#sf21vus

How Smart Are My "Things"?

A Traffic Analysis of IoT Devices

Simone Mainardi, PhD
mainardi@ntop.org

About me

- Simone Mainardi
- Computer Engineer, PhD
- Joined ntop in late 2015
- Used to be a pure data scientist
- Now more close to a software developer

Funding & Sponsorships

- This project has received funding from
  - ntop
  - The European Union's Horizon 2020 research and innovation programme under the NGI_TRUST grant agreement no 825618

ntop

Horizon 2020 Programme
DG CNECT
Next-generation Internet

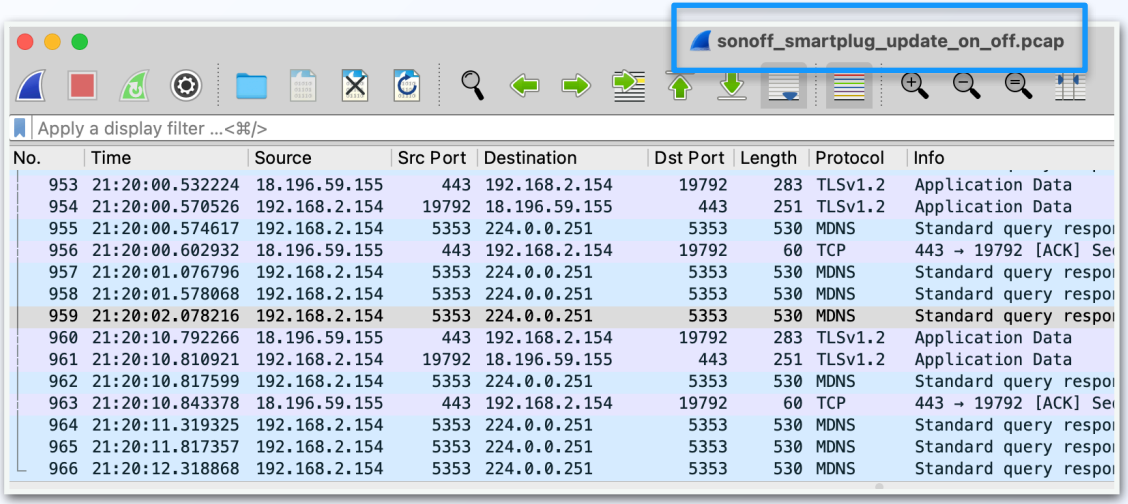
NGI TRUST

Agenda

- Introduction and motivation
- Tracking device network activities
  - "Idle" - Really idle?
  - Operating
  - Over-The-Air (OTA) firmware updates - How do they work? Are they secure?
- Discussion and conclusion

Following Along

- **pcaps** and supplementary material available at https://bit.ly/2X4bwSq
- Screenshots shown during the presentation, look at the filename!
- **"In Depth:"** badges with references to supplementary material



In Depth: File ezviz_cam_private_data_via_udp_sampled.pcap

Introduction

- The Internet of Things (IoT) is quickly becoming a relevant part of our everyday life
- IoT devices are pervasive in industrial environments and are poised to remake our homes
- Security cameras, smart sensors, and light bulbs are just a few examples of IoT devices that we use everyday to control homes, and automate personal or industrial tasks

Motivation

- IoT advantages don’t come for free
- IoT devices directly connected to the networks we use to run our businesses or manage our personal activities at home
- This opens up to a series of implications as IoT devices are in general low-cost, weakly-secured devices able to create shortcuts between the Internet and our private networks

Focus of This Talk

- Focus of this talk is the **network behavior** of a group of IoT devices operated on a testbed network
- More than one month worth of packets is analyzed to investigate their activities in the short and in the long run

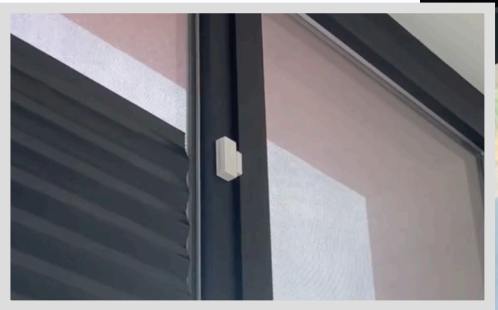
What is this Talk About

- This talk is about the **network behavior** of a set of IoT Devices
  - Contacted hosts (LAN/Internet/cloud)
  - Protocols used to communicate and the extent to which encryption is used
  - Over-The-Air (OTA) firmware updates are performed - and if they are secure
- Emphasis on **security**

What is this Talk NOT About

- This talk is NOT about
  - Exploitation of vulnerabilities
    - Not carrying out attacks, just giving pointers and ideas
  - Reverse engineering
    - Peek into some firmware binaries, but no disassemble-modify-reassemble

The Testbed



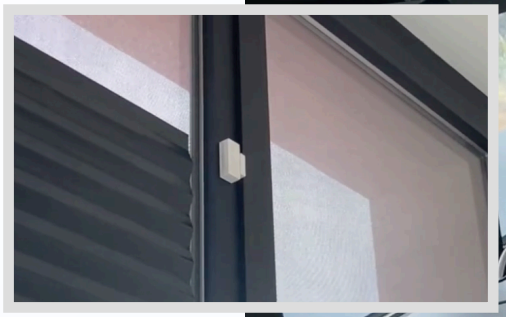
The Testbed: EzViz Security Cameras

The Testbed: TP-Link Smart Plugs

#sf21vus - Material: https://bit.ly/2X4bwSq

The Testbed: Sonoff Smart Plugs

#sf21vus - Material: https://bit.ly/2X4bwSq

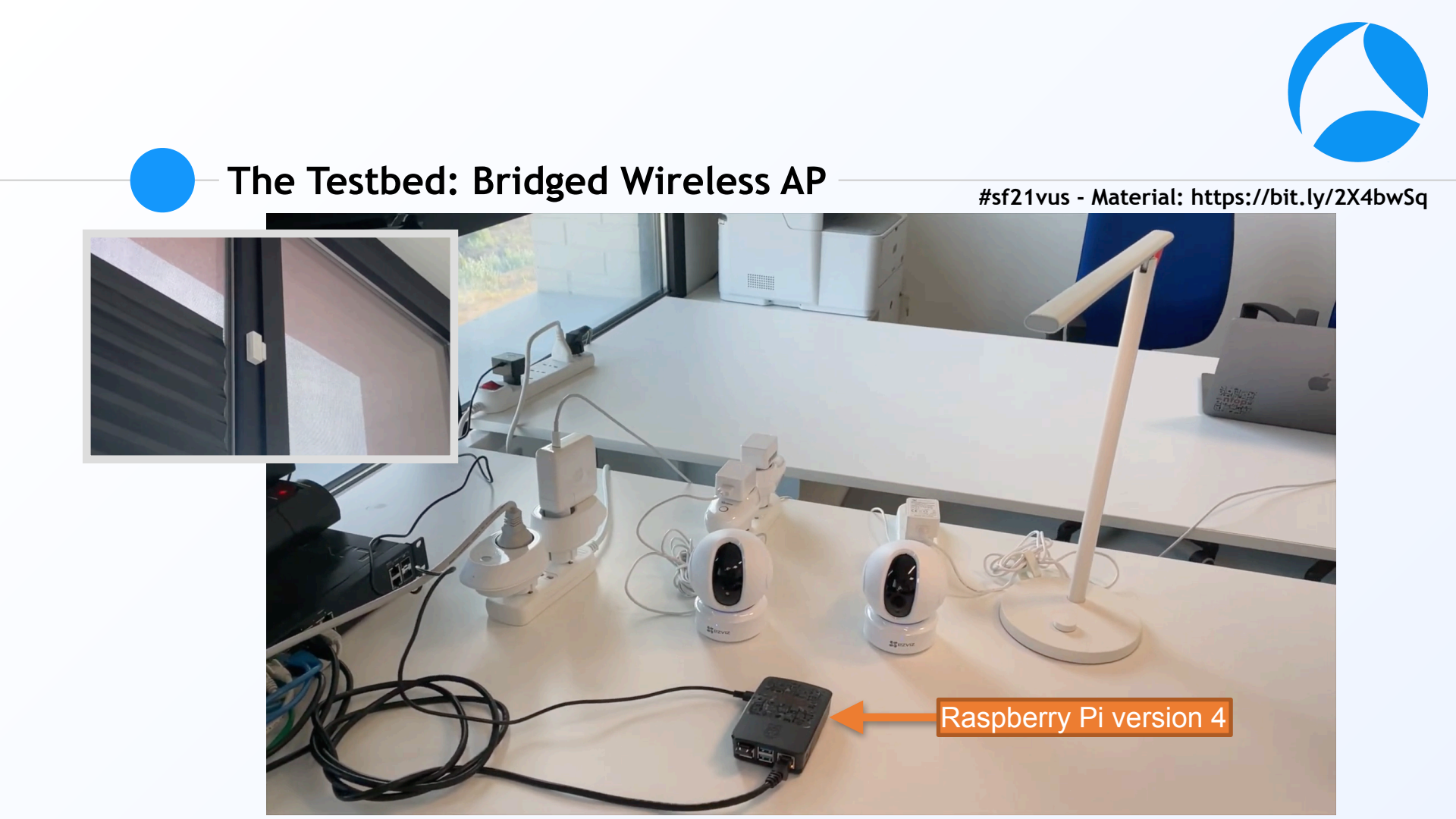


The Testbed: Lamp & Door Sensor

#sf21vus - Material: https://bit.ly/2X4bwSq

Sonoff DW2
Door Sensor

Xiaomi Mi LED Desk Lamp 1S
Smart Lamp



The Testbed: Bridged Wireless AP

#sf21vus - Material: https://bit.ly/2X4bwSq

Raspberry Pi version 4

The Bridged Wireless AP

- Give the testbed **LAN** and **Internet** access **while sniffing** the traffic
- Used a **Raspberry Pi** version 4
- Set up a **Linux bridge** between the devices and the rest of the network - including the Internet
- A Linux bridge is a **Layer-2** software-implemented **network switch**
  - It forwards packets between interfaces that are connected to it
- Packets forwarded through the bridge sniffed with **tcpdump**

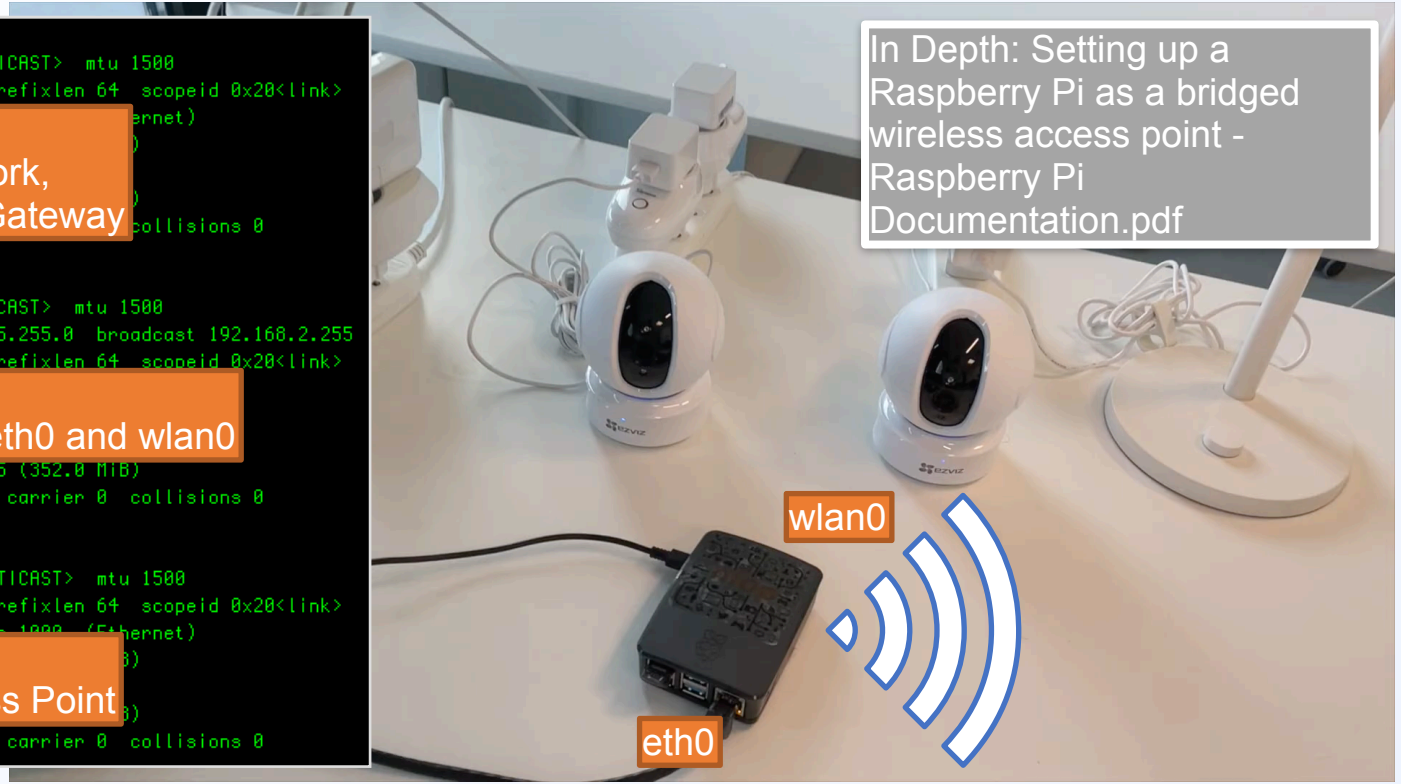
Bridged Wireless AP Interfaces


```
pi@raspberrypi:~ $ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::dea6:32ff:fedf:c0fa  prefixlen 64  scopeid 0x20<link>
                                                    ernet)
                                                    )
                                                    )
                                                    collisions 0

pi@raspberrypi:~ $ ifconfig br0
br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.210  netmask 255.255.255.0  broadcast 192.168.2.255
        inet6 fe80::2ff7:9104:4ffd:734a  prefixlen 64  scopeid 0x20<link>
        TX packets 3036208  bytes 369110015 (352.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pi@raspberrypi:~ $ ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::dea6:32ff:fedf:c0fc  prefixlen 64  scopeid 0x20<link>
                                                    thernet)
                                                    )
                                                    )
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

eth0
To the rest of the network, including the Internet Gateway

br0
Linux bridge between eth0 and wlan0

wlan0
For the Wireless Access Point



Running tcpdump On The Bridged Wireless AP

- Created a **systemd** service to run **tcpdump** and dump **1 pcap / hour**
- A few MBs per pcap with idle devices

```
$ ls -lha | head
total 28G
drwxr-xr-x 3 pi   pi   132K Aug 11 17:34 .
drwxr-xr-x 6 root root 4.0K May  4 13:13 ..
-rw-r--r-- 1 root root 5.3M May  4 15:41 iotdump_2021-05-04__14.pcap
-rw-r--r-- 1 root root 2.9M May  4 16:41 iotdump_2021-05-04__15.pcap
-rw-r--r-- 1 root root 3.2M May  4 17:41 iotdump_2021-05-04__16.pcap
[...]

$ mergecap  -w iotdump_month.pcap iotdump_2021-0*
```

In Depth: Appendix of these slides

Tracking Devices “Idling”

- Run for 5 days without any interaction
  - No app use, no video streaming, …
- Quantify the traffic generated by devices just to stay **idle**

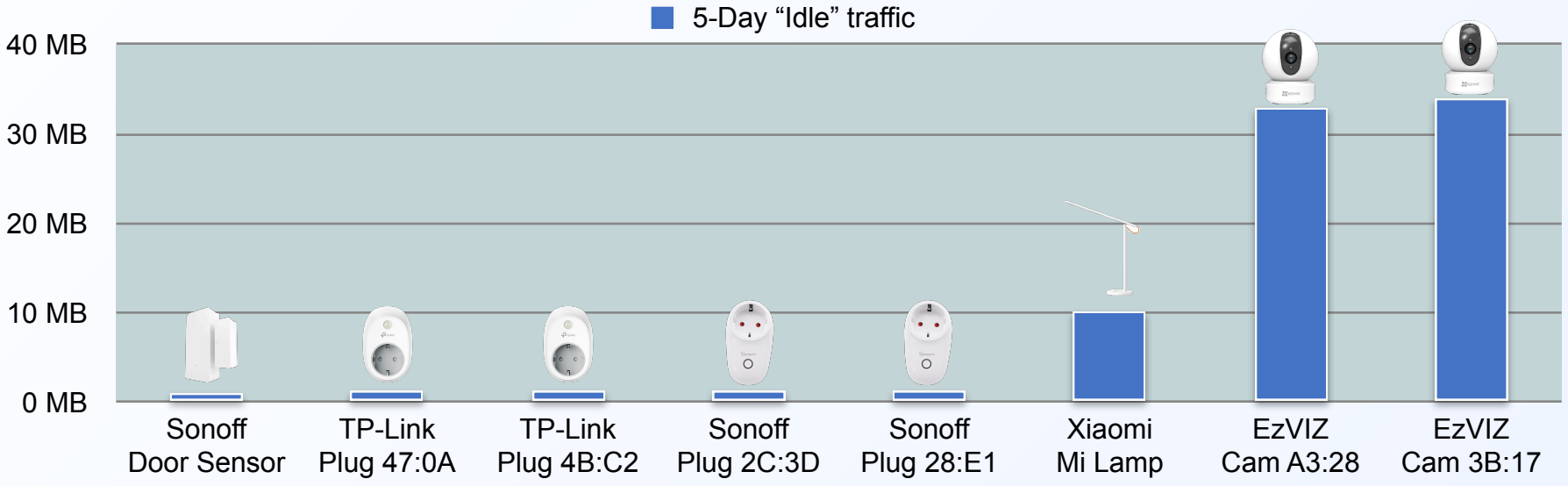
Devices “Idling” Traffic Volume

- Use Wireshark **capinfos** CLI tool
- Idle devices generate ~ **0.7 MB / hour**

```
$ capinfos week_idle_iot.pcap
File name:           week_idle_iot.pcap
[…]
Number of packets:   760 k
File size:           96 MB
Data size:           84 MB
Capture duration:    431994.169478 second
First packet time:   2021-05-06 00:41:13.744666
Last packet time:    2021-05-11 00:41:07.914144
Data byte rate:      194 bytes/s
Data bit rate:       1558 bits/s
Average packet size: 110.64 bytes
Average packet rate: 1 packets/s
```

Who's The Most Chatty when "Idle"

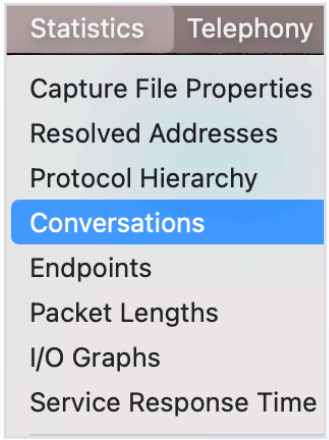
- All devices doing **less than 2 MB** except for the lamp and the two cameras

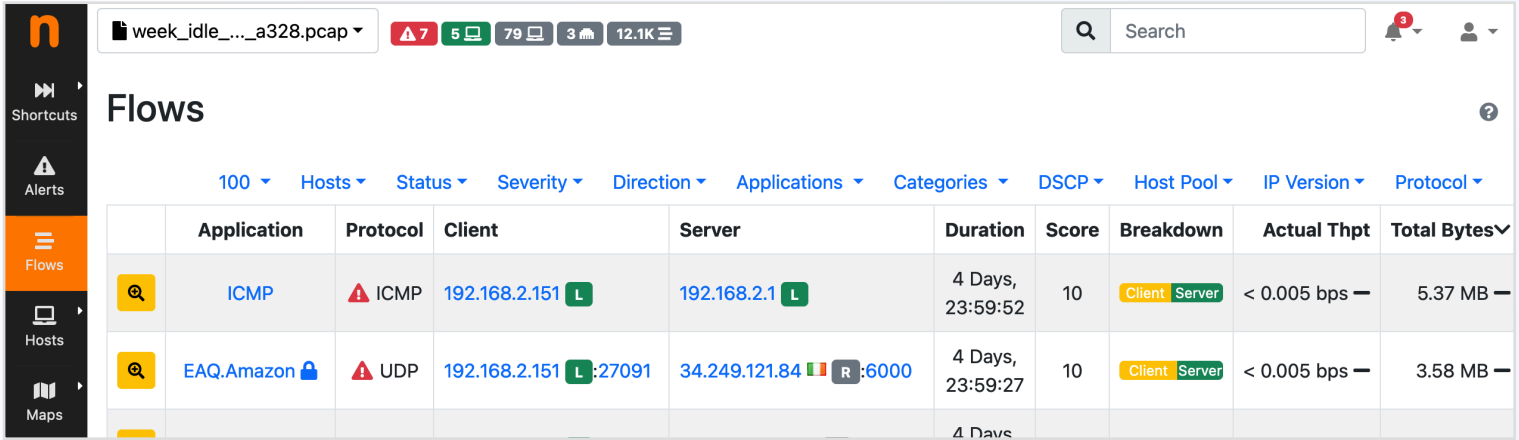


EzVIZ Cam: What's Inside 30+ MB of "idle" Traffic

#sf21vus - Material: https://bit.ly/2X4bwSq

- Seems EzVIZ cameras are not that idle
- Traffic analysis with **Wireshark** and **ntopng**





EzVIZ Cam: “Idle” LAN Traffic

#sf21vus - Material: https://bit.ly/2X4bwSq

D-Link NOT in the testbed!

week_idle_..._a328.pcap | 7 | 5 | 79 | 3 | 12.1K | Search

Shortcuts | Alerts | Flows | Hosts | Maps | Interface | Settings | Developer | Help

All Hosts

Significant traffic to other hosts in the LAN

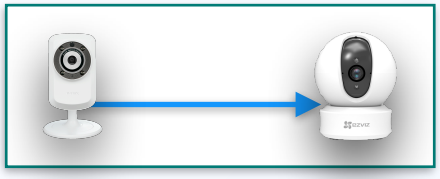
100 | IP Versio

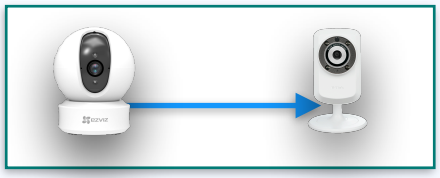
IP Address	Flows	MAC Address	Name	Seen Since	Breakdown	Throughput	Total Bytes
192.168.2.151 L	12089	80:9F:9B:45:A3:28	e25105525	100 Days, 14:42:16	Sent Rcvd	0.0 bit/s	32.73 MB
192.168.2.44 L	6446	D-LinkIn_1C:34:2C		100 Days, 14:40:32	Sent Rcvd	0 bit/s	7.09 MB
192.168.2.1 L	239	Ubiquiti_06:B3:5A		100 Days, 14:42:16	Sent Rcvd	0.0 bit/s	5.44 MB
162.62.52.181 R	685	Ubiquiti_06:B3:5A		100 Days, 14:35:28	Se Rcvd	0 bit/s	5.31 MB
34.249.121.84 R	1	Ubiquiti_06:B3:5A		100 Days, 14:41:57	Sent Rcvd	0.0 bit/s	3.58 MB
34.254.0.145 R	1	Ubiquiti_06:B3:5A		100 Days, 14:41:57	Sent Rcvd	0.0 bit/s	3.51 MB
52.16.156.56 R	2	Ubiquiti_06:B3:5A		100 Days, 14:42:06	Sent Rcvd	0.0 bit/s	2.42 MB
239.255.255.250 M	3136	IPv4mcast_7F:FF:FA		100 Days, 14:40:38	Rcvd	0 bit/s	2.37 MB

EzVIZ Cam: Lateral Movements

#sf21vus - Material: https://bit.ly/2X4bwSq

1. The EzVIZ Cam uses **SSDP** to discover services in the LAN
2. Another D-Link Cam in the LAN responds
3. The EzVIZ Cam uses **HTTP** to fetch the service description from the D-Link Cam





week_idle_ezviz_cam_a328.pcap

Source	Src Port	Destination	Dst Port	Length	Protocol	Info
192.168.2.151	58002	239.255.255.250	1900	179	SSDP	M-SEARCH * HTTP/1.1
192.168.2.151	58002	239.255.255.250	1900	174	SSDP	M-SEARCH * HTTP/1.1

week_idle_ezviz_cam_a328.pcap

2.168.2.44 && !(tcp.port==8815)

Source	Src Port	Destination	Dst Port	Length	Protocol	Info
192.168.2.44	3075	192.168.2.151	58002	261	SSDP	HTTP/1.1 200 OK
192.168.2.44	3075	192.168.2.151	45760	261	SSDP	HTTP/1.1 200 OK

week_idle_ezviz_cam_a328.pcap

ddr==192.168.2.44

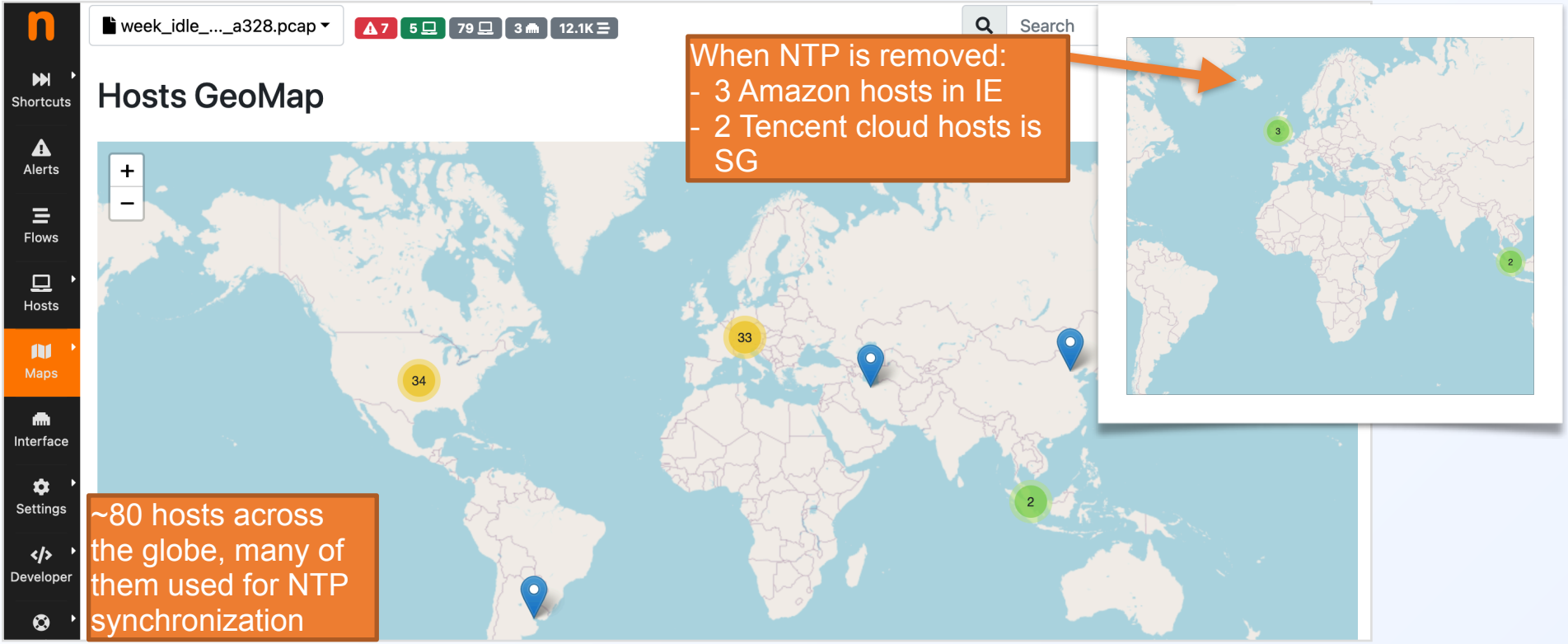
Source	Src Port	Destination	Dst Port	Length	Protocol	Info
192.168.2.151	43412	192.168.2.44	8815	191	HTTP	GET /rootdesc.xml HTTP/1.1
192.168.2.44	8815	192.168.2.151	43412	1039	HTTP/XML	HTTP/1.1 200 OK
192.168.2.151	43413	192.168.2.44	8815	191	HTTP	GET /rootdesc.xml HTTP/1.1
192.168.2.44	8815	192.168.2.151	43413	1039	HTTP/XML	HTTP/1.1 200 OK

EzVIZ Cam: Lateral Movements Implications

- The EzVIZ cam discovers services in the network and **moves laterally by default** without nothing explicitly configured
- An attacker can respond to SSDP, advertise fake services and expect the EzVIZ camera to connect with HTTP requests

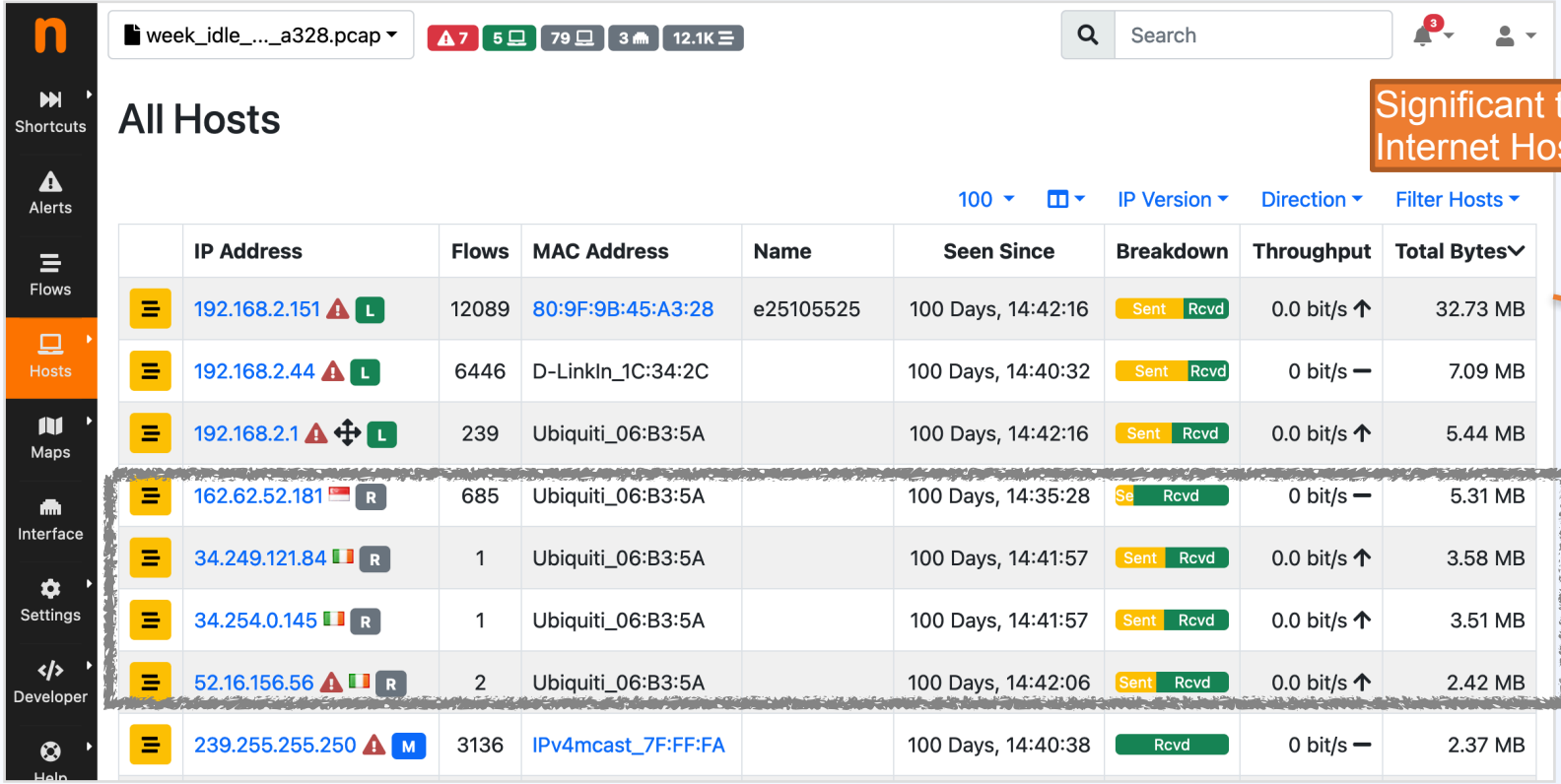
EzVIZ Cam: Geography of Internet Hosts

#sf21vus - Material: https://bit.ly/2X4bwSq



EzVIZ Cam: “Idle” Internet Traffic

#sf21vus - Material: https://bit.ly/2X4bwSq



EzVIZ Cam: Top Internet Talker

- The EzVIZ cam exchanges **5.31 MB** of **UDP** traffic with **Tencent Cloud** host 162.62.52.181
  - Host geolocated in **Singapore**
- Although the application protocol is unknown, **private data is transmitted in cleartext**…

EzVIZ Cam: Cleartext Private Data

#sf21vus - Material: https://bit.ly/2X4bwSq

EzVIZ Cam: Other Internet Talkers

- With similar analyses it can be found that the EzVIZ cam exchanges unencrypted data also with other Internet talkers
- Among the protocols used are MQTT, HTTP
- Found also an HTTP backup

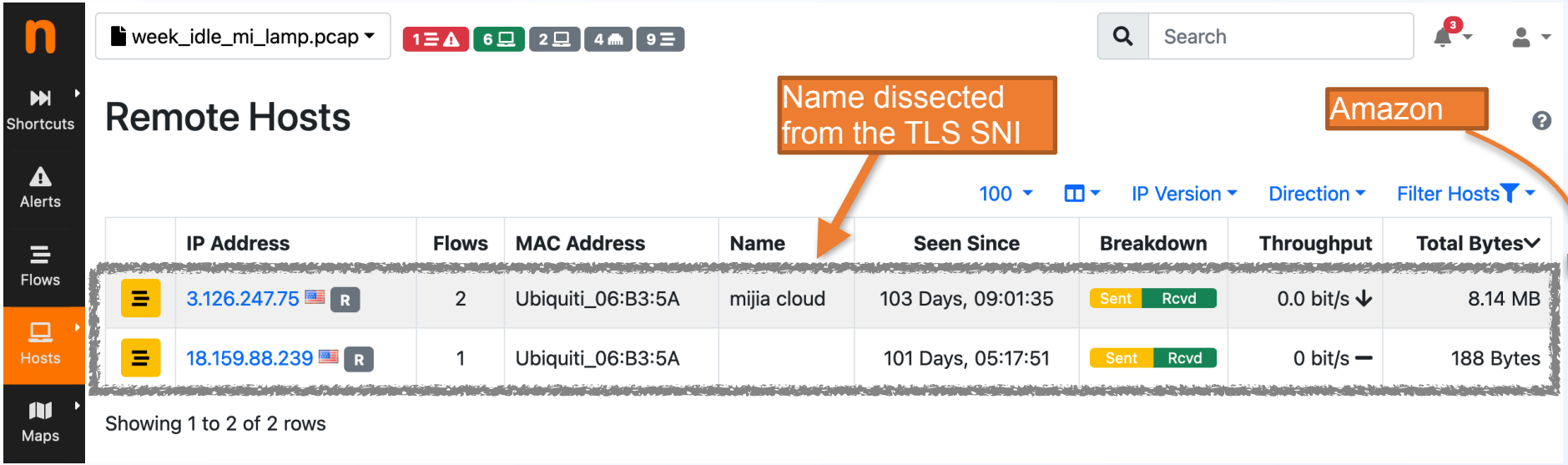
Flows
Hosts
Maps

	Application	Protocol	Client	Server	Duration	Score	Breakdown	Actual Thpt	Total Bytes	Info
	HTTP.Amazon	TCP	192.168.2.151 L:46074	34.253.31.111 R:http-alt	< 1 sec	10	Client	0 bps	42.4 KB	backupserver:8080/sdk.po...

In Depth: File ezviz_cam_http_backupserver.pcap

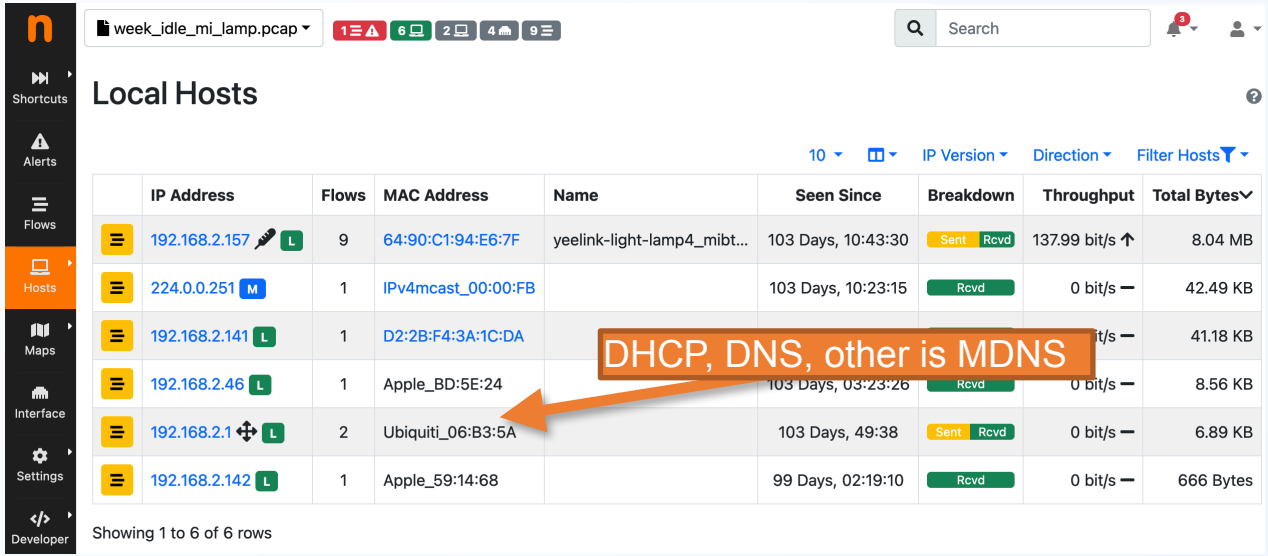
Mi Lamp: “Idle” Internet Traffic

- Internet traffic to Amazon, **mostly TLS** and a couple of Unknown UDP packets



Mi Lamp: “Idle” LAN Traffic

- Mostly ARP, also several MDNS responses (NO discovery), DHCP, DNS



Other Devices: “Idle” Traffic

- Activity significantly reduced
  - A bunch of **TLS** sessions with **Amazon**
  - Some **service traffic** with the gateway (DNS, DHCP)
  - TP-Link Smart Plug syncs with ~90 **NTP** hosts worldwide

In Depth: Appendix of these slides

Tracking Devices “Idling”: Conclusions

#sf21vus - Material: https://bit.ly/2X4bwSq

	Cloud Services	Cloud Countries	NTP Sync	Lateral Movements
EzVIZ Cameras	Amazon (EAQ + MQTT w/ cleartext) Tencent Cloud (Unknown w/ cleartext)		✓	Service Discovery (SSDP + HTTP)
Xiaomi Smart Lamp	Amazon (TLS + Unknown)			
Sonoff Door Sensor	Amazon (TLS)			
Sonoff Smart Plugs	Amazon (TLS)			
Tp-Link Smart Plugs	Amazon (TLS)		✓	

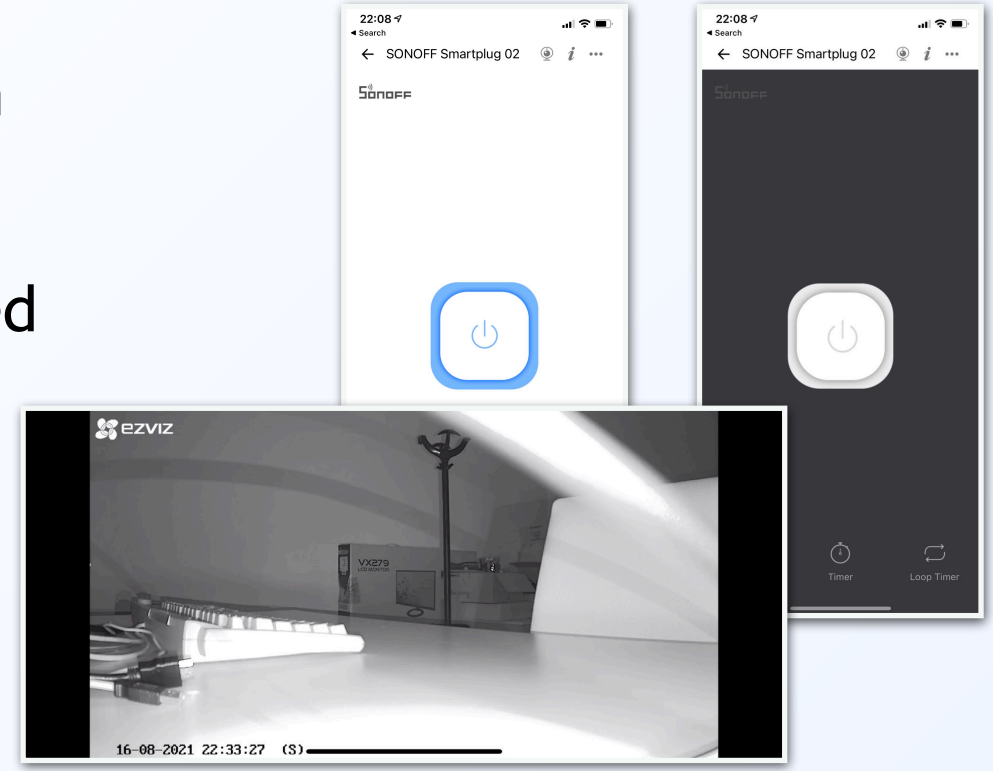
Tracking Devices “Idling”: Conclusions

- All devices rely on **Amazon cloud services**
  - EzVIZ cameras with EAQ and MQTT
  - All other devices initiate and keep long-lived TLS connections open
- EzVIZ cameras and TP-Link plugs contacts **NTP servers**
  - 80 / 90 NTP servers geographically distributed
- EzVIZ cameras **move laterally** in the **LAN**
  - SSDP to discover services
  - HTTP to discovered services

Tracking Devices “Operating” Activities

#sf21vus - Material: https://bit.ly/2X4bwSq

- What happens when
  - A plug is toggled on/off
  - A lamp is switched on/off
  - A door is open/ closed
  - A video stream is open



Sonoff Smart Plug: Switching it On/Off

#sf21vus - Material: https://bit.ly/2X4bwSq

- On/Off exhibit a symmetric pattern
  - 3 **TLS** packets + 4 Unsolicited **MDNS** responses

sonoff_smartplug_update_on_off.pcap

Apply a display filter ...<⌘/>

HTTP > .5

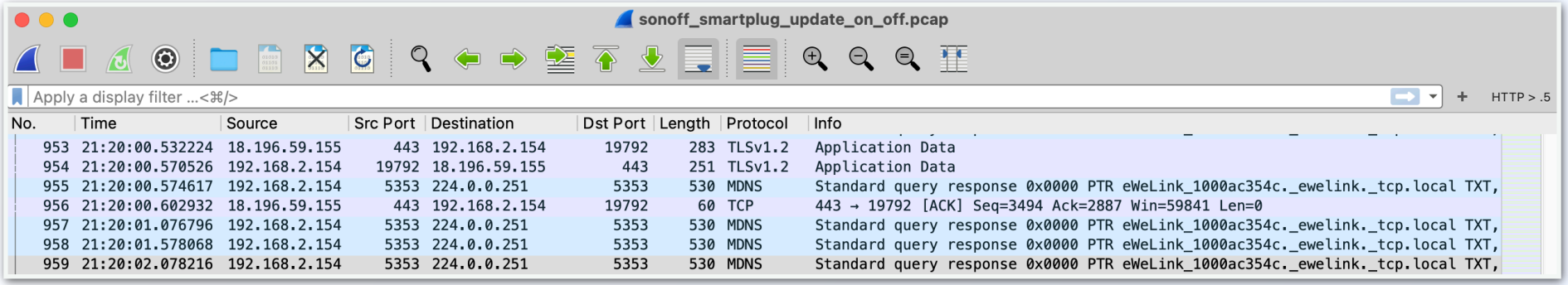
No.	Time	Source	Src Port	Destination	Dst Port	Length	Protocol	Info
953	21:20:00.532224	18.196.59.155	443	192.168.2.154	19792	283	TLSv1.2	Application Data
954	21:20:00.570526	192.168.2.154	19792	18.196.59.155	443	251	TLSv1.2	Application Data
955	21:20:00.574617	192.168.2.154	5353	224.0.0.251	5353	530	MDNS	Standard query response 0x0000 PTR eWeLink_1000ac354c._ewelink._tcp.local TXT,
956	21:20:00.602932	18.196.59.155	443	192.168.2.154	19792	60	TCP	443 → 19792 [ACK] Seq=3494 Ack=2887 Win=59841 Len=0
957	21:20:01.076796	192.168.2.154	5353	224.0.0.251	5353	530	MDNS	Standard query response 0x0000 PTR eWeLink_1000ac354c._ewelink._tcp.local TXT,
958	21:20:01.578068	192.168.2.154	5353	224.0.0.251	5353	530	MDNS	Standard query response 0x0000 PTR eWeLink_1000ac354c._ewelink._tcp.local TXT,
959	21:20:02.078216	192.168.2.154	5353	224.0.0.251	5353	530	MDNS	Standard query response 0x0000 PTR eWeLink_1000ac354c._ewelink._tcp.local TXT,
960	21:20:10.792266	18.196.59.155	443	192.168.2.154	19792	283	TLSv1.2	Application Data
961	21:20:10.810921	192.168.2.154	19792	18.196.59.155	443	251	TLSv1.2	Application Data
962	21:20:10.817599	192.168.2.154	5353	224.0.0.251	5353	530	MDNS	Standard query response 0x0000 PTR eWeLink_1000ac354c._ewelink._tcp.local TXT,
963	21:20:10.843378	18.196.59.155	443	192.168.2.154	19792	60	TCP	443 → 19792 [ACK] Seq=3723 Ack=3084 Win=59644 Len=0
964	21:20:11.319325	192.168.2.154	5353	224.0.0.251	5353	530	MDNS	Standard query response 0x0000 PTR eWeLink_1000ac354c._ewelink._tcp.local TXT,
965	21:20:11.817357	192.168.2.154	5353	224.0.0.251	5353	530	MDNS	Standard query response 0x0000 PTR eWeLink_1000ac354c._ewelink._tcp.local TXT,
966	21:20:12.318868	192.168.2.154	5353	224.0.0.251	5353	530	MDNS	Standard query response 0x0000 PTR eWeLink_1000ac354c._ewelink._tcp.local TXT, cache

OFF

ON

Sonoff Smart Plug: On/Off TLS

- TLS with an **Amazon** host (existing conn.)
  1. Amazon tells something to the plug
  2. The smart plug responds
  3. Amazon ACKs



Sonoff Smart Plug: On/Off MDNS

#sf21vus - Material: https://bit.ly/2X4bwSq

- 3 unsolicited MDNS responses
  - Contain encoded (base64) data

sonoff_smartplug_update_on_off.pcap

Apply a display filter ...<⌘/>

No.	Time	Source	Src Port	Destination	Dst Port	Length	Protocol	Info
953	21:20:00.532224	18.196.59.155	443	192.168.2.154	19792	283	TLSv1.2	Application Data
954	21:20:00.570526	192.168.2.154	19792	18.196.59.155	443	251	TLSv1.2	Application Data
955	21:20:00.574617	192.168.2.154	5353	224.0.0.251	5353	530	MDNS	Standard query response 0x0000 PTR eWeLink_1000ac354c._ewelink._tcp.local TXT, cache

```
TXT: id=1000ac354c
TXT Length: 9
TXT: type=plug
TXT Length: 9
TXT: apivers=1
TXT Length: 5
TXT: seq=4
TXT Length: 12
TXT: encrypt=true
TXT Length: 27
TXT: iv=MTAwMTQ0NzQ0ODM4MDMzMw==
TXT Length: 198
TXT: data1=QX++6...a1ALnU/6v5gNxAUIq0IMMc4mCal5tbSJRrdqbTMC4MGKsr6e1q5sEpskwQg3gB22KRVM1x6P021m4riNyB8nxziDAx+mH4n5xtFxUyJLNR3kQhyaLgnGzL1Ii8KFFXT/PT9EcnFs+rTIs9B
```

iv = Initialization Vector?
base64 decoded: 1001447448380333

data1= encrypted data?

EzVIZ Cam: Video Stream

- An **MQTT Publish** message arrives from **Amazon**
- The **camera ACKs** the message and start sending data on another unknown (encrypted?) TCP stream

((frame.number >= 2438 && frame.number <= 2458) || (frame.number >= 4044 && frame.number <= 4064)) || (tcp.stream == 3)

MQTT

No.	Time	Source	Src Port	Destination	Dst Port	Length	Protocol	Info
24…	18:01:19.136761	54.195.28.223	31006	192.168.2.151	42533	892	MQTT	Publish Message [/E25105525/3100/12545]
24…	18:01:19.176513	192.168.2.151	42533	54.195.28.223	31006	66	TCP	42533 → 31006 [ACK] Seq=24063 Ack=4319 Win=22690 Len=0 TSval=4294951690 TSecr=1945619951
24…	18:01:19.252724	49.51.168.13	7760	192.168.2.151	58828	66	TCP	7760 → 58828 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1424 SACK_PERM=1 WS=512
24…	18:01:19.254948	192.168.2.151	58828	49.51.168.13	7760	54	TCP	58828 → 7760 [ACK] Seq=1 Ack=1 Win=14000 Len=0
24…	18:01:19.257986	192.168.2.151	58828	49.51.168.13	7760	122	TCP	58828 → 7760 [PSH, ACK] Seq=1 Ack=1 Win=14000 Len=68
24…	18:01:19.289182	49.51.168.13	7760	192.168.2.151	58828	60	TCP	7760 → 58828 [ACK] Seq=1 Ack=69 Win=14848 Len=0
24…	18:01:19.313152	192.168.2.151	42533	54.195.28.223	31006	354	MQTT	Publish Message [/3100/12546]
24…	18:01:19.343015	192.168.2.151	58828	49.51.168.13	7760	1454	TCP	58828 → 7760 [ACK] Seq=69 Ack=1 Win=14000 Len=1400
24…	18:01:19.343384	192.168.2.151	58828	49.51.168.13	7760	62	TCP	58828 → 7760 [PSH, ACK] Seq=1469 Ack=1 Win=14000 Len=8
24…	18:01:19.343845	192.168.2.151	58828	49.51.168.13	7760	1454	TCP	58828 → 7760 [ACK] Seq=1477 Ack=1 Win=14000 Len=1400
24…	18:01:19.343904	192.168.2.151	58828	49.51.168.13	7760	1454	TCP	58828 → 7760 [ACK] Seq=2877 Ack=1 Win=14000 Len=1400
24…	18:01:19.347004	192.168.2.151	58828	49.51.168.13	7760	1454	TCP	58828 → 7760 [ACK] Seq=4277 Ack=1 Win=14000 Len=1400
24…	18:01:19.347062	192.168.2.151	58828	49.51.168.13	7760	1454	TCP	58828 → 7760 [ACK] Seq=5677 Ack=1 Win=14000 Len=1400

In Depth: File ezviz_cam_mqtt_video_stream.pcap

Other Devices: “Operating” Activities

- All exhibiting an almost identical pattern
- Packet exchange 1 or 2 Amazon hosts over already existing connections

In Depth: Appendix of these slides

Tracking Devices “Operating” Activities: Conclusions

#sf21vus - Material: https://bit.ly/2X4bwSq

	Cloud Services	Cloud Countries	First Packet	LAN
EzVIZ Cameras Video Stream	Amazon (MQTT w/ cleartext) Tencent Cloud (Unknown)	🇮🇪 🇩🇪	Cloud→Device	
Xiaomi Smart Lamp On/Off	Amazon (TLS)	🇩🇪	Cloud→Device	
Sonoff Door Sensor Open/Close	Amazon (TLS)	🇩🇪	Device→Cloud	
Sonoff Smart Plugs On/Off	Amazon (TLS)	🇩🇪	Cloud→Device	Unsolicited MDNS
Tp-Link Smart Plugs On/Off	Amazon (TLS)	🇺🇸	Cloud→Device	

- All devices rely on already-open sessions with **Amazon** hosts
- All devices use **TLS** except for the cameras
  - Cameras use MQTT and another UDP unknown communication likely transporting the actual Video
- **Unsolicited MDNS** generated by the Sonoff Smart Plugs

Over-The-Air Updates

- Keeping devices up to date is fundamental
  - Updates carry **new features** but also **security fixes**
- Generally, the only way for an end-user to keep IoT devices up to date is to let them fetch firmware and software **Over-The-Air** (OTA)
  - **Wirelessly** (as in the case of the testbed)
  - Directly **from the Internet**, without the user having to connected the device to a computer via USB, or plugging SD cards, etc.

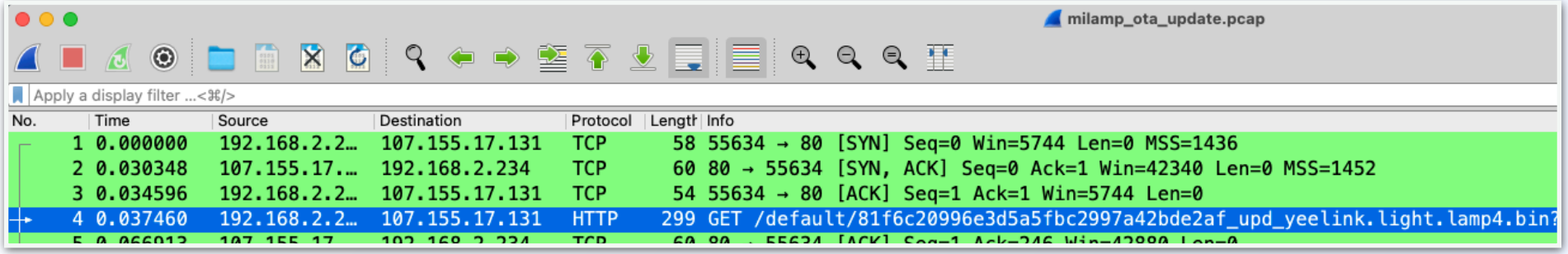
Over-The-Air Updates: Edge-To-Cloud

- All IoT devices in the testbed fetch their updates straight from the **cloud**
- This update mechanism is also known as **Edge-To-Cloud OTA Update**
- The IoT device on the **Edge** of the network acts as an **Update** dispatcher and processor

In Depth: OTA Architectures for IoT Devices.pdf

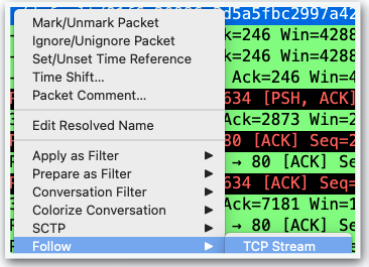
Mi Lamp: Edge-To-Cloud OTA Updates

- Mi Lamp updated from the Mi Home IoS app
- The lamp (192.68.2.234) has reached a server (107.155.17.131) in the **cloud** via **HTTP**
- Server belongs to Zenlayer's edge cloud services and is located in France



Mi Lamp: HTTP Edge-To-Cloud OTA Update

#sf21vus - Material: https://bit.ly/2X4bwSq



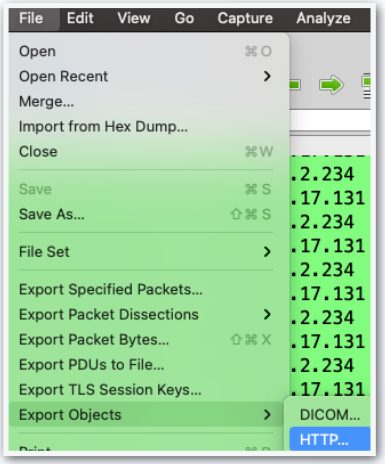
Wireshark · Follow TCP Stream (tcp.stream eq 0) · milamp_ota_update.pcap

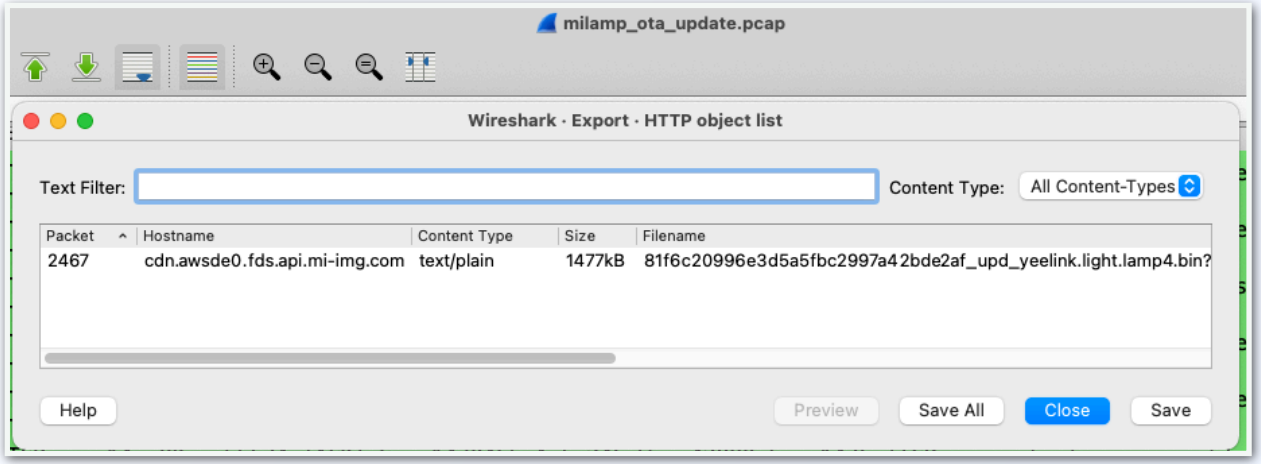
```
GET /default/81f6c20996e3d5a5fbc2997a42bde2af_upd_yeelink.light.lamp4.bin?
GalaxyAccessKeyId=5721718224520&Expires=9223372036854775807&Signature=Bk0xVkxXGqpGrrLwczFXyfIHpSY=&uniqRequestId=40995047 HTTP/1.1
host: cdn.awsde0.fds.api.mi-img.com

HTTP/1.1 200 OK
Date: Thu, 29 Apr 2021 08:01:37 GMT
Content-Type: text/plain
Content-Length: 1477604
Connection: keep-alive
Server: nginx
Last-Modified: Sun, 21 Mar 2021 01:59:33 GMT
x-xiaomi-meta-content-length: 1477604
ETag: "81f6c20996e3d5a5fbc2997a42bde2af"
Content-MD5: 81f6c20996e3d5a5fbc2997a42bde2af
x-xiaomi-hash-crc64ecma: -5868401979149786662
x-xiaomi-server-side-encryption: AES256
Cache-Control: max-age=86400
x-xiaomi-request-id: 986a8ae2-b5dc-0150-0000-01791ca5c713
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: content-md5, upload-time, x-xiaomi-meta-content-length
X-Ser: BC75_US-California-santa-clara-1-cache-2, BC130_FR-Paris-Paris-1-cache-3
X-Cache: MISS from BC130_FR-Paris-Paris-1-cache-3(baishan)

... d..@.............. .@?....2T.............ac38ad7.........................miio_app.......................09:54:38........Mar 21
2021.....ac38ad7.......................... ...`Fo5%Y6eF.`h&.^..HB.T:..>.
+.....................................................................esp_task_wdt_init(CONFIG_ESP_TASK_WDT_TIMEOUT_S, false)./home/
auto_build/ylk_auto_build/build_dir/esp32_mi2x/esp-idf/components/esp32/cpu_start.c.esp_task_wdt_add(idle_0)./dev/uart/0.err == ESP_OK && "Failed
to init pthread module!".res == pdTRUE.....................main_task.start_cpu0_default.clk..[0;33mW (%d) %s: 32 kHz XTAL not found, switching
to internal 150 kHz oscillator.[0m
.rtc_clk_xtal_freq_get() != RTC_XTAL_FREQ_AUTO./home/auto_build/ylk_auto_build/build_dir/esp32_mi2x/esp-idf/components/esp32/
clk.c.res.......esp_clk_init.core_id<portNUM_PROCESSORS./home/auto_build/ylk_auto_build/build_dir/esp32_mi2x/esp-idf/components/esp32/
crosscore_int.c.err == ESP_OK.esp_crosscore_int_send.esp_crosscore_int_init.!((vd->flags&VECDESC_FL_SHARED)&&(vd->flags&VECDESC_FL_NONSHARED))./
home/auto_build/ylk_auto_build/build_dir/esp32_mi2x/esp-idf/components/esp32/intr_alloc.c.svd !=
```

Packet 6.1 client pkt, 1, 029 server pkts, 1 turn. Click to select.

Mi Lamp: Extracting the .bin File

- To extract the .bin file with Wireshark (Make sure to have "Allow subdissector to reassemble TCP stream")





Mi Lamp: Peeking into the .bin File with file

- Utility **file** tells the .bin **DOS Executable**, also known as **COM**

```
Simones-Mac-mini:Downloads simone$ file 81f6c20996e3d5a5fbc2997a42bde2af_upd_yeelink.light.lamp4.bin
81f6c20996e3d5a5fbc2997a42bde2af_upd_yeelink.light.lamp4.bin: DOS executable (COM)
```

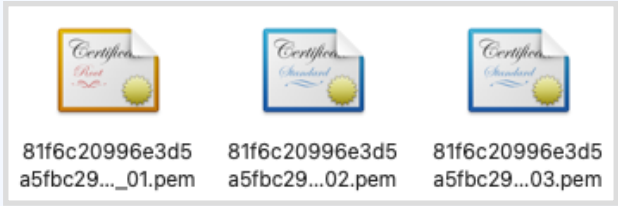
- Chances are the .bin file is a **binary firmware image** composed of multiple pieces
- Utility **binwalk** to identify embedded code, files, ...

```
$ binwalk 81f6c20996e3d5a5fbc2997a42bde2af_upd_yeelink.light.lamp4.bin | grep -v "Unix
path"

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
66012         0x101DC         CRC32 polynomial table, little endian
69924         0x11124         PEM certificate
78856         0x13408         PEM certificate
95468         0x174EC         PEM certificate
158652        0x26BBC         SHA256 hash constants, little endian
208024        0x32C98         Neighborly text, "neighbor entry"
212481        0x33E01         PEM certificate
218304        0x354C0         Base64 standard index table
224876        0x36E6C         SHA256 hash constants, little endian
```

PEM-encoded TLS certificates

Mi Lamp: TLS Certificates

- Three TLS certificates extracted (the fourth cannot be decoded)
  - Two intermediate
  - One **root**
- Very long validity periods
- Likely used in the IoT device chains of trust

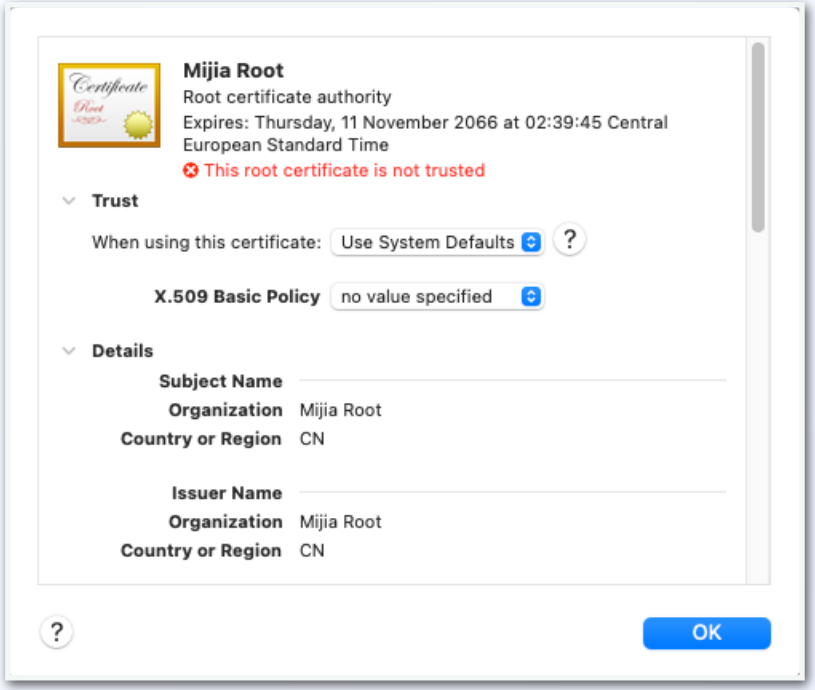


In Depth: Files 81f6c20996e3d5a5fbc2997a42bde2af_upd_yeelink.light.lamp4.bin.certificate_{01,02,03}.pem

Mi Lamp: Root TLS Certificate

#sf21vus - Material: https://bit.ly/2X4bwSq

- A root certificate is trusted because it is assumed to be delivered **by some trustworthy procedure**
  - Delivered to the device via HTTP!
- Untrusted
- Unknown issuer **Mijia Root**



In Depth: Files 81f6c20996e3d5a5fbc2997a42bde2af_upd_yeelink.light.lamp4.bin.certificate_01.pem

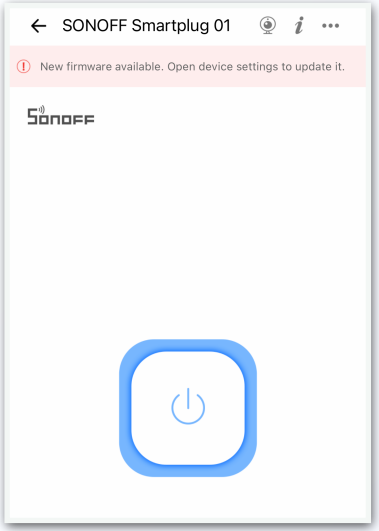
- Search for IP addresses, domain names, etc.

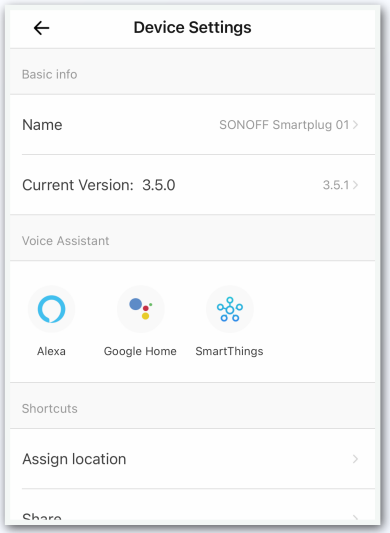
```
Simones-Mac-mini:Downloads simone$ strings
81f6c20996e3d5a5fbc2997a42bde2af_upd_yeelink.light.lamp4.bin | egrep -ri '\.com'
(standard input):ot.io.mi.com
(standard input):cloud.yeelight.com
(standard input):ot.io.mi.com
(standard input):ots.io.mi.com
(standard input):otc.io.mi.com
(standard input):dlg.io.mi.com
(standard input):http://dlg.io.mi.com/v1/ot/upload
(standard input):dns.io.mi.com
(standard input):dk.io.mi.com
Simones-Mac-mini:Downloads simone$ strings
81f6c20996e3d5a5fbc2997a42bde2af_upd_yeelink.light.lamp4.bin | egrep -ri '^(?:[0-9]{1,3}\.)
{3}[0-9]{1,3}$'
(standard input):0.0.0.0
(standard input):110.43.0.83
(standard input):110.43.0.85
(standard input):127.0.0.1
```

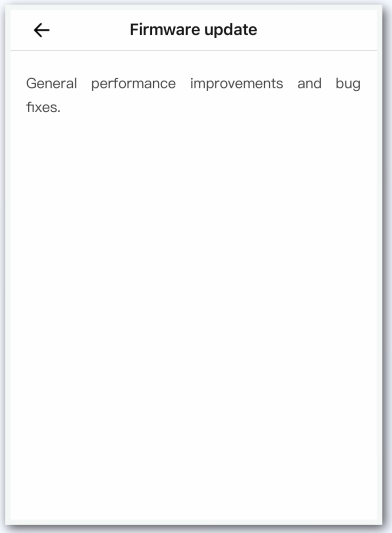
Sonoff Devices: Edge-To-Cloud OTA Updates

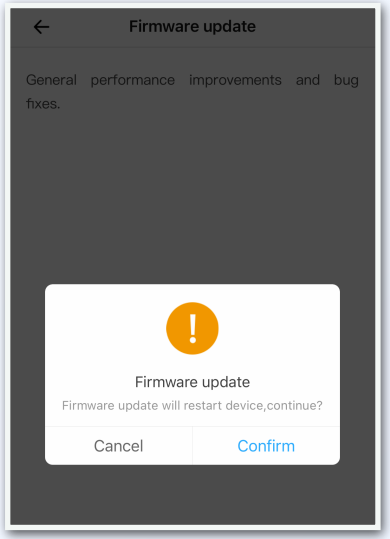
#sf21vus - Material: https://bit.ly/2X4bwSq

- Sonoff devices updated manually from the IoS app









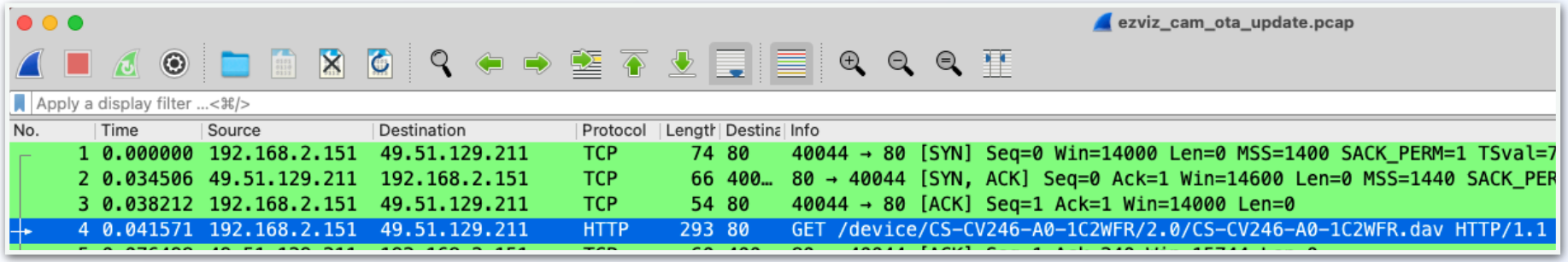
- Very similar behavior to the one observed with the Mi Lamp
- Firmware sent over plain HTTP
- **Root certificates** found for both the smart plug and the door sensor
- Seemingly a **private key** carried with the smart plug firmware

```
$ binwalk  user2.1024.new.2.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------
[...]
440123        0x6B73B         PEM RSA private key
441134        0x6BB2E         PEM certificate
[...]
```

In Depth: Appendix of these slides

EzVIZ Cam: Edge-To-Cloud OTA Updates

- EzVIZ Camera updated manually from the eWeLink app
- The camera (192.68.2.151) has reached a **Tencent Cloud** server (49.51.129.211) via **HTTP**



EzVIZ Cam: Peeking Into The .bin File

- Binary file extracted with Wireshark and inspected with binwalk
- A whole **filesystem** shipped in it
- Filesystem extracted with **binwalk -Mer**

```
$ binwalk CS-CV246-A0-1C2WFR.dav

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
196           0xC4            Squashfs filesystem, little endian, version 4.0 [...]
4059420       0x3DF11C        uImage header [...] image type: OS Kernel Image [...] "Linux-3.0.8"
6248761       0x5F5939        Certificate in DER format (x509 v3), header length: 4, sequence
[...]
```

EzVIZ Cam: The Filesystem Found in The Firmware

#sf21vus - Material: https://bit.ly/2X4bwSq

- The root of the filesystem contains kernel modules, shared objects, executable scripts, binary applications, ...

```
simone@devel:/storage/ezviz_cam/_CS-CV246-A0-1C2WFR.dav.extracted/squashfs-root$ ls -la
total 0
..
ASC16
ASC32.bin
HZK16
da_info
dhcpd.conf
execSystemCmd
ezapp
ezdsp
font_asc16.hex
font_gb2312.hex
hostapd
initrun.sh
jxf23_mipi_day_hdr.hex
jxf23_mipi_day_lnr.hex
jxf23_mipi_nig_lnr.hex
libdadsp.so
libdata_pack.so
libdewarp.so
libf2_isp.so
libnl-3.so.200
libnl-genl-3.so.200
libsys.so
load_modules_y8.sh
logoblack.hex
logowhite.hex
mav_cal.conf
mfgutil
mlan.ko
model_human_86185422.bin
model_human_86185770.bin
nuvoton.bin
sd8801.ko
```

EzVIZ Cam: The Filesystem dhcpd.conf

- A dhcpd.conf file containing network configuration for a bridged interface

```
../squashfs-root$ cat dhcpd.conf
ddns-update-style none;
subnet 192.168.8.0 netmask 255.255.255.0{
interface br0
option router 192.168.8.1
option subnet 255.255.255.0
option dns 192.168.8.1
option broadcast 192.168.8.255
option lease 864000
start 192.168.8.2
end 192.168.8.254
max_leases 250
}
```

- All devices use HTTP to fetch firmware binaries
- Trying to implement security over an insecure **HTTP** channel
  - Can't trust the server
  - Can't trust the contents of the firmware
- Typical attack can be the widely known Man-in-The-Middle (MITM)

OTA Conclusions: Firmware Data [1/2]

- Downloaded firmware contain TLS-data such as certificates
  - An attacker can replace such data and have it installed on the device
- EzVIZ cameras firmware carries not only TLS-data but also a **whole filesystem**
  - An attacker can **unpack**, **edit**, and **repack** the filesystem into the firmware image

OTA Conclusions: Firmware Data [2/2]

- A private keys seems also to be transmitted in cleartext to the Sonoff smart plug
  - Really a false positive?
  - Just errors in the firmware reassembly from the network?

OTA Conclusions: Cloud Services

#sf21vus - Material: https://bit.ly/2X4bwSq

	Cloud Services	Cloud Countries
EzVIZ Cameras	Tencent Cloud (HTTP)	🇩🇪
Xiaomi Smart Lamp	Zenlayer's Edge Cloud (HTTP)	🇫🇷
Sonoff Door Sensor	Amazon (HTTP)	🇩🇪
Sonoff Smart Plugs	Amazon (HTTP)	🇩🇪
Tp-Link Smart Plugs	*n/a: no device updates in the observation period*	

Take-Home: Internet Traffic

- Cloud Services
  - All devices rely heavily on **Amazon cloud services** when idle, operating, and also for OTA updates
  - EzVIZ cameras also rely on Tencent cloud
- NTP synchronization causes certain devices to contact tens of hosts across the globe

Take-Home: LAN Traffic

- Devices also actively **interact** with other hosts in **the LAN**
  - **Expected** service traffic such as DHCP and DNS
- EzVIZ cameras have been found to **move laterally** in the LAN with SSDP and HTTP

Take-Home: Protocols

- Mostly standard, widely known protocols such as HTTP and TLS
- The MQTT “standard” protocol for IoT messaging only used by EzVIZ cameras
- IPv6 almost non-existent

Take-Home: Security

- Encryption is used significantly with TLS being a first-citizen under normal device activities (idle and operating)
- OTA firmware updates are done via HTTP and this opens up to a series of security issues

#sf21vus

Appendix

Sniffing the Traffic: Hardware Used

- The Linux bridge has been setup on a **Raspberry Pi version 4**
  - Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
  - 8GB LPDDR4-3200 SDRAM
  - Full specs: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

Sniffing the Traffic: Software Used

- Raspbian GNU/Linux 10 (buster)

Sniffing the Traffic: tcpdump

- Created a **systemd** service to dump **1 pcap / hour**

```
pi@raspberrypi:~ $ cat /etc/systemd/system/tcpdump.service
[Unit]
After=network.target

[Service]
Restart=always
RestartSec=30
Environment="TCPDUMP_FORMAT=%%Y-%%m-%%d__%%H"
ExecStartPre=/bin/mkdir -p /storage/pcaps/
ExecStart=/sbin/tcpdump -i eth0 not port 22 -s 0 -G 3600 -w '/storage/pcaps/
iotdump_${TCPDUMP_FORMAT}.pcap'
ExecStop=/bin/kill -s QUIT $MAINPID

[Install]
WantedBy=multi-user.target
```

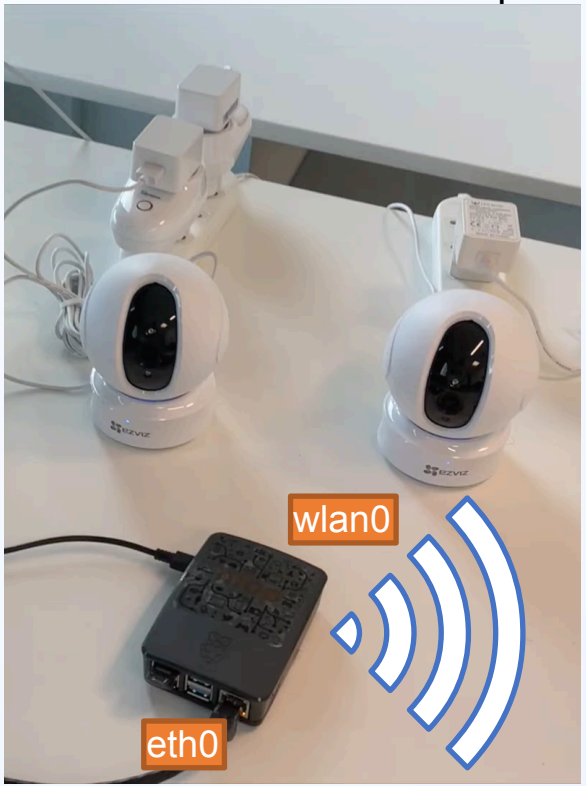
Sniffing the Traffic: pcap files

- One pcap per hour, a few MBs with “idle” devices
- Merged with **mergecap**

```
$ ls -lha | head
total 28G
drwxr-xr-x 3 pi   pi   132K Aug 11 17:34 .
drwxr-xr-x 6 root root 4.0K May  4 13:13 ..
-rw-r--r-- 1 root root 5.3M May  4 15:41 iotdump_2021-05-04__14.pcap
-rw-r--r-- 1 root root 2.9M May  4 16:41 iotdump_2021-05-04__15.pcap
-rw-r--r-- 1 root root 3.2M May  4 17:41 iotdump_2021-05-04__16.pcap
-rw-r--r-- 1 root root 2.8M May  4 18:41 iotdump_2021-05-04__17.pcap
-rw-r--r-- 1 root root 3.0M May  4 19:41 iotdump_2021-05-04__18.pcap
-rw-r--r-- 1 root root 2.8M May  4 20:41 iotdump_2021-05-04__19.pcap
-rw-r--r-- 1 root root 2.9M May  4 21:41 iotdump_2021-05-04__20.pcap

$ mergecap  -w iotdump_month.pcap iotdump_2021-0*
```

Sniffed pcaps: What's Inside

- pcaps sniffed on an **interface of a Linux bridge**
- Not only IoT devices traffic
- pcaps contain also
  - Multicast/Broadcast traffic entering the the bridge from eth0
  - Traffic originated by the Raspberry PI 4 itself



pcaps Cleanup with IoT Devices MACs

- Used **tcpdump with BFP** to extract traffic using IoT devices MACs
  - Can add BFP filters straight into the systemd service
- Example

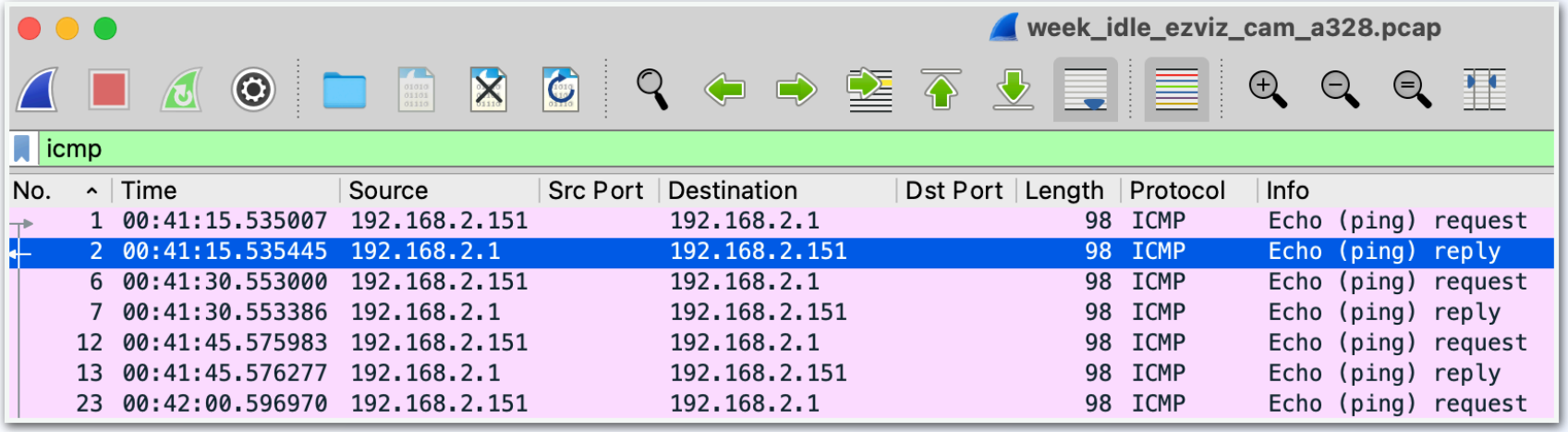
```
$ tcpdump -r iotdump_2021-05-06_to_2021-05-10.pcap -s0 ether host 80:9f:9b:45:a3:28
-w week_idle_ezviz_cam_a328.pcap
```

Traffic Analysis: Wireshark vs ntopng

- Tools are complimentary
- **Wireshark** is more **oriented to the packets**
  - Fantastic to inspect payloads, up to the single bit
- **ntopng** is more **oriented to the conversations** - aka **flows**
  - Very good for an overview of the network activity
- Recommendation is to **start with ntopng** and then **jump to Wireshark**
  - Especially with large trace files

EzVIZ Cam: ICMP to the Gateway

- PINGs to the gateway every 15 seconds
- Accounted for 5+ MB over 5 days



EzVIZ Cam: Idle Traffic Conclusions

- Significant amount of traffic towards hosts in the LAN (lateral movements)
  - Discovery of services
  - Attempts at fetching HTTP resources
- Significant data exchanged with the internet in plaintext
- Contacted 79 hosts across the globe just to stay idle

Sonoff Door Sensor: “Idle” Traffic

- The door sensor is much quieter

week_idle_...ensor.pcap | 131 | 2 | 1 | 2 | 275 | Search

Shortcuts | Alerts | Flows | Hosts | Maps

All Hosts

100 | IP Version | Direction | Filter Hosts

	IP Address	Flows	MAC Address	Name	Seen Since	Breakdown	Throughput	Total Bytes
	192.168.2.190 L	275	D0:27:01:E2:E3:CE		101 Days, 09:39:32	Sent Rcvd	0 bit/s	951.35 KB
	52.57.118.192 R	133	Ubiquiti_06:B3:5A	eu-api.coolkit.cc	101 Days, 09:39:32	Sent Rcvd	0 bit/s	910.7 KB
	192.168.2.1 L	142	Ubiquiti_06:B3:5A		101 Days, 08:42:19	Sent Rcvd	0 bit/s	40.65 KB

Showing 1 to 3 of 3 rows

In Depth: File week_idle_sonoff_door_sensor.pcap

Sonoff Door Sensor: “Idle” Internet Traffic

TLS with Amazon

week_idle_...ensor.pcap | 131 | 2 | 1 | 2 | 275 | Search

Shortcuts | Alerts | Flows | Hosts | Maps

All Hosts

TLS traffic with an Amazon host in Germany

Name dissected from the TLS SNI

100 | IP Version | Direction | Filter Hosts

IP Address	Flows	MAC Address	Name	Seen Since	Breakdown	Throughput	Total Bytes
192.168.2.190 L	275	D0:27:01:E2:E3:CE		101 Days, 09:39:32	Sent Rcvd	0 bit/s	951.35 KB
52.57.118.192 R	133	Ubiquiti_06:B3:5A	eu-api.coolkit.cc	101 Days, 09:39:32	Sent Rcvd	0 bit/s	910.7 KB
192.168.2.1 L	142	Ubiquiti_06:B3:5A		101 Days, 08:42:19	Sent Rcvd	0 bit/s	40.65 KB

Showing 1 to 3 of 3 rows

In Depth: File week_idle_sonoff_door_sensor.pcap

Sonoff Door Sensor: “Idle” LAN Traffic

Service traffic with the gateway

week_idle_...ensor.pcap | 131 | 2 | 1 | 2 | 275 | Search

Shortcuts | Alerts | Flows | Hosts | Maps

All Hosts

- DHCP
- DNS traffic with the gateway (misconfiguration?). The gateway responds with ICMP port unreachable.

IP Version | Direction | Filter Hosts

IP Address	Flows				Breakdown	Throughput	Total Bytes
192.168.2.190 L	2?5			9:32	Sent Rcvd	0 bit/s	951.35 KB
52.57.118.192 R	133	Ubiquiti_06:B3:5A	eu-api.coolkit.cc	101 Days, 09:39:32	Sent Rcvd	0 bit/s	910.7 KB
192.168.2.1 L	142	Ubiquiti_06:B3:5A		101 Days, 08:42:19	Sent Rcvd	0 bit/s	40.65 KB

Showing 1 to 3 of 3 rows

In Depth: File week_idle_sonoff_door_sensor.pcap

Sonoff Smart Plug: “Idle” Traffic

#sf21vus - Material: https://bit.ly/2X4bwSq

week_idle_..._28e1.pcap

Search

Shortcuts
Alerts
Flows
Hosts
Maps
Interface

All Hosts

TLS traffic with an Amazon host in Germany

SSDP responses (EzVIZ discovery)

DHCP

100 | IP Version | Direction | Filter Hosts

	IP Address	Flows	MAC Address	Name	Seen Since	Breakdown	Throughput	Total Bytes
	192.168.2.241 L	4	Espressi_B6:28:E1	esp_b628e1	101 Days, 10:15:00	Sent Rcvd	0 bit/s ↓	882.67 KB
	52.29.59.123 R	2	Ubiquiti_06:B3:5A			Sent Rcvd	0 bit/s ↓	866.2 KB
	224.0.0.251 M	1	IPv4mcast_00:00:FB		100 Days, 22:52:31	Rcvd	0 bit/s —	9.32 KB
	192.168.2.1 L	1	Ubiquiti_06:B3:5A		101 Days, 09:20:51	Sent Rcvd	0 bit/s —	7.15 KB

Showing 1 to 4 of 4 rows

In Depth: File week_idle_sonoff_smart_plug_28e1.pcap

TP-Link Smart Plug: “Idle” Traffic

#sf21vus - Material: https://bit.ly/2X4bwSq

- Quiet, < 2MB, mostly Internet

week_idle_..._4bc2.pcap | 2 | 2 | 2 | 95 | 2 | 494 | Search

Shortcuts | Alerts | Flows | Hosts | Maps | Interface | Settings

All Hosts

100 | IP Version | Direction | Filter Hosts

	IP Address	Flows	MAC Address	Name	Seen Since	Breakdown	Throughput	Total Bytes
	192.168.2.155 L	494	1C:3B:F3:4D:4B:C2		101 Days, 09:56:39	Sent Rcvd	0 bit/s ↓	938.64 KB
	54.229.4.116 R	1	Ubiquiti_06:B3:5A	n-devs.tplinkcloud.com	99 Days, 06:12:40	Sent Rcvd	0 bit/s ↓	463.25 KB
	52.16.226.187 R	1	Ubiquiti_06:B3:5A		101 Days, 09:56:39	Sent Rcvd	0 bit/s —	352.73 KB
	192.168.2.1 L	247	Ubiquiti_06:B3:5A		101 Days, 09:31:41	Sent Rcvd	0 bit/s —	66.0 KB
	162.159.200.123 R	25	Ubiquiti_06:B3:5A		101 Days, 08:30:36	Sent Rcvd	0 bit/s —	5.37 KB
	34.192.244.186 R	1	Ubiquiti_06:B3:5A	n-deventry.tplinkcloud.c...	99 Days, 06:12:43	Sent Rcvd	0 bit/s —	5.1 KB

In Depth: File week_idle_tplink_smart_plug_4bc2.pcap

TP-Link Smart Plug: “Idle” Internet Traffic

#sf21vus - Material: https://bit.ly/2X4bwSq

week_idle_..._4bc2.pcap

Search

Shortcuts
Alerts
Flows
Hosts
Maps
Interface
Settings

All Hosts

TLS traffic with Amazon hosts

100 | IP Version | Direction | Filter Hosts

IP Address	Flows	MAC Address	Name	Seen Since	Breakdown	Throughput	Total Bytes
192.168.2.155 L	94	1C:3B:F3:4D:4[illegible]:C2		101 Days, 09:56:39	Sent Rcvd	0 bit/s ↓	938.64 KB
54.229.4.116 R	1	Ubiquiti_[illegible]:B3:5A	n-devs.tplinkcloud.com	99 Days, 06:12:40	Sent Rcvd	0 bit/s ↓	463.25 KB
52.16.226.187 R	1	Ubiquiti_06:B3:5A		101 Days, 09:56:39	Sent Rcvd	0 bit/s —	352.73 KB
192.168.2.1 L	247	Ubiquiti_06:B3:5A		101 Days, 09:31:41	Sent Rcvd	0 bit/s —	66.0 KB
162.159.200.123 R	[illegible]	Ubiquiti_06:B3:5A		101 Days, 08:30:36	Sent Rcvd	0 bit/s —	5.37 KB
34.192.244.186 R	1	Ubiquiti_06:B3:5A	n-deventry.tplinkcloud.c...	99 Days, 06:12:43	Sent Rcvd	0 bit/s —	5.1 KB

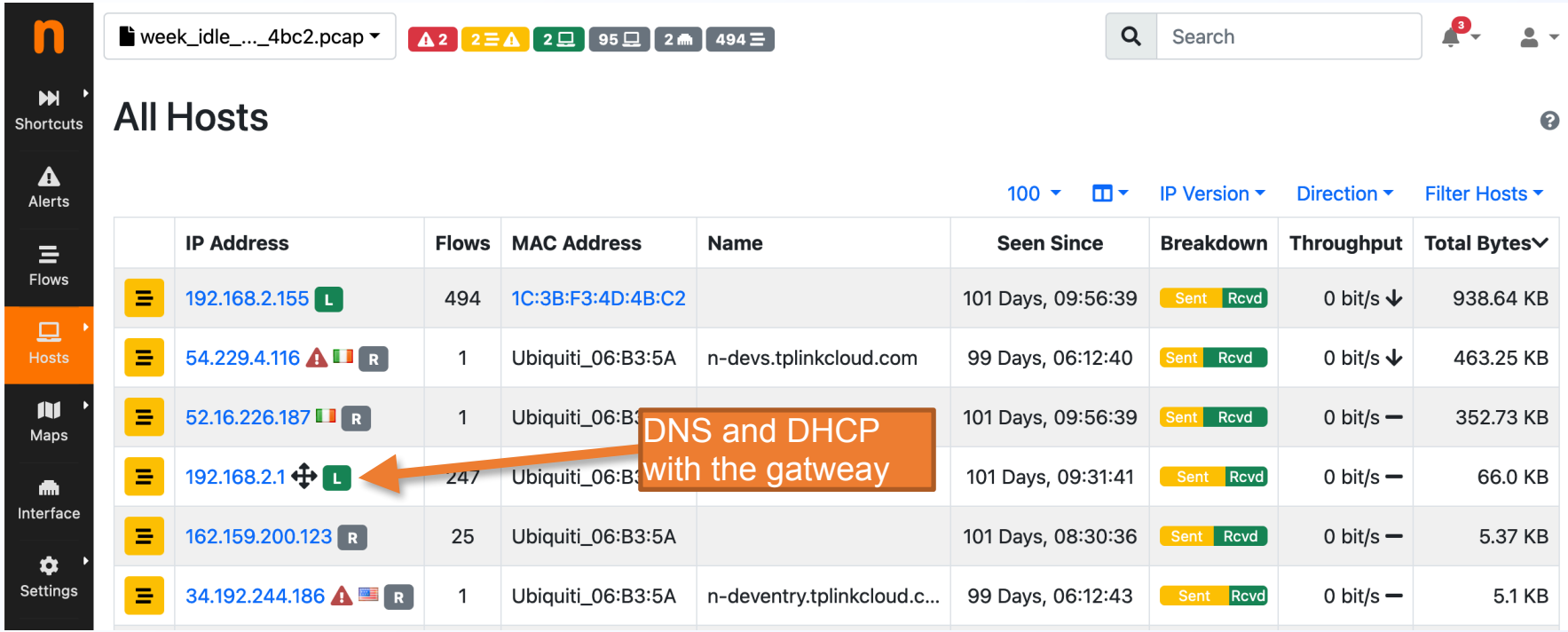
In Depth: File week_idle_tplink_smart_plug_4bc2.pcap

TP-Link Smart Plug: Geography of Internet Hosts

#sf21vus - Material: https://bit.ly/2X4bwSq

TP-Link Smart Plug: “Idle” LAN Traffic

#sf21vus - Material: https://bit.ly/2X4bwSq



In Depth: File week_idle_tplink_smart_plug_4bc2.pcap

Sonoff Door Sensor: Open/Close

#sf21vus - Material: https://bit.ly/2X4bwSq

- 4 packets on an already-open **TLS** connection with an **Amazon** host

sonoff_door_sensor_pairing_open_close.pcap

tcp.stream eq 2

No.	Time	Source	Src Port	Destination	Dst Port	Length	Protocol	Info
72	15:15:40.493589	192.168.2.190	52332	52.57.118.192	8080	54	TCP	[TCP Dup ACK 70#1] 52332 → 8080 [ACK] Seq=912 Ack=3273 Win=11312 Len=0
74	15:15:40.507174	192.168.2.190	52332	52.57.118.192	8080	54	TCP	[TCP Dup ACK 70#2] 52332 → 8080 [ACK] Seq=912 Ack=3273 Win=11312 Len=0
79	15:17:36.486884	192.168.2.190	52332	52.57.118.192	8080	507	TLSv1.2	Application Data
80	15:17:36.511668	52.57.118.192	8080	192.168.2.190	52332	60	TCP	8080 → 52332 [ACK] Seq=3273 Ack=1365 Win=30016 Len=0
81	15:17:36.644739	52.57.118.192	8080	192.168.2.190	52332	331	TLSv1.2	Application Data
82	15:17:40.100803	192.168.2.190	52332	52.57.118.192	8080	54	TCP	52332 → 8080 [ACK] Seq=1365 Ack=3550 Win=11680 Len=0
91	15:18:00.463415	192.168.2.190	52332	52.57.118.192	8080	507	TLSv1.2	Application Data
92	15:18:00.488291	52.57.118.192	8080	192.168.2.190	52332	60	TCP	8080 → 52332 [ACK] Seq=3550 Ack=1818 Win=31088 Len=0
93	15:18:00.567685	52.57.118.192	8080	192.168.2.190	52332	331	TLSv1.2	Application Data
94	15:18:03.858378	192.168.2.190	52332	52.57.118.192	8080	54	TCP	52332 → 8080 [ACK] Seq=1818 Ack=3827 Win=11403 Len=0
97	15:18:17.315193	192.168.2.190	52332	52.57.118.192	8080	507	TLSv1.2	Application Data
98	15:18:17.340256	52.57.118.192	8080	192.168.2.190	52332	60	TCP	8080 → 52332 [ACK] Seq=3827 Ack=2271 Win=32160 Len=0
99	15:18:17.431501	52.57.118.192	8080	192.168.2.190	52332	331	TLSv1.2	Application Data
100	15:18:20.859622	192.168.2.190	52332	52.57.118.192	8080	54	TCP	52332 → 8080 [ACK] Seq=2271 Ack=4104 Win=11680 Len=0

OPEN

CLOSE

Tp-Link Smart Plug: On/Off TLS

#sf21vus - Material: https://bit.ly/2X4bwSq

- On/Off involves exchanging **TLS** with two **Amazon** hosts
- Exchange is initiated from Amazon
- Connections already open

OFF

ON

```
22:40:55.598704 IP 192.168.2.155.63982 > ec2-107-23-25-187.compute-1.amazonaws.com.https: Flags [P.], seq 1263:1348, ack 964, win 11928, length 85
22:40:55.600903 IP 192.168.2.155.63984 > ec2-52-206-238-222.compute-1.amazonaws.com.https: Flags [P.], seq 3397:3674, ack 7895, win 12288, length 277
22:40:55.688341 IP 192.168.2.155.63982 > ec2-107-23-25-187.compute-1.amazonaws.com.https: Flags [P.], seq 1263:1348, ack 964, win 11928, length 85
22:40:55.689130 IP 192.168.2.155.63984 > ec2-52-206-238-222.compute-1.amazonaws.com.https: Flags [P.], seq 3397:3674, ack 7895, win 12288, length 277
22:40:55.714977 IP ec2-107-23-25-187.compute-1.amazonaws.com.https > 192.168.2.155.63982: Flags [.], ack 1348, win 457, length 0
22:40:55.720313 IP 192.168.2.155.63982 > ec2-107-23-25-187.compute-1.amazonaws.com.https: Flags [P.], seq 1348:1481, ack 964, win 11928, length 133
22:40:55.754915 IP ec2-52-206-238-222.compute-1.amazonaws.com.https > 192.168.2.155.63984: Flags [.], ack 3674, win 155, length 0
22:40:55.759024 IP 192.168.2.155.63984 > ec2-52-206-238-222.compute-1.amazonaws.com.https: Flags [P.], seq 3674:4847, ack 7895, win 12288, length 1173
22:40:55.800907 IP ec2-52-206-238-222.compute-1.amazonaws.com.https > 192.168.2.155.63984: Flags [.], ack 3674, win 155, length 0
22:40:55.804669 IP ec2-107-23-25-187.compute-1.amazonaws.com.https > 192.168.2.155.63982: Flags [.], ack 1348, win 457, length 0
22:40:55.836718 IP ec2-107-23-25-187.compute-1.amazonaws.com.https > 192.168.2.155.63982: Flags [.], ack 1481, win 469, length 0
22:40:55.871479 IP ec2-52-206-238-222.compute-1.amazonaws.com.https > 192.168.2.155.63984: Flags [.], ack 4847, win 164, length 0
22:40:55.878575 IP ec2-52-206-238-222.compute-1.amazonaws.com.https > 192.168.2.155.63984: Flags [P.], seq 7895:9100, ack 4847, win 164, length 1205
22:40:55.878686 IP ec2-52-206-238-222.compute-1.amazonaws.com.https > 192.168.2.155.63984: Flags [P.], seq 9100:9169, ack 4847, win 164, length 69
22:40:55.882912 IP 192.168.2.155.63984 > ec2-52-206-238-222.compute-1.amazonaws.com.https: Flags [.], ack 9169, win 11651, length 0
22:40:59.278013 IP ec2-107-23-25-187.compute-1.amazonaws.com.https > 192.168.2.155.63982: Flags [P.], seq 964:1209, ack 1481, win 469, length 245
22:40:59.282248 IP 192.168.2.155.63982 > ec2-107-23-25-187.compute-1.amazonaws.com.https: Flags [P.], seq 1481:1566, ack 1209, win 11806, length 85
22:40:59.284498 IP 192.168.2.155.63984 > ec2-52-206-238-222.compute-1.amazonaws.com.https: Flags [P.], seq 4847:5124, ack 9169, win 11651, length 277
22:40:59.398485 IP ec2-107-23-25-187.compute-1.amazonaws.com.https > 192.168.2.155.63982: Flags [.], ack 1566, win 469, length 0
22:40:59.402362 IP 192.168.2.155.63982 > ec2-107-23-25-187.compute-1.amazonaws.com.https: Flags [P.], seq 1566:1699, ack 1209, win 11806, length 133
22:40:59.438795 IP ec2-52-206-238-222.compute-1.amazonaws.com.https > 192.168.2.155.63984: Flags [.], ack 5124, win 173, length 0
22:40:59.442988 IP 192.168.2.155.63984 > ec2-52-206-238-222.compute-1.amazonaws.com.https: Flags [P.], seq 5124:6297, ack 9169, win 11651, length 1173
22:40:59.518923 IP ec2-107-23-25-187.compute-1.amazonaws.com.https > 192.168.2.155.63982: Flags [.], ack 1699, win 480, length 0
22:40:59.555737 IP ec2-52-206-238-222.compute-1.amazonaws.com.https > 192.168.2.155.63984: Flags [.], ack 6297, win 182, length 0
22:40:59.565351 IP ec2-52-206-238-222.compute-1.amazonaws.com.https > 192.168.2.155.63984: Flags [P.], seq 9169:10374, ack 6297, win 182, length 1205
22:40:59.565352 IP ec2-52-206-238-222.compute-1.amazonaws.com.https > 192.168.2.155.63984: Flags [P.], seq 10374:10443, ack 6297, win 182, length 69
22:40:59.568269 IP 192.168.2.155.63984 > ec2-52-206-238-222.compute-1.amazonaws.com.https: Flags [.], ack 10443, win 11014, length 0
```

Mi Lamp: On/Off TLS

- On/Off involves a **TLS** conversation with **Amazon**
- Similar to what has been seen for the plugs
- No MDNS, more TLS packets

OFF

```
22:50:07.397262 IP ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https > 192.168.2.157.53903: Flags [P.], seq 2039:2176, ack 1878, win 65535, length 137
22:50:07.499462 IP 192.168.2.157.53903 > ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https: Flags [P.], seq 1878:1983, ack 2176, win 5550, length 105
22:50:07.568149 IP ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https > 192.168.2.157.53903: Flags [.], ack 1983, win 65535, length 0
22:50:07.629267 IP 192.168.2.157.53903 > ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https: Flags [P.], seq 1983:2104, ack 2176, win 5550, length 121
22:50:07.658270 IP ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https > 192.168.2.157.53903: Flags [.], ack 2104, win 65535, length 0
22:50:07.658720 IP ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https > 192.168.2.157.53903: Flags [P.], seq 2176:2265, ack 2104, win 65535, length 89
22:50:07.888531 IP 192.168.2.157.53903 > ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https: Flags [.], ack 2265, win 5461, length 0
```

ON

```
22:50:10.863833 IP ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https > 192.168.2.157.53903: Flags [P.], seq 2265:2402, ack 2104, win 65535, length 137
22:50:10.980443 IP 192.168.2.157.53903 > ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https: Flags [P.], seq 2104:2209, ack 2402, win 5324, length 105
22:50:11.049253 IP ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https > 192.168.2.157.53903: Flags [.], ack 2209, win 65535, length 0
22:50:11.677768 IP 192.168.2.157.53903 > ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https: Flags [P.], seq 2209:2330, ack 2402, win 5324, length 121
22:50:11.707236 IP ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https > 192.168.2.157.53903: Flags [.], ack 2330, win 65535, length 0
22:50:11.707431 IP ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https > 192.168.2.157.53903: Flags [P.], seq 2402:2491, ack 2330, win 65535, length 89
22:50:11.882031 IP 192.168.2.157.53903 > ec2-3-126-247-75.eu-central-1.compute.amazonaws.com.https: Flags [.], ack 2491, win 5235, length 0
```

- Use the utility **strings** to find plaintext inside the .bin file
- Can look for almost everything, including
  - TLS certificates
  - IP addresses
  - Domain names
  - etc.

Use strings to look for **TLS certificates**

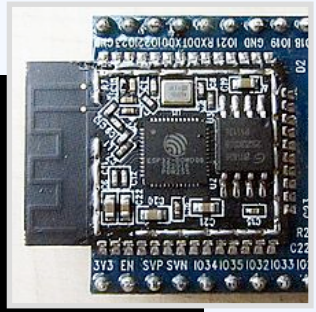
```
Simones-Mac-mini:Downloads simone$ strings
81f6c20996e3d5a5fbc2997a42bde2af_upd_yeelink.light.lamp4.bin  | egrep -ri
certificate
(standard input):A?-----BEGIN CERTIFICATE-----
(standard input):-----END CERTIFICATE-----
(standard input):-----END CERTIFICATE-----
(standard input):@-----BEGIN CERTIFICATE-----
(standard input):-----END CERTIFICATE-----
(standard input):-----BEGIN CERTIFICATE-----
(standard input):-----END CERTIFICATE-----
(standard input):[1;%dmRead the Accessory Certificate failed
(standard input):[1;%dmFailed to read MFi Certificate
(standard input):-----BEGIN CERTIFICATE-----
```

PEM-encoded
TLS certificates

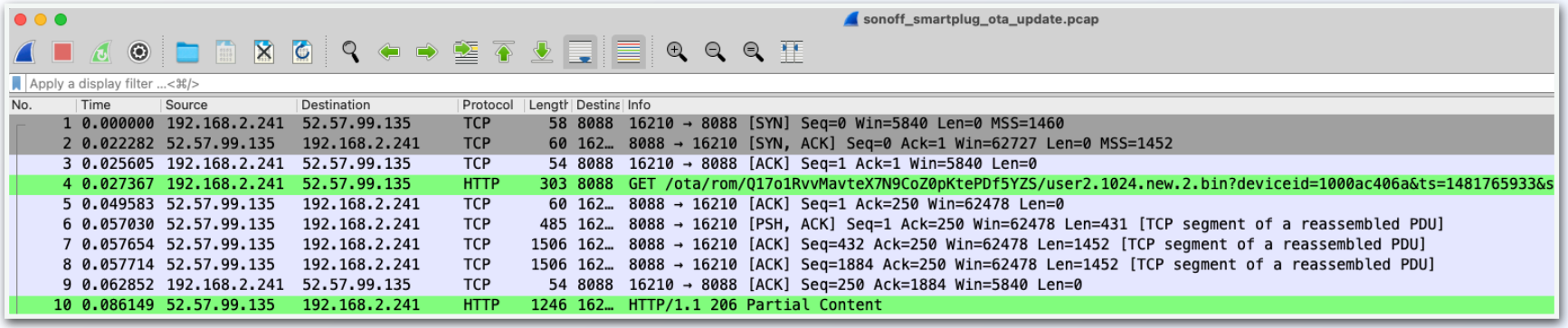
- Use the utility **strings** to find plaintext inside the .bin file

```
Simones-Mac-mini:Downloads simone$ strings
81f6c20996e3d5a5fbc2997a42bde2af_upd_yeelink.light.lamp4.bin  | head -n20
ac38ad7
miio_app
09:54:38
Mar 21 2021
ac38ad7
`Fo5%Y6eF
esp_task_wdt_init(CONFIG_ESP_TASK_WDT_TIMEOUT_S, false)
/home/auto_build/ylk_auto_build/build_dir/esp32_mi2x/esp-idf/components/esp32/
cpu_start.c
esp_task_wdt_add(idle_0)
/dev/uart/0
```

ESP32 SoC microcontroller initialization functions



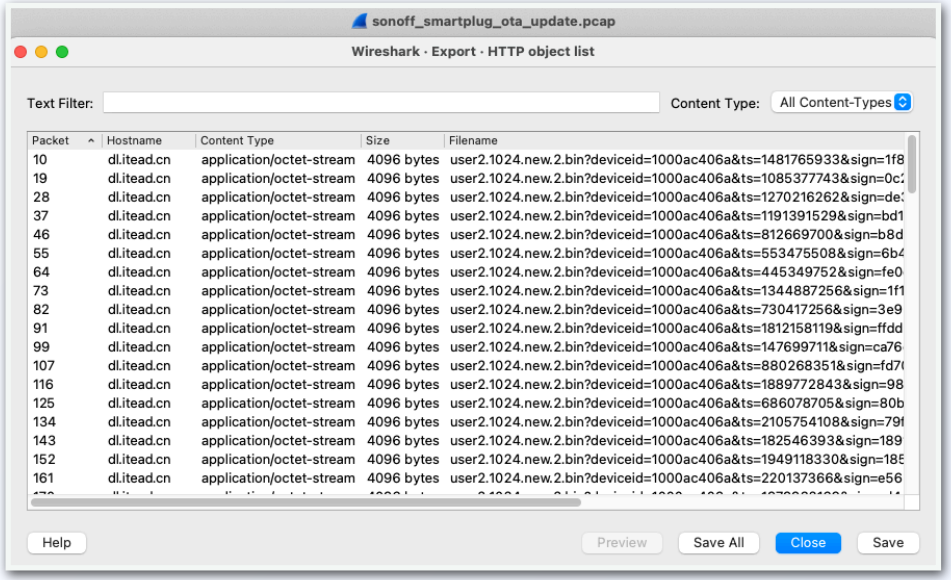
- The lamp (192.68.2.241) has reached an **Amazon** server (52.57.99.135) in via **HTTP**
- Requests and responses use **Range:** to fetch data in chunks of 4096 Bytes



Sonoff Smart Plug: HTTP Range Data

#sf21vus - Material: https://bit.ly/2X4bwSq

- Wireshark couldn't reassemble chunks into a single .bin with Wireshark
- Reassembled with **bash**

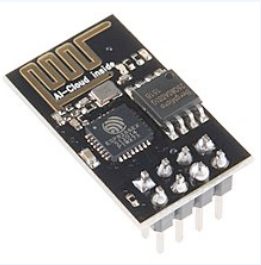



```
$ for fl in `find . -type f -name "user2.1024.new.2.b*" -print0 | xargs -0 ls -thtU | tail
-r`; do cat $fl >> user2.1024.new.2.bin; done
$ ls -lha user2.1024.new.2.bin
-rw-r--r--  1 simone  staff   433K Aug 12 19:06 user2.1024.new.2.bin
```

In Depth: File user2.1024.new.2.bin

Sonoff Smart Plug: Peeking Into The .bin File

- Look at the file with **binwalk** and **strings** to understand its content
  - Firmware image with TLS data
- Can guess the microcontroller

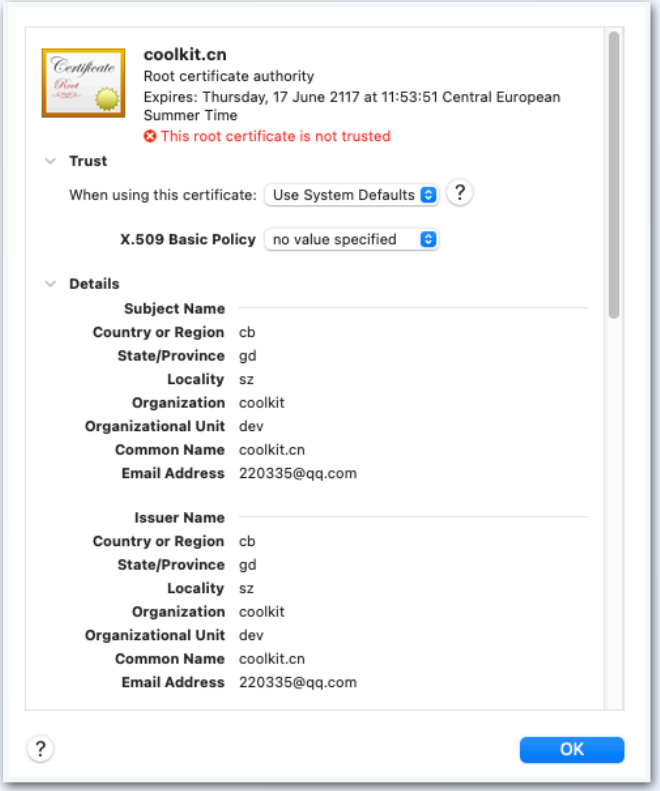


```
$ binwalk  user2.1024.new.2.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
379382        0x5C9F6         Certificate in DER format (x509 v3), header length: 4,
sequence length: 708
394624        0x60580         Base64 standard index table
394816        0x60640         SHA256 hash constants, little endian
440123        0x6B73B         PEM RSA private key
441134        0x6BB2E         PEM certificate
$ strings user2.1024.new.2.bin | egrep -ri firm
(standard input):Firmware ONLY supports ESP8266!!!
```

Sonoff Smart Plug: TLS Data

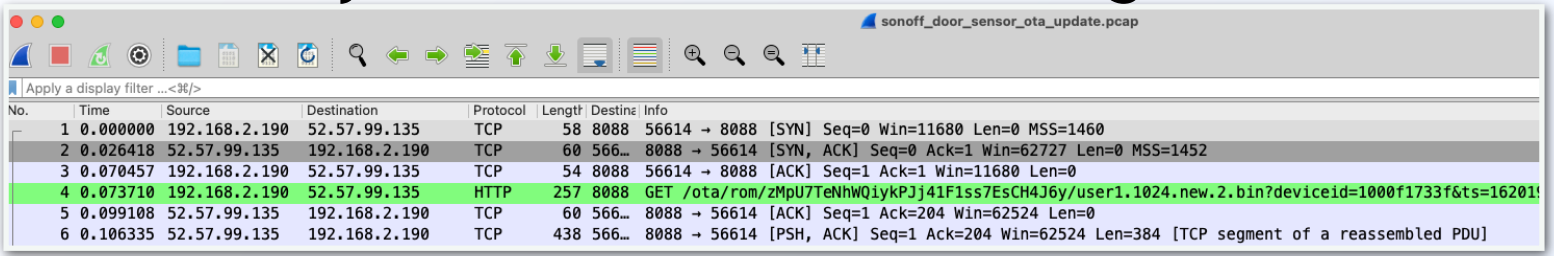
- Extracted a TLS **root** certificate
  - Expires in 2117
- The **private key** luckily seems to be a **binwalk false positive** with just garbled data

In Depth: File 5C9F6.crt



- Sonoff door sensor updated manually from the Mi Home IoS app
- Same update procedure as done for the smart plug

- The lamp (192.68.2.190) has reached an **Amazon** server (52.57.99.135) via **HTTP**
- **Similarities** with the Sonoff Smart Plug Update
  - Same **HTTP server**
  - URL pattern looks the same (/ota/rom/...)
- Binary file transferred with single HTTP



Sonoff Door Sensor: Peeking Into The .bin File:

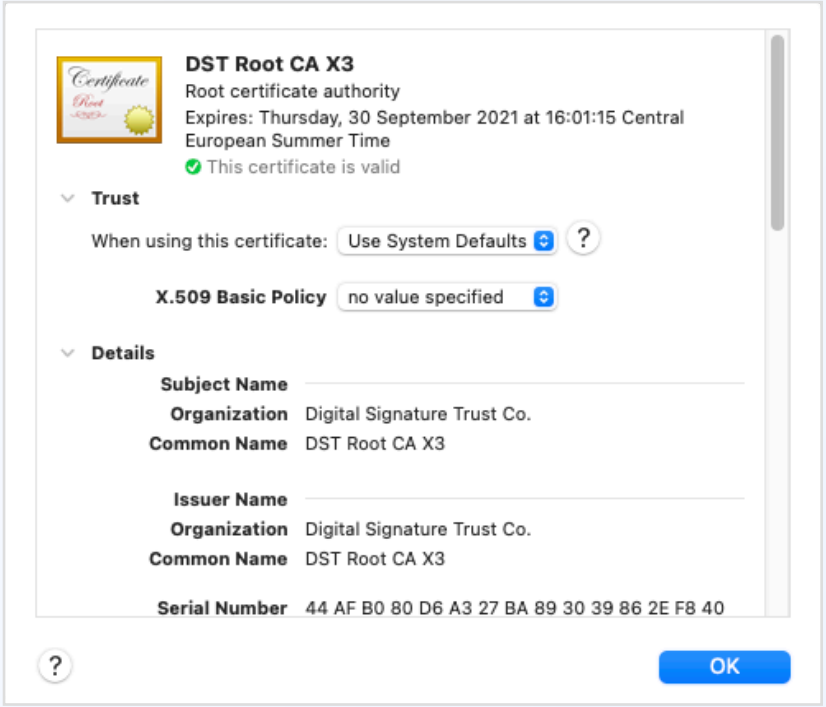
- Binary file extracted with Wireshark
  - Tiny **165K**
- Incremental patch (PTCH?)
- Used strings to look for data, including certificate files

```
$ binwalk sonoff_door_sensor.user1.1024.new.2.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
115132        0x1C1BC         Unix path: /api/user/device/update
115404        0x1C2CC         PEM certificate
119248        0x1D1D0         Base64 standard index table
$ strings sonoff_door_sensor.user1.1024.new.2.bin | head -n2
PTCH
PTCH
$ strings sonoff_door_sensor.user1.1024.new.2.bin | egrep -ri
certificate
(standard input):Failed to verify peer certificate! Flags = %d
(standard input):Certificate verified.
(standard input):-----BEGIN CERTIFICATE-----
(standard input):-----END CERTIFICATE-----
```

Sonoff Door Sensor: Peeking Into The .bin File

#sf21vus - Material: https://bit.ly/2X4bwSq

- Found a **Root** certificate
- Close to its expiration date
- Again, the root of a chain of trust has been sent over **HTTP**



In Depth: File sonoff_door_sensor.user1.1024.new.2.bin.certificate_01.pem