Security in IoT Ecosystem

Module 7

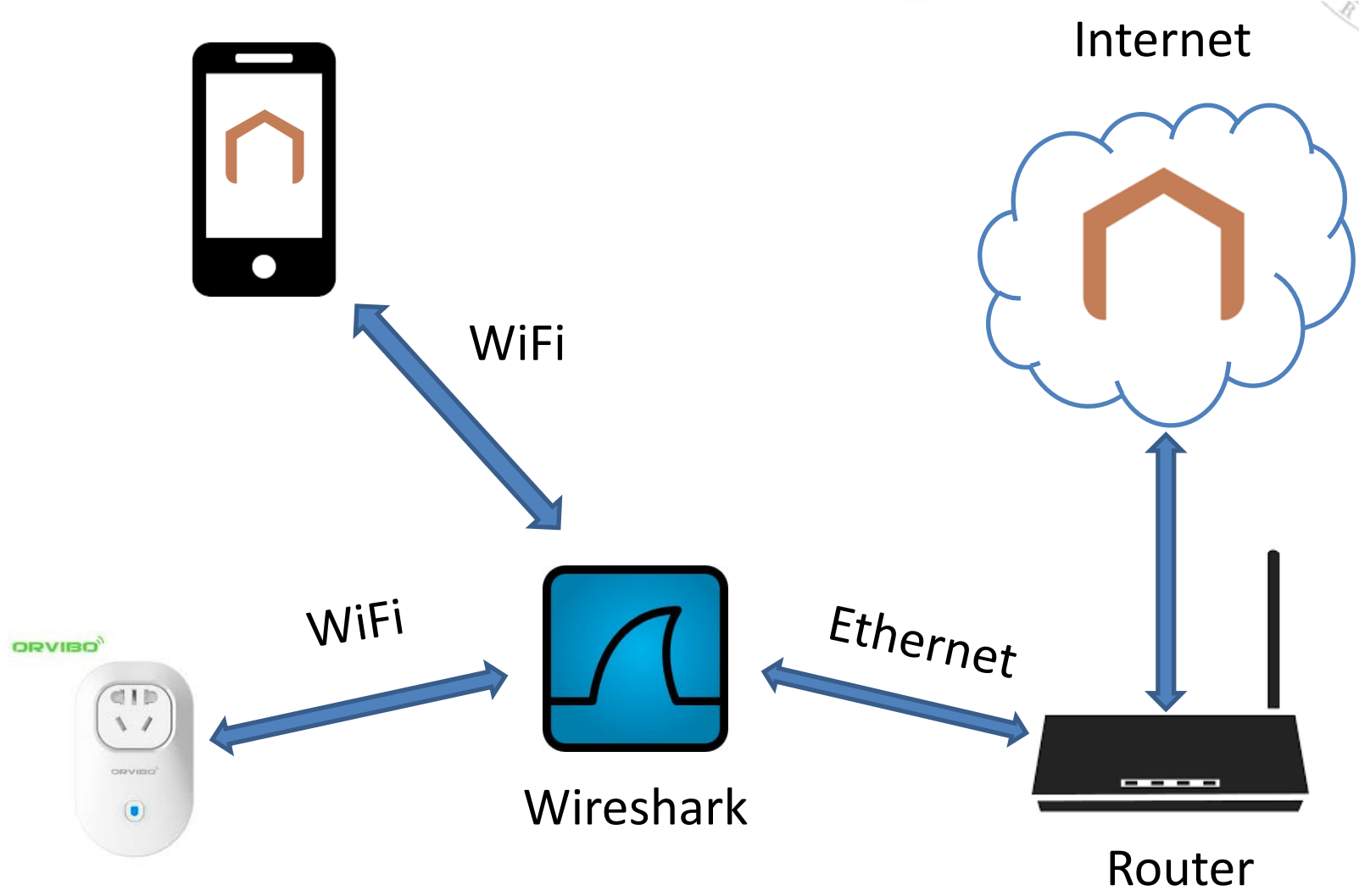
Smart Socket Pentest
Part IV

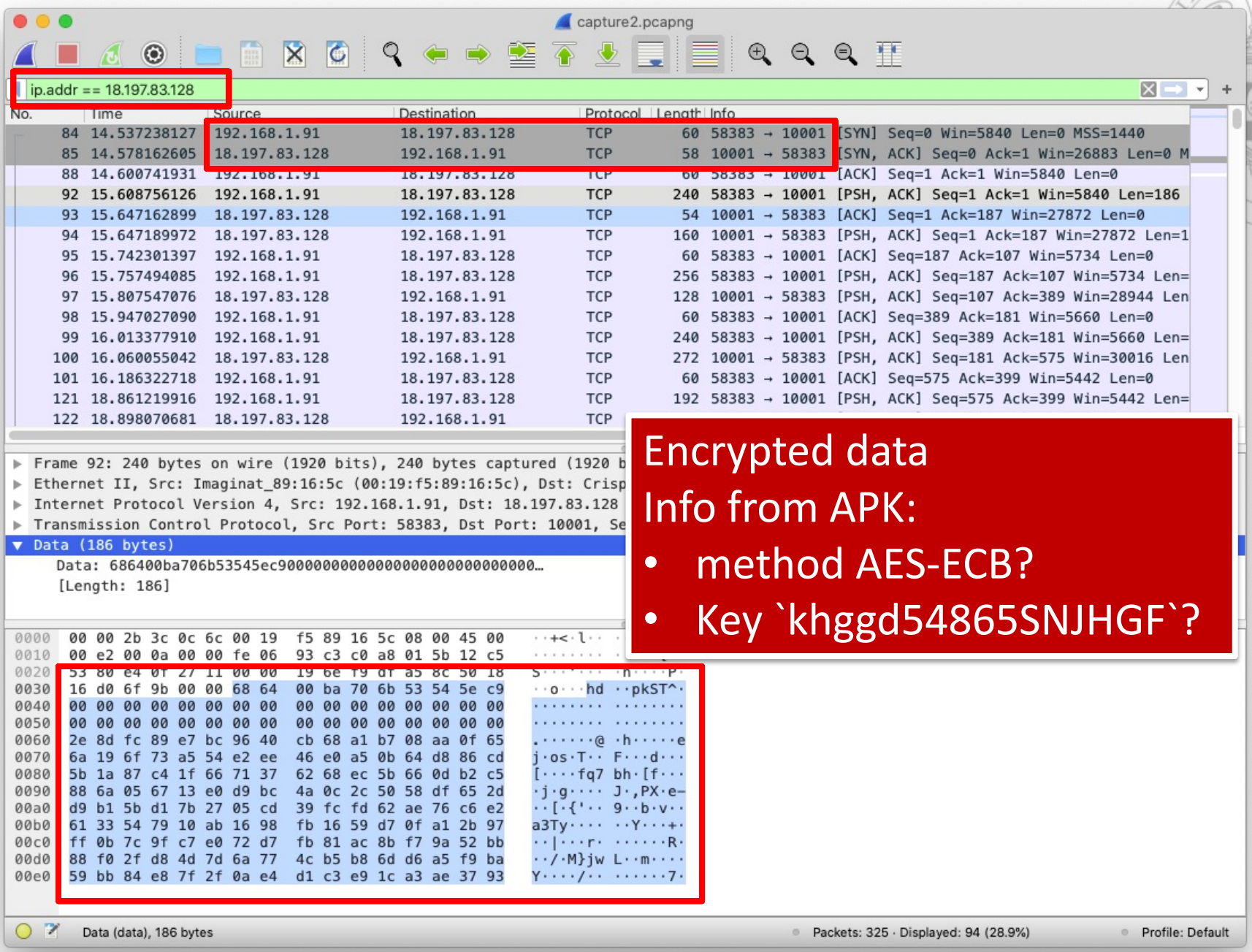
Prof.: Joaquín Recas

Smart Socket Initial Setup

1. Initial Setup
2. Capture Traffic
3. Traffic analysis
4. Apk Analysis
5. Decrypt traffic
6. Take control of the device (MiM)
7. Pentest Project tasks

Study communications:





Encrypted data

Info from APK:

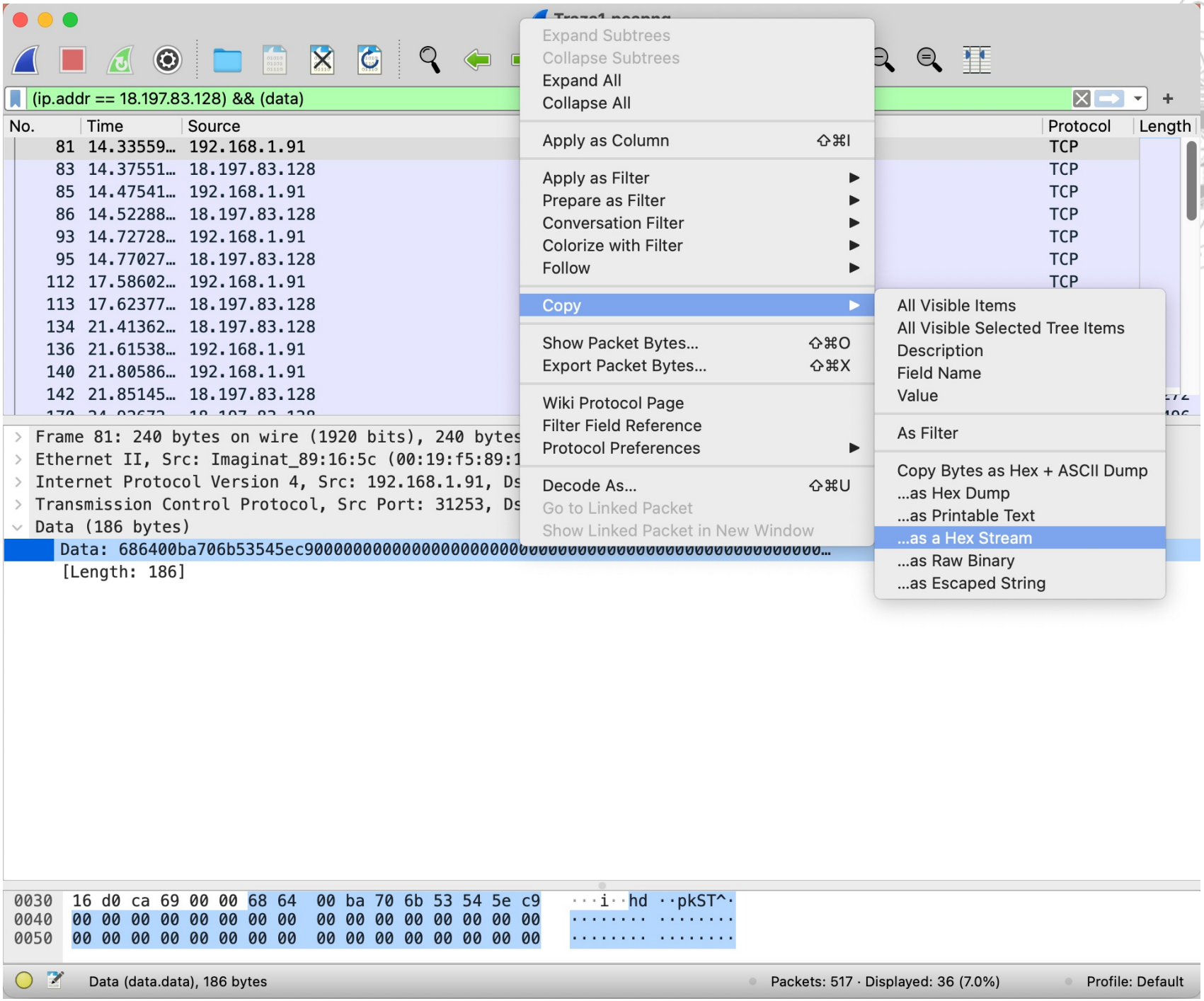
- method AES-ECB?
- Key `khggd54865SNJHGF`?

Python Crypto

- Install crypto library:

```
$ sudo pip install pycrypto
```

- Select packages that go to/from Orvibo server and have payload
- Copy the payload from WireShark
- Decrypt using AES-ECB + key


(ip.addr == 18.197.83.128) && (data)
No. Time Source Protocol Length
81 14.33559… 192.168.1.91 TCP
83 14.37551… 18.197.83.128 TCP
85 14.47541… 192.168.1.91 TCP
86 14.52288… 18.197.83.128 TCP
93 14.72728… 192.168.1.91 TCP
95 14.77027… 18.197.83.128 TCP
112 17.58602… 192.168.1.91 TCP
113 17.62377… 18.197.83.128
134 21.41362… 18.197.83.128
136 21.61538… 192.168.1.91
140 21.80586… 192.168.1.91
142 21.85145… 18.197.83.128
Frame 81: 240 bytes on wire (1920 bits), 240 bytes
Ethernet II, Src: Imaginat_89:16:5c (00:19:f5:89:1
Internet Protocol Version 4, Src: 192.168.1.91, Ds
Transmission Control Protocol, Src Port: 31253, Ds
Data (186 bytes)
Data: 686400ba706b53545ec9
[Length: 186]
0030 16 d0 ca 69 00 00 68 64 00 ba 70 6b 53 54 5e c9 ···i··hd ··pkST^·
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ···· ····
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ···· ····
Data (data.data), 186 bytes
Packets: 517 · Displayed: 36 (7.0%)
Profile: Default

Expand Subtrees
Collapse Subtrees
Expand All
Collapse All
Apply as Column ⇧⌘I
Apply as Filter
Prepare as Filter
Conversation Filter
Colorize with Filter
Follow
Copy
Show Packet Bytes... ⇧⌘O
Export Packet Bytes... ⇧⌘X
Wiki Protocol Page
Filter Field Reference
Protocol Preferences
Decode As... ⇧⌘U
Go to Linked Packet
Show Linked Packet in New Window
All Visible Items
All Visible Selected Tree Items
Description
Field Name
Value
As Filter
Copy Bytes as Hex + ASCII Dump
...as Hex Dump
...as Printable Text
...as a Hex Stream
...as Raw Binary
...as Escaped String

Python Crypto

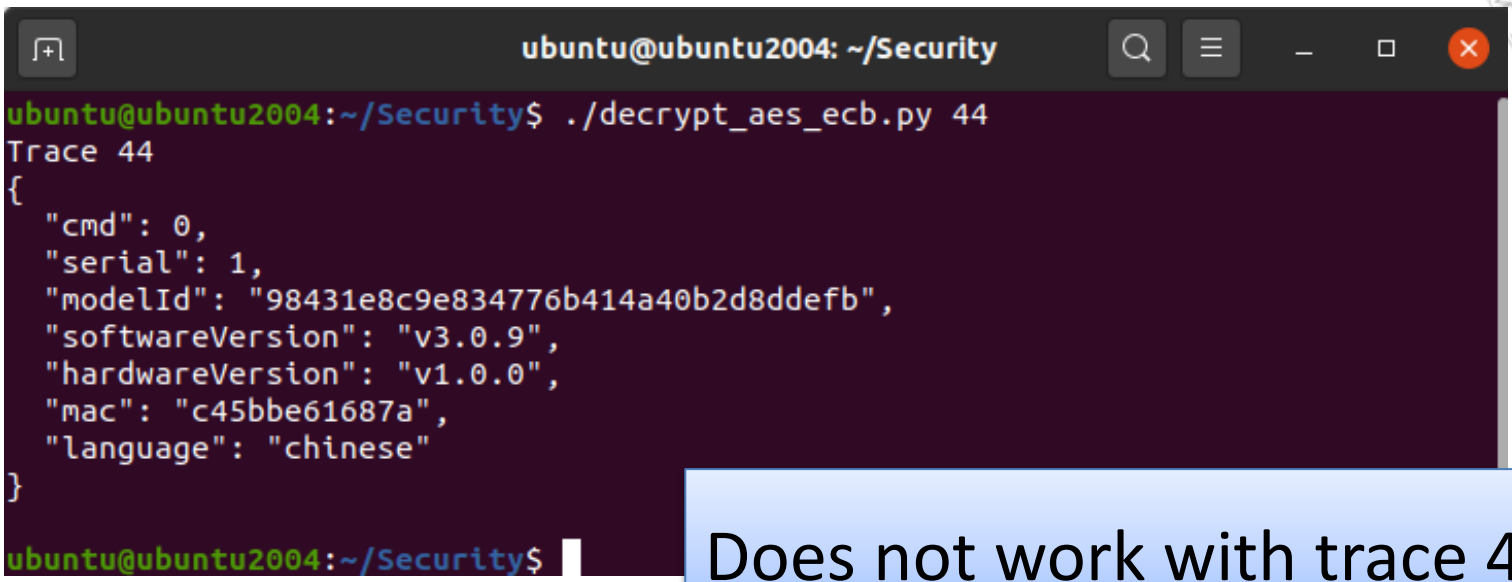
decrypt_AES_ECB.py — UNREGISTERED

decrypt_AES_ECB.py

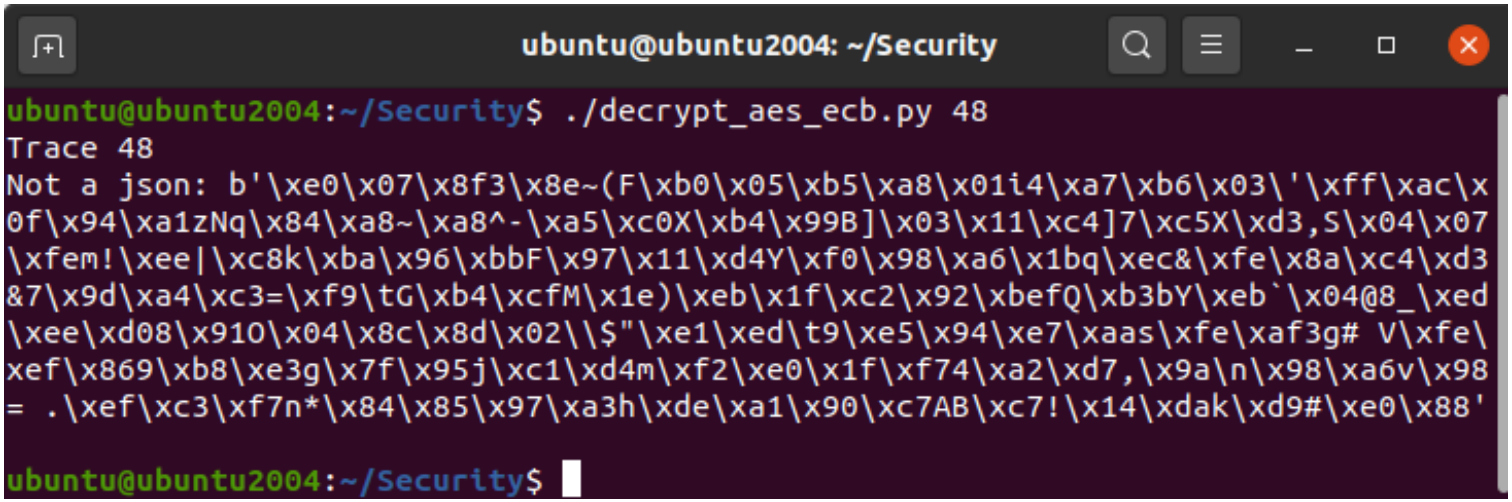
```
1  #!/usr/bin/python3
2
3  from Crypto.Cipher import AES
4  from binascii import unhexlify
5
6  aeskey="KEY"
7
8  wiresharkpacket = unhexlify("WireShark_Data_as_HexStream")
9
10 aesobj = AES.new(aeskey, AES.MODE_ECB)
11 print(aesobj.decrypt(wiresharkpacket[42:]))
```

Line 1, Column 1 Tab Size: 4 Python

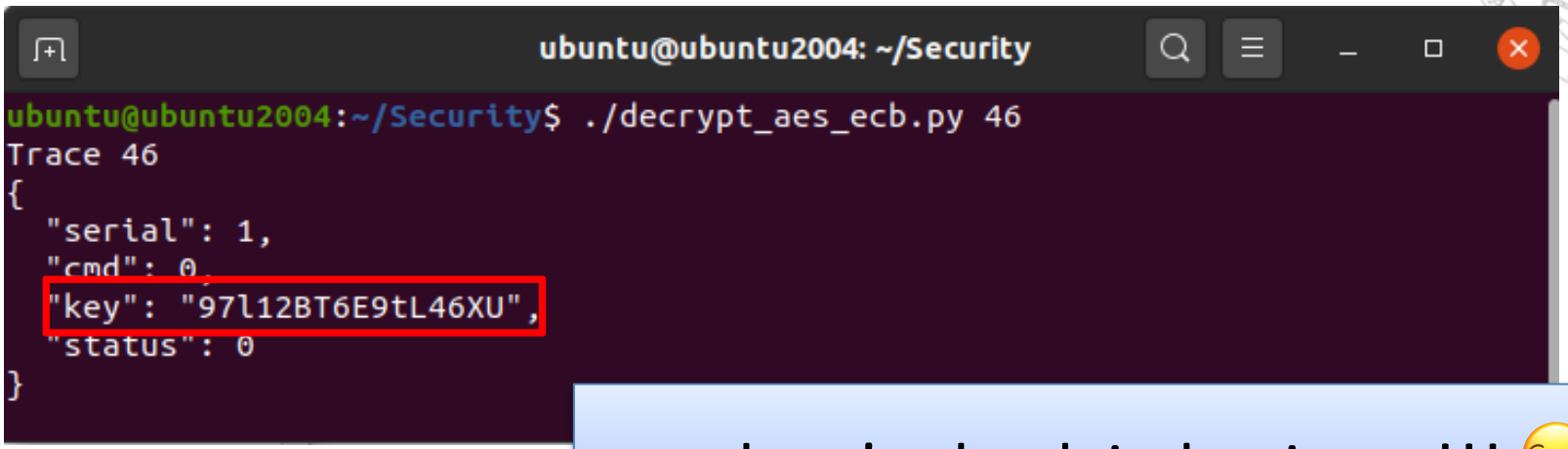
We decrypt data



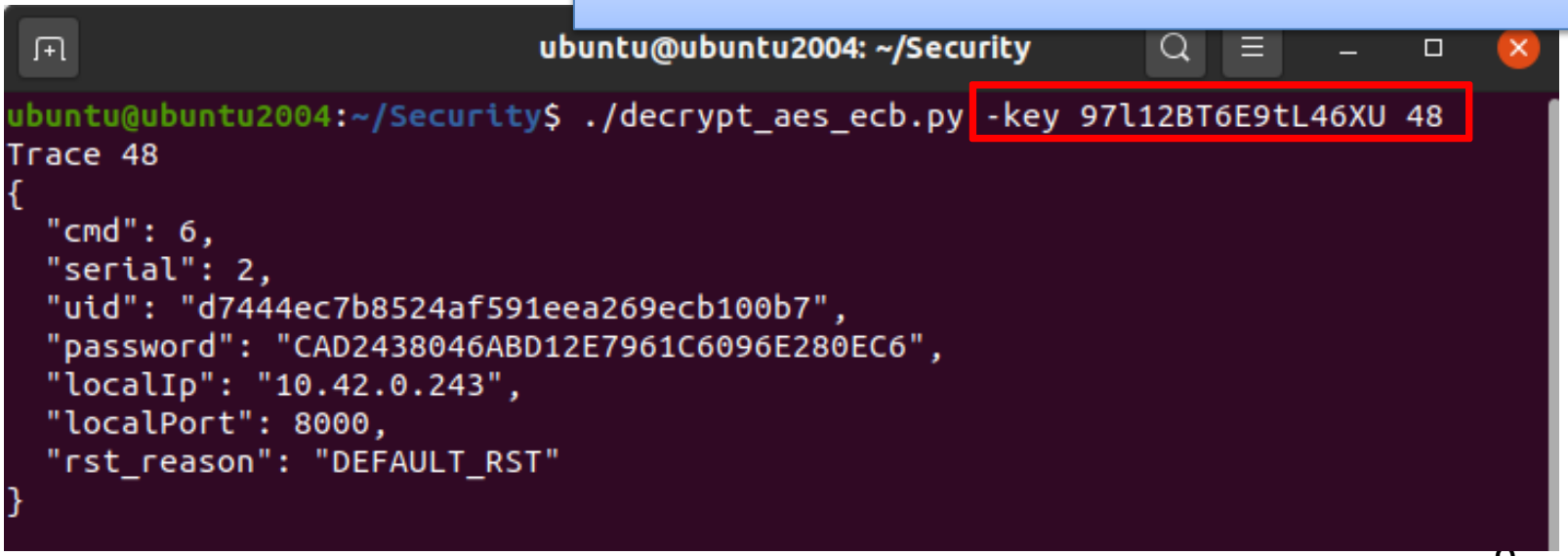
Does not work with trace 48 😭



Find new AES-ECB key



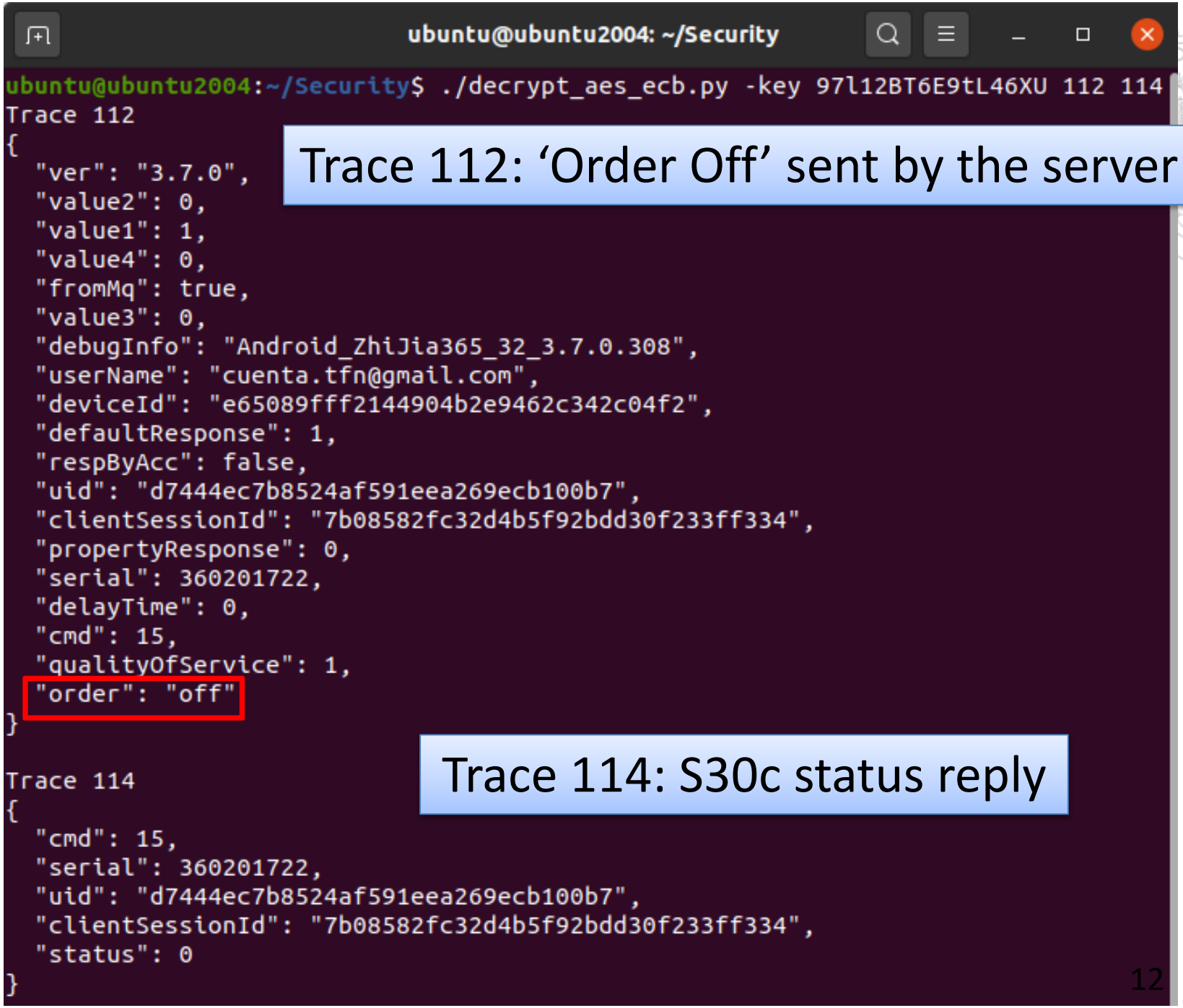
…and we're back in business!!! 😏



Status info sent to Orvibo

Trace 85: ‘Order On’ sent by the server

Trace 89: S30c status reply



Trace 112: ‘Order Off’ sent by the server

Trace 114: S30c status reply

Information on operation

- On and off:
  - Android app request server to turn on/off plug
  - The server communicates with the socket:
    - "order":"on"
    - "order":"off"
  - The socket responds with a status message
- ...in addition, the device periodically sends status information to the server, among other things...

Smart Socket Initial Setup

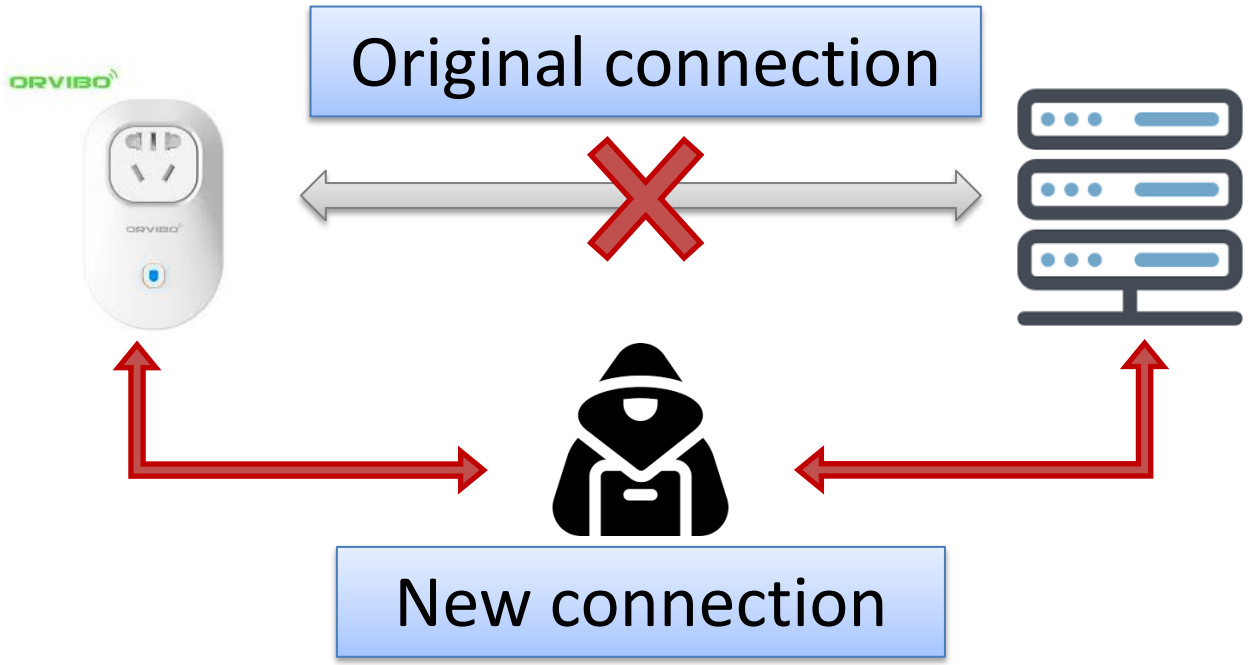
1. Initial Setup
2. Capture Traffic
3. Traffic analysis
4. Apk Analysis
5. Decrypt traffic
6. Take control of the device (MiM)
7. Pentest Project tasks

Can we take control of the device?

- We know how to encrypt/decrypt messages (symmetric encryption)
- We know the format of the on/off messages
- We know IPs and ports used by device and server
- We can do a MiTM (Man in the Middle) type attack

Man in the Middle

- Messages from the server (and their responses) reach the device by passing through the proxy.



- The proxy can inject new messages to the device, responses to these messages should be discarded.

Proxy *orvibocontrol.py*

1. Unplug the Smarsocket
2. Configure the system to redirect traffic from the device to the server through port 8080 and launch `orvibocontrol.py`:

```
iot@raspy-iot-da: ~
File  Edit  View  Search  Terminal  Help
iot@raspy-iot-da:~$ sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp
 -d 18.197.83.128 --dport 10001 -j REDIRECT --to-port 8080
[sudo] password for iot:
iot@raspy-iot-da:~$ ./orvibocontrol.py
[+] Main
```

Proxy *orvibocontrol.py*

3. Plug the Smartsocket

```
iot@raspy-iot-da: ~
File  Edit  View  Search  Terminal  Help
iot@raspy-iot-da:~$ sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp -d 18.197.83.128 --dport 10001 -j REDIRECT --to-port 8080
[sudo] password for iot:
iot@raspy-iot-da:~$ ./orvibocontrol.py
[+] Main
[+] Client connected
[+] HTTP Server started on 5555

[+] socket -> server
{u'hardwareVersion': u'v1.0.0', u'language': u'chinese', u'cmd': 0, u'softwareVersion': u'v3.0.9', u'mac': u'c45bbe61687a', u'ser
ial': 1, u'modelId': u'98431e8c9e834776b414a40b2d8ddefb'}
[+] server -> socket
{u'status': 0, u'serial': 1, u'cmd': 0, u'key': u'3MKe0y2YF0hgbJ1Q'}


[+] socket -> server
{u'rst_reason': u'DEFAULT_RST', u'localPort': 8000, u'cmd': 6, u'localIp': u'10.42.0.243', u'serial': 2, u'password': u'CAD243804
6ABD12E7961C6096E280EC6', u'uid': u'd7444ec7b8524af591eea269ecb100b7'}
[+] server -> socket
{u'rst_reason': u'DEFAULT_RST', u'status': 0, u'uid': u'd7444ec7b8524af591eea269ecb100b7', u'cmd': 6, u'localIp': u'10.42.0.243',
 u'serial': 2, u'password': u'CAD2438046ABD12E7961C6096E280EC6', u'localPort': 8000}
```

Proxy *orvibocontrol.py*

4. Open Firefox in the Raspi on 127.0.0.1:5555

Smart Socket Initial Setup

1. Initial Setup
2. Capture Traffic
3. Traffic analysis
4. Apk Analysis
5. Decrypt traffic
6. Take control of the device (MiM)
7. Pentest Project tasks

Smart Socket Pentest Tasks

Create a document with the following points, for this use screenshots and comment all the steps:

1. Get the key and encryption method:
    1. Download Home Mate Apk
    2. Analyze it with JADX to find the `decryptByte` function
    3. Unzip the APK and locate the library where the message encryption function is located (rgrep decryptByte)
    4. Disassemble the library file using Binary Ninja and locate the encryption key and method

Smart Socket Pentest Tasks

2. Decrypt messages:
    2. Record an On/off sequence trace with WireShark
    3. Use the cipher key of Step 1 and decryptaes.py script to decrypt the first SmartPlug messages
    4. Find the new cipher key
    5. Decipher an on/off messages and the response from the SmartPlug

Smart Socket Pentest Tasks

3. Analyze and comment `orvibocontrol.py` proxy code:
   1. How many threads and queues are created?
   2. How many sockets are used? What's its purpose?
   3. Create a block diagram of the code
4. Comment possible measures to improve security of the Smartsocket

This practice represents a total of 30% of the grade.
Send me the corresponding memory until June 3 (recas@ucm.es).