

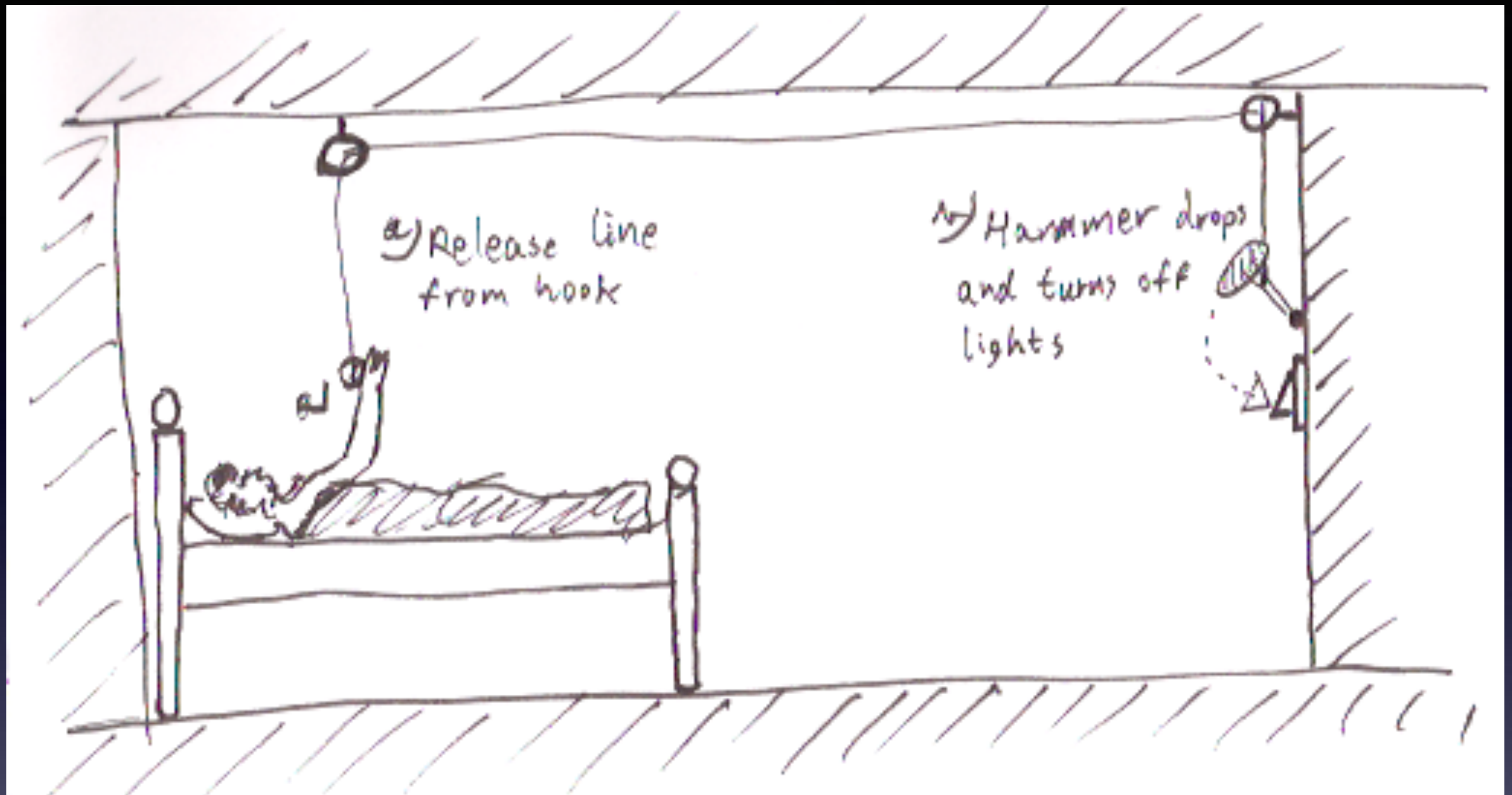
Exploring Security of IoT and Embedded devices

es - HITCON CMT 2018

about

- es
- knows a little about:
 - web
 - security
 - mechatronics
- UCCU / 好想工作室 / 若渴計劃
- Developer @ Positive Grid





- How to turn off the lights without getting out of bed?
- <https://lifehacks.stackexchange.com/questions/7696/how-to-turn-off-the-lights-without-getting-out-of-bed>



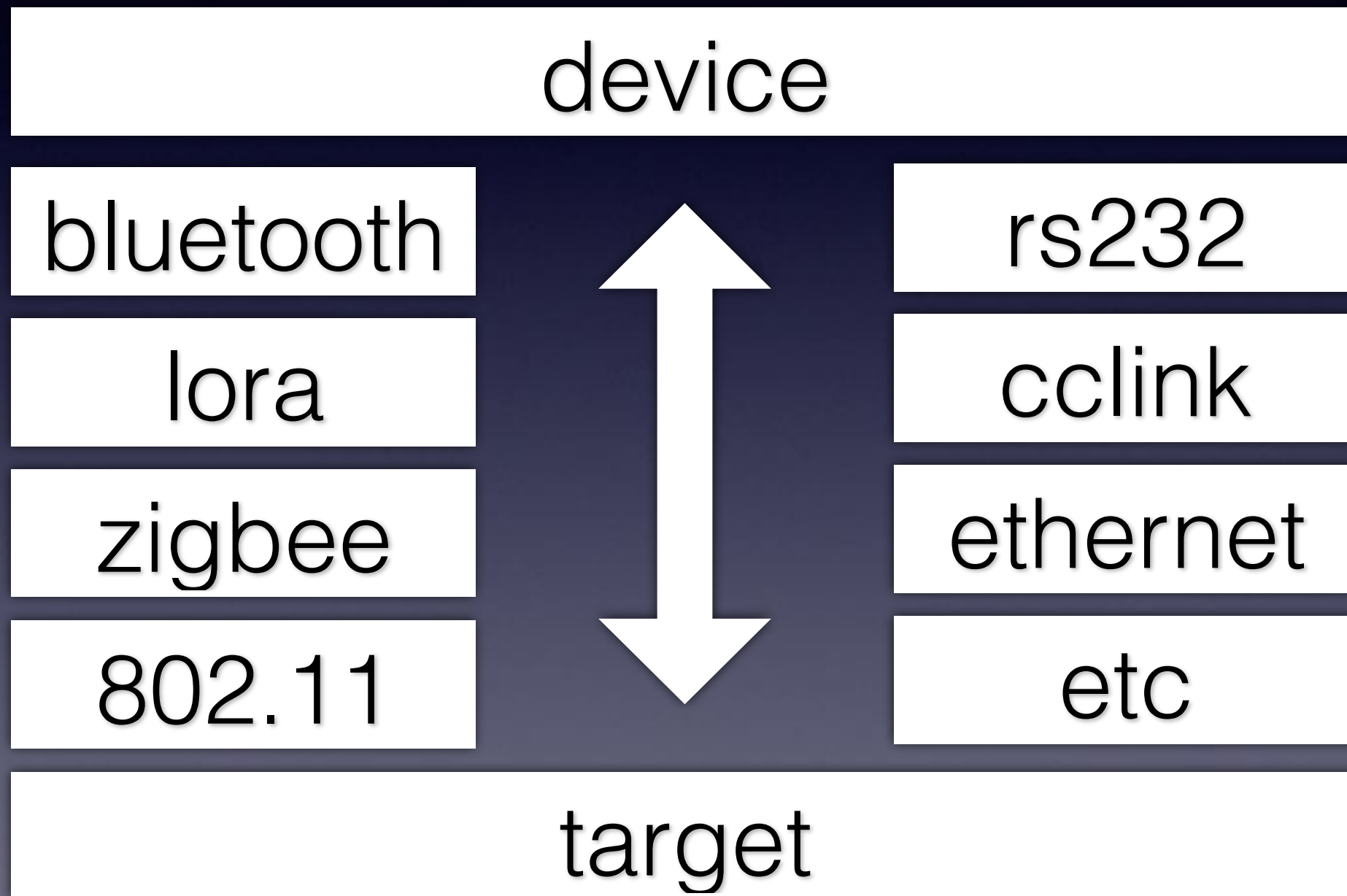
embedded device / iot

- embedded
 - computer system, dedicated function within a mechanical / electrical system
 - watch / mp3 player / traffic light / vehicles etc
 - bluetooth / lora / zigbee / wifi
- iot
 - embedded device
 - ability to inter-operate with internet

embedded device / iot

- embedded
 - computer system, dedicated function within a mechanical / electrical system
 - watch / mp3 player / traffic light / vehicles etc
 - bluetooth / lora / zigbee / wifi
- iot
 - embedded device
 - ability to inter-operate with internet

type of communication



simplified

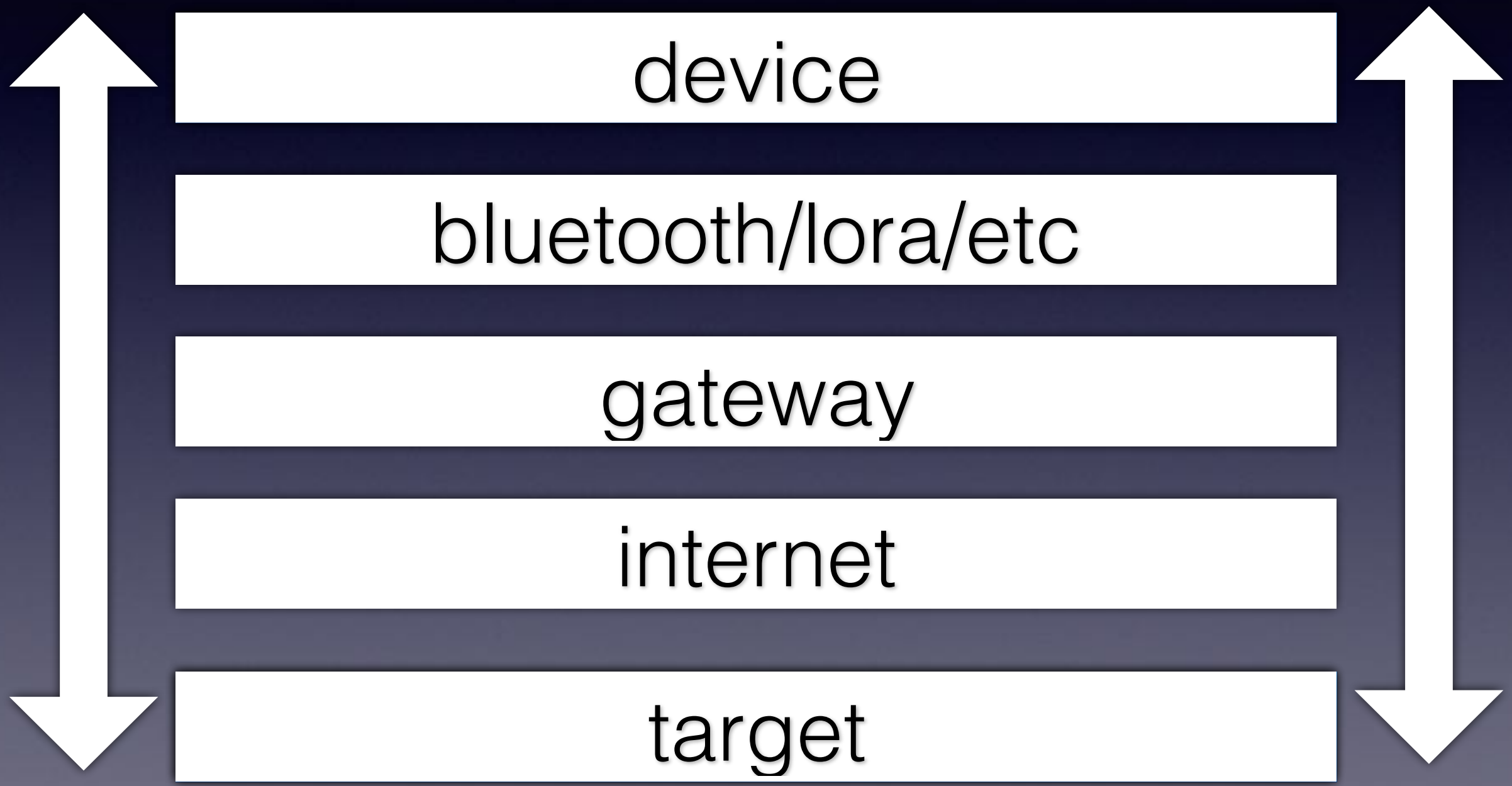
device

communication

target



iot



device

bluetooth/lora/etc

gateway

internet

target

* border is blurry between iot/embedded these days

embedded problems

embedded problem

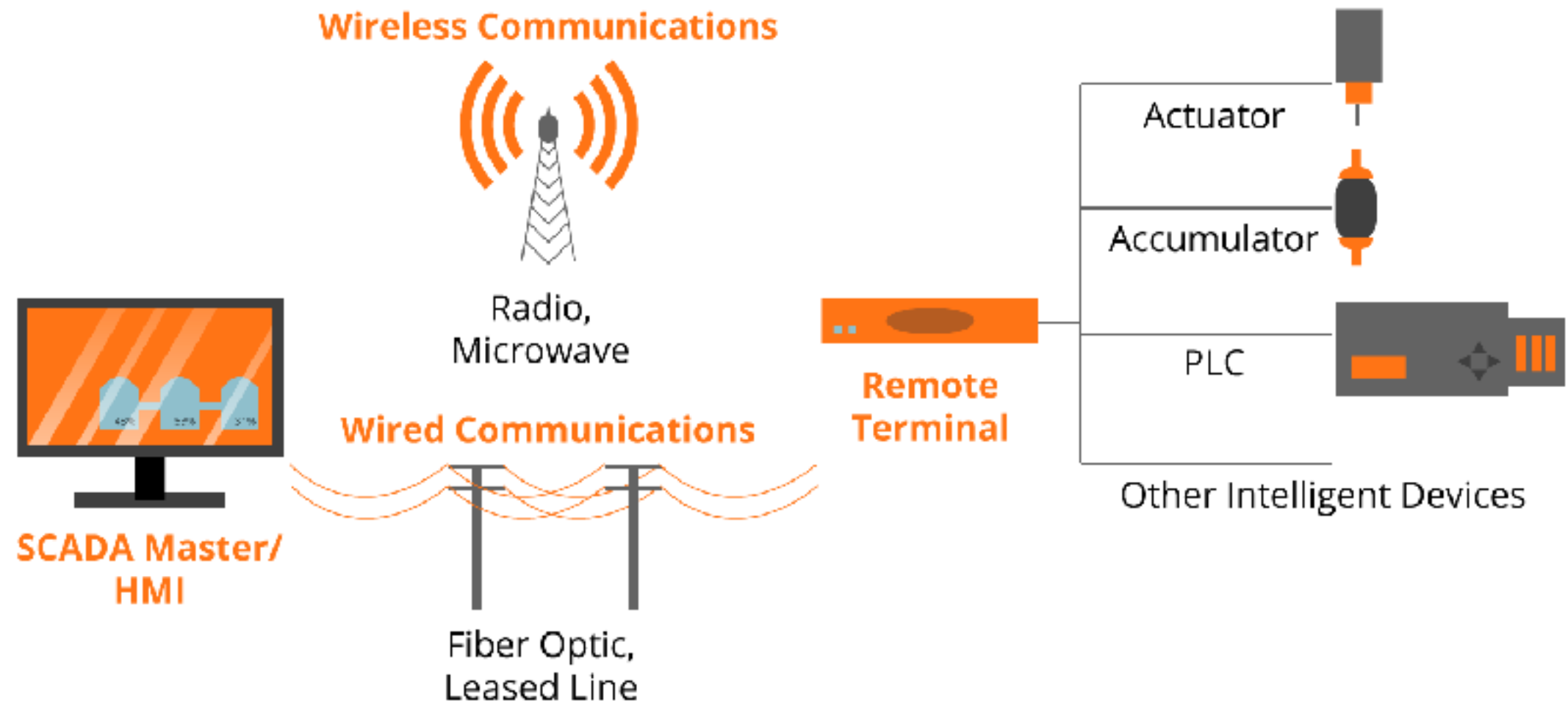
- lots of facilities and equipments
- needs personal to visit/inspect/adjust/collect data

SCADA

- supervisory control & data acquisition
- allowing control of processes locally / remotely
- monitor / gather / process real-time data

SCADA

GENERAL SCADA SYSTEM LAYOUT



attack on SCADA

- water purification plant
 - leaking raw sewage to local waterway
- power grid
 - no power (\$\$\$\$\$)
- sensitive industrial application
 - impact on productivity

enough accidents with hacked SCADA
what about IoT?

what if it is

- a scooter battery charger?



(image for reference only)

what if

- ...we hit the scooter itself? (or hit a car?)
- car: obd2 dongle?
 - Fast and Vulnerable: A Story of Telematic Failures [1]
- bricking via OTA updates?
- weird APIs?

[1] <http://cseweb.ucsd.edu/~savage/papers/WOOT15.pdf>

what if it is

- ...a toaster?



<https://www.pentestpartners.com/security-blog/iot-aga-cast-iron-security-flaw/>

what if it is

- ...a stove knob? :)



conclusion:

any device has its hazards, but we need to think
about mitigation and it's necessities

conclusion:

any device has its hazards, but we need to think
about mitigation and its necessities

let's talk about network attack patterns

network attack - patterns

- interception
- **unauthorized party gained access** to an asset
- attack its **confidentiality**
- **capture data** in a network

network attack - patterns

- modification
- **unauthorized access + data tampering**
- attack its **integrity**
- **modify** message content
- **alter** the program

network attack - patterns

- injection
- **inserts counterfeit data**
- attack its **authenticity**
- **spurious** message

defend against network attacks

- common weaknesses with embedded / IoT
- no native encryption support
- need to add some new hardware
- \$ / latency / technical difficulty

current solution

- SSL/TLS
- public key cryptography
- embed encryption inside SoC

problems with IoT

- consider that you just got a
 - air purifier
 - rice cooker
 - door lock
 - ?

problems with IoT



problems with IoT

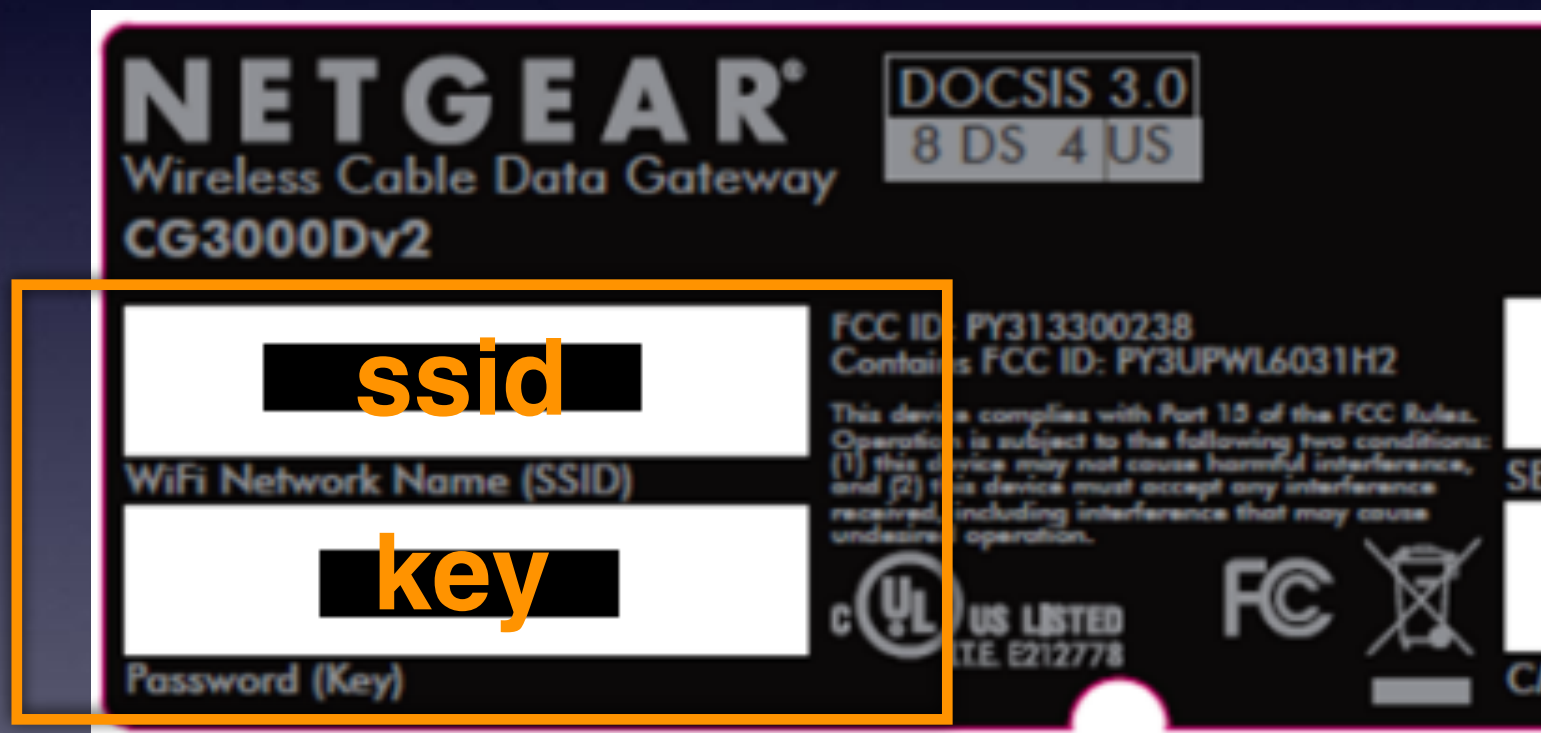
- needs to be connected to network
- it cannot connect to internet by itself
- it must use your network at home / work

problems with IoT

1. device turns on AP mode
2. connect to this AP
3. send it your network credentials
4. device turns off AP and connect to the network

why unencrypted?

- how to share this pre-shared key?
 - predefined
 - keyboard
 - out of band
 - no key at all



network attack “entry”

device

bluetooth / zigbee / wifi



gateway

tcp / udp

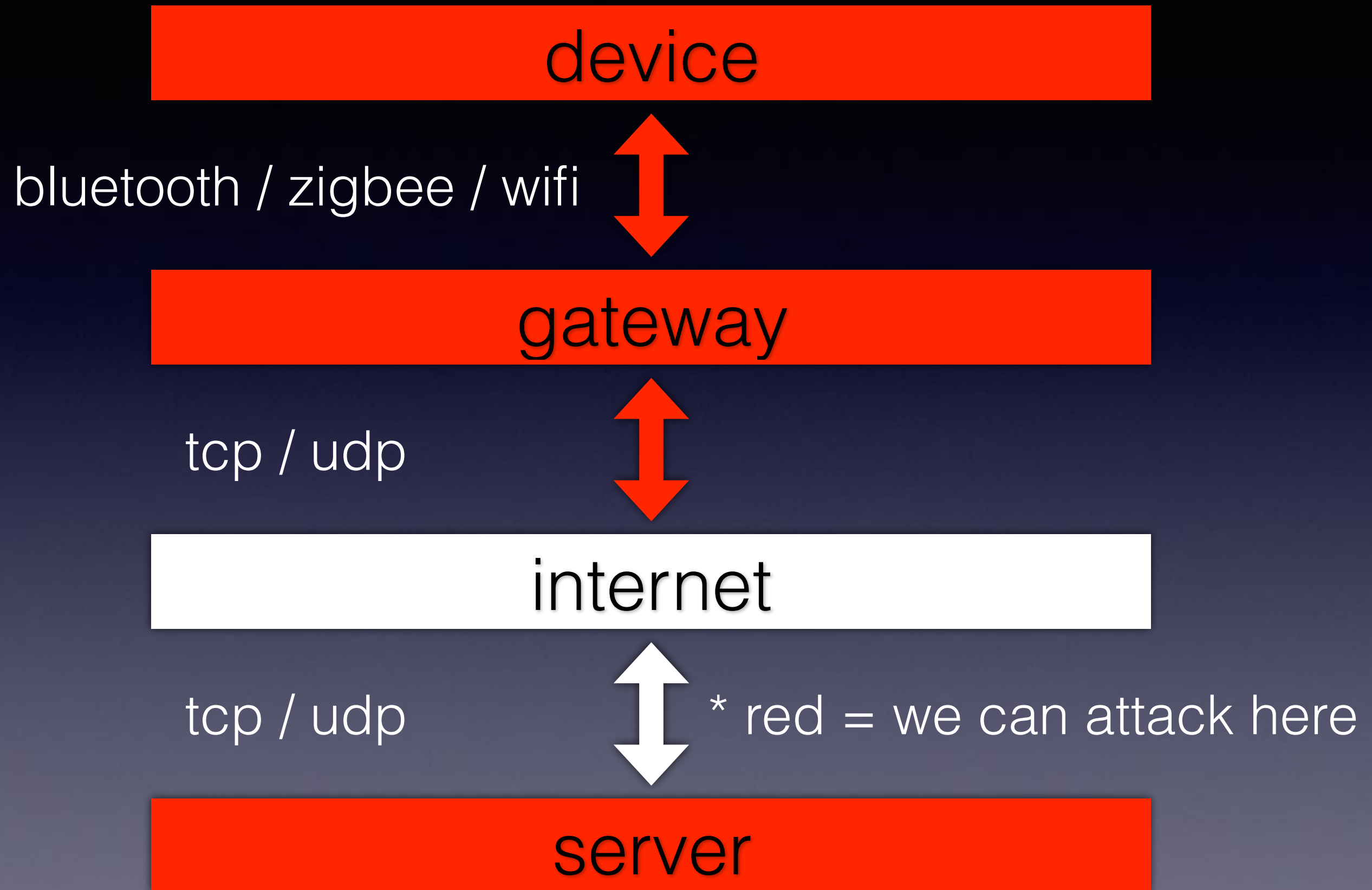


internet

tcp / udp

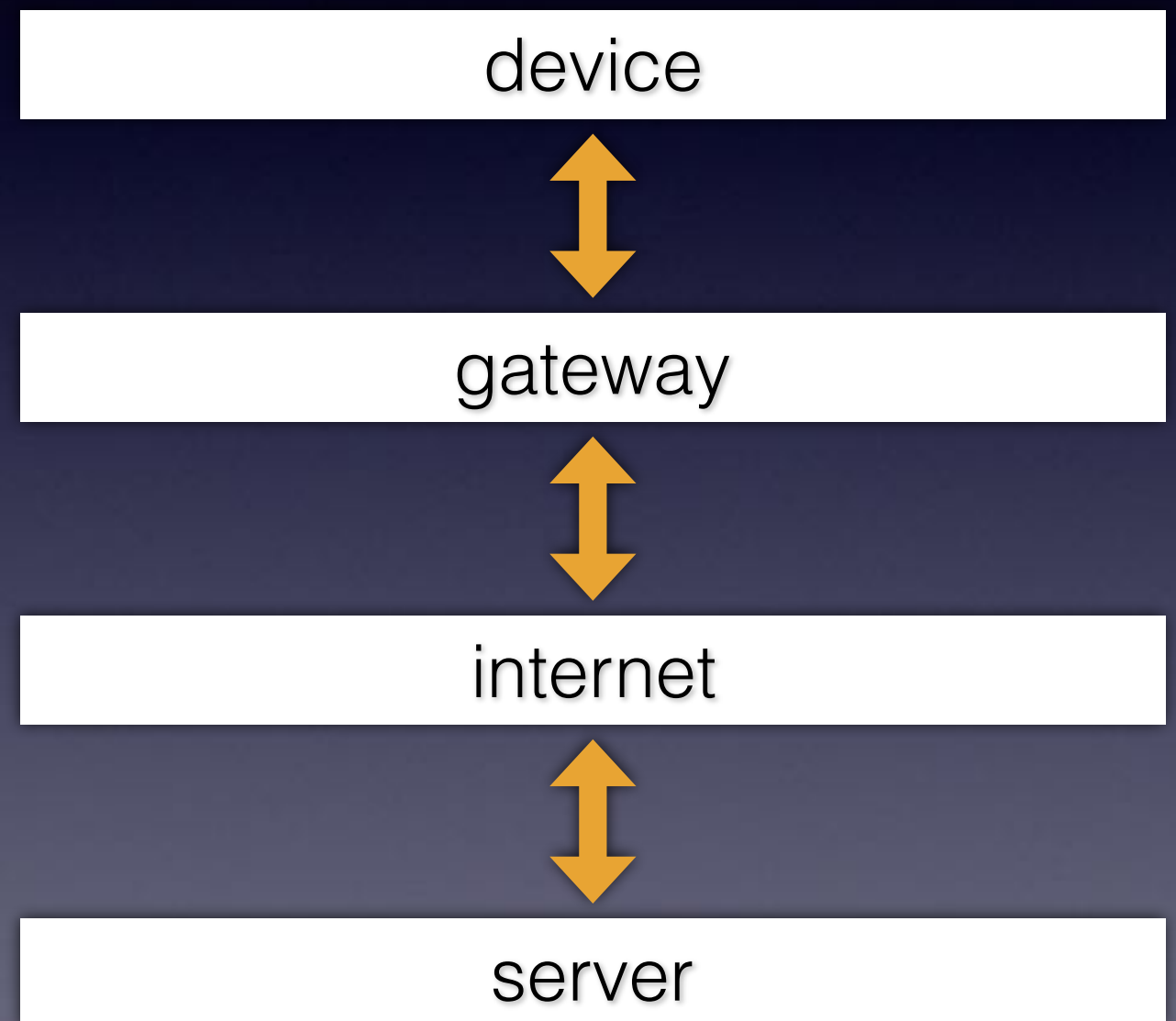


server

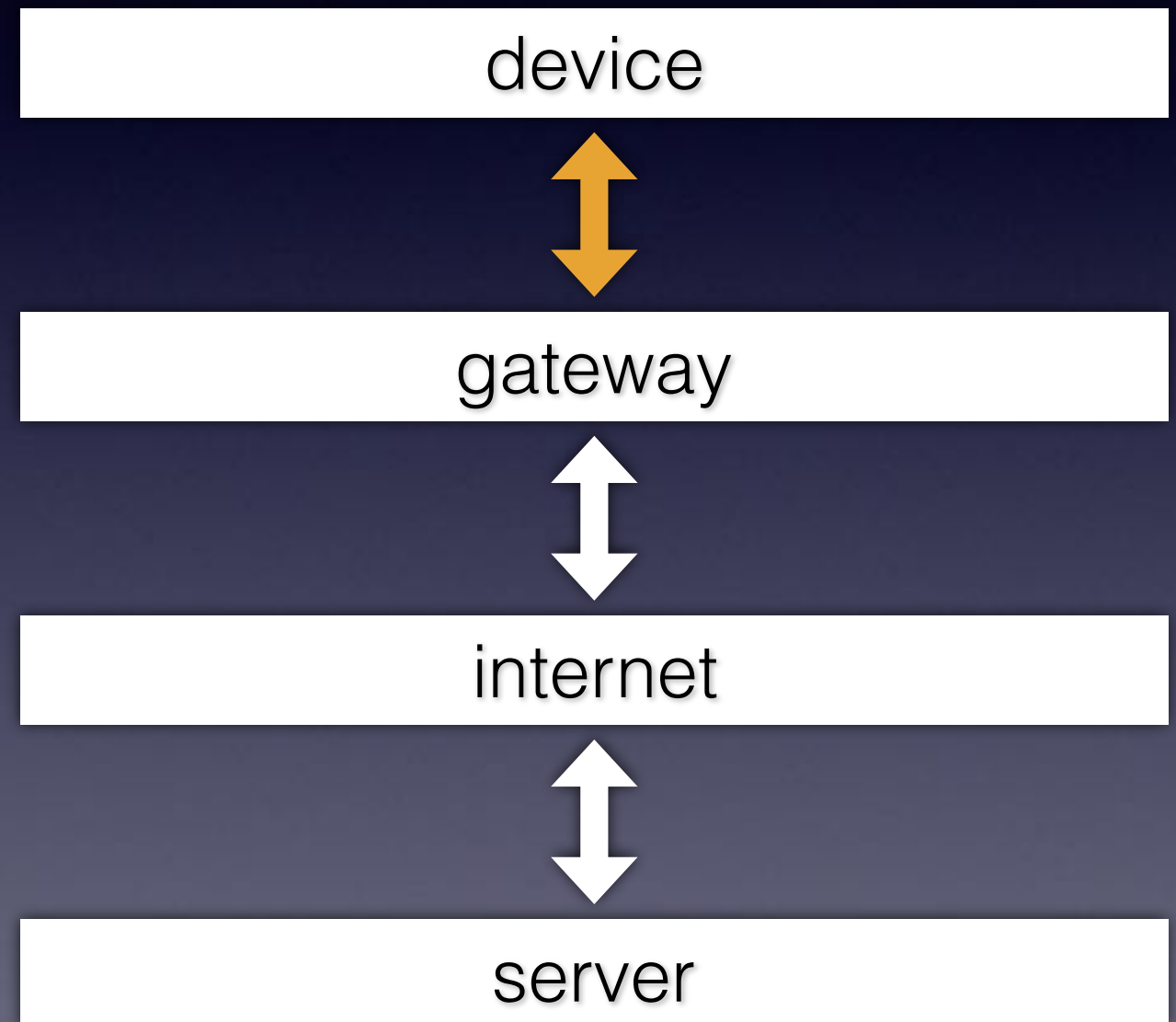
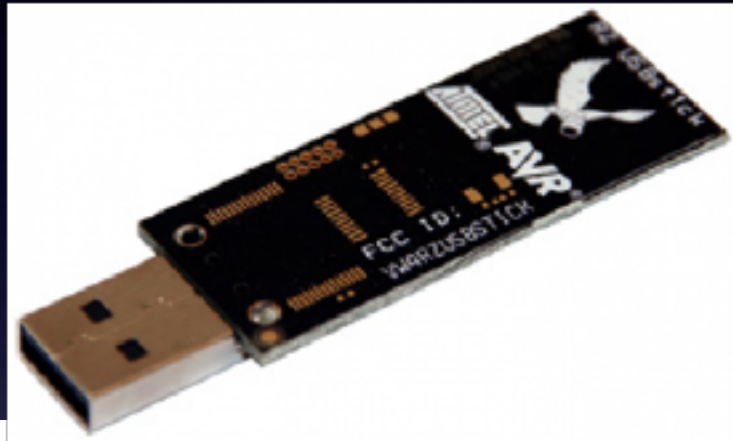


attacking transmission

- interception
- modification
- injection

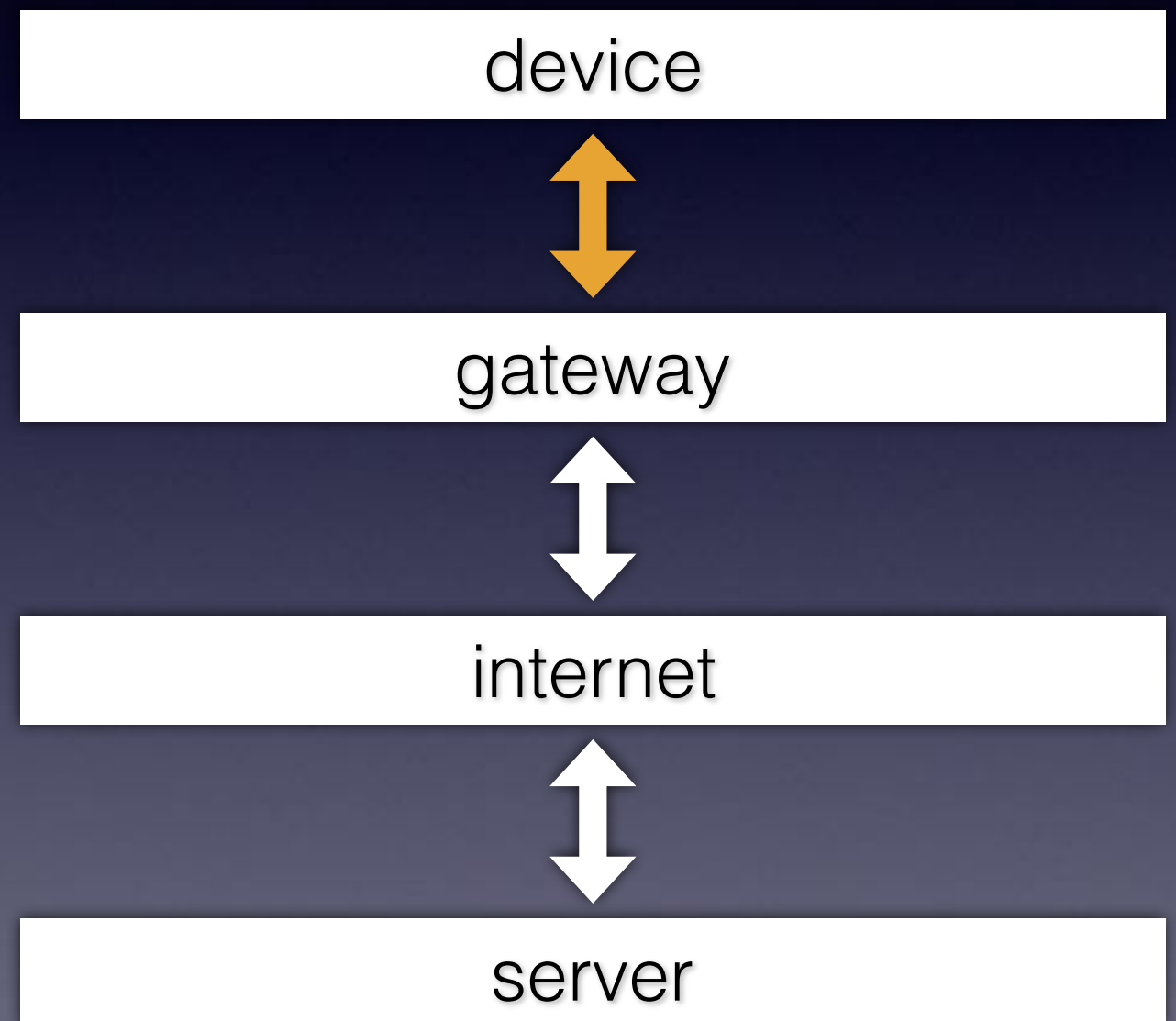


attacking transmission



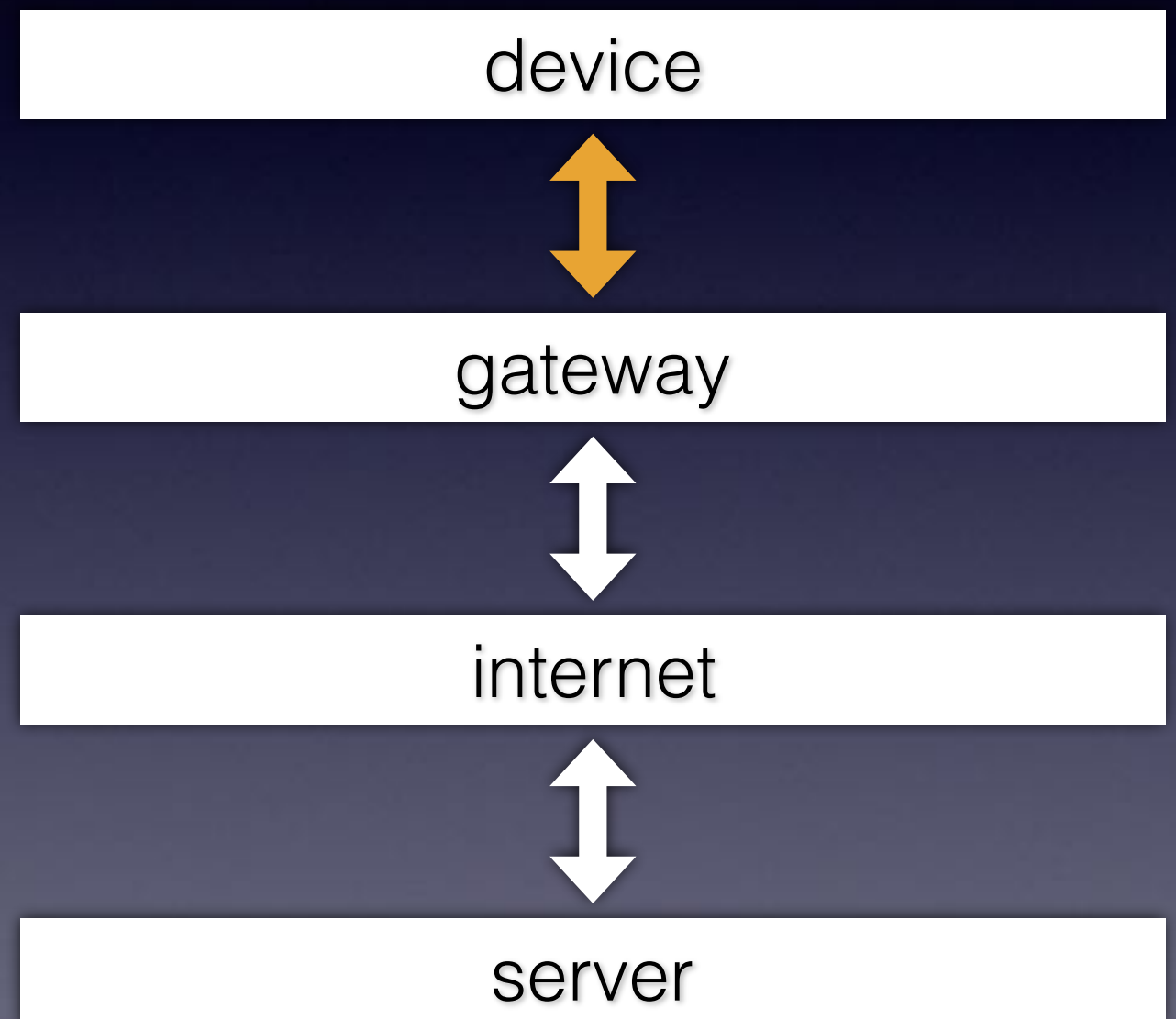
attacking transmission - problems

- tcp / udp
 - tcp sequence (forging)
- https
- bluetooth, zigbee etc
- ≥ 4.2 : secure connection



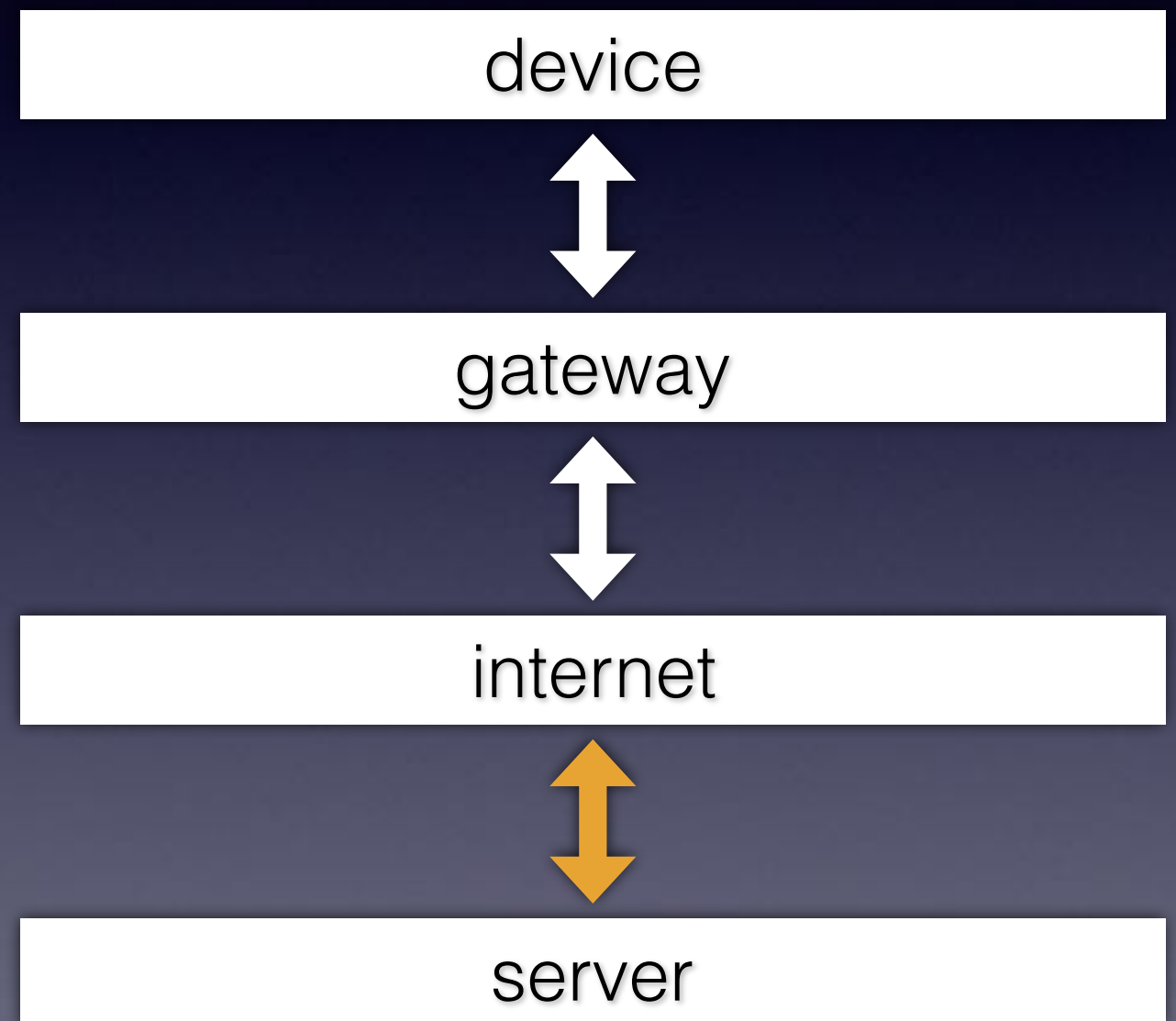
attacking transmission

- rouge AP / evil twin
- lan tap
- packet injection



attacking transmission

- http/https
- certificate pinning
- BGP / DNS

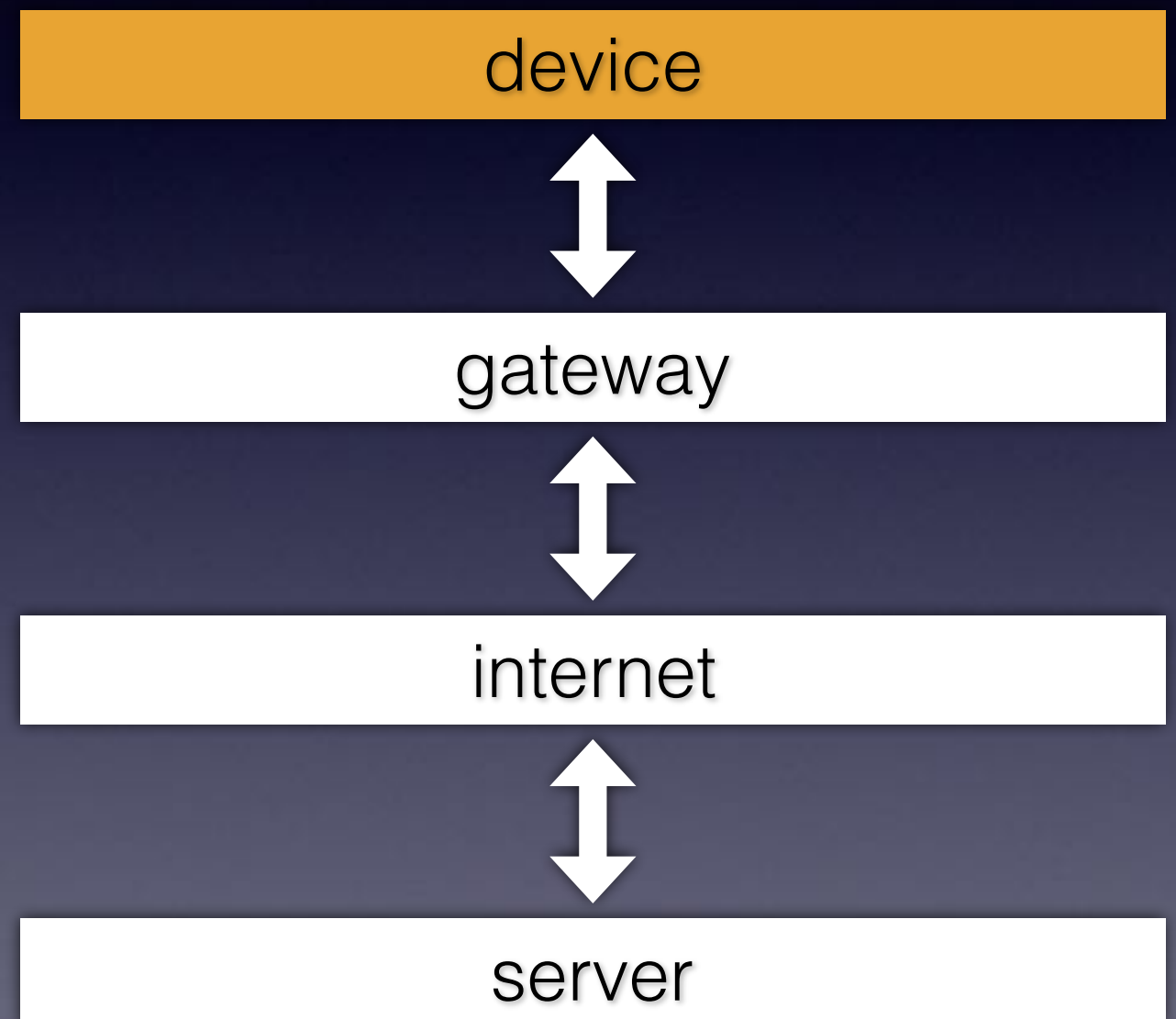


attacking device

- implementation error
 - insufficient auth
 - shit logic

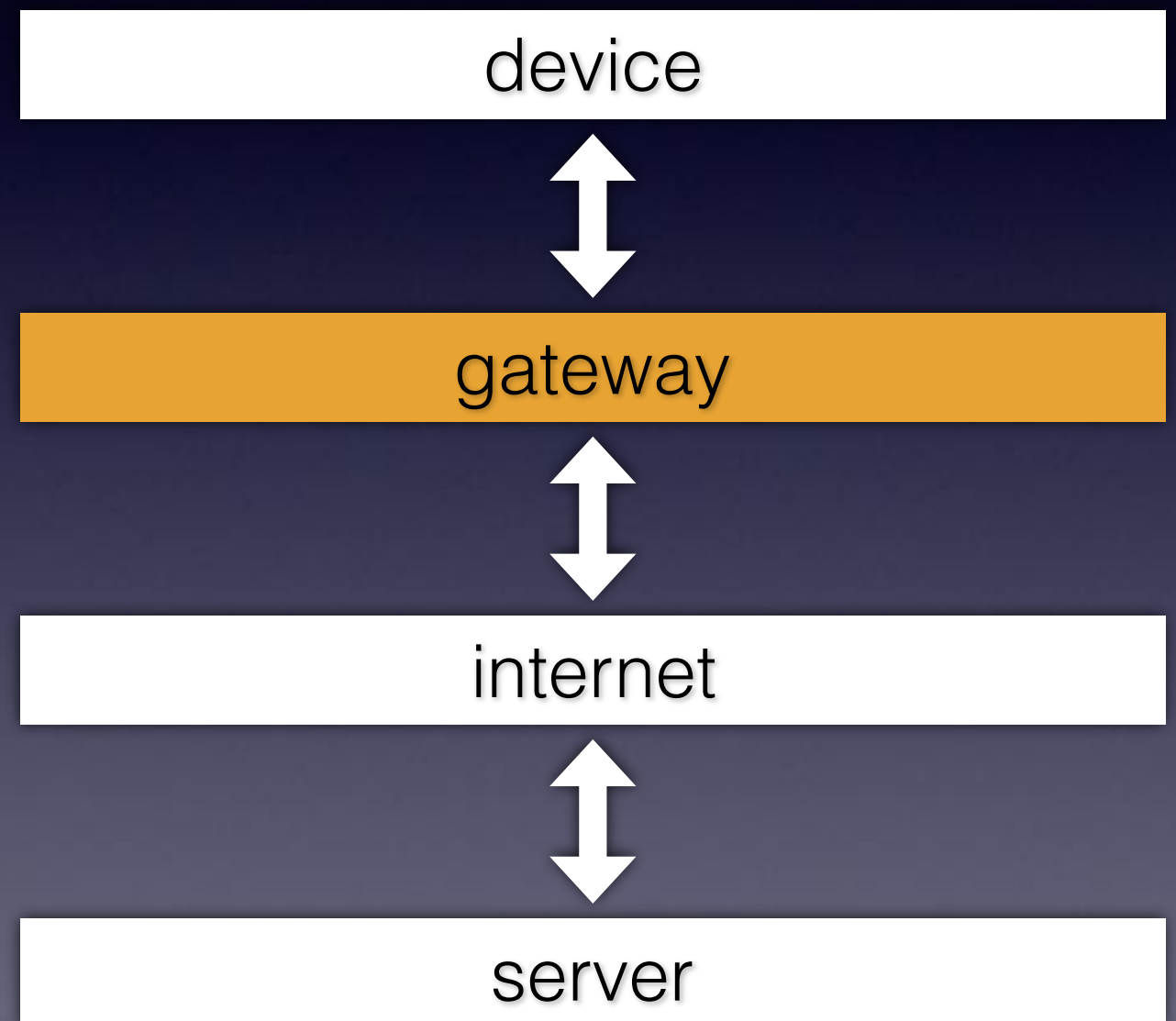


- protocol vulnerability



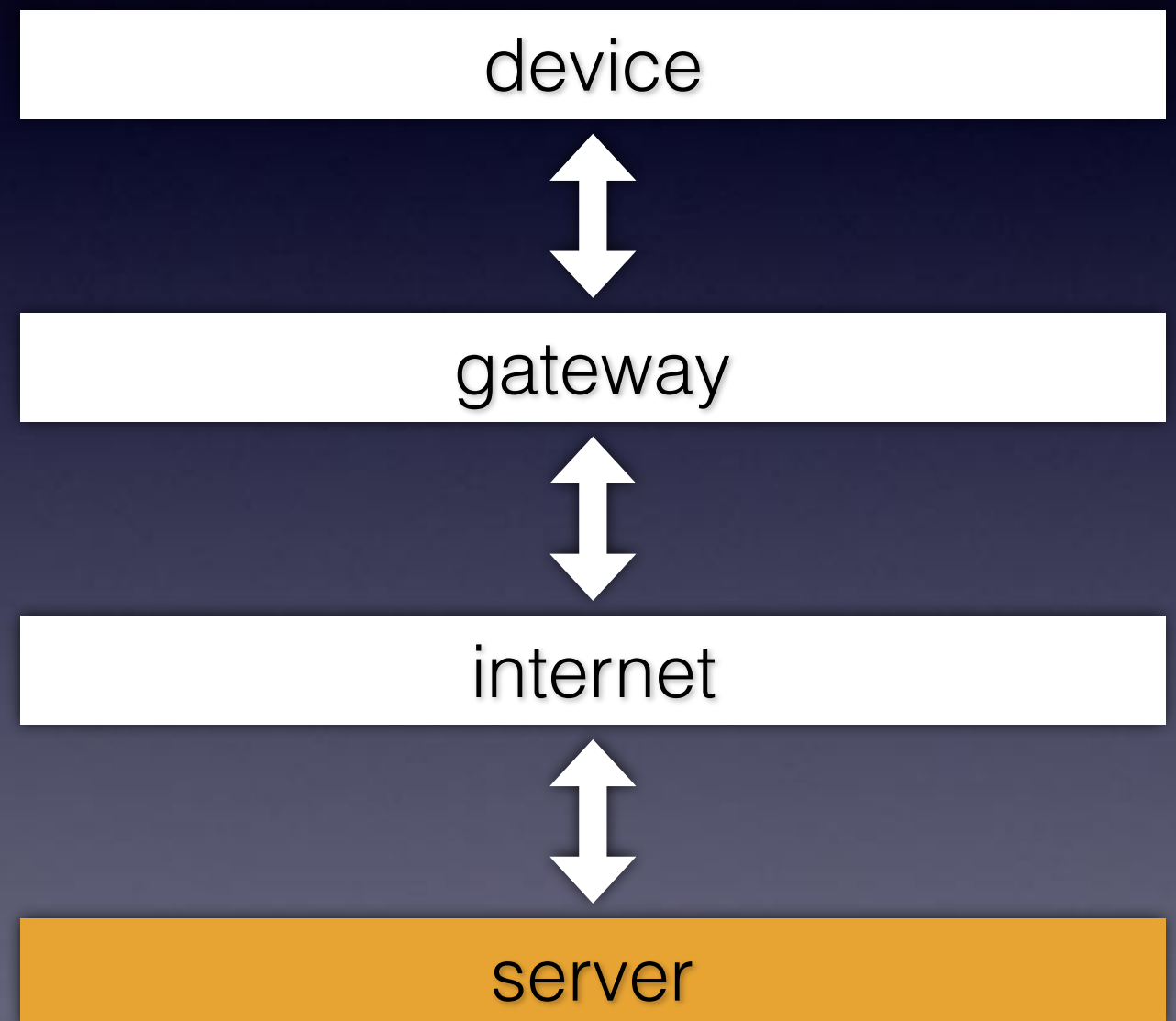
attacking gateway

- some interesting service leading to escalation
- weak password / known CVE



attacking server

- denial of service
- information leakage
- unauthorized use of device
- shit logic
- insufficient auth
- exploits / known CVE / SQL injection / etc



real world example

real world example

```
Raw Params Headers Hex
POST HTTP/1.1
Host:
Content-Type: application/json
Connection: close
Accept: */*
User-Agent:
Content-Length: 124
Accept-Language: en-us
Accept-Encoding: gzip, deflate

{"credential":{"authtype":"01","username":"@gmail.com"},"accessid":"","password":"
```

plaintext credentials over http

real world example

- GET /devices/?userid=xxx

HTTP/1.1 200 OK

```
[{"deviceid": "1234567890", "devicename": "iPhone12", "model": "iPhone12,1", "MAC": "08:00:2E:01:1A:5B",  
  "SNO": "1234567890", "scope": "owner", "deviceowner": "john.doe@gmail.com"}]
```

real world example

- Devices can be shared between users on this particular service
- Able to {edit,view,set} {owner,guest} of a device regardless of ownership
 - Set owner to “guest” on owned device

real world example

- Devices can be shared between users on this particular service
- Able to {edit,view,set} {owner,guest} of a device regardless of ownership
- Set owner to “guest” on owned device
 - Owner gets locked out :(

device study



device study

- app
 - certificate pinning - can be circumvented
 - security is quite good
 - attribute “sign” - md5 ((all keys sorted) + (secret key))

device study

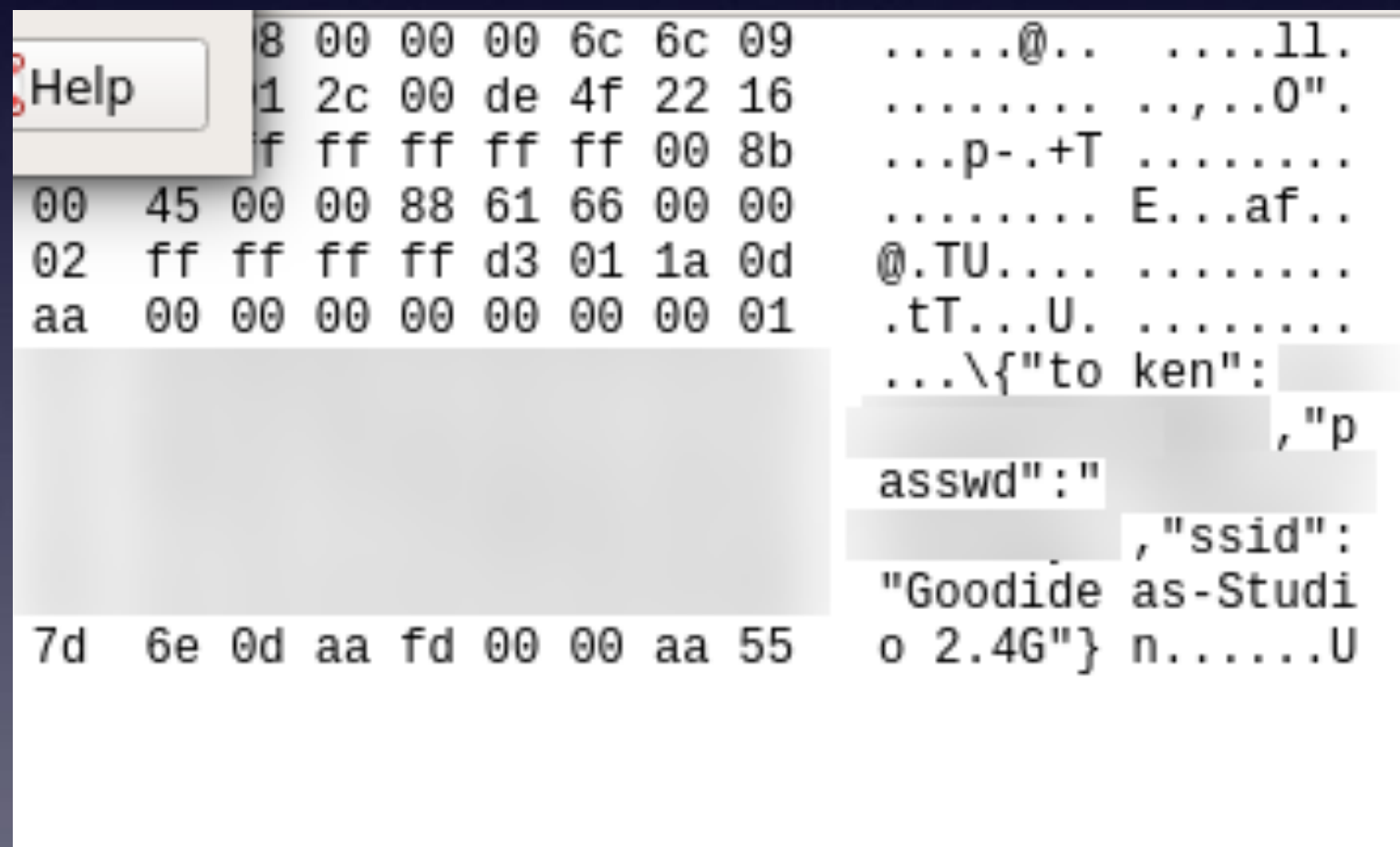
- some difficulties...
 - i have a rt3070-based wifi dongle
 - used wireshark to decrypt wpa2 packets
 - driver is unstable & dongle overheats
 - ordered an ALFA AWUS036ACH
 - time wasted while looking for equipment...

device study

- manual says device has two pairing modes
 - bluetooth
 - wifi AP
- both needs to send credentials over
- bluetooth mode doesn't work actually

device study

- y u no encryption :(



device study

- what it looks like when a command (e.g. power on) is issued
 - phone -> Publish Message -> remote A
 - remote B -> Publish Message -> plug
 - remote A -> Message ACK -> phone
 - plug -> Message ACK -> remote B
- wonder why 2 remote servers are needed

wtf is this?

0000	d0	22	be	23	7e	21	00	c0	ca	96	82	43	08	00	45	00	.".#~!... ..C..E.
0010	01	06	67	1f	40	00	e2	06	85	d3	22	d0	7e	26	0a	00	..g.@... ..".~&..
0020	00	09	07	5b	89	ac	73	84	b4	a6	d5	ae	c1	34	80	18	...[...s.4..
0030	00	6e	0a	27	00	00	01	01	08	0a	00	a7	fb	75	00	05	.n.'....u..
0040	bf	f9	32	cf	01	00	20	73	6d	61	72	74	2f	6d	62	2f	..2... s mart/mb/
0050	69	6e	2f	30	32	32	30	30	34	39	30	64	63	34	66	32	in/02200 490dc4f2
0060	32	31	36	31	62	64	30	00	09	32	2e	31	63	34	30	32	2161bd0. .2.1c402
0070	61	65	63	33	32	39	36	39	32	33	37	38	2b	48	2f	79	aec32969 2378+H/y
0080	37	4c	4d	45	76	4d	6c	78	64	54	37	6b	49	78	42	35	7LMEvM1x dT7kIxBS
0090	31	39	6a	72	47	2b	6d	70	57	51	69	4d	57	59	6b	2b	19jrG+mp wQiMwYk+
00a0	6a	49	41	57	74	4e	4f	61	6b	38	54	4b	36	74	56	31	jIAWtNOa k8TK6tV1
00b0	36	4f	30	33	41	50	64	64	31	4c	57	38	2f	49	39	51	6003APdd 1LW8/I9Q
00c0	4c	51	6e	73	47	46	63	54	79	4d	4c	38	37	41	57	5a	LQnsGFcT yML87AWZ
00d0	49	34	34	77	2b	47	38	64	79	76	6d	37	44	2f	47	66	I44w+G8d yvm7D/Gf
00e0	56	53	31	75	35	68	52	52	6a	49	58	4a	6f	72	54	4d	VS1u5hRR jIXJorTM
00f0	6a	58	61	50	49	56	59	68	4a	78	6f	75	31	6a	56	6b	jXaPIVYh Jxou1jVk
0100	4e	4c	31	70	54	49	5a	59	49	7a	42	79	38	31	2b	47	NL1pTIZY IzBy81+G
0110	45	77	3d	3d													Ew==

No.: 8 · Time: 1.290315644 · Source: 34.208.126.38 · Destination: 10.0.0.9 · Protocol: MQTT · Length: 276 · Info

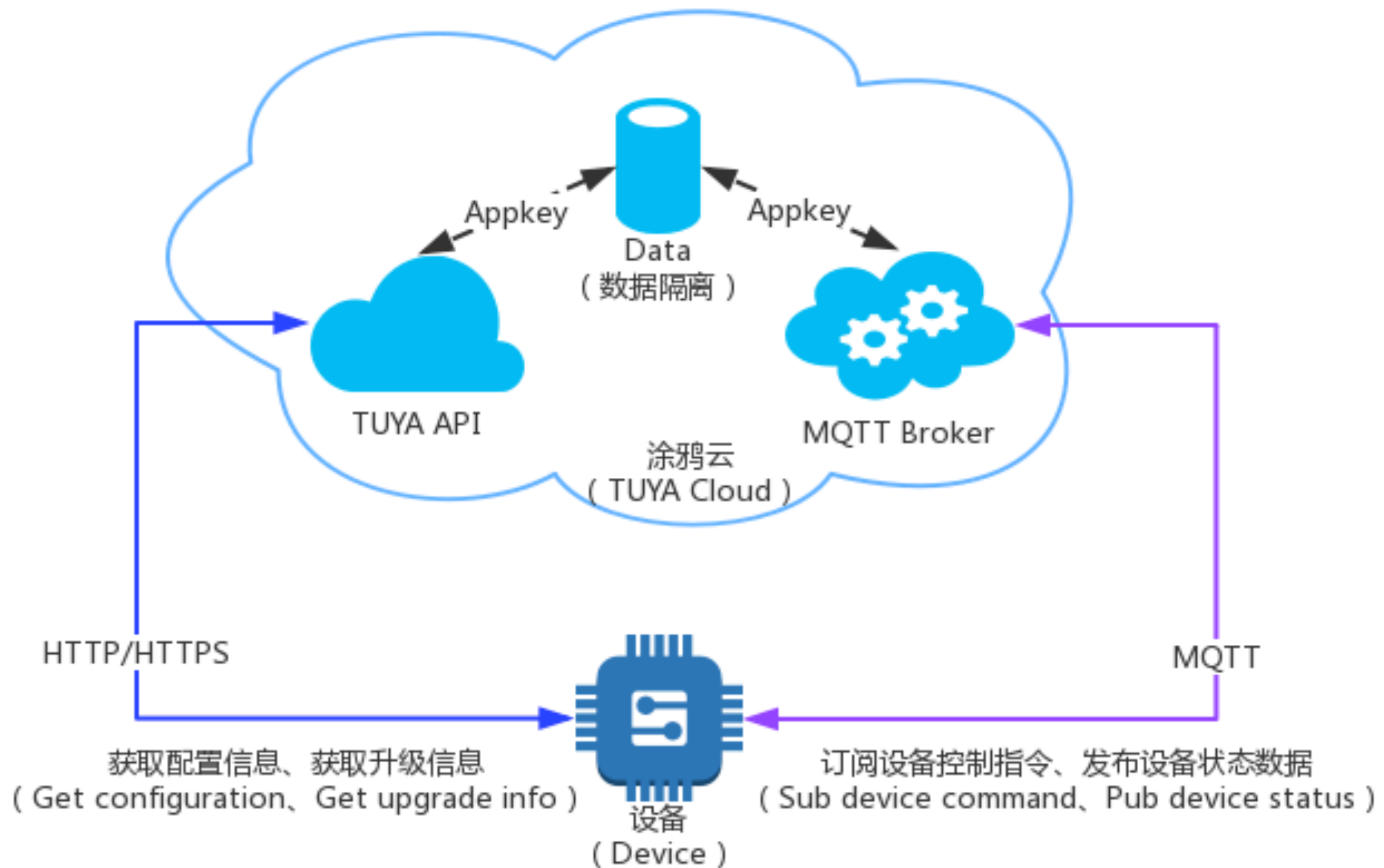
device study

- so much security involved in this \$10 plug
 - cert pinning in app
 - web framework built-in, not *that* difficult
 - mqtt message is somehow encrypted
- remote domain name: `a1.tuya.us`

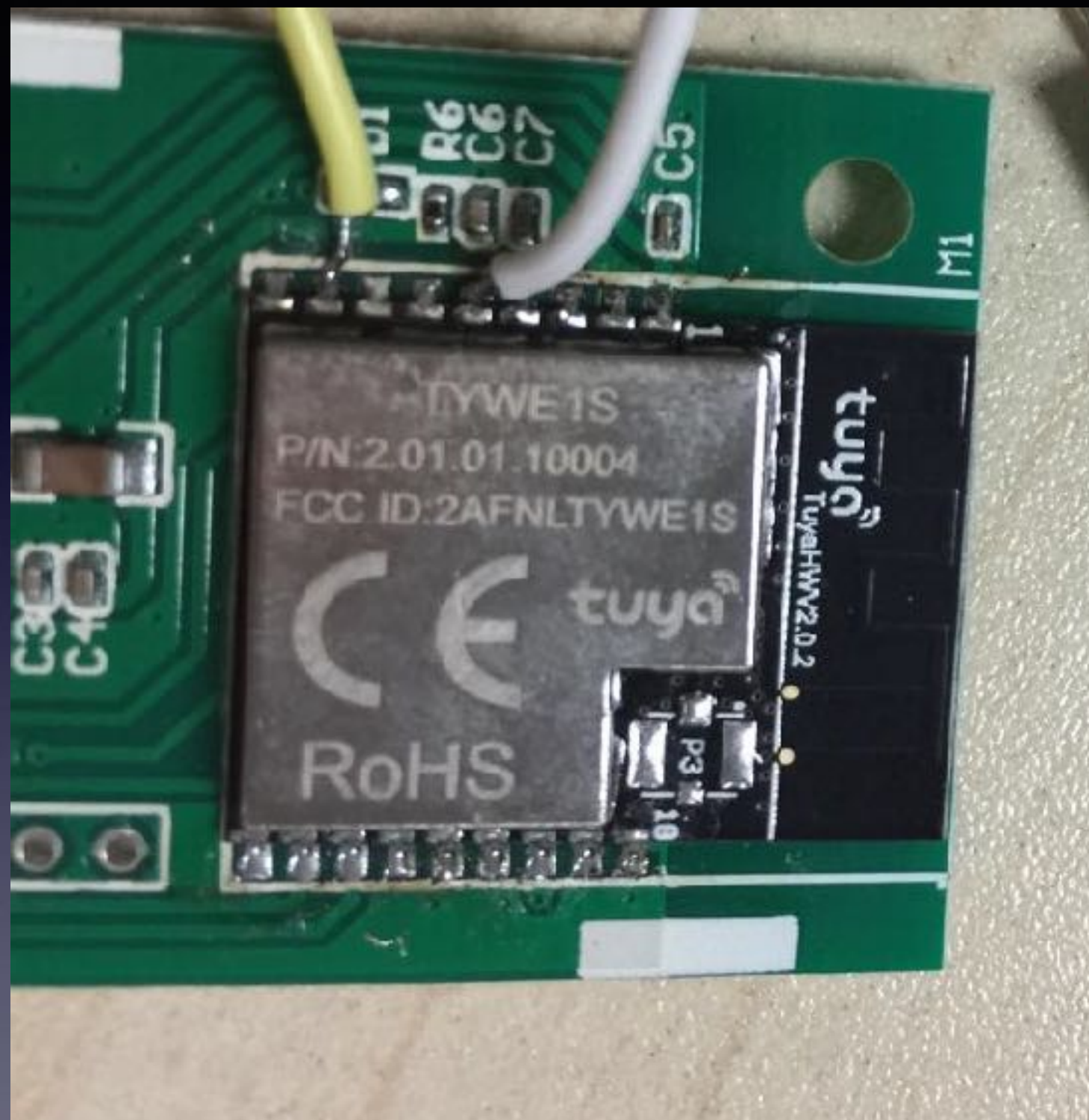
tuya

- “TaaS” (IoT as a Service)
- MCU + Cloud Support
- device maker buys MCU from tuya
 1. integrates with their original product
 2. gains internet connectivity
 3. cloud support

tuya



* note: i am not associated with tuna



back to the payload

- tuya supports HTTP/HTTPS and MQTT
- device ID and localKey is required to construct/decrypt the payload

2018-05-25 01:29:27 **POST** https://52.89.155.66/api.json?appVersion=2.9.11&appRnVersion=2.91&s75f5c9d61309a371cddf19f13...
← **200 OK** application/json 2.02k 427ms

Request

Response

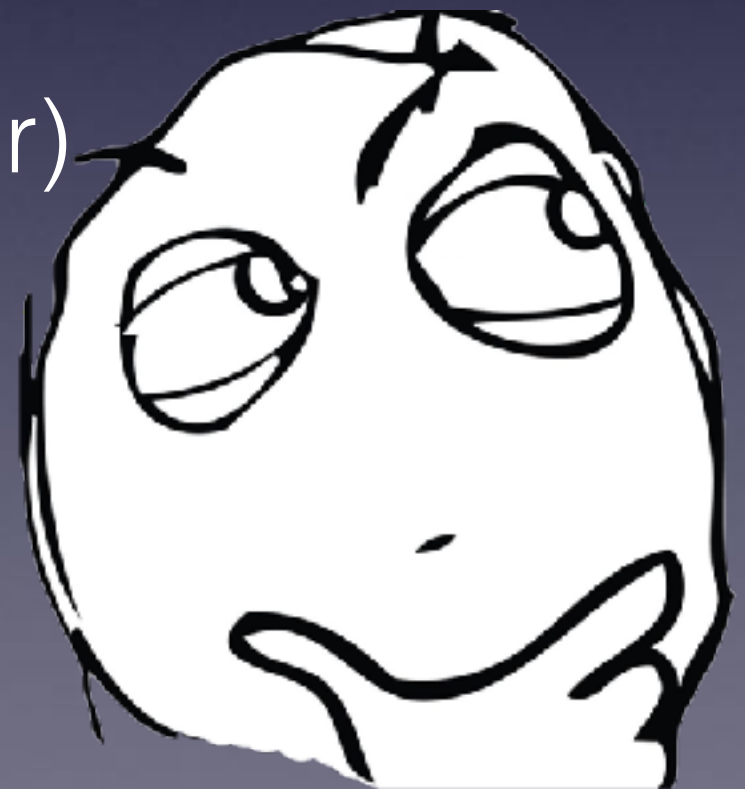
Detail

```
"v": "1.0"
},
{
  "a": "tuya.m.device.my.list",
  "result": [
    {
      "activeTime": 1527175972,
      "devId": "02200490dc4f22161bd0",
      "dpMaxTime": 1527181365357,
      "dps": {
        "1": false,
        "2": 0
      },
      "iconUrl": "https://images.tuyaus.com/smart/icon/1496461963_0.jpeg",
      "ip": "111.255.25.196",
      "isShare": false,
      "lat": "",
      "localKey": "69e85a5b8b16a850",
      "lon": "",
      "moduleMap": {
        "mcu": {
```

← localKey

localKey

is there an another way to ~~steal~~ obtain
this localKey?
(without pwning tuya server)



obtaining localKey

- device sends devID and a password as MQTT username/password
 - not localKey, sadly
- how about pairing process?

obtaining localKey - pairing process, detailed

- phone sends udp broadcast for your ssid/key and `token`
- device then make requests with this `token` in HTTP
 - POST /gw.json?a=s.gw.token.get
 - POST /gw.json?a=s.gw.dev.pk.active
 - here we get our localKey :)
 - POST /gw.json?a=s.gw.update

what's next?

- use tuya SDK?
 - has authorization (per user)
 - need to know a bit of java / android dev
- here comes teh power of GitHub

pytuya

python-tuya

build passing

Python 2.7 and Python 3.6.1 interface to ESP8266MOD WiFi smart devices from Shenzhen Xenon. If you are using the Jinvo Smart App, this allows local control over the LAN. NOTE requires the devices to have already been **activated** by Jinvo Smart App (or similar).

(or forge your own packet)

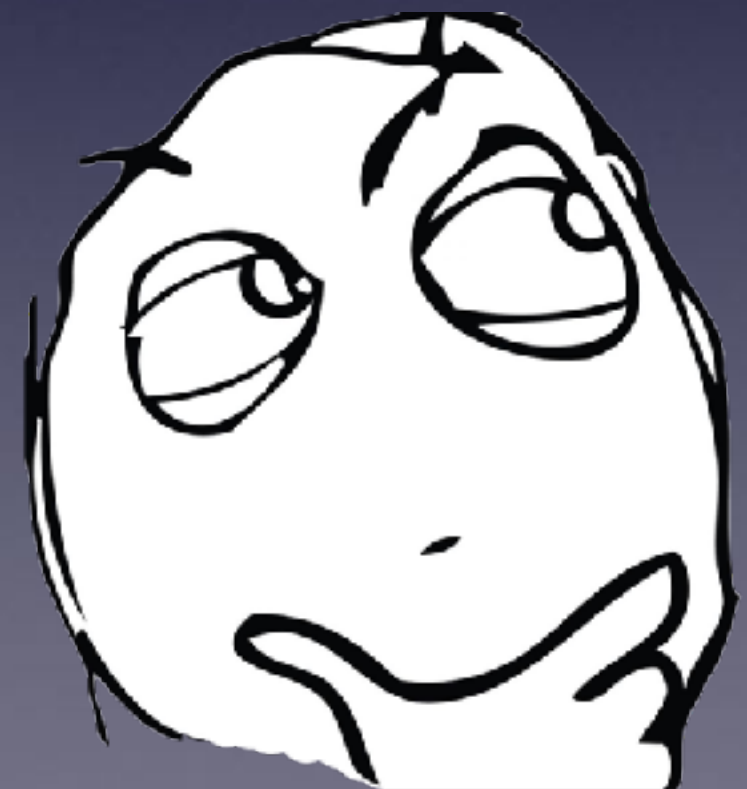
```

payload_dict = {
    "device": {
        "status": {
            "hexByte": "0a",
            "command": {"gwId": "", "devId": ""}
        },
        "set": {
            "hexByte": "07",
            "command": {"devId": "", "uid": "", "t": ""}
        },
        "prefix": "000055aa0000000000000000", # Next byte
        "suffix": "00000000000000aa55"
    }
}

buffer = hex2bin( payload_dict[self.dev_type]['prefix'] +
    payload_dict[self.dev_type][command]['hexByte']
    '000000' +
    postfix_payload_hex_len ) + postfix_payload

```

- this looks like serial
- serial-over-wifi?!



Field	Length (byte)	Description
Frame header	2	0x55aa
Version	1	0x00
Command word	1	0x00
Data length	2	0x0000
Data	0	无None
Checksum	1	Sum by byte from frame header, and the results to be divided by 256 for the remainder

some commands...

- heartbeat
- normal commands (e.g. get state, set state)
- get product info
- get wifi state

some interesting commands...

- heartbeat
- normal commands (e.g. get state, set state)
- get product info
- get wifi state
- reset -> overtake
- start updating -> brick



conclusion

capture

~~death~~

pwn

we shall have a live demo of what we can do

conclusion: what can i do better

- when in doubt (and hungry), go for food
- don't waste too much time on equipment
- learn more about protocol reverse engineering

discussion

- a friend give me one of these....
- <https://garrettmiller.github.io/meross-mss110-vuln/>
- telnet, firmware .bin, etc



even car cams have wifi?



any questions?

es@evsfy.com