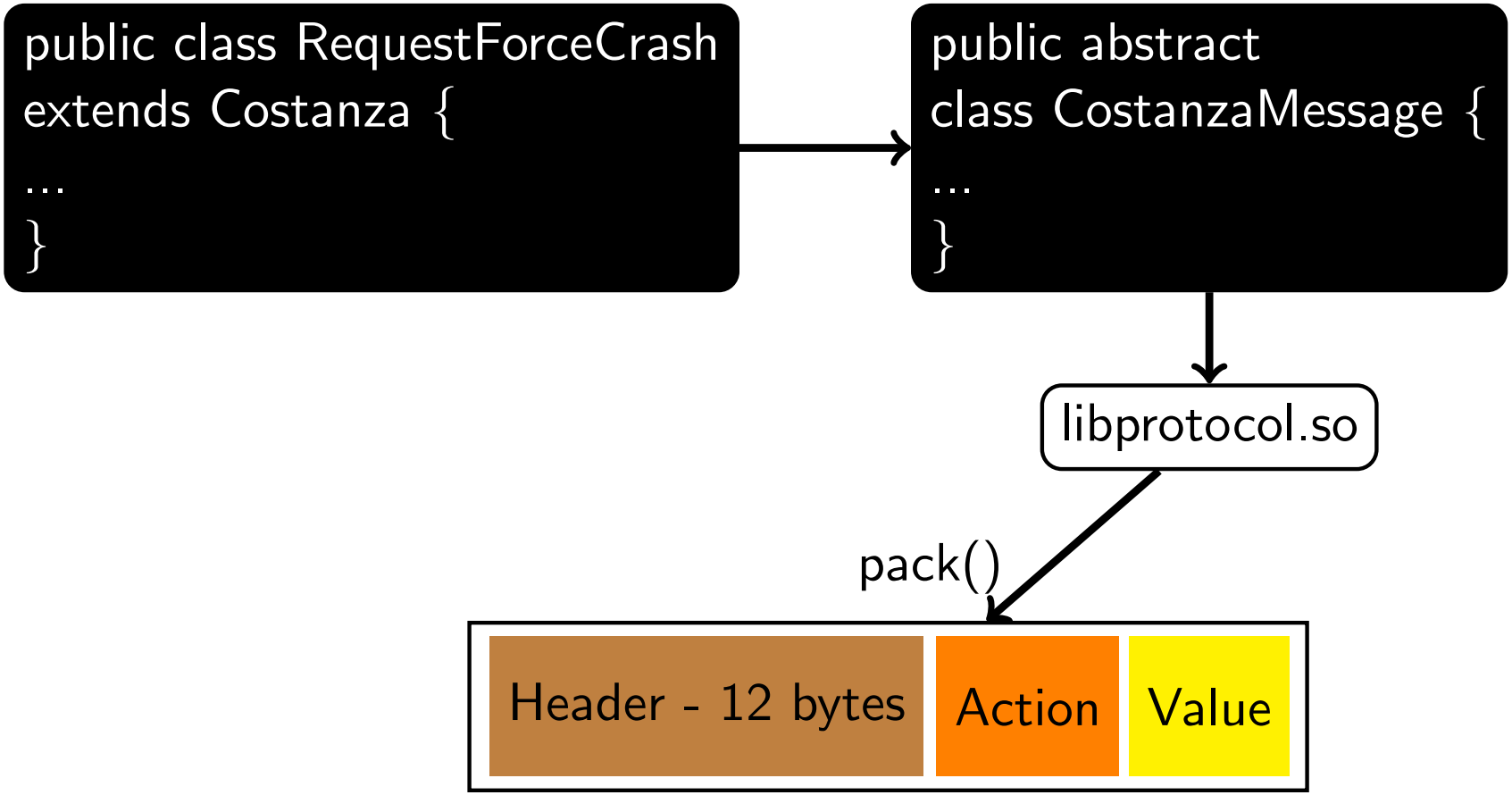
```
public class RequestForceCrash extends CostanzaMessage
    public static final int FORCE_CRASH_REQUEST_MAGIC
        = 0xC057A72A;
    private int mMagic;

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

666 → Number of the Beast

C057A72A → Costanza

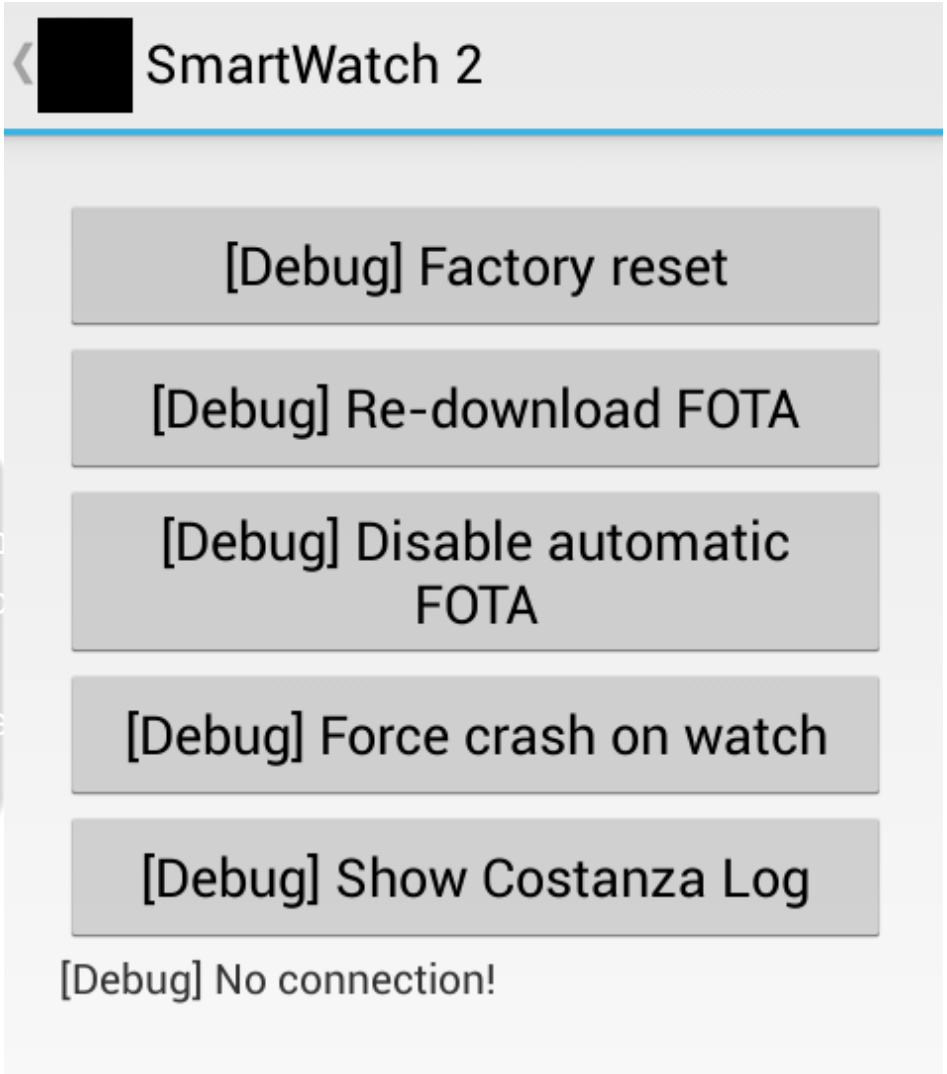
Sending Costanza messages



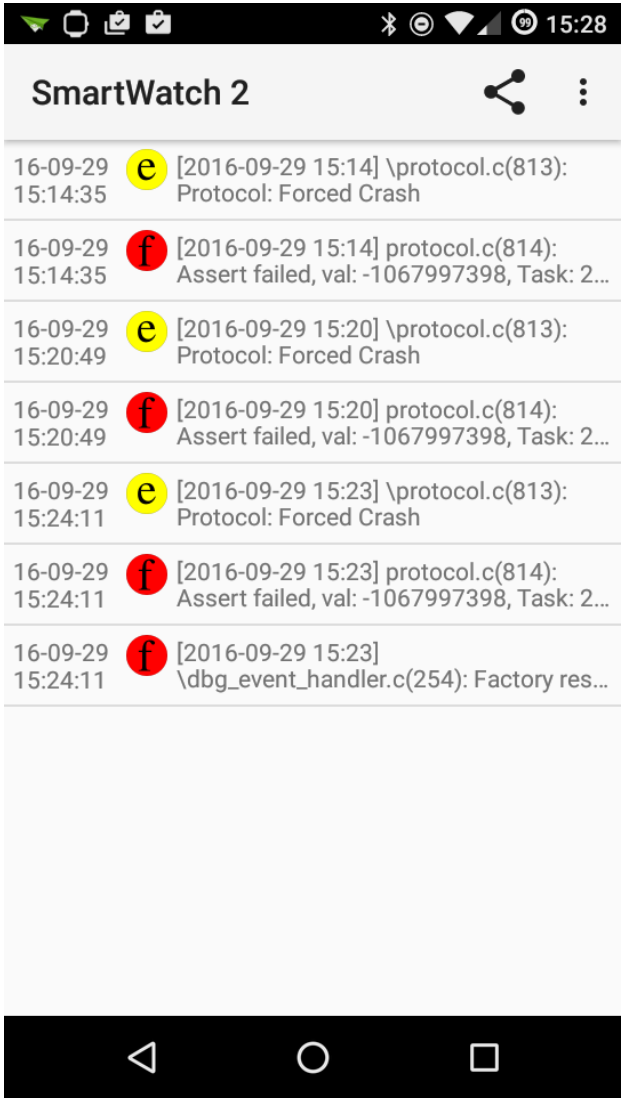
Hidden screen

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

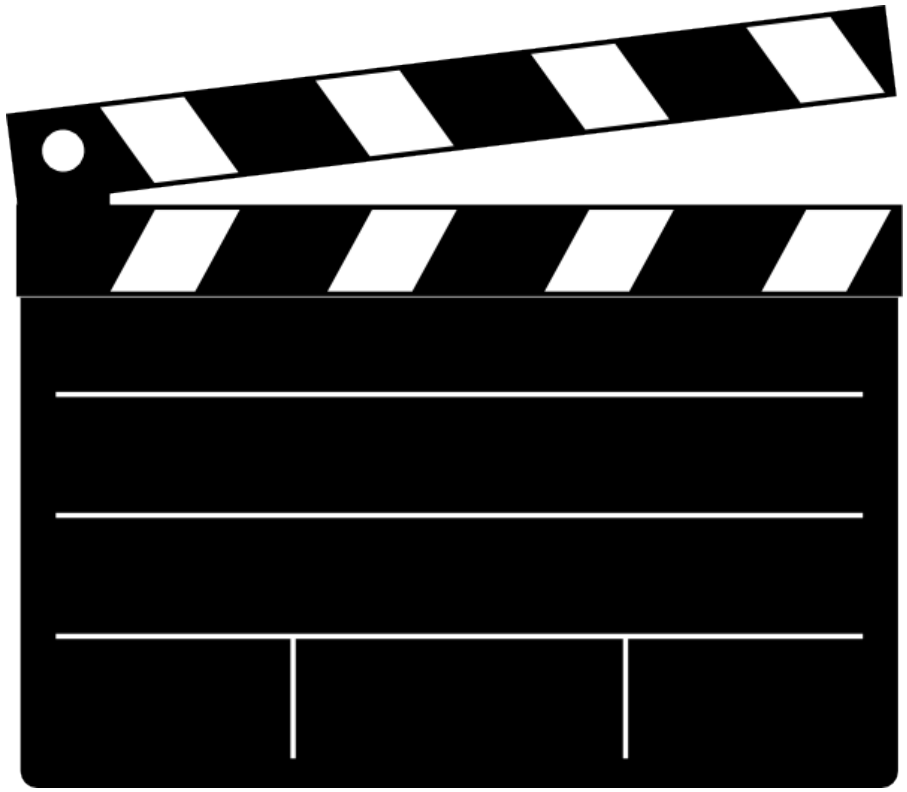


Debug console

```
$ 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:
```

Video of hidden functionalities



Outline

There's an Android app for the alarm

- Protect your house against burglars
- Controllable by SMS

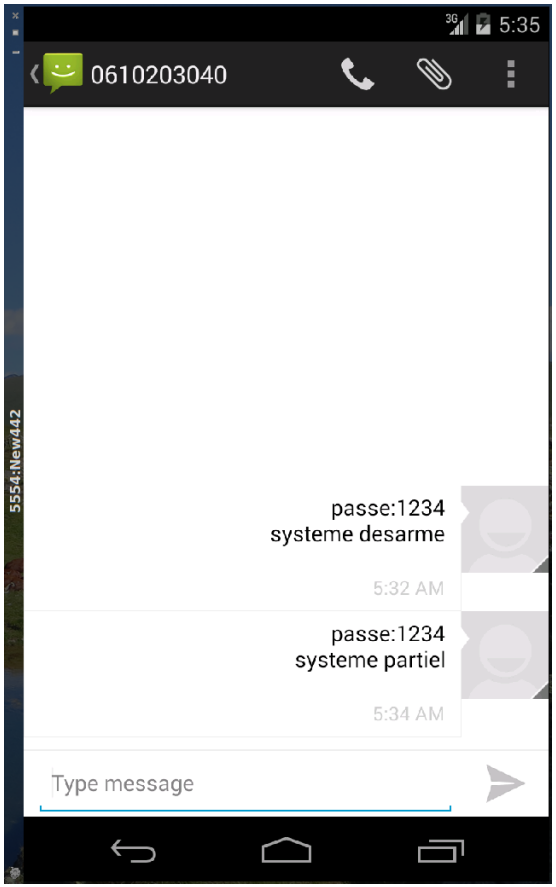
But it's not very user friendly...

Comply to a strict SMS formatting

So, they created an **Android app** to assist end-users

Outbox is not secure

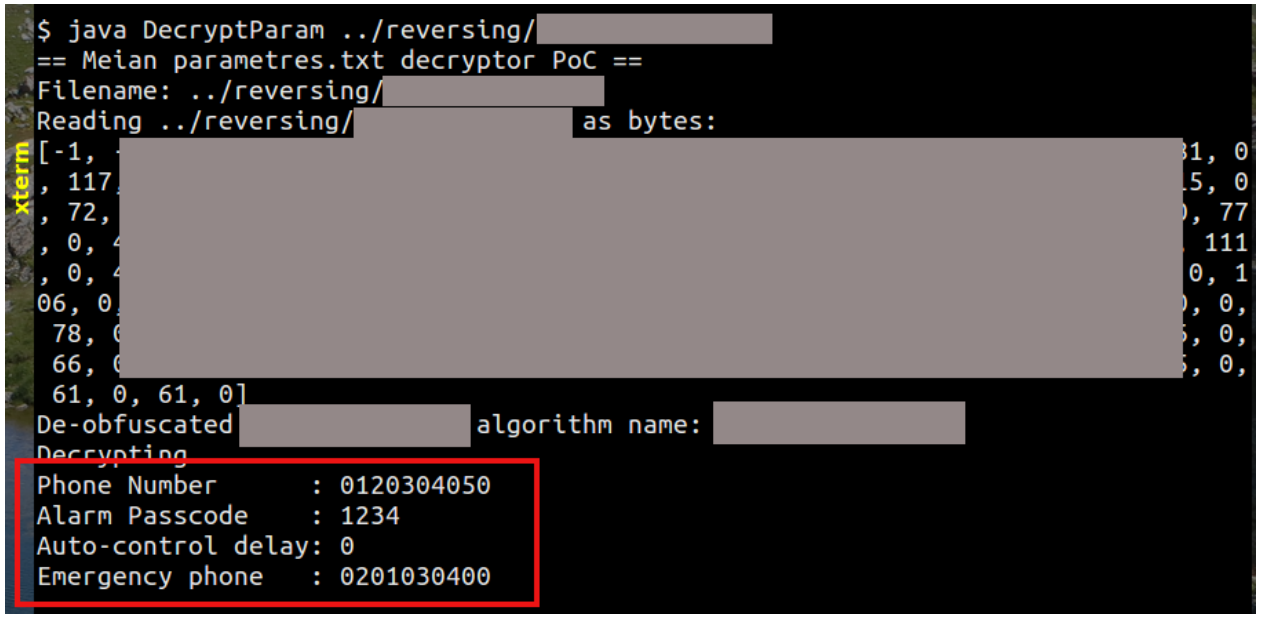
In the **outbox**, the SMS contains the **password** and **phone number** of the alarm.
You get it? You control the alarm!



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**!



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

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

Contact

@cryptax - aapvrille@fortinet.com

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