Sneak into buildings with KNXnet/IP

using BOF

whoami

Claire Vacherot

Senior pentester @ Orange Cyberdefense France

Random info:

- Writing tools (and then discover they already exist)
- Part of GreHack's organization team
- Penetration testing on "unusual" environments

Best conference (after Defcon)

PROGRAMMING

SECURE PROGRAMMING

... ON EMBEDDED/ICS DEVICES

EMBEDDED/ICS DEVICE SECURITY

ICS SECURITY

Disclaimer

Please be careful ☺

Testing industrial systems is dangerous

- Control of physical processes

Impact on people's safety

- Accidents
- Disabling safety controls

Test in controlled environments

BMS

KNX

BOF

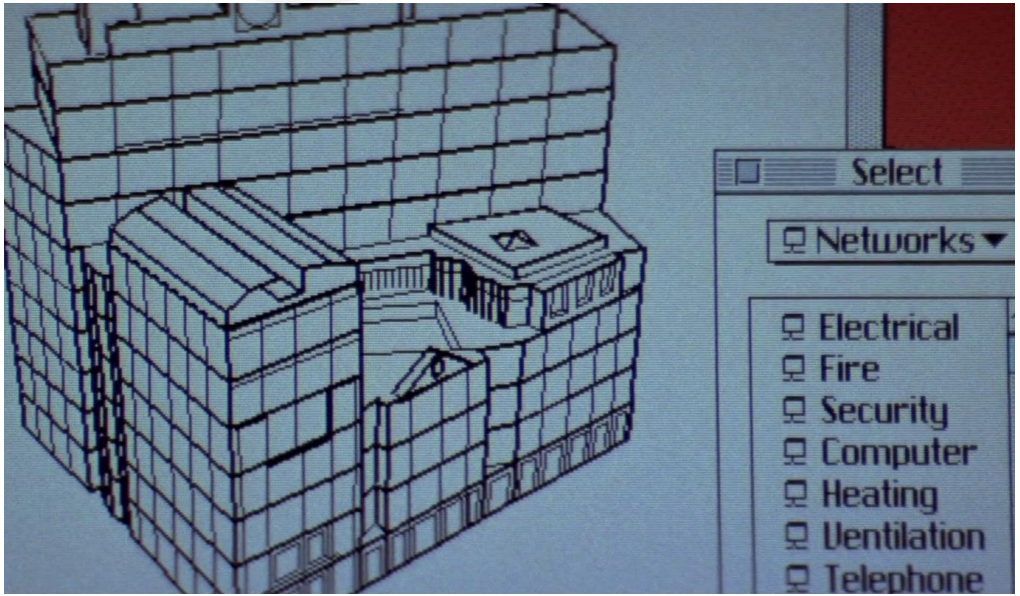
BMS

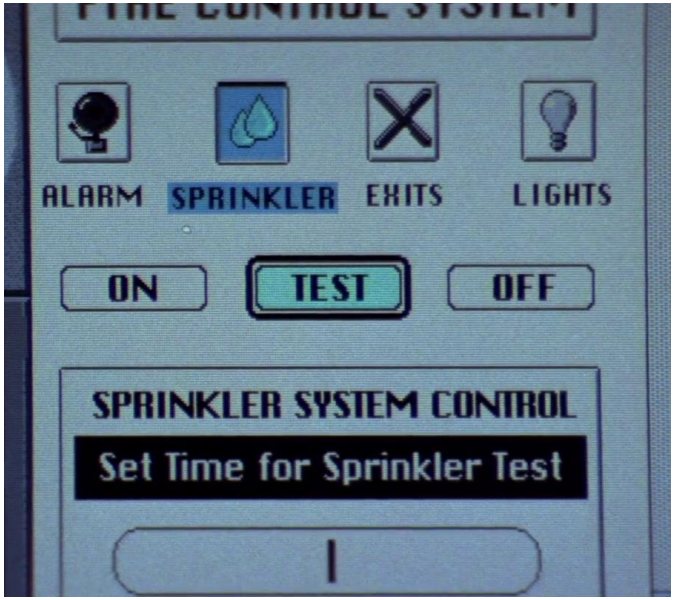
Building Management/Automation System

- Components automation & control
- Home, buildings, factories, hospitals, offices, …

- HVAC
- Lighting
- Shutters
- Elevators
- Access control
- Intrusion detection
- Safety and security

BMS

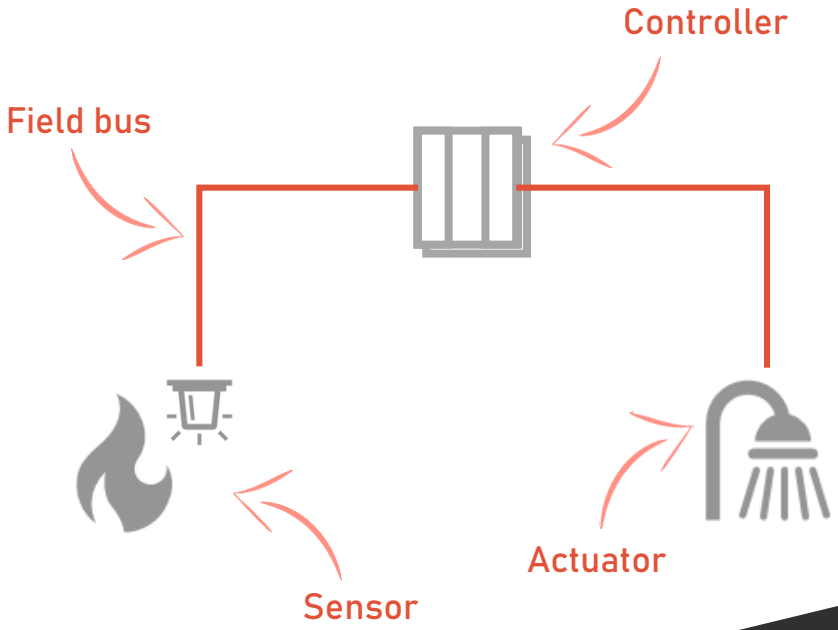




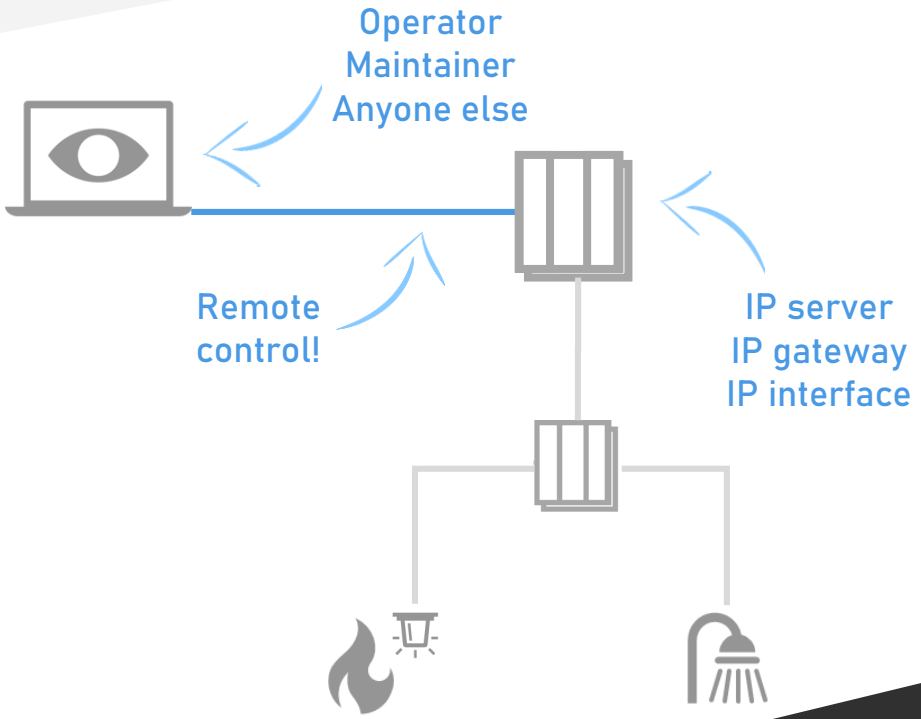
* "Hackers“ (1995) has a BMS hacking scene

« The pool on the roof has a leak. »

The « field » network



The « IP » network

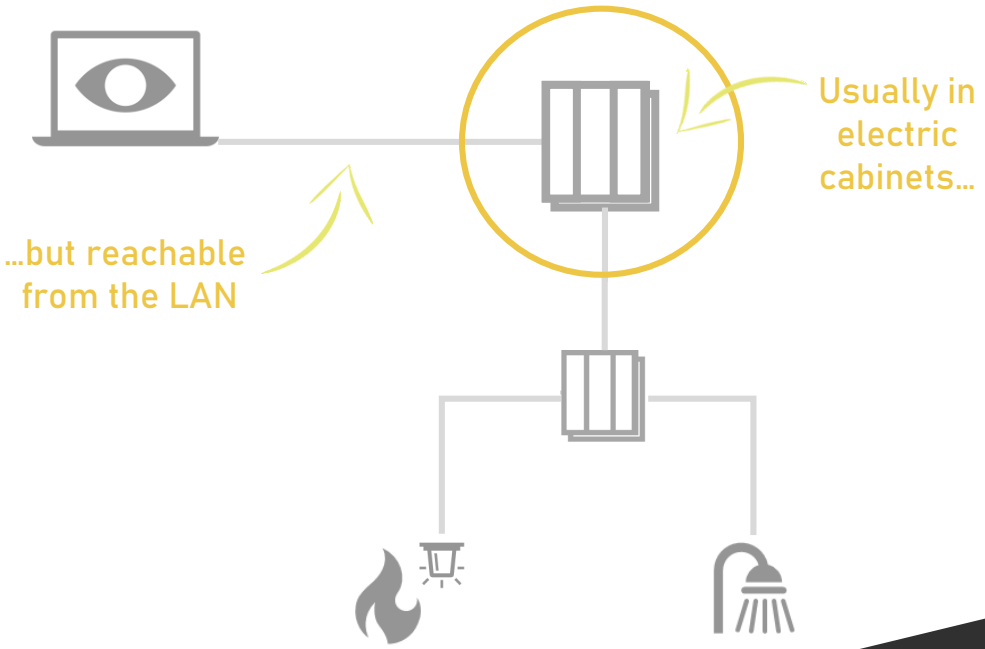


Why should we take a look?

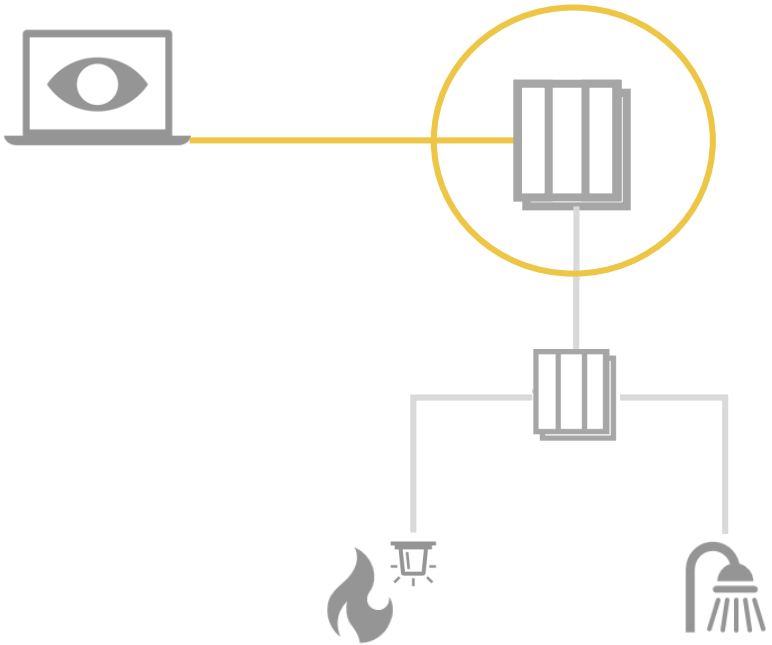
Exposing « industrial » protocols and devices

- Legacy software and protocols
- Not designed to handle cybersecurity issues
- Not operated with cybersecurity in mind

The « interface » device



Let's scan it!



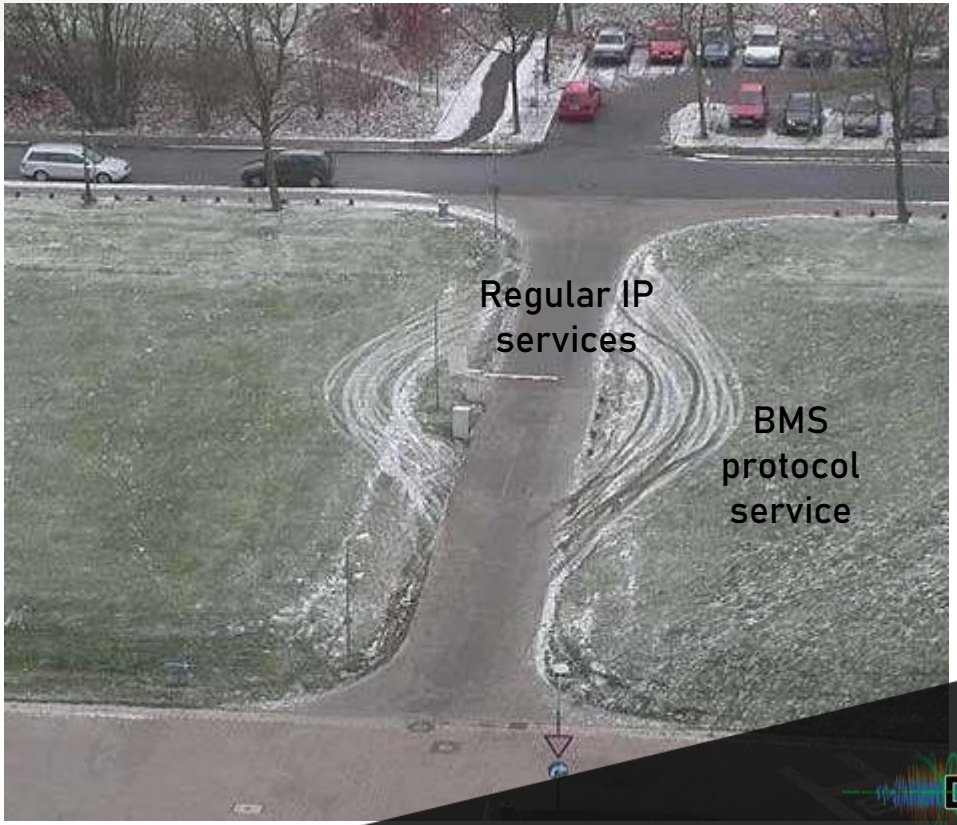
- 21/tcp ftp
- 22/tcp ssh
- 23/tcp telnet
- 80/tcp http
- 443/tcp https

Admin

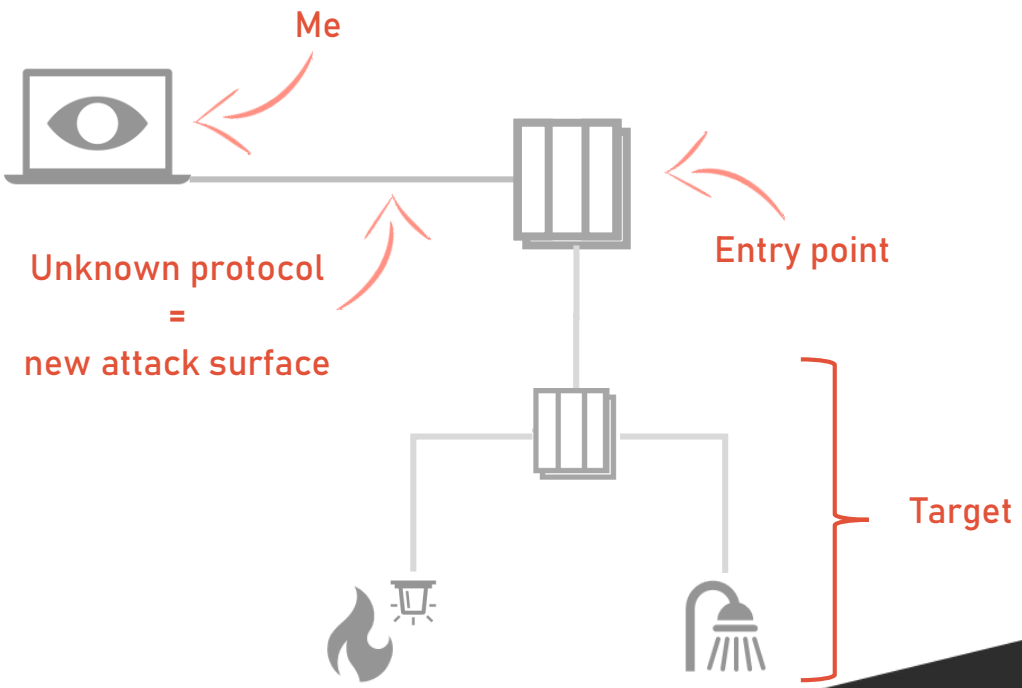
- 3671/udp knxnet
- 47808/udp bacnet

BMS link

Why should we take a look?



BMS security wrap up



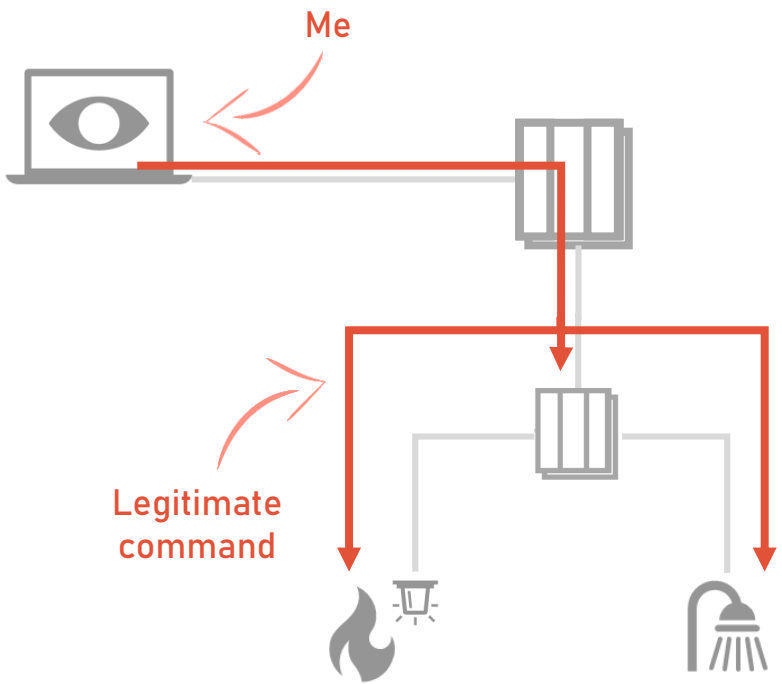
BMS security wrap up

What can we do with that?

#1 Send valid stuff

#2 Send invalid stuff

Attack scenario #1
Change BMS behavior



- Enable sprinklers
- Disable fire detection
- Change thresholds
- Turn everything off
- …

Attack scenario #1
Change BMS behavior

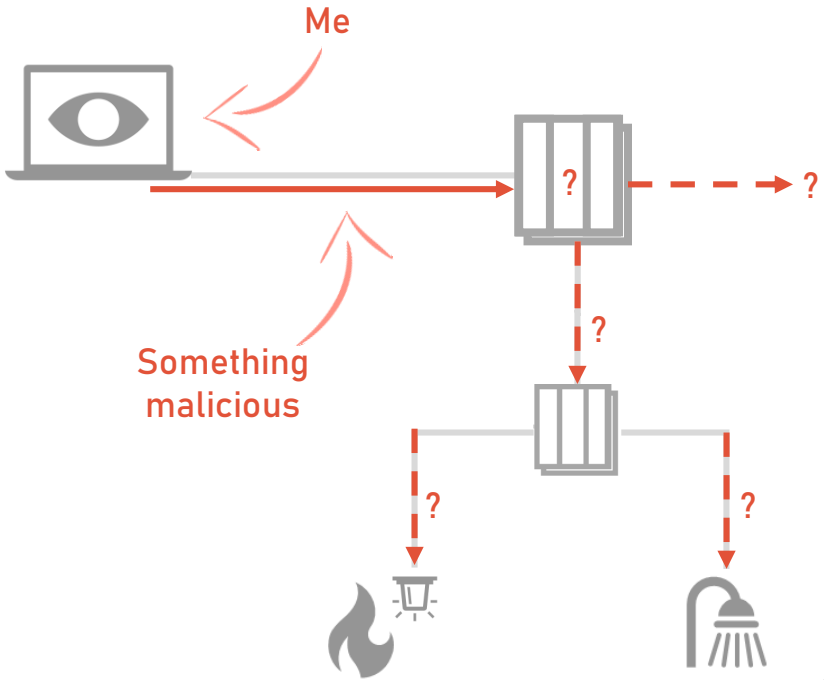
Example

I have no idea what they mean

- Listen to the traffic
- Replaying BACnet frames
- Turning off that HVAC

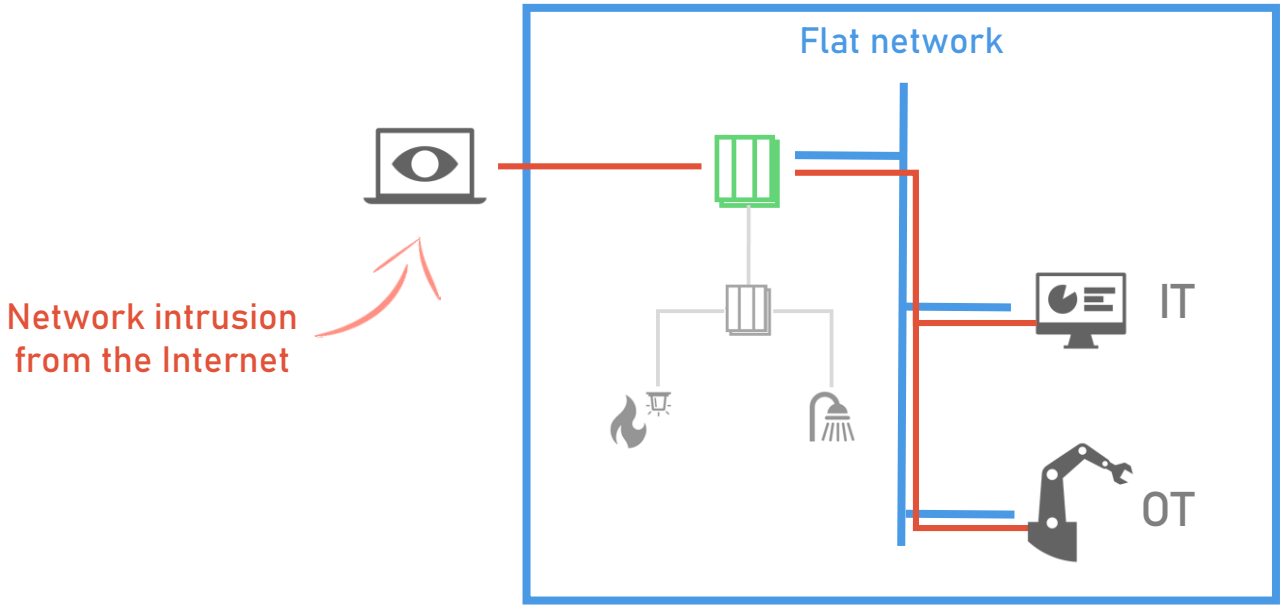
```
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # UDP
s.connect(("192.168.1.1", 47808))
try:
    while 1:
        s.send(payload_ventilation_off)
        sleep(1) # (Different kind of) DoS if we don't wait
except KeyboardInterrupt:
    pass
s.close()
```

Attack scenario #2
Unintended use of devices

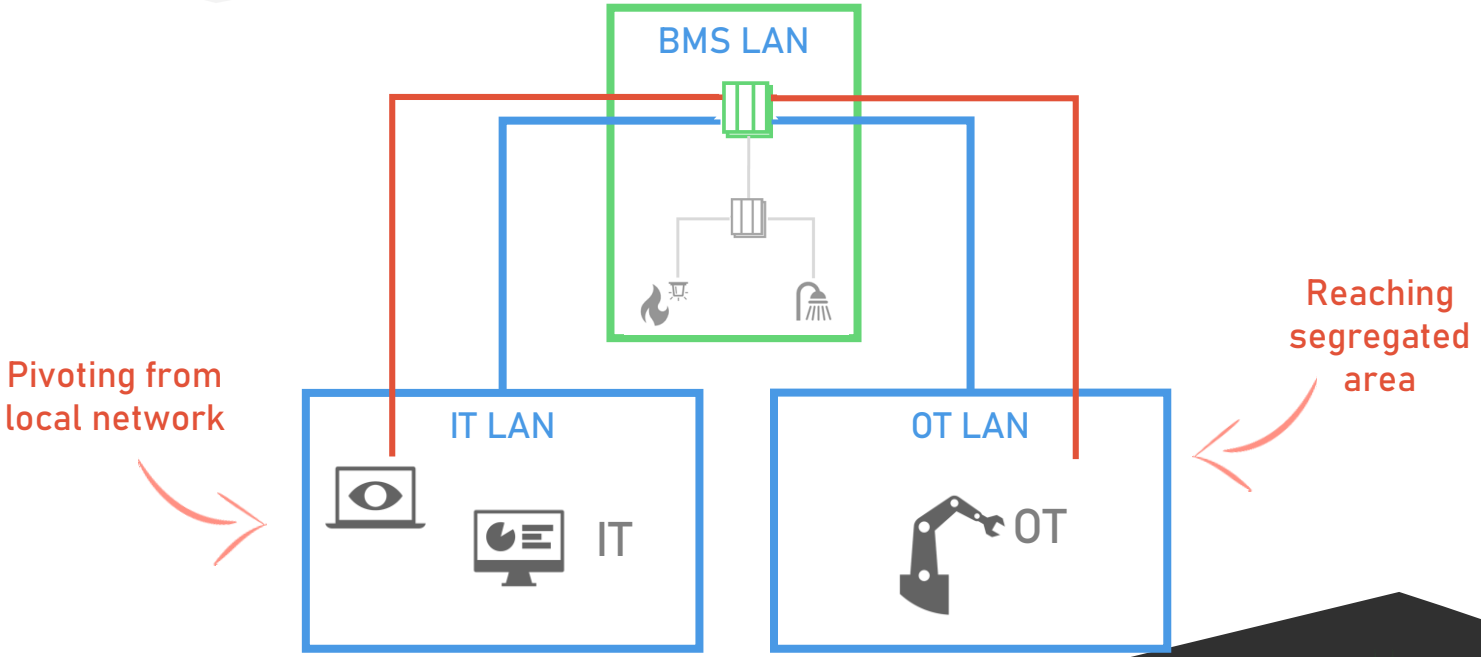


- ???

Attack scenario #2
Unintended use of devices



Attack scenario #2
Unintended use of devices



What we know so far

Introduction to BMS security

- InSecurity in Building Automation

  Thomas Brandstetter @ Defcon 25 (2017)

  KNX security or how to steal a skyscraper - Yegor Litvinov (2015)

  Pwning KNX & ZigBee Networks - HuiYu Wu, YuXiang Li & Yong Yang (2018)

You should watch it!

BMS exploitation talks (discovery) – Attack Scenario #1

- Learn How to Control Every Room at a Luxury Hotel Remotely

  Jesus Molina @ Defcon 22 (2014)

What we know so far

Cool stuff on fuzzing BACnet

Advanced attacks / fuzzing – Attack scenario #2

- HVACking Understand the Delta Between Security and Reality
  Douglas McKee & Mark Bereza @ Defcon 27

Attack remediation, detection

- Anomaly Detection in BACnet/IP managed Building Automation Systems
  Matthew Peacock, 2019

What we know so far

… where is KNX? ☹

BMS

KNX

BOF

And now... KNX!

- SEVERAL EUROPEAN STANDARDS (1980s)
- 3 STANDARDS MERGED INTO KNX (1999)
- KNXNET/IP (2007)
- 1ST SECURITY EXTENSION (2013)
- KNX STANDARD FREE (2016)

And now... KNX!

- Hard to find / use documentation
- Few research & work about KNX security

That does not mean there is nothing to say about it ☺

And now... KNX!

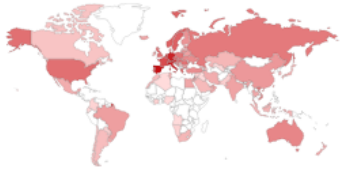
Checked 2021-07-01

SHODAN | Explore | Downloads | Pricing | knx

TOTAL RESULTS

14,980

TOP COUNTRIES



Spain	2,438
Italy	1,839
Germany	1,660
Austria	1,015
Belgium	734

More...

KNXnet/IP port

TOP PORTS

3671	13,558
22	391
80	183
443	138
1024	97

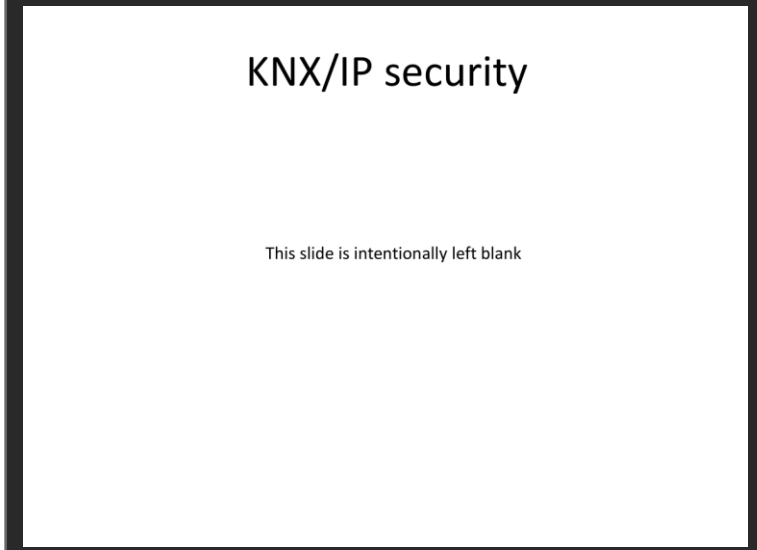
More...

« A minor concern »

« For KNX, security is a minor concern, as any breach of security requires local access to the network »

- Authentication as an option
  - Disabled by default (when implemented)
- Security extensions
  - KNX IP Secure, KNX Data Secure

Security is optional!



Tribute to Molina

« A minor concern »

"It is quite unlikely that legitimate users of a network would have the means to intercept, decipher, and then tamper with the KNXnet/IP without excessive study of the KNX Specifications."

- KNX Standard v2.1

« A minor concern »

"It is quite unlikely that legitimate users of a network would have the means to intercept, decipher, and then tamper with the KNXnet/IP without excessive study of the KNX Specifications."

- KNX Standard v2.1

HOLD MY BEER KNX

Where to start
The boring way

- KNX specifications free since 2016
  - You just need a ~~fake~~ account on KNX's website
  - …and also good nerves

148 PDF files \o/

Only 33 PDF files!

- « Volume 3 – System Specifications » is the useful part

Where to start
The hacker way

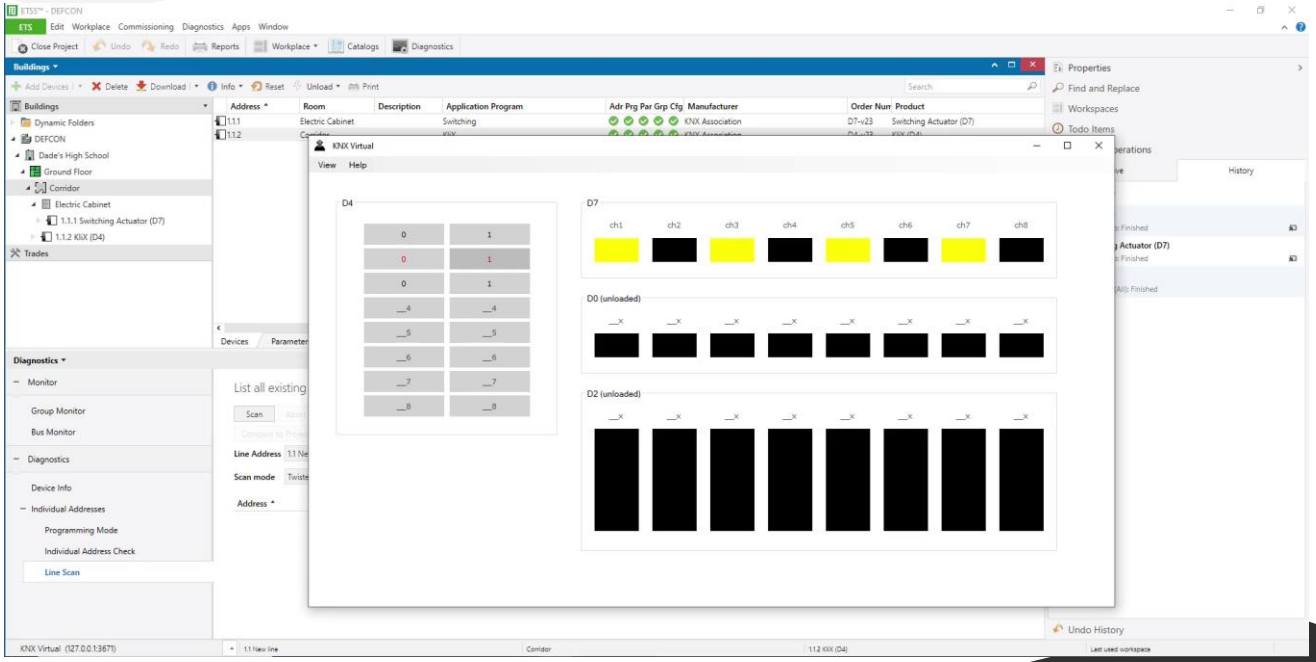
1. Set up a test environment with KNX Virtual and ETS
2. Listen to the traffic and learn
3. …or just replay it

} Works with (almost) any industrial/BMS network protocol

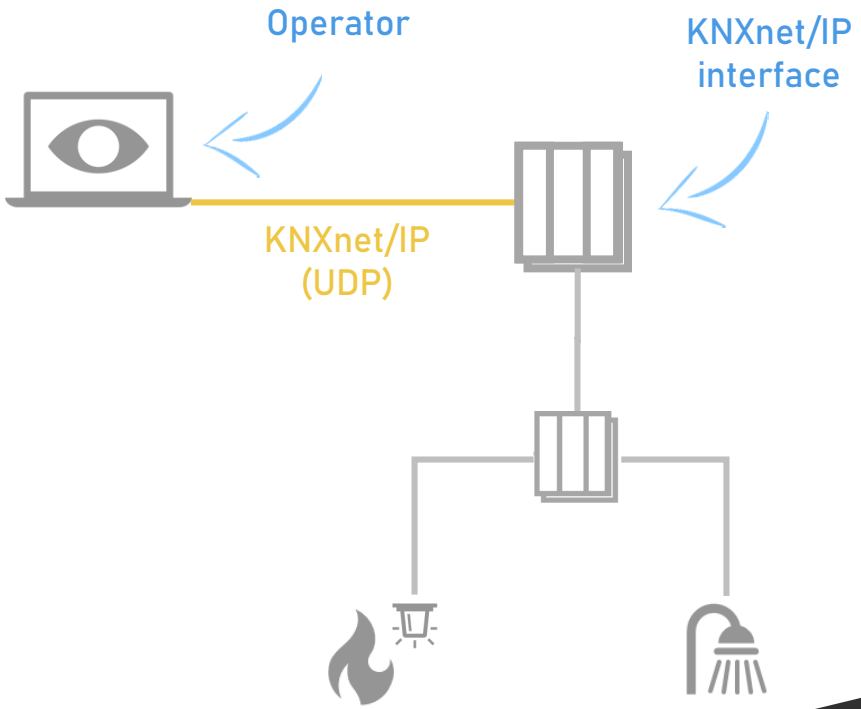
- « Engineering Tool Software » (ETS) provided by KNX association
  « The best hacking tool », according to Thomas Brandstetter

- Wireshark has a KNXnet/IP dissector

DEMO : Setting up a test environment



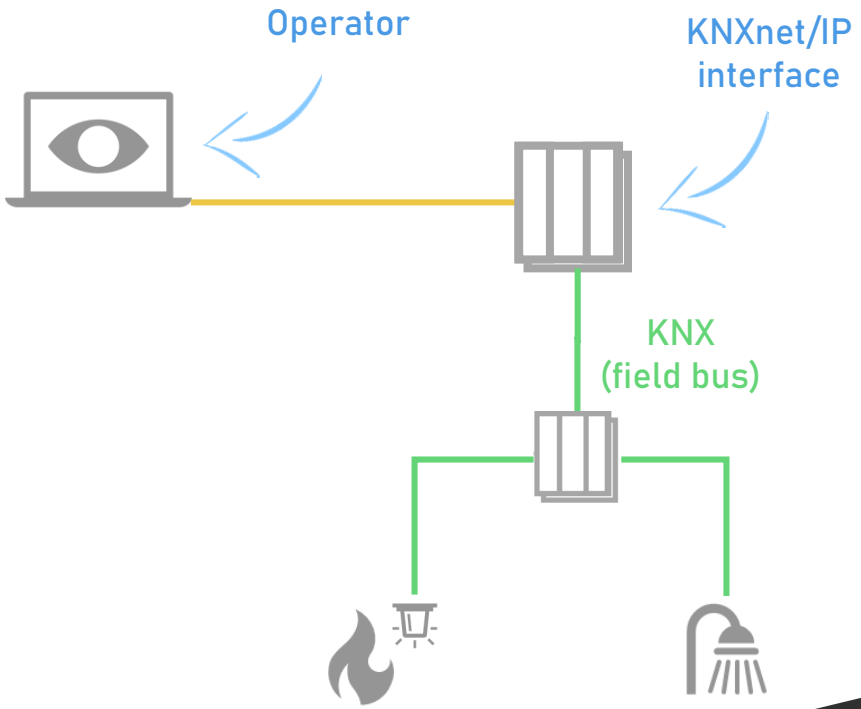
« Deciphering » KNX

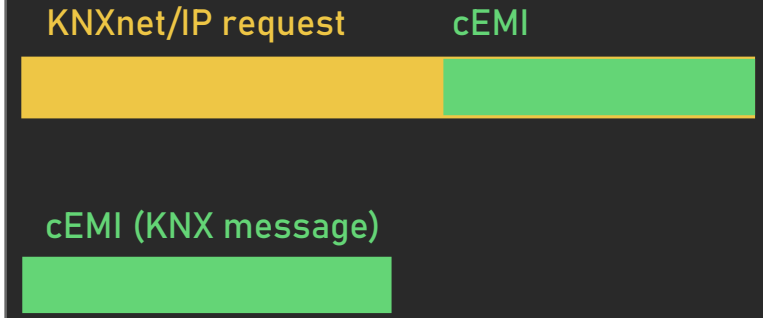


KNXnet/IP request

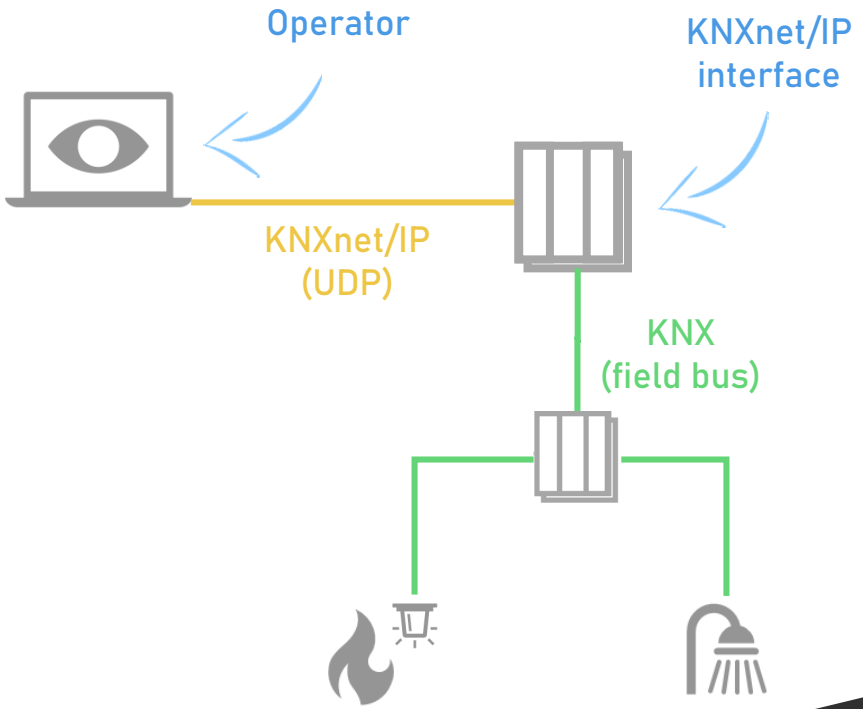
KNXnet/IP request cEMI

« Deciphering » KNX





« Deciphering » KNX



KNXnet/IP

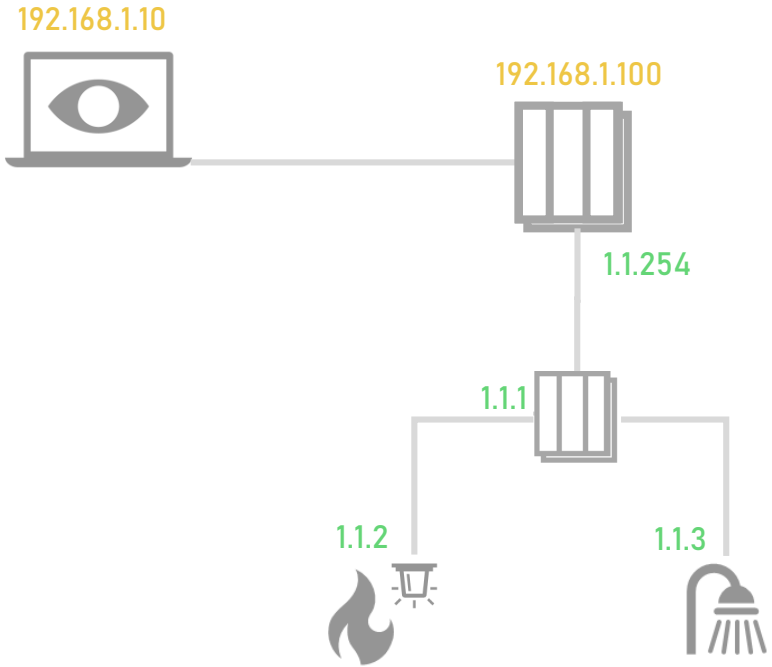
+

KNX (cEMI)

=

Not one, but 2 protocols to test!

« Deciphering » KNX



- KNX Individual Address (1.1.1) = devices
  Ex: scan KNX network
- KNX Group Address (1/1/1) = « functions »
  Ex: run commands

KNXnet/IP embedding a KNX frame

Tooling to start testing

- ETS
- KNXmap
  https://github.com/takeshixx/knxmap
- New : KNXnet/IP layer for Scapy
  - https://github.com/secdev/scapy
  - Layer by Julien Bedel @ Orange Cyberdefense

Also, thanks to Scapy maintainers for their support and kindness ☺

DEMO : KNXmap

```
lex  DEFCON29  Demo  knxmap scan 192.168.1.242
Scanning 1 target(s)

192.168.1.242
   Port: 3671
   MAC Address: 00:6[redacted]
   KNX Bus Address: 1.1.254
   KNX Device Serial[redacted]
   KNX Medium: KNX Tr
   Device Friendly Name: boiboite
   Device Status:
      Programming Mode: disabled
      Link Layer: disabled
      Transport Layer: disabled
      Application Layer: disabled
      Serial Interface: disabled
      User Application: disabled
      BC DM: 0
   Project Install Identifier: 0
   Supported Services:
      KNXnet/IP Core
      KNXnet/IP Device Management
      KNXnet/IP Tunnelling
      KNXnet/IP Routing

Scan took 0.0052754878997802734 seconds
lex  DEFCON29  Demo
```

Tooling to start testing

- ETS
- KNXmap

  https://github.com/takeshixx/knxmap
- New : KNXnet/IP layer for Scapy
  - https://github.com/secdev/scapy
  - Layer by Julien Bedel @ Orange Cyberdefense

Also, thanks to Scapy maintainers for their support and kindness ☺

Tooling to start testing

- Suitable for basic interaction
- Limitations for extensive testing

= Opportunity for a new tool! :D

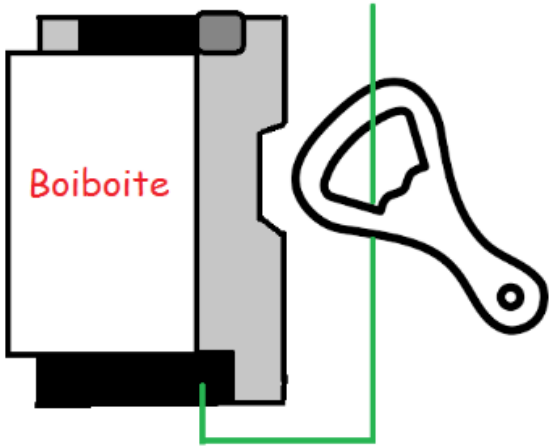
BMS

KNX

BOF

BOF: Boiboite Opener Framework

For discovery, basic interaction and advanced testing via industrial network protocols (including KNXnet/IP)



- Python 3.6+ library
- Originally created to write attack scripts
  - Change devices' behavior (#1)
  - Test protocol implementations on devices (#2)

- https://github.com/Orange-Cyberdefense/bof

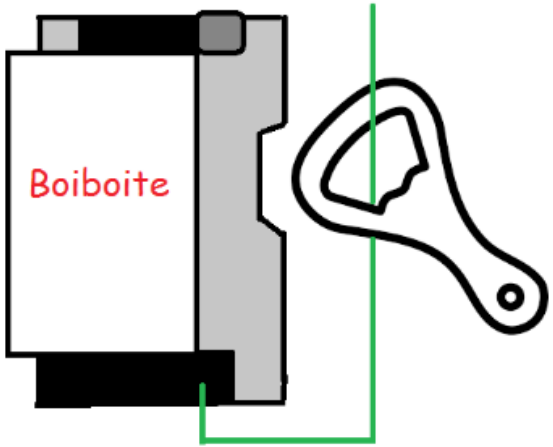
There is at least one user (me)

I use it during pentests…

- For basic network and devices discovery
- To send basic commands
- To write attack scripts

…and for my own vulnerability research

- Dumb and not-so-dumb fuzzing
- To write very specific attack scripts



Scapy + BOF = <3

« Why not use Scapy? »

Very good question

Pros:

- Nothing better for protocol implementations

Cons:

- Incompatibilities with BOF's expected behavior
- Willing to keep BOF's script syntax

Scapy + BOF = <3

« Why not use Scapy? »

Very good question

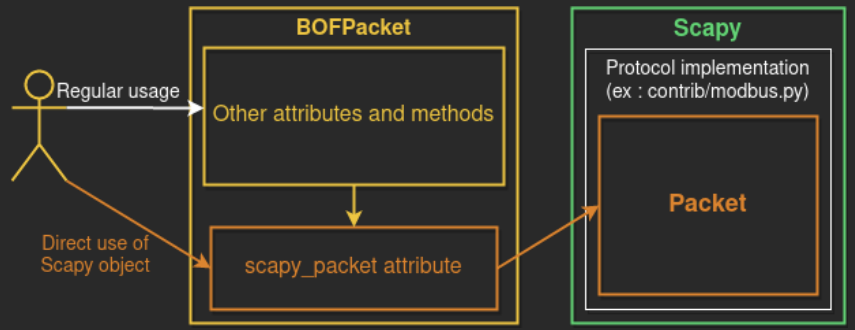
Pros:

- Nothing better for protocol implementations

Cons:

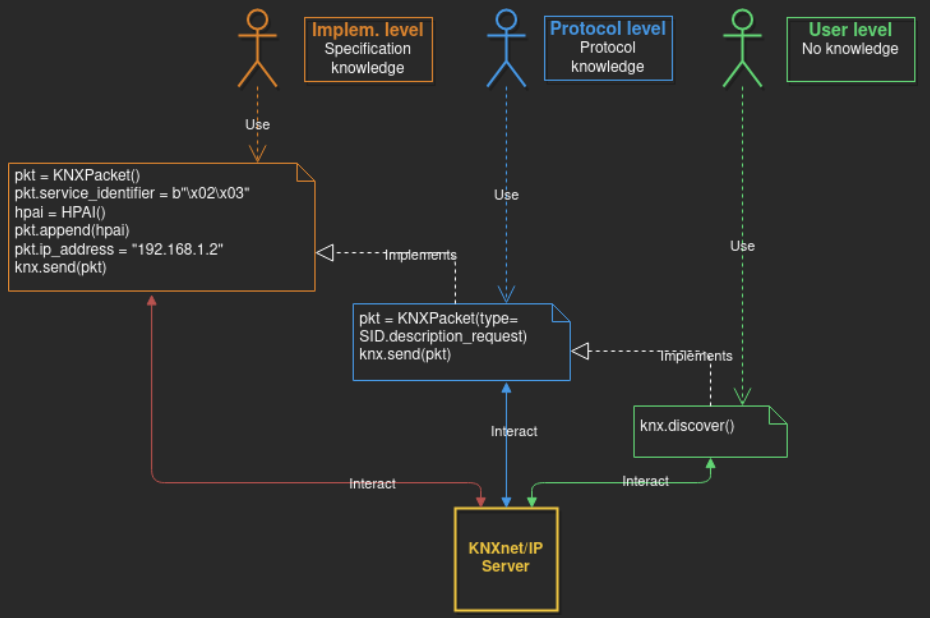
- Incompatibilities with BOF's expected behavior
- Willing to keep BOF's script syntax

Now that you mention it...

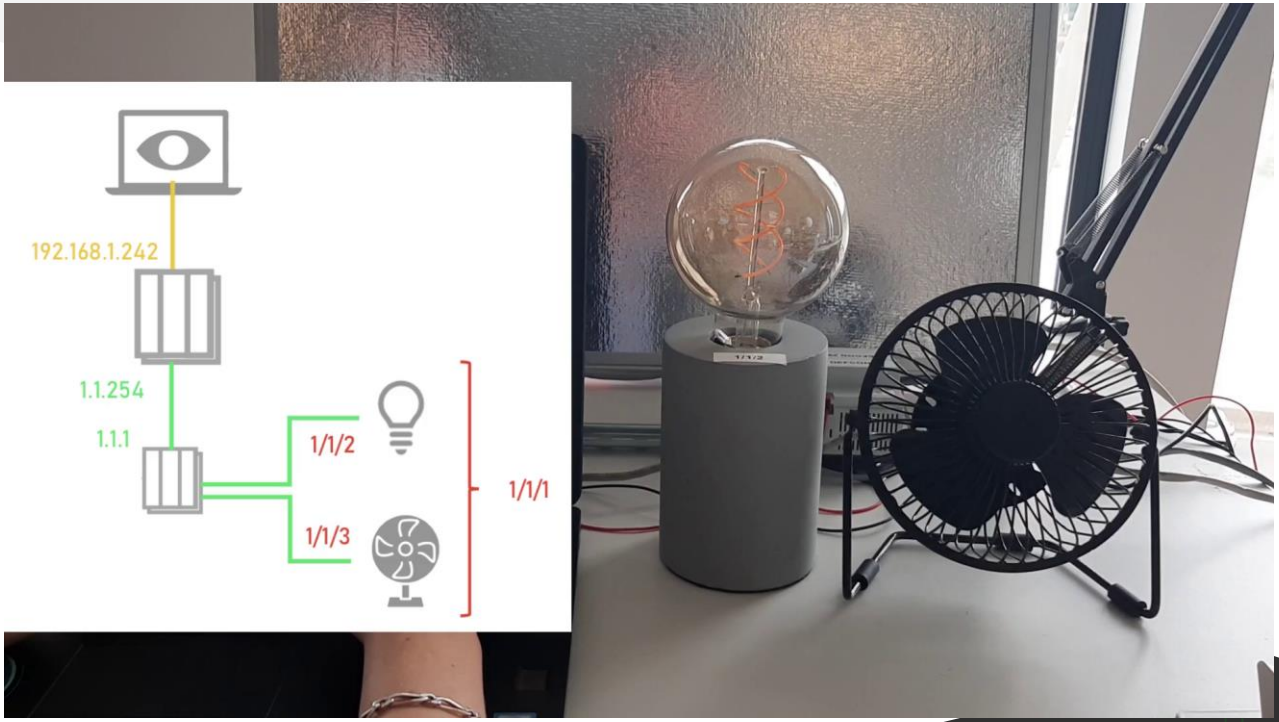


How does this work?

- High-level discovery
- Intermediate usage
- Low-level testing



DEMO : Discovery

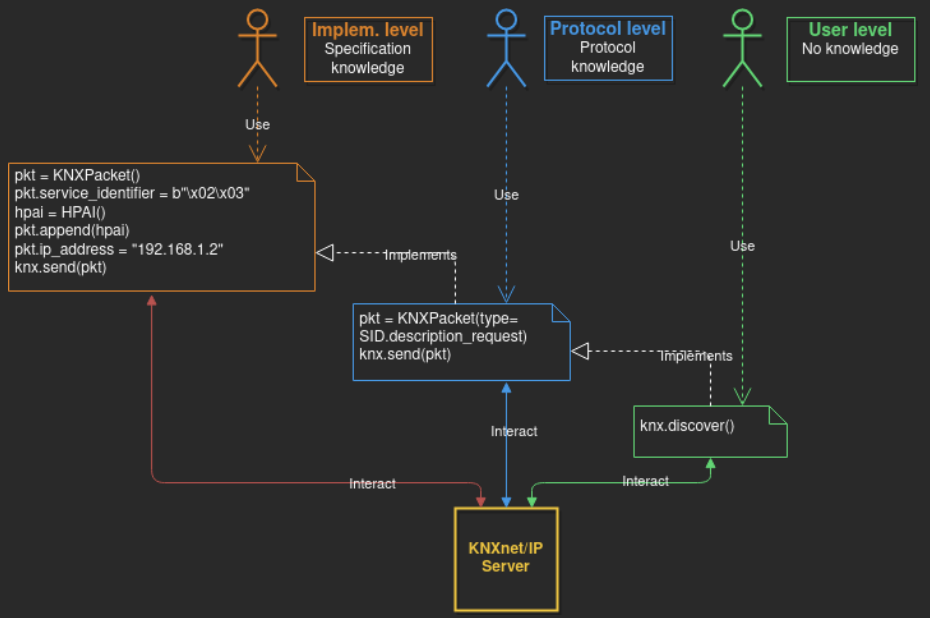


DEMO : Discovery

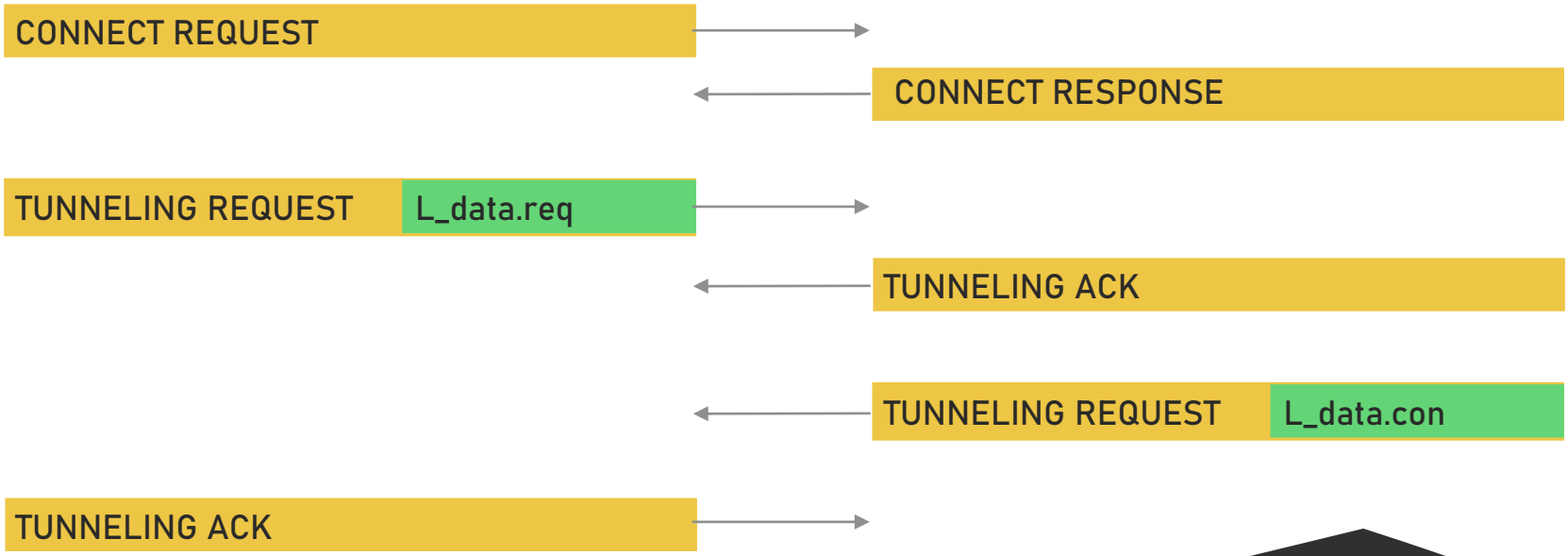
```
lex  DEFCON29  Demo  python
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from bof.layers import knx
>>> knx.group_write("192.168.1.242", "1/1/1", 1)
<bof.layers.knx.knx_packet.KNXPacket object at 0x7f4a8eb7ad68>
>>>
```

How does this work?

- High-level discovery
- Intermediate usage
- Low-level testing



Turning off the lights

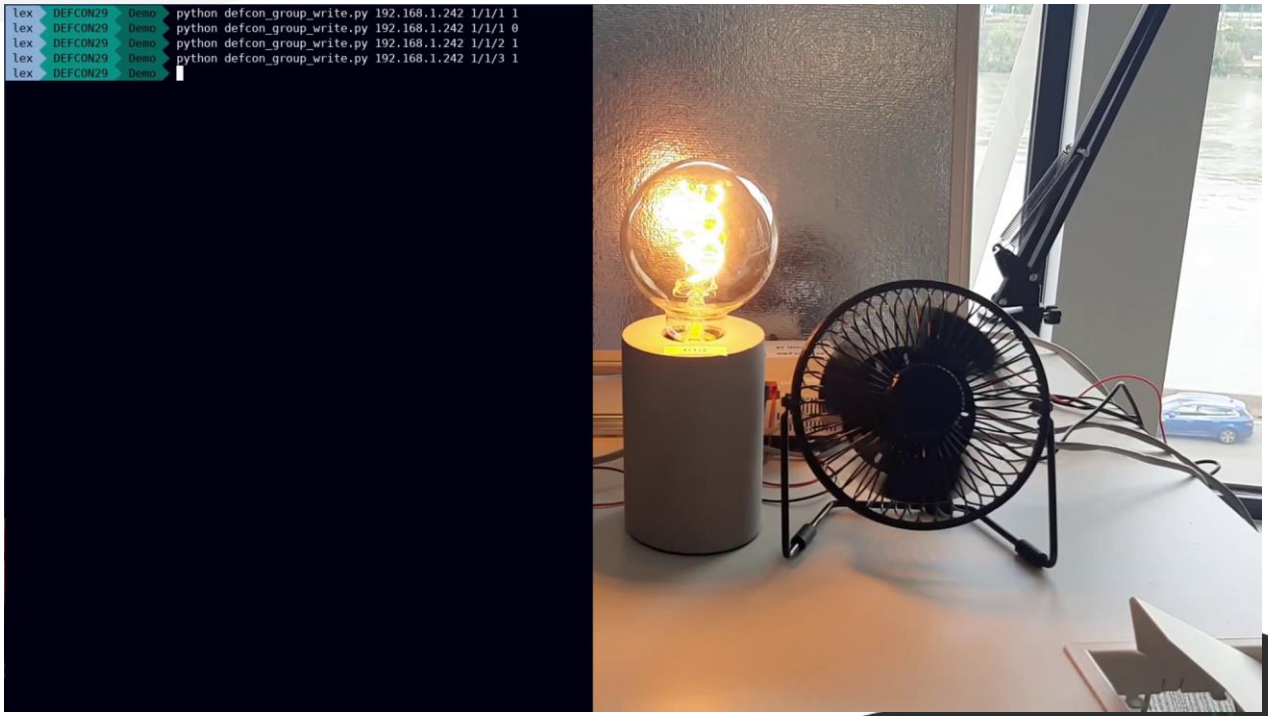


Yes its UDP

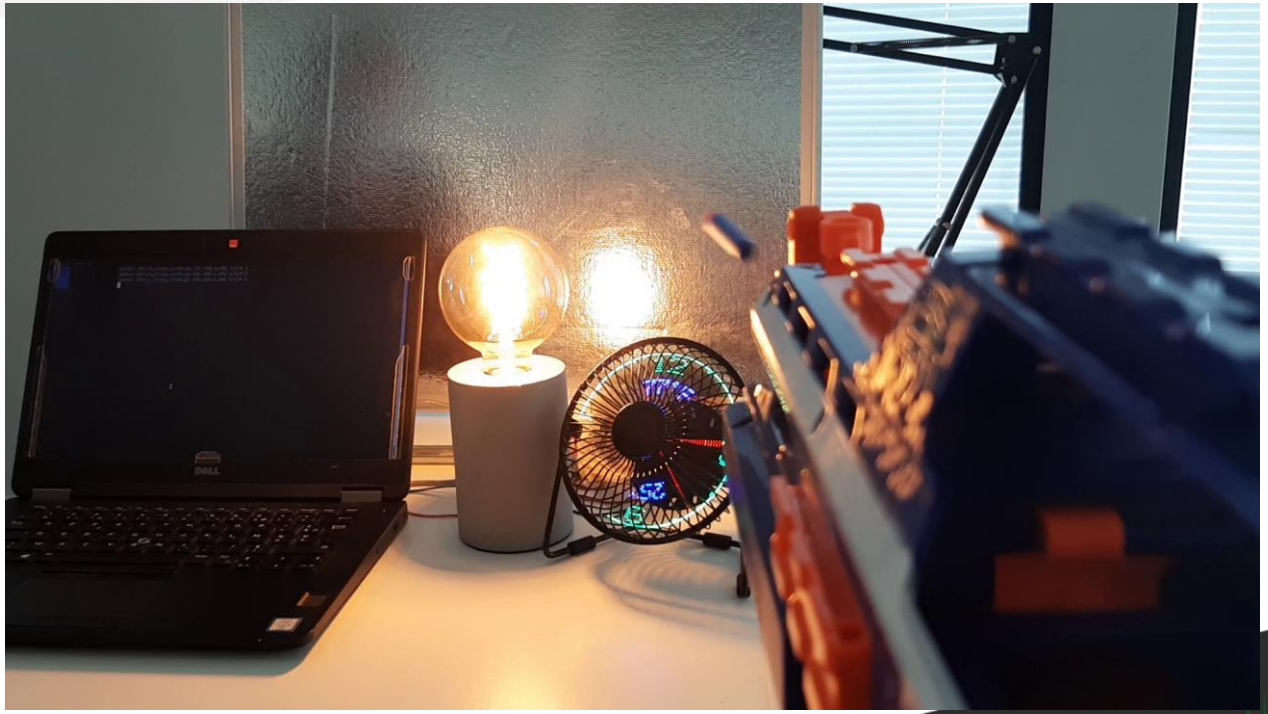
Turning off the lights

```
# CONNECT REQUEST
channel, knx_source_addr = connect_request_tunneling(knxnet)

# cEMI
cemi = cemi_group_write(knx_source_addr, KNX_GROUP_ADDR, VALUE)

# TUNNELING REQUEST (broken down)
tun_req = KNXPacket(type=SID.tunneling_request)
tun_req.communication_channel_id = channel
tun_req.cemi = cemi

# SEND and wait for ACK and RESPONSE
ack, _ = knxnet.sr(tun_req)
response, source = knxnet.receive()

# SEND ACK and DISCONNECT REQUEST
...
```

DEMO : Basic operation

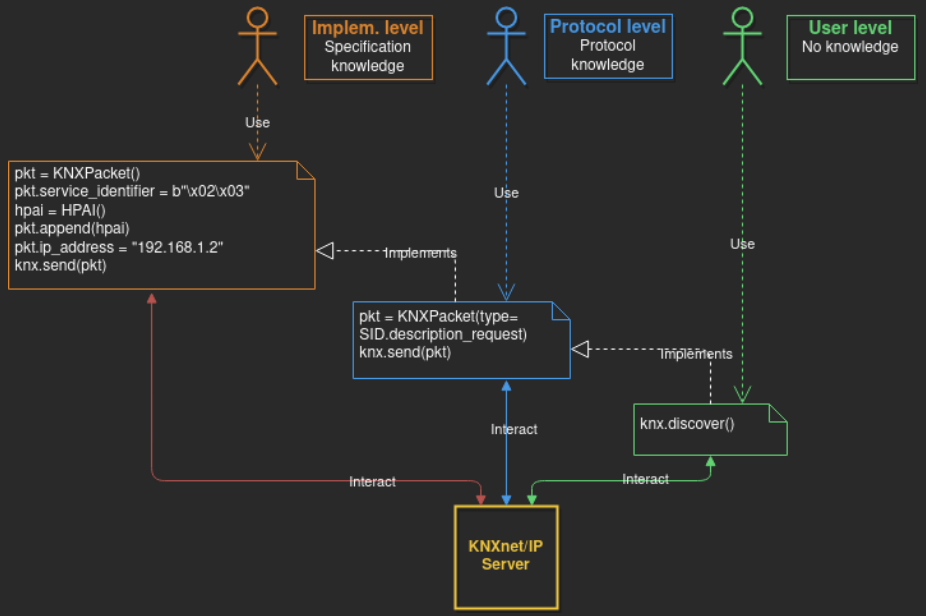


DEMO : Basic operation



How does this work?

- High-level discovery
- Intermediate usage
- Low-level testing



DEMO : Advanced testing

```
###[ KNXnet/IP ]###
  header_length= 6
  protocol_version= 0x10
  service_identifier= CONFIGURATION_REQUEST
  total_length= 17
###[ CONFIGURATION_REQUEST ]###
     structure_length= 4
     communication_channel_id= 1
     sequence_counter= 0
     reserved  = 0
     \cemi      \
      |###[ CEMI ]###
      |  message_code= M_PropRead.req
      |  \cemi_data \
      |   |###[ DP_cEMI ]###
      |   |  object_type= 0
      |   |  object_instance= 1
      |   |  property_id= 0
      |   |  number_of_elements= 1
      |   |  start_index= 0

*** START: 21-07-11-19:06:39 ***
535 requests sent, 0 event(s)... (Ctrl+C to stop)
```

udp && ip.addr == 192.168.1.242

No.	Time	Source	Destination	Protocol	Length	Info
1856	48.075742619	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:200 M_PropRead.req OT=0 OI=0 I
1857	48.076337340	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:200 OK
1858	48.076574824	192.168.1.242	192.168.1.33	KNXnet/IP	61	? ConfigReq #01:200 M_PropRead.con OT=0 OI=0 I
1859	48.078668264	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:200 OK
1860	48.081507589	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:201 M_PropRead.req OT=0 OI=0 I
1861	48.081984082	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:201 OK
1862	48.082260925	192.168.1.242	192.168.1.33	KNXnet/IP	60	? ConfigReq #01:201 M_PropRead.con OT=0 OI=0 I
1863	48.084303926	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:201 OK
1864	48.086616856	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:202 M_PropRead.req OT=0 OI=0 I
1865	48.087189111	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:202 OK
1866	48.087413668	192.168.1.242	192.168.1.33	KNXnet/IP	61	? ConfigReq #01:202 M_PropRead.con OT=0 OI=0 I
1867	48.089352980	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:202 OK
1868	48.091464479	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:203 M_PropRead.req OT=0 OI=0 I
1869	48.092037600	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:203 OK
1870	48.092291889	192.168.1.242	192.168.1.33	KNXnet/IP	60	? ConfigReq #01:203 M_PropRead.con OT=0 OI=0 I
1871	48.094225860	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:203 OK
1872	48.096381219	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:204 M_PropRead.req OT=51600 O
1873	48.096930376	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:204 OK
1874	48.097184577	192.168.1.242	192.168.1.33	KNXnet/IP	60	? ConfigReq #01:204 M_PropRead.con OT=51600 O
1875	48.099405401	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:204 OK
1876	48.101913860	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:205 M_PropRead.req OT=0 OI=0 I
1877	48.102464120	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:205 OK
1878	48.102667235	192.168.1.242	192.168.1.33	KNXnet/IP	61	? ConfigReq #01:205 M_PropRead.con OT=0 OI=0 I
1879	48.104860460	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:205 OK
1880	48.107907502	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:206 M_PropRead.req OT=0 OI=0 I
1881	48.107549188	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:206 OK
1882	48.107798012	192.168.1.242	192.168.1.33	KNXnet/IP	61	? ConfigReq #01:206 M_PropRead.con OT=0 OI=0 I
1883	48.110593777	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:206 OK
1884	48.112865061	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:207 M_PropRead.req OT=56933 O
1885	48.113351954	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:207 OK
1886	48.113623308	192.168.1.242	192.168.1.33	KNXnet/IP	60	? ConfigReq #01:207 M_PropRead.con OT=56933 O
1887	48.117251010	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:207 OK
1888	48.119474730	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:208 M_PropRead.req OT=0 OI=0 I
1889	48.119976586	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:208 OK
1890	48.120267781	192.168.1.242	192.168.1.33	KNXnet/IP	60	? ConfigReq #01:208 M_PropRead.con OT=0 OI=0 I
1891	48.122401095	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:208 OK
1892	48.124645861	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:209 M_PropRead.req OT=0 OI=0 I
1893	48.125206975	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:209 OK
1894	48.125446170	192.168.1.242	192.168.1.33	KNXnet/IP	61	? ConfigReq #01:209 M_PropRead.con OT=0 OI=0 I
1895	48.127671721	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:209 OK
1896	48.130179274	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:210 M_PropRead.req OT=0 OI=0 I
1897	48.130688962	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:210 OK
1898	48.130998520	192.168.1.242	192.168.1.33	KNXnet/IP	60	? ConfigReq #01:210 M_PropRead.con OT=0 OI=0 I
1899	48.132963003	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:210 OK
1900	48.135073320	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:211 M_PropRead.req OT=0 OI=0 I
1901	48.135702337	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:211 OK
1902	48.135849023	192.168.1.242	192.168.1.33	KNXnet/IP	60	? ConfigReq #01:211 M_PropRead.con OT=0 OI=0 I
1903	48.138269911	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:211 OK
1904	48.140454362	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:212 M_PropRead.req OT=39226 O
1905	48.141091643	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:212 OK
1906	48.141346717	192.168.1.242	192.168.1.33	KNXnet/IP	60	? ConfigReq #01:212 M_PropRead.con OT=39226 O
1907	48.143324099	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:212 OK
1908	48.145698433	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:213 M_PropRead.req OT=0 OI=0 I
1909	48.146212403	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:213 OK
1910	48.146523262	192.168.1.242	192.168.1.33	KNXnet/IP	61	? ConfigReq #01:213 M_PropRead.con OT=0 OI=0 I
1911	48.149220176	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:213 OK
1912	48.152534422	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:214 M_PropRead.req OT=0 OI=15
1913	48.153015572	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:214 OK
1914	48.153302990	192.168.1.242	192.168.1.33	KNXnet/IP	60	? ConfigReq #01:214 M_PropRead.con OT=0 OI=15
1915	48.155768879	192.168.1.33	192.168.1.242	KNXnet/IP	52	ConfigAck #01:214 OK
1916	48.158068074	192.168.1.33	192.168.1.242	KNXnet/IP	59	? ConfigReq #01:215 M_PropRead.req OT=0 OI=0 I
1917	48.158544582	192.168.1.242	192.168.1.33	KNXnet/IP	60	ConfigAck #01:215 OK
1918	48.158828483	192.168.1.242	192.168.1.33	KNXnet/IP	61	? ConfigReq #01:215 M_PropRead.con OT=0 OI=0 I

What do we expect?

Crashes == Something is not handled correctly

Error in KNXnet/IP frame (anywhere)

Service or other software interpreting frames

- Possibly compromise the interface

Error in KNX frame (cEMI)

KNX layer on interface or on devices

- Possible denial of service on devices
- Possibly compromise the interface

Wrap up

TODO

So far

- Major impact, minor concern
- No need to « bypass » protections yet

If we go further…

- What's inside widely-used implementations?
- What about KNXnet/IP security extensions?
- How to secure efficiently?

FIXME

Vendors

- Stop assuming security is the user's problem

Users

- Stop assuming security is the vendor's problem

FIXME

Attackers

- Brand new attack surface
  - Maybe someone will learn something
  - Reminder: Be careful and take care of people!

Defenders

- Brand new defense surface ☺
  - Quick win: don't expose devices

Thank you!

BOF

https://github.com/Orange-Cyberdefense/bof

https://bof.readthedocs.io/en/latest/

…and a huge thanks to DEFCON, Scapy maintainers, Olivier Gervais, Judicaël Courant, Baptiste Cauchard, Julien Bedel, and Leon Jacobs for their help and support!