Scorched Earth

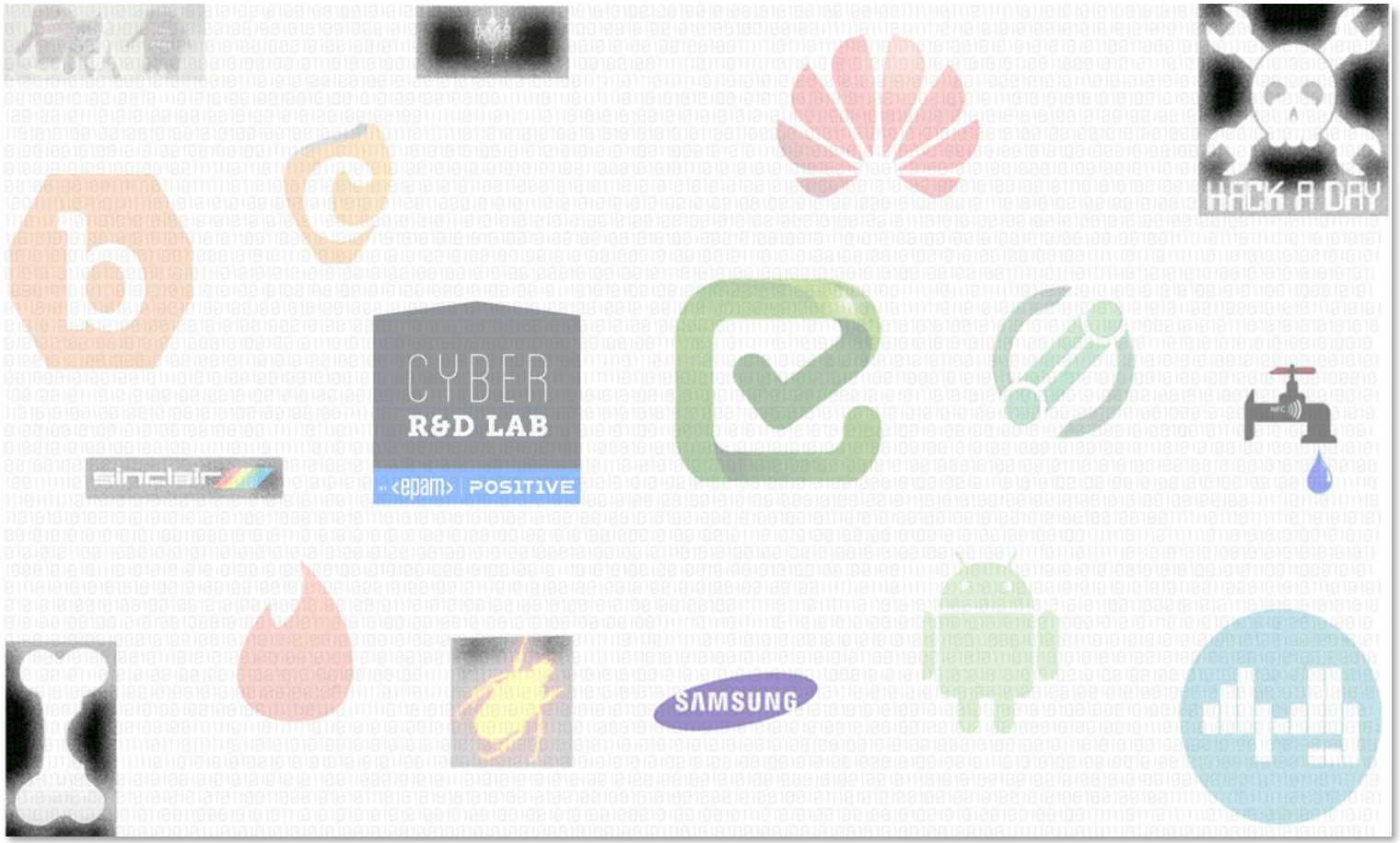
IoT Security Landscape

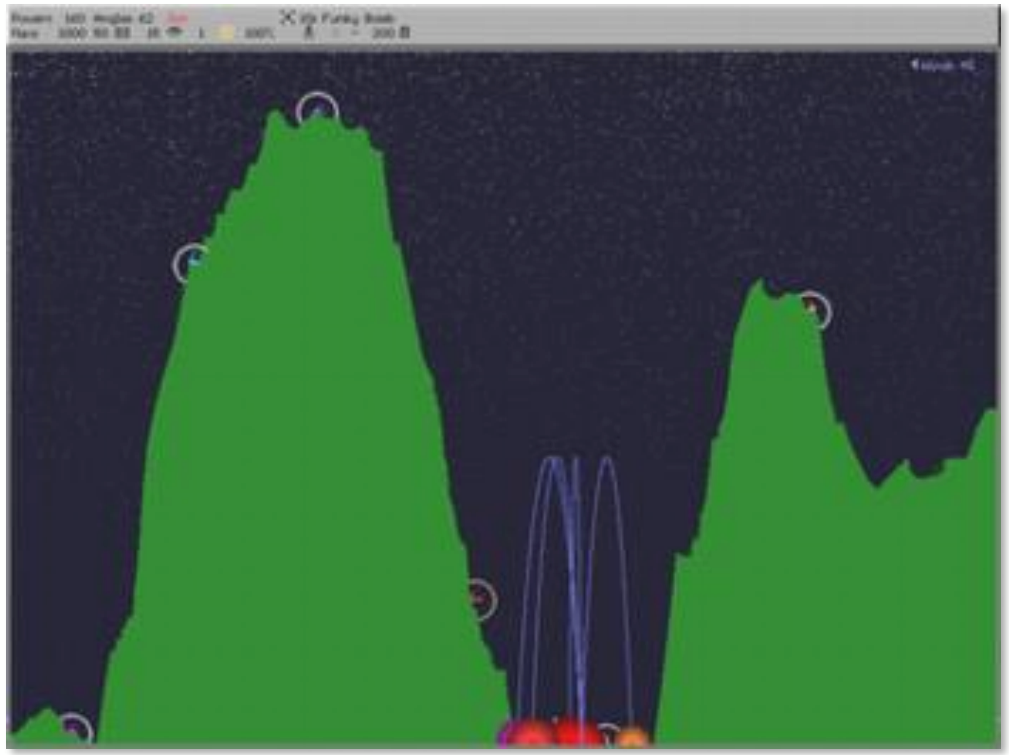
Pedro Umbelino
Senior Security Researcher / Char49

Pedro Umbelino

- Senior Security Researcher

- Email –pedro@char49.com
- IRC – **kripthor** irc.overthewire.org
- Twitter - **@kripthor**





- **IoT Security Landscape**
  - **What changed and what not…**
  - Common mistakes
  - Classic mistakes
  - Recurrent mistakes
  - Strange mistakes
  - Stupid mistakes
  - All of the above, not necessarily in that order

Motivation

- The Internet has descended into every single device
- Recap some of the recurring mistakes that IoT vendors make
- Look at some real world examples throughout the years
- Share the information in hopes of keeping the pressure on vendors to take security seriously

- **Agenda**
- **Example driven presentation**
  - A Smart Doorbell
  - A Smart Vacuum Cleaner
  - A Camera Security Solution
- **Final Words**

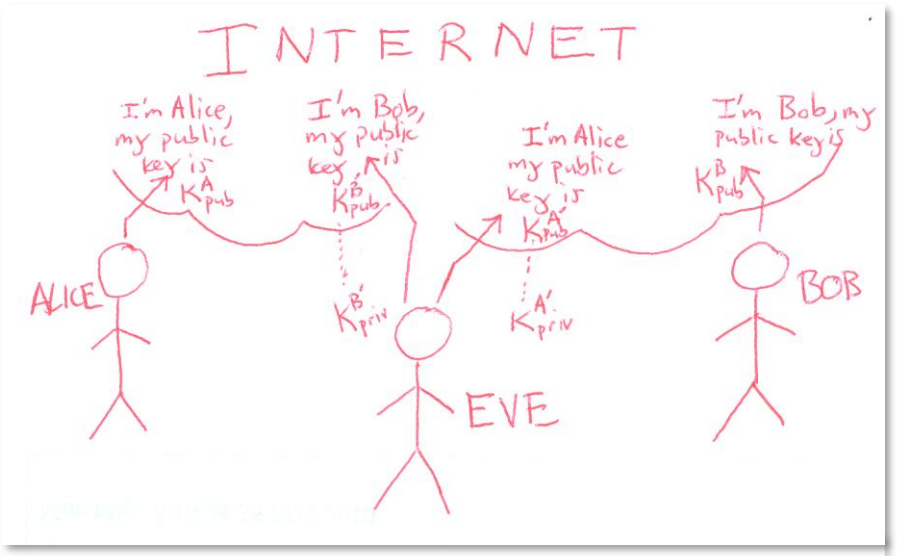
<Example 1>

A Smart Video Doorbell

Doorbell features

- Wi-Fi connectivity, Alexa and Google Assistant integration
- Real Time video (in App)
- Secure Local Storage (military-grade encryption *cough*)
- Smart Detection Zones
- AI technology to detect body shapes and faces

- **Communications security (common mistake)**



- <u>The Doorbell fails to properly validate the HTTPS certificate chain</u>, the job of analyzing HTTPS requests and server responses is trivial since it accepts any certificate.

- False sense of security, adversaries capable of observing traffic can usually intercept and modify it.

○ Server-side Security (classic mistake)

- The Doorbell regularly uploads photos to the Amazon servers every X minutes. (Why? Dunno… GDPR anyone?)
- When the smartphone app wants access to latest snapshot, it requests the signed URL:

https://security-app.XXXXXXX.com/v1/s/file_url?key={"MYDOORBELLSERIAL":"/special/1970/01/01/station/MYDOORBELLSERIAL/title_pic.jpg"}

- The user validation stored in cookies is made against the serial highlighted yellow, but the server retrieves another serial number

- **A path traversal in the green serial number makes it possible for an attacker to access all camera snapshots given a serial number.**
- How hard is to find valid serial numbers? Well… they are sequential so... ¯_(ツ)_/¯

- **Firmware security(recurring mistake)**

- In hardware, the firmware update procedure is always a target to gain access to the device.
- This device firmware is a set of files fetched from the server.
- There is no code signing. Since there is also no certificate validation, implanting a persistent backdoor is easy.

```
[kripthor@zorba orig]$ ls -sh1

3,8M zImage
 32M doorbell_app.bin
 17M rootfs.squashfs
312K m5s_bootimage_sf.bin
284K u-boot.bin
 24K m5s.dtb

[kripthor@zorba orig]$ nc 10.42.0.13 12345

/mnt/flash # id

uid=0(root) gid=0(root)
```

Storage security (stupid mistake)

- If the device is compromised or if a device is going to be easily reachable by a potential attacker, the information stored should not be easily retrievable.
- The devices uses AES 128 for local storage. It generates a key, encrypts it using the server public key and sends it back to the server. No keys are stored device side.
- The files are all stored as:

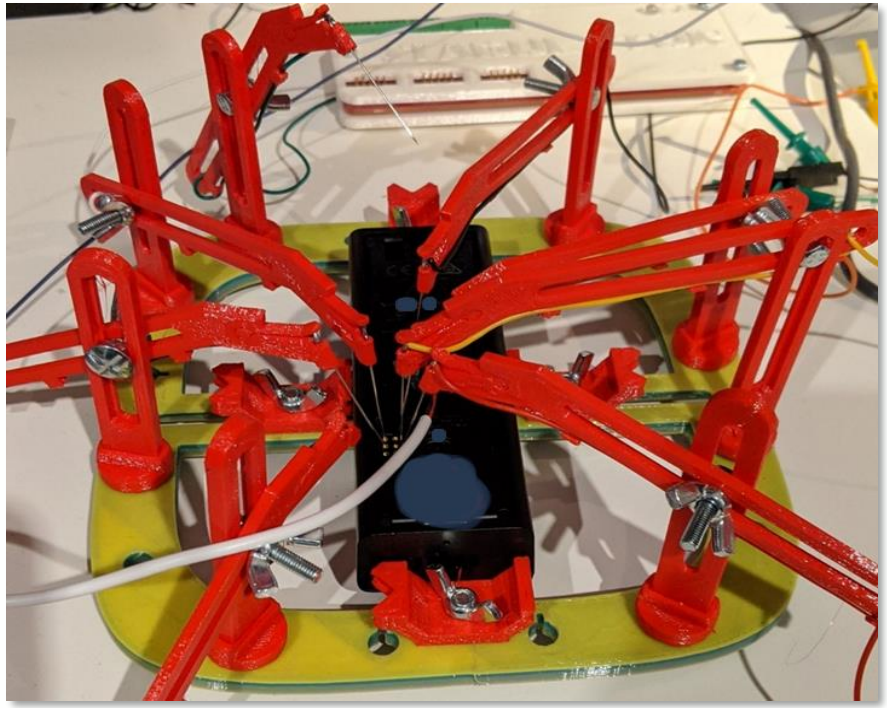
  /mnt/userdata/video/h264_video_<date>_<time>.data

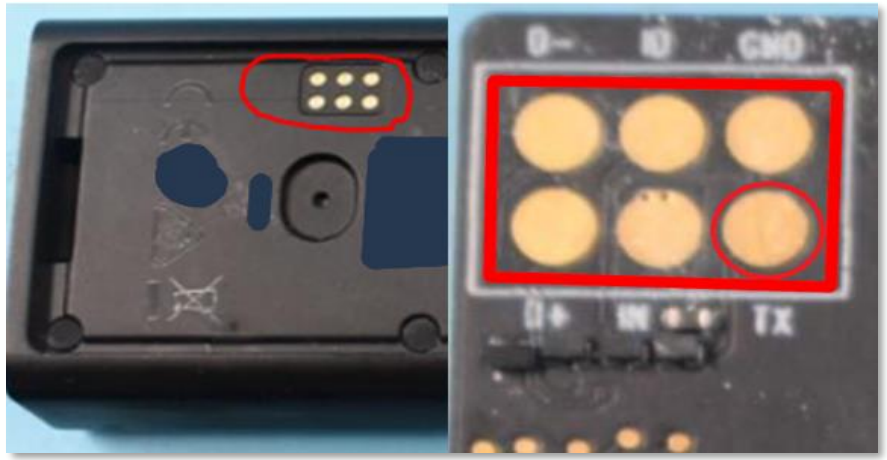
The key generation algorithm is deeply broken.

```
unsigned __int8 * rand_str(unsigned __int8 *str, const int len) {

 const char ascii_str[] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz<!#?$>";

 int i;

 srand(time(0));

 for ( i = 0; i < len; ++i ) str[i] = ascii_str[rand() % 68u];
 str[len] = 0;
 return str;
}
```

- **Physical security (common mistake)**

- Exposed pins on the back of the doorbell are part of a USB-OTG pinout
- It is possible to stop autoboot and enter into fastboot mode
- With a computer and a special cable, it takes roughly one minute to flash a backdoor onto this doorbell, given physical access.
- The attacker does not have to open the device!





<Example 2>

A Smart Vaccum

Vacuum features

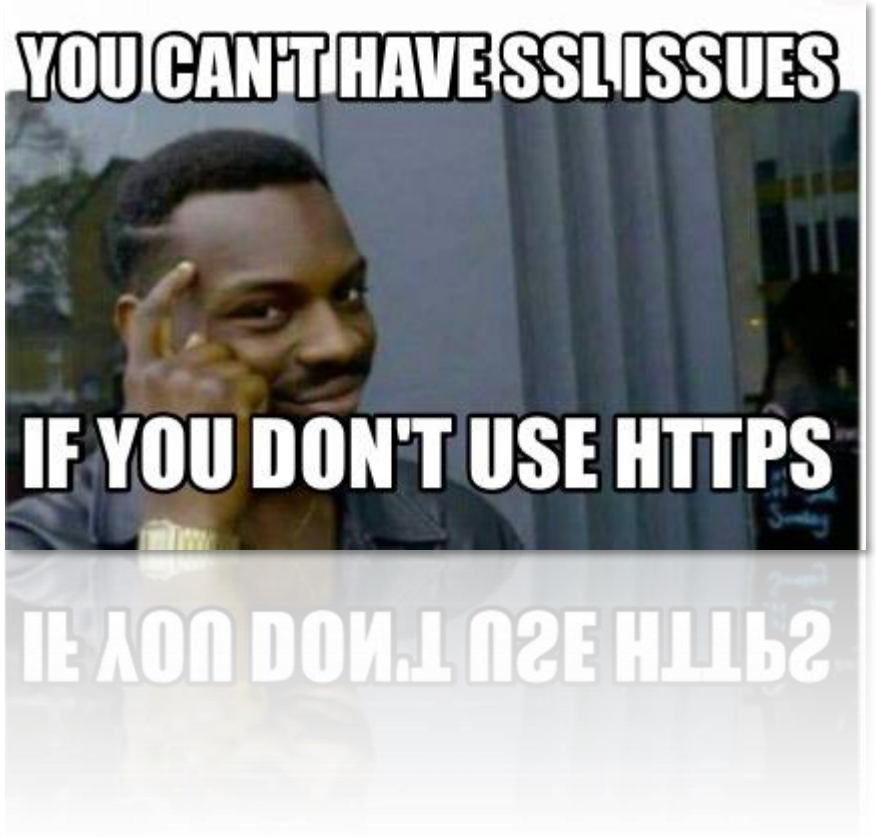
- Wi-Fi connectivity
- Remotely controllable
- AI technology
- Real time map
- Real Time video (in App)
- A vacuum and a home surveillance solution

- **Communications security (both strange and classic mistakes)**

- SSL used for HTTP, except for the app updates location server ¯_(ツ)_/¯

- SSL used for MQTT, but the MQTT servers also listen on plain non-SSL ports

- No SSL for RTMP camera feeds



Server-side Security (common mistake)

- The MQTT supporting servers allows to subscribe to all topics '+'
- This allows an attacker to get information about ALL currently online vacuums
- Among other information, an attacker will get:
  - The SSID of the Wi-Fi network currently used (wiggle.net anyone?)
  - The MAC address of the vacuum
  - The password protected URL of the RTMP stream of the vacuum

The password to view any RTMP stream is only dependent on the **MAC** and **current timestamp**. It is trivial for an attacker to generate the password and view all camera feeds.

Server-side Security (stupid mistake)

- All calls to the API servers are protected via a signature that accompanies the request.
- The signing process is the following:

sign = MD5(appId+appToken+tt)

- Both “appId” and “tt” are fields that are already inside the request. The appToken is a hardcoded key that can be extracted from the APK.
- This ‘signing’ process may add complexity but does not add security.

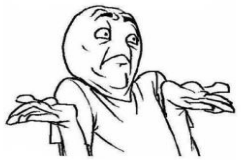
```
POST /v2/personuser/email/login HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/2.1.0 (Linux; U;
Android 8.0.0; SM-G950F Build/R16NW)
Host: XXXXXXXX.XXXX.XXX

{"data":{"password":"<MD5(realpassword)>",
. . .},"req":
{"tt":"1562592342299","appId":"2bd0d9c698f
a1d7dd43393f65f27fef6","sign":"14122c6a0c0
4765ce4dc4dee6b76ee14"}}
```

- **App/Comms security(strange mistake)**

- The Android app contains a self update procedure that contacts a specific HTTP server on a non-standard port for the latest APK location

- It’s easy to intercept the request and have the Android app to install a rogue application, since the App asks for installation permissions for APKs outside the Play Store. !?

- Why circumvent Playstore for updates?



- **Device security (classic mistake)**

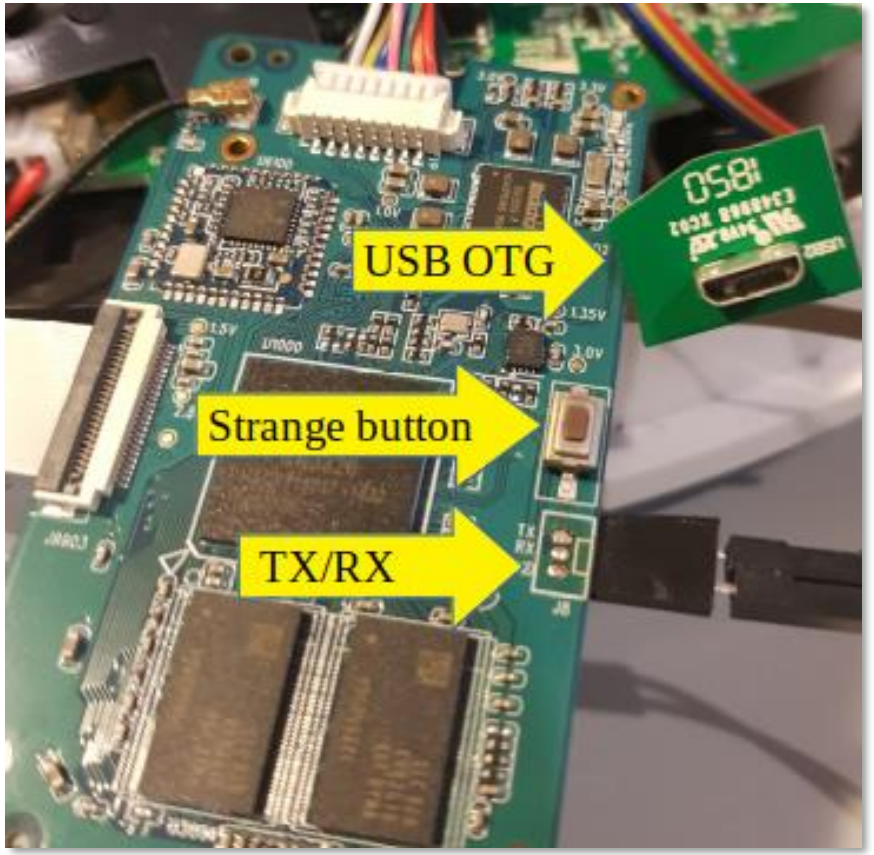
- It’s possible to an attacker that can reach the IP of the vacuum (usually local network) to directly connect to the video feed.

- The RTSP server running on the device has no password and the endpoint URL is the device MAC address.

- It’s classic for IoT devices to (wrongly) trust the local network

Physical security

- Exposed accessible USB port, but off by default
- By pressing the “strange button” at a particular time at boot, the vacuum boots into recovery mode
- It’s a Rockchip board, so one can use either Rockusb or, in this case, also enter UMS.
- With an USB cable, using UMS is just like browsing an USB pen drive. Getting persistent root becomes easy.



<Example 3>

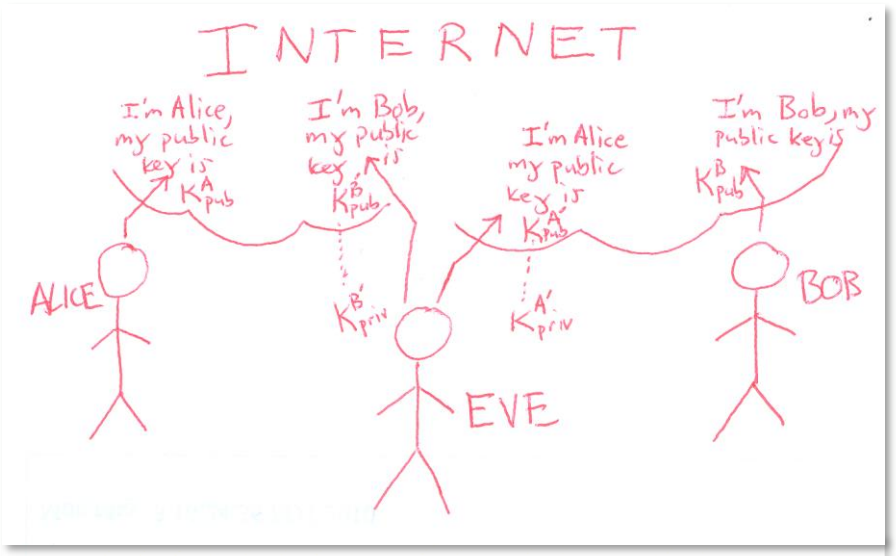
A Smart Camera

Camera features

- Wi-Fi connectivity
- Real Time video (in App)
- Both Local and Cloud Storage
- Motion Sensor
- Geofencing with App

Communications security (common mistake)

- The camera uses SSL to secure all communications
- Well, all, except one…



- **At boot**, the camera checks the configuration server for an updated list of all endpoints (API, web, RTSP, etc…)
- <u>This critical check is done via plain HTTP</u>
- Tampering this request can give an attacker absolute control of the camera.

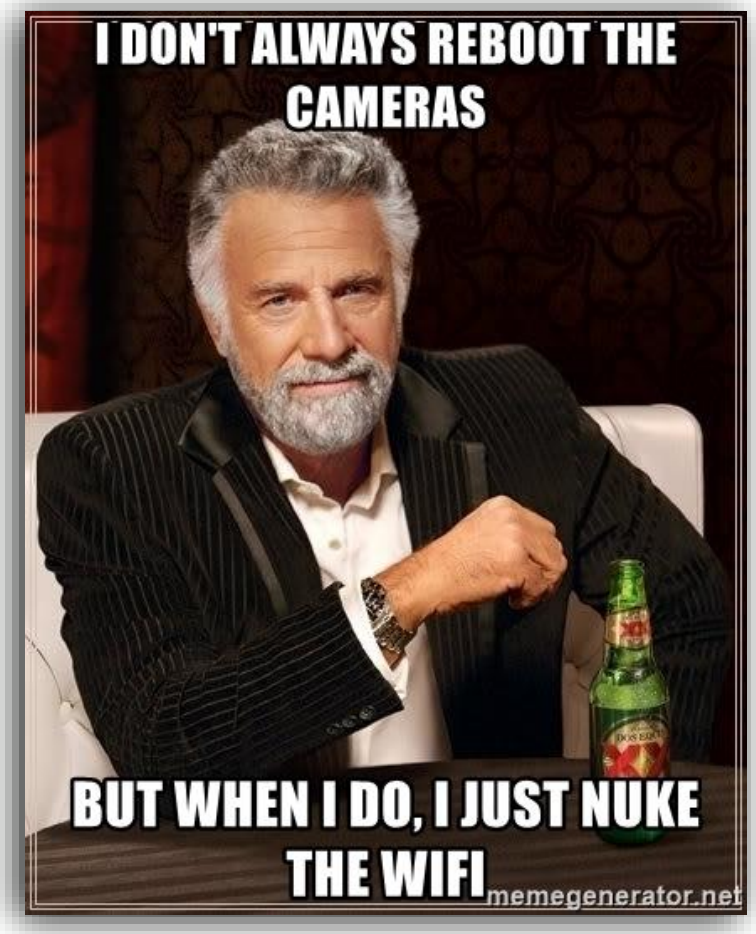
- **Device security (design mistake)**

- For the camera initial setup, the user has to use the smartphone App and generate a QR-Code with the Wi-Fi credentials
- But the credentials setup routine is also active at every boot of the camera
- An attacker that is in line of sight with the camera can show its own Wi-Fi credentials and make the camera connect to an Access-Point of his choosing

- **Device security (design mistake)**

- Last two issues only happen at boot.
- Can an attacker make the camera reboot?
- TL;DR -> Yes

- There are several code paths that makes the camera reboot. One of them is the failure to connect to the video stream servers for more than 120 seconds.
- That’s fairly easy to achieve in a Wi-Fi device…



Server-side Security (classic mistake)

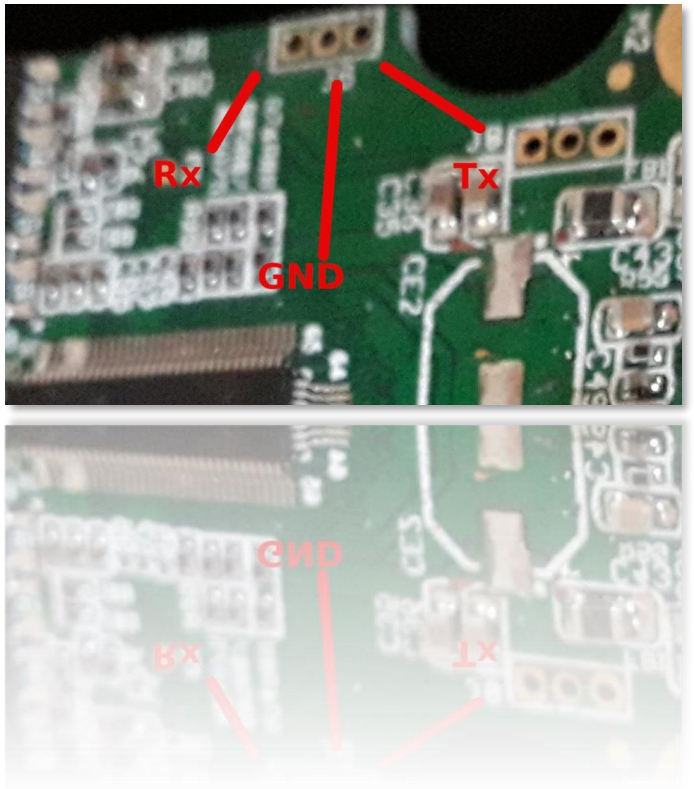
- The streaming servers allows a user to preview a snapshot of what was recorded as an image.
- The height and width of an image can be arbitrarily manipulated and are not limited
- The server, if asked, will try to generate an image with 65536x65536 pixels (4Gb) and render itself useless.

Storage security (design mistake)

- The device has an undocumented accessible SD card slot.
- If a card is present, the device automatically starts to save video recordings to the SD card, without warning or any indication to the user.
- If an attacker is able to insert and later retrieve an SD-Card, this can present a major privacy concern.
- There is no way to disable this behavior.

- **Physical security (common mistake)**

- The board has 2 marked TX/RX pins that implement a serial port and yield a root console that is not password protected
- **For this attack to work, one must open the device but….**



- That’s too much trouble…
- The device accepts an SDCard to expand local video storage AND when booting, the device first tries to boot from the SDCard…
- <u>An attacker only has to put the proper boot files in the SDCard to pwn the camera…</u>

~$ cat ./final-words.txt

- **I've been doing (paid) IoT research for a long time**

- **As many of you, I keep seeing the same mistakes over and over again.**

- **A non extensive list includes:**
- Faulty or no encryption at all securing communications
- Bad cryptography practices
- Vulnerable firmware updates and/or lack of signing
- Default credentials
- Insufficient or inexistant physical security
- Basic design or logic mistakes

~$ cat ./final-words.txt

- **I've been doing (paid) IoT research for a long time**

- **Feature-wise, there was a fantastic a boom in IoT, with all the good and bad.**
- **Security-wise, not much has changed.**

- **The industry can not be in a permanent status of repeating the same mistakes over and over again.**

- **Most security issues described in the examples are prevalent throughout the IoT landscape and there is an urgent need to start to address them in a more systematic way.**

~$ cat ./final-final-words.txt

- There will be a point in time when I will want to buy toaster without Wi-Fi and there will be none in the market.
- And I really don’t want to die in a fire because of my toaster had a bad firmware update…
- In the end, we all are going to need to play our part in fixing this mess.

phdays.com

Positive Hack Days

ORIGIN

Thank you for attention!