The return of FAIFA and HomePlugPWN

Make Power-Line Communication hacks great again!

By Sébastien Dudek

leHack

July 6th 2019

Working team on the subject

- Xavier Carcelle
- Joffrey Czarny (@_Sn0rkY)
- And myself

Still a lot of work to do!

About me

- Sébastien Dudek (@FlUxIuS)
- Working at Synacktiv: pentests, red team, audits, and vuln researches
- Likes radio and hardware
- And to confront theory vs. practice

1 Introduction

2 State of the art

3 Current attacks

4 Attacking HomePlug GP

5 Smart Grids

6 Non documented things

7 Work in progress

8 Conclusion

Introduction

- PLC: Powerline Communication
- Principle discovered by Edward Davy in 1838
- Released in the early 2000s for home applications
- Evolves a lot in therms of speed

Could be found in various applications.

Applications

Classical: domestic

- Use HomePlug specifications (Ex. HomePlug AV)
- Extend a local network
- Depending on the context cheaper than buying multiple repeaters
- Generally more reliable than Wi-Fi

Other cases

Applications

Classical: domestic

Other cases

- Electrical counters:
  - Like Cenélec (3-148.5 kHz low voltage) are used : meter readings, intruder alarms, fire detection, gaz leak detection, and so on.
  - Linky G3, G1 specs, etc.
  - But some countries use HomePlug specifications for their counters
- Smart grid → recently found in missions
- Home automation
- And so on.

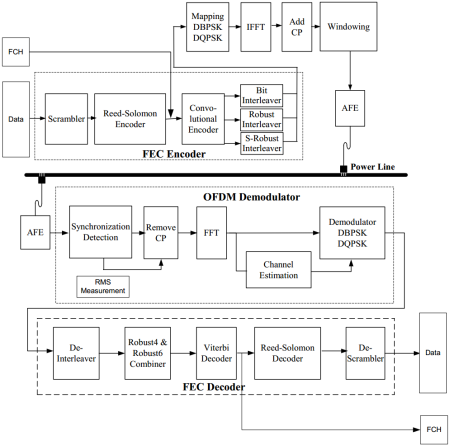
Data propagation: reminders

- AC voltage is 50 Hz → a signal do 50 cycles/s
- Could be represented by the formula: $Ps = A\sqrt{2}sin(2\pi ft)$ (f: frequency in Hz; t: time)
- The data (Da) is superposed to this carrier → $Td = Ps + da)$

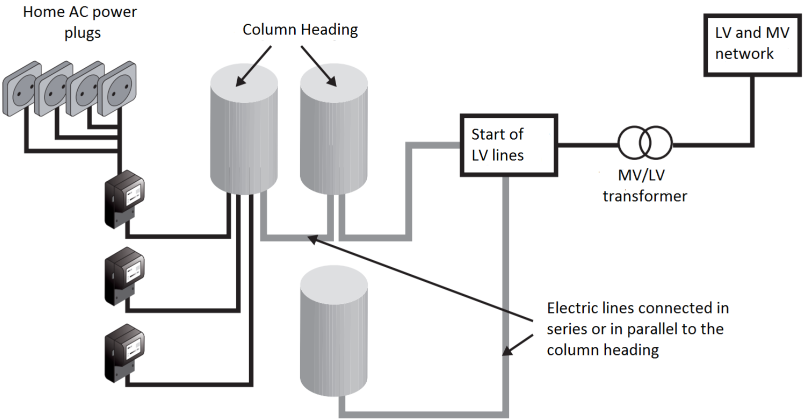
But before being sum to the power supply → need error detection, code mapping, multi-carrier modulation

Data propagation: DSP

1. data scrambling
2. turbo encoding
3. modulation of control and data frames
4. form OFDM symbols
5. windowing
6. etc.

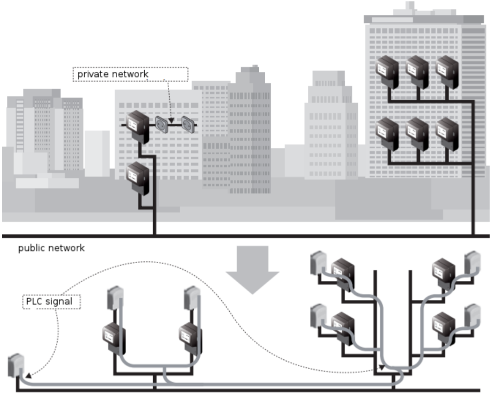


Data transmission at home



source: PLC in Practice by Xavier Carcelle

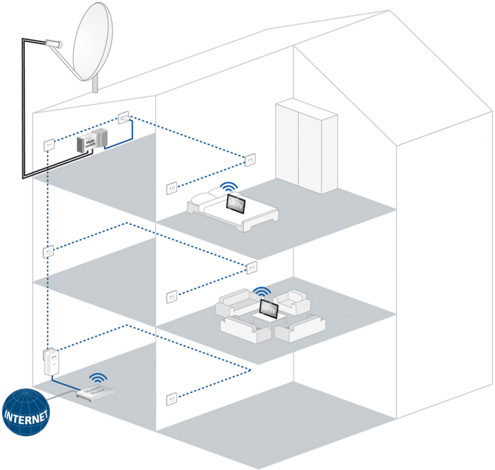
Private vs Public network



source: PLC in Practice by Xavier Carcelle

- In reality: no choc-coil → no real private network

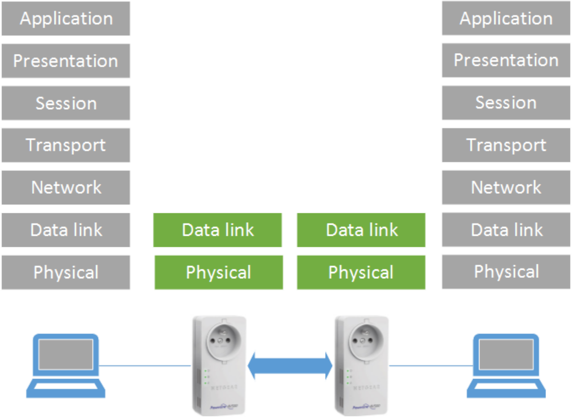
Data transmission at home



source: Devolo

PLC layers

A PLC uses layer 1 and 2 of the OSI model → IEEE 802.3



Communications

Computer ↔ PLC

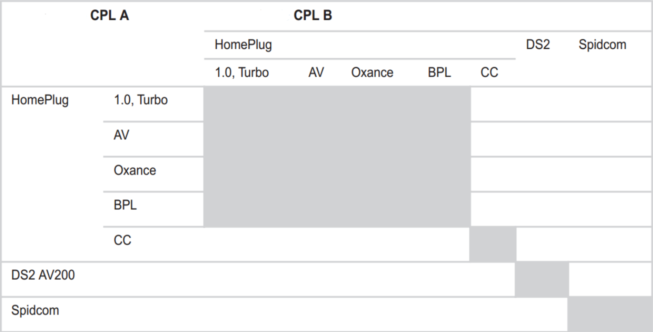
- Communicate through Ethernet on MAC layer
- Clear text (no ciphering)

PLC ↔ PLC

- Communicate through powerline
- Data is encrypted (using AES CBC 128 bits on new PLCs)

Everything is defined in HomePlug AV specifications

Interroperability

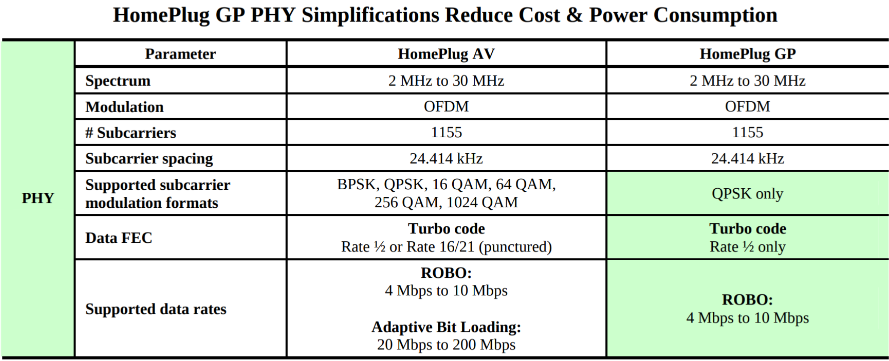


CPL A		CPL B						
		HomePlug					DS2	Spidcom
		1.0, Turbo	AV	Oxance	BPL	CC		
HomePlug	1.0, Turbo	▨	▨	▨	▨			
	AV	▨	▨	▨	▨			
	Oxance	▨	▨	▨	▨			
	BPL	▨	▨	▨	▨			
	CC					▨		
DS2 AV200							▨	
Spidcom								▨

But also with HomePlug Green PHY

HomePlug AV and GP

Homeplug GP (Green PHY) → subset of HomePlug AV



HomePlug GP PHY Simplifications Reduce Cost & Power Consumption

	Parameter	HomePlug AV	HomePlug GP
PHY	**Spectrum**	2 MHz to 30 MHz	2 MHz to 30 MHz
	Modulation	OFDM	OFDM
	# Subcarriers	1155	1155
	Subcarrier spacing	24.414 kHz	24.414 kHz
	Supported subcarrier modulation formats	BPSK, QPSK, 16 QAM, 64 QAM, 256 QAM, 1024 QAM	QPSK only
	Data FEC	**Turbo code** Rate ½ or Rate 16/21 (punctured)	**Turbo code** Rate ½ only
	Supported data rates	**ROBO:** 4 Mbps to 10 Mbps **Adaptive Bit Loading:** 20 Mbps to 200 Mbps	**ROBO:** 4 Mbps to 10 Mbps

HomePlug AV and Green PHY

- HomePlug Green PHY (HPGP) → subset of HomePlug AV
- HomePlug AV used to extend domestic local network
- HPGP Intended to be used for ”smart” grid or other automation systems
- Throughput decreased → use of QPSK instead of high order QAM
- HomePlug AV higher peak rate than HomePlug Green PHY

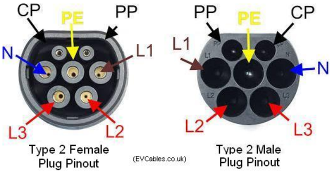
Into the wild

- Charging connector → Control Pilot line for HomePlug GP transfers

The Combined Charging System connectors

Different types of connectors exist, like IEC 62196 in UE:

- PP: Proximity pilot for pre-insertion signalling
- CP: Control Pilot for post-insertion signalling
- PE: Protective earth
- N: Neutral (single/3 phase AC/DC-mid)
- L1, L2 and L3 three phase AC/DC-mid



HGPG data multiplexed onto the Control Pilot and ground lines

1 Introduction

2 State of the art

3 Current attacks

4 Attacking HomePlug GP

5 Smart Grids

6 Non documented things

7 Work in progress

8 Conclusion

Publication

- Power Line Communications in Practice by Xavier Carcelle → a must read!
- HomePlug AV Security Mechanisms by Richard Newman, Larry Younge, Sherman Gavette, and Ross Anderson, published in 2007
- MISC #37 HomePlug Security by Xavier Carcelle
- HomePlugAV PLC: Practical attacks and backdooring, at NoSuchCon 2014, by Sébastien Dudek → introducing a flaw in Direct Access Key (DAK) generation
- V2G Injector: Whispering to cars and charging units through the Power-Line, at SSTIC 2019, by Sébastien Dudek → introducing a new flaw in HomePlug Green PHY

Tools

- plconfig → manage PLCs over the network
- FAIFA[1] by Xavier Carcelle (similar to plconfig) → first Open source PLC tool
- Vendors' softwares
- open-plc-utils[2] by Qualcomm Atheros, published after FAIFA
- Wireshark has a dissector for HomePlugAV, but not for HomePlug GP
- HomePlugPWN[3] by Sébastien Dudek: Scapy dissectors for HomePlug AV / GP**(new)**, attack DAK keys and collect HomePlug GP secrets**(new)**

[1] `https://github.com/ffainelli/faifa`

[2] `https://github.com/qca/open-plc-utils`

[3] `https://github.com/FlUxIuS/HomePlugPWN`

This presentation

- Remindings: Power-Line Communications and previous found vulnerabilities
- Methodologies to attack those devices nowadays
- A new vulnerability found on the HomePlug Green PHY
- Hidden secrets of HomePlug devices
- New areas of research
- Surprises with the use of HomePlug in power meters :)

1 Introduction

2 State of the art

3 **Current attacks**

4 Attacking HomePlug GP

5 Smart Grids

6 Non documented things

7 Work in progress

8 Conclusion

Identification of devices

2 techniques:

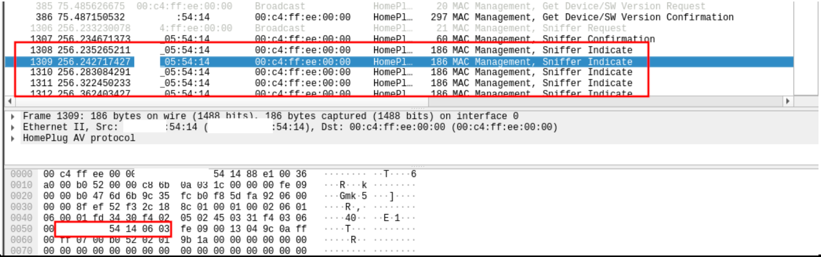
1. NetworkInfo Req → Confirmations → Station informations
2. Enable Sniff Mode → get MME of Central Coordinators (CCo)
    - A detected CCo = potential AV logical network

But *NetworkInfo* confirmation messages list stations of the same AVLN only → need to be smarter

Detection of HomePlug AV/GP devices with sniff mode

To detect Central Coordinator (CCo) devices → same old tricks are still possible:

1. Enabling sniff mode with *plcmon.py* provided in HomePlugPWN tool
2. See all EVSE that appears as CCo devices reported by Sniff indicate packets



```
 385 75.485626675  00:c4:ff:ee:00:00   Broadcast           HomePl…   20 MAC Management, Get Device/SW Version Request
 386 75.487150532  [redacted]:54:14    00:c4:ff:ee:00:00   HomePl…  297 MAC Management, Get Device/SW Version Confirmation
1306 256.233230078 [redacted]4:ff:ee:00:00 Broadcast       HomePl…   21 MAC Management, Sniffer Request
1307 256.234671373 [redacted]_05:54:14 00:c4:ff:ee:00:00   HomePl…   60 MAC Management, Sniffer Confirmation
1308 256.235265211 [redacted]_05:54:14 00:c4:ff:ee:00:00   HomePl…  186 MAC Management, Sniffer Indicate
1309 256.242717427 [redacted]_05:54:14 00:c4:ff:ee:00:00   HomePl…  186 MAC Management, Sniffer Indicate
1310 256.283084291 [redacted]_05:54:14 00:c4:ff:ee:00:00   HomePl…  186 MAC Management, Sniffer Indicate
1311 256.322450233 [redacted]_05:54:14 00:c4:ff:ee:00:00   HomePl…  186 MAC Management, Sniffer Indicate
1312 256.362403427 [redacted]_05:54:14 00:c4:ff:ee:00:00   HomePl…  186 MAC Management, Sniffer Indicate

▸ Frame 1309: 186 bytes on wire (1488 bits), 186 bytes captured (1488 bits) on interface 0
▸ Ethernet II, Src: [redacted]:54:14 ([redacted]:54:14), Dst: 00:c4:ff:ee:00:00 (00:c4:ff:ee:00:00)
▸ HomePlug AV protocol

0000  00 c4 ff ee 00 0[redacted]  54 14 88 e1 00 36   ········ ··T···6
0010  a0 00 b0 52 00 00 c8 6b  0a 03 1c 00 00 00 fe 09   ···R···k ········
0020  00 00 b0 47 6d 6b 9c 35  fc b0 f8 5d fa 92 06 00   ···Gmk·5 ···]····
0030  00 00 8f ef 52 f3 2c 18  8c 01 00 01 00 02 06 01   ····R·,· ········
0040  06 00 01 fd 34 30 f4 02  05 02 45 03 31 f4 03 06   ····40·· ·E·1···
0050  00 [redacted] 54 14 06 03  fe 09 00 13 04 9c 0a ff   ····T··· ········
0060  00 ff 07 00 b0 52 02 01  9b 1a 00 00 00 00 00 00   ·····R·· ········
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
```

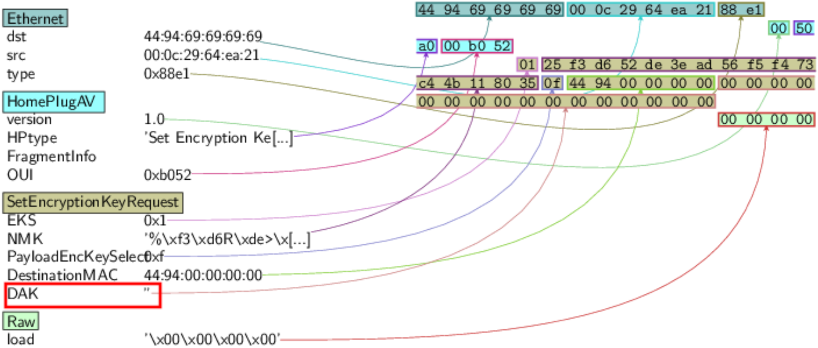
HomePlug AV and Green PHY keys

2 kinds of keys to manage and encrypt data:

- Network Membership Key (NMK): to encrypt the communication using 128-bit AES CBC
- Direct Access Key (DAK): to remotely configure the NMK of a targeted PLC device over the Power-Line interface

Configuring the NMK

- if local → DAK can be empty
- remotely the DAK of the targeted device should be included

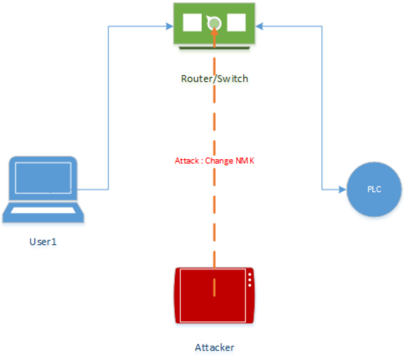


```
Ethernet
  dst                      44:94:69:69:69:69
  src                      00:0c:29:64:ea:21
  type                     0x88e1
HomePlugAV
  version                  1.0
  HPtype                   'Set Encryption Ke[...]
  FragmentInfo
  OUI                      0xb052
SetEncryptionKeyRequest
  EKS                      0x1
  NMK                      '%\xf3\xd6R\xde>\x[...]
  PayloadEncKeySelec0xf
  DestinationMAC           44:94:00:00:00:00
  DAK                      ''
Raw
  load                     '\x00\x00\x00\x00'
```

```
44 94 69 69 69 69  00 0c 29 64 ea 21  88 e1
                                               00  50
a0  00 b0 52
                01  25 f3 d6 52 de 3e ad 56 f5 f4 73
c4 4b 11 80 35  0f  44 94 00 00 00 00  00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00
                                       00 00 00 00
```

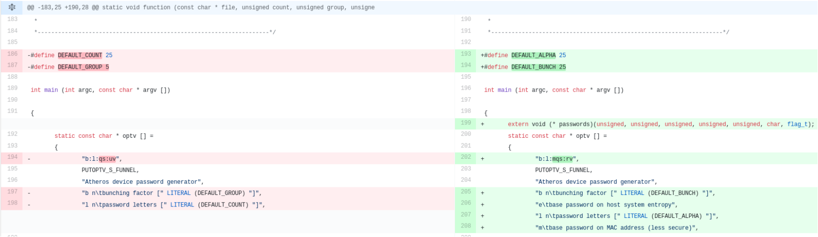
Attacking the local interface

- Ethernet interface: allowed to perform privileged operations
- If an attacker is on the LAN → backdoor the device:
  - Program it's own NMK
  - Replace device's firmware



DAK generation status

- Qualcomm devices had a weak DAK → see our research paper presented at NSC 2014[4]
- In Feb 2015: Qualcomm patched their utility, refering to their GitHub:



```
@@ -183,25 +190,28 @@ static void function (const char * file, unsigned count, unsigned group, unsigne
```

```
  *
  *-----------------------------------------------------------------*/

-#define DEFAULT_COUNT 25
-#define DEFAULT_GROUP 5

 int main (int argc, const char * argv [])

 {

        static const char * optv [] =
        {
-               "b:l:qs:uv",
                PUTOPTV_S_FUNNEL,
                "Atheros device password generator",
-               "b n\tbunching factor [" LITERAL (DEFAULT_GROUP) "]",
-               "l n\tpassword letters [" LITERAL (DEFAULT_COUNT) "]",
```

```
  *
  *-----------------------------------------------------------------*/

+#define DEFAULT_ALPHA 25
+#define DEFAULT_BUNCH 25

 int main (int argc, const char * argv [])

 {
+       extern void (* passwords)(unsigned, unsigned, unsigned, unsigned, unsigned, char, flag_t);
        static const char * optv [] =
        {
+               "b:l:mqs:rv",
                PUTOPTV_S_FUNNEL,
                "Atheros device password generator",
+               "b n\tbunching factor [" LITERAL (DEFAULT_BUNCH) "]",
+               "e\tbase password on host system entropy",
+               "l n\tpassword letters [" LITERAL (DEFAULT_ALPHA) "]",
+               "m\tbase password on MAC address (less secure)",
```

But still devices from 2015 and older + chineese and some other devices remain vulnerable

Attacking vulnerable devices

- Discover CCo to get a MAC address:

```
python plcmon.py
[+] Enabling sniff mode
Sent 1 packets.
[+] Listening for CCo station...
        Found CCo: 44:94:fc:56:ff:34 (DAK: RMHT-ILPO-TYMN-IIXY)
        [...]
```

- Run K.O.DAK attack to reconfigure the NMK remotely:

```
python quickKODAK.py -i eth0 -t 4494fc56ff34
Sent 1 packets.
```

- Configure our PLC to connect to the targeted AVLN

We can then use the internet connection (so much QoS than attacking Wi-Fi network), or attack computers in this network.

1 Introduction

2 State of the art

3 Current attacks

4 Attacking HomePlug GP

5 Smart Grids

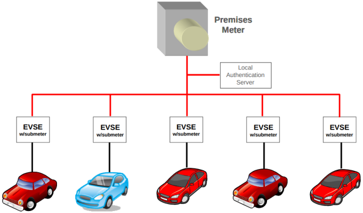
6 Non documented things

7 Work in progress

8 Conclusion

Plug-in Electrical Vehicle (PEV) Association

- PEV can be charged everywhere (public, home, etc.)
- It leaves unconfigurated in new AVLN (AV Logical Network)
- So it needs to join the AVLN of the corresponding EVSE once plugged with a charging connector



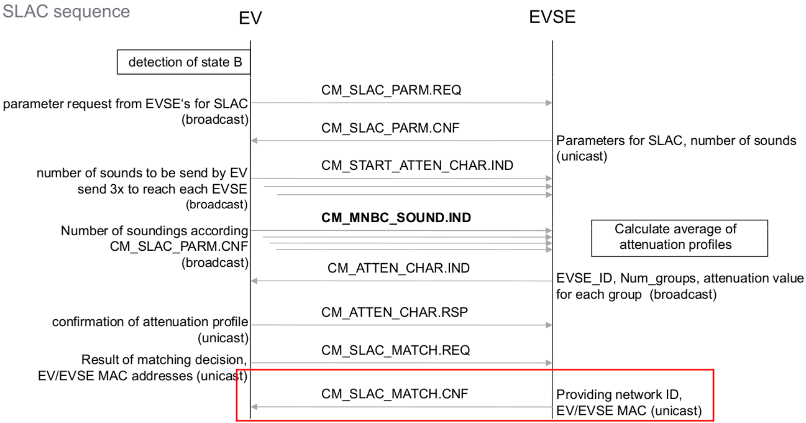
source: HomePlug Green PHY white paper

But PLC packets are broadcast in the Power-Line...

SLAC procedure

- SLAC: Signal Level Attenuation Characterization
- Aimed to avoid bad association (avoid billing errors, etc)
- Principle:
  1. PEV broadcast unacknowledged SOUNDING packets
  2. Stations (EVSE) around measure the received power and send it to the PEV
  3. PEV finally select the EVSE with the best result
  4. Then EVSE provides network (how???)

SLAC procedure (2)



SLAC sequence

EV

EVSE

detection of state B

CM_SLAC_PARM.REQ

parameter request from EVSE‘s for SLAC (broadcast)

CM_SLAC_PARM.CNF

Parameters for SLAC, number of sounds (unicast)

CM_START_ATTEN_CHAR.IND

number of sounds to be send by EV send 3x to reach each EVSE (broadcast)

CM_MNBC_SOUND.IND

Number of soundings according CM_SLAC_PARM.CNF (broadcast)

Calculate average of attenuation profiles

CM_ATTEN_CHAR.IND

EVSE_ID, Num_groups, attenuation value for each group (broadcast)

CM_ATTEN_CHAR.RSP

confirmation of attenuation profile (unicast)

CM_SLAC_MATCH.REQ

Result of matching decision, EV/EVSE MAC addresses (unicast)

CM_SLAC_MATCH.CNF

Providing network ID, EV/EVSE MAC (unicast)

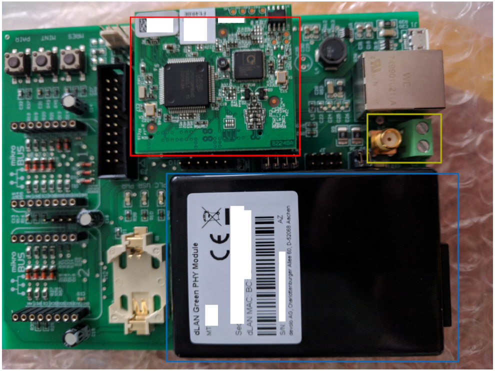
source: HomePlug Green PHY whitepaper

Our contribution

- Developed Scapy layers for HomePlug GP
- Found a new flaw in HPGP SLAC procedure → intrude AVLN of charging station for example

Our first device to test it

dLAN Green PHY eval board EU II → multiple interfaces



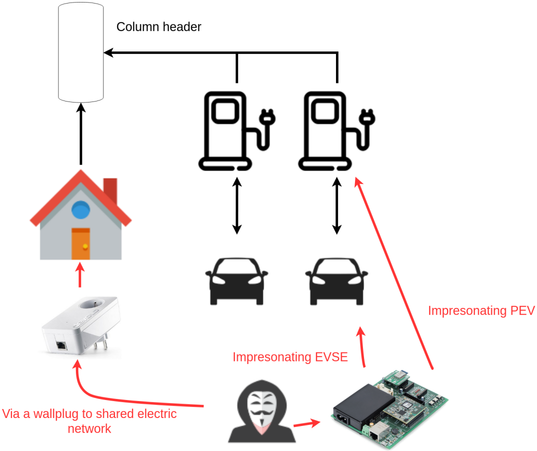
But cheaper alternative exist

Cheapest way: the wallplug

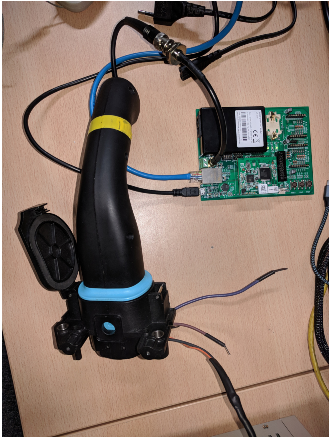
- Any QCA 7k will do the work
- Ex: Devolo 1200+ works like a charm
- No modification needed if charging stations share the same electrical network
- Otherwise some rework should be done on the coupler

We are actually working on some modular rework with this adaptor

How to interface

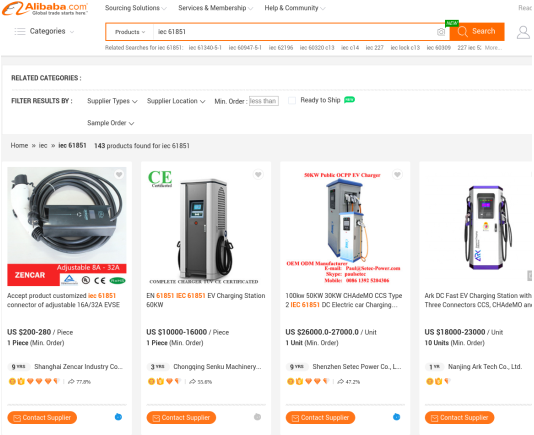


With a charging station connector



Where can we find those connectors?

You can really find everything in Alibaba, even charging stations...



iec 61851

Related Searches for iec 61851: iec 61340-5-1 iec 60947-5-1 iec 62196 iec 60320 c13 iec c14 iec 227 iec lock c13 iec 60309 227 iec 5: More...

RELATED CATEGORIES :

FILTER RESULTS BY : Supplier Types Supplier Location Min. Order : less than ☐ Ready to Ship NEW Sample Order

Home » iec » **iec 61851** **143** products found for iec 61851

Accept product customized iec 61851 connector of adjustable 16A/32A EVSE
US $200-280 / Piece
1 Piece (Min. Order)
9 YRS Shanghai Zencar Industry Co...
77.8%

EN 61851 IEC 61851 EV Charging Station 60KW
US $10000-16000 / Piece
1 Piece (Min. Order)
3 YRS Chongqing Senku Machinery...
55.6%

100kw 50KW 30KW CHAdeMO CCS Type 2 IEC 61851 DC Electric car Charging...
US $26000.0-27000.0 / Unit
1 Unit (Min. Order)
9 YRS Shenzhen Setec Power Co., L...
47.2%

Ark DC Fast EV Charging Station with Three Connectors CCS, CHAdeMO an
US $18000-23000 / Unit
10 Units (Min. Order)
1 YR Nanjing Ark Tech Co., Ltd.

HomePlug Green PHY modes

Can be set in 3 specific modes:

- Unconfigured
- PEV: can see HPGP specific packets from EVSE
- EVSE: see HGPG specific packets from PEV

Each mode allows or disallow to intercept certain HomePlug GP packets at MAC Layer 2

HomePlug Green PHY modes

Can be set in 3 specific modes:

- Unconfigured
- PEV: can see HPGP specific packets from EVSE
- EVSE: see HGPG specific packets from PEV

Each mode allows or disallow to intercept certain HomePlug GP packets at MAC Layer 2

Warning

Need the correct mode to collect MME packets of a specific device

Changing SLAC mode

Change SLAC mode into PEV modifying byte 0x1653 with “setpib“ after dumping it with *plctool*[5]:

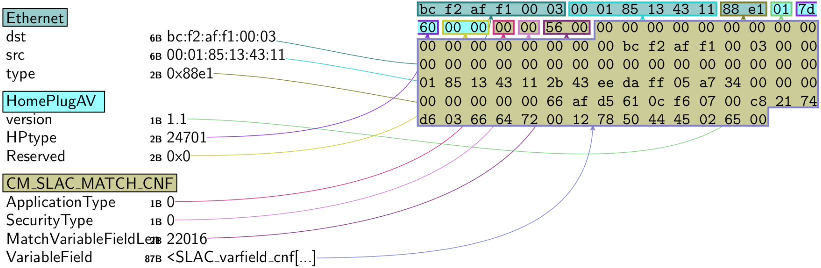
```
$ setpib PIBdump.pib 1653 byte 1
```

Then → capture packets coming from EVSEs

[5] https://github.com/qca/open-plc-utils

Flaw in the SLAC procedure

When analysing the SLAC procedure → surprise!



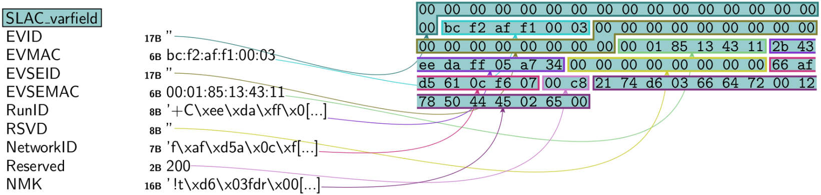
```
Ethernet
  dst                        6B bc:f2:af:f1:00:03
  src                        6B 00:01:85:13:43:11
  type                       2B 0x88e1
HomePlugAV
  version                    1B 1.1
  HPtype                     2B 24701
  Reserved                   2B 0x0
CM_SLAC_MATCH_CNF
  ApplicationType            1B 0
  SecurityType               1B 0
  MatchVariableFieldLen      2B 22016
  VariableField             87B <SLAC_varfield_cnf[...]
```

```
bc f2 af f1 00 03 00 01 85 13 43 11 88 e1 01 7d
60 00 00 00 00 56 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 bc f2 af f1 00 03 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01 85 13 43 11 2b 43 ee da ff 05 a7 34 00 00 00
00 00 00 00 00 66 af d5 61 0c f6 07 00 c8 21 74
d6 03 66 64 72 00 12 78 50 44 45 02 65 00
```

It was supposed to be a unicast packet, isn't it? → but it is broadcasted in the Power-Line!

Getting keys of AVLNs

By decoding the different fields of the *CM_SLAC_MATCH.CNF* message:



SLAC_varfield		
EVID	17B	''
EVMAC	6B	bc:f2:af:f1:00:03
EVSEID	17B	''
EVSEMAC	6B	00:01:85:13:43:11
RunID	8B	'+C\xee\xda\xff\x0[...]
RSVD	8B	''
NetworkID	7B	'f\xaf\xd5a\x0c\xf[...]
Reserved	2B	200
NMK	16B	'!t\xd6\x03fdr\x00[...]

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 bc f2 af f1 00 03 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 01 85 13 43 11 2b 43
ee da ff 05 a7 34 00 00 00 00 00 00 00 00 66 af
d5 61 0c f6 07 00 c8 21 74 d6 03 66 64 72 00 12
78 50 44 45 02 65 00
```

Our PLC can be easily set by changing “slac/pev.ini” profile and used with “pev” tool[6]

[6] https://github.com/qca/open-plc-utils

Into the AVLN

- Once part of an AVLN → we can talk to every possible device into the same AVLN
- Reach services exposed by devices
- Intercept exchanged data EV ↔ charging station

More about: V2G Injector

- Available: https://github.com/FlUxIuS/V2GInjector
- Paper, slides and recording: click here (SSTIC 2019)

1 Introduction

2 State of the art

3 Current attacks

4 Attacking HomePlug GP

5 **Smart Grids**

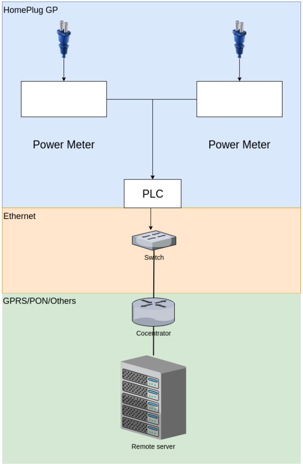
6 Non documented things

7 Work in progress

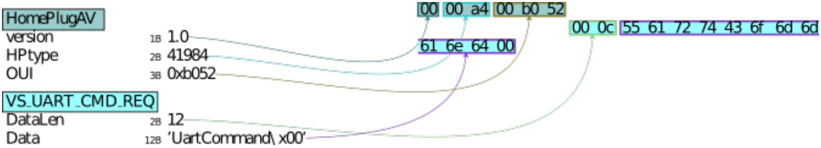
8 Conclusion

HomePlug applied to Smart Grids

- For Smart Grids use HomePlug GP
- Sends UART commands through PowerLine → WTF?!



Very simple to generate with Scapy



```
HomePlugAV
version     1B  1.0
HPtype      2B  41984
OUI         3B  0xb052
VS_UART_CMD_REQ
DataLen     2B  12
Data       12B  'UartCommand\ x00'

00 00 a4 00 b0 52 00 0c 55 61 72 74 43 6f 6d 6d
61 6e 64 00
```

- You can test it on detected devices → it will reply with a confirmation message
- Implemented in HomePlugPWN[7]

[7] urlhttps://github.com/FlUxIuS/HomePlugPWN/blob/master/layer-scapy/HomePlugSG.py

Smart cities = UART cmds everywhere?!

But you know...

1 Introduction

2 State of the art

3 Current attacks

4 Attacking HomePlug GP

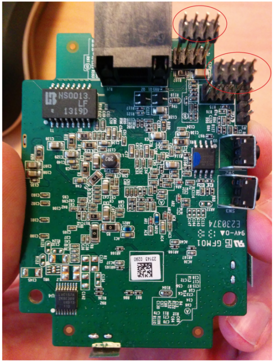
5 Smart Grids

6 Non documented things

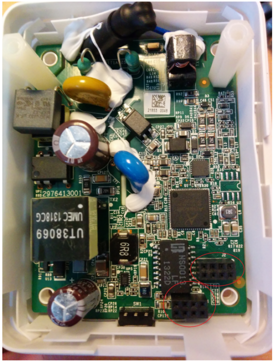
7 Work in progress

8 Conclusion

Remember?



Vendor part



PLC part

Other examples

Program Information Blocks (PIB)

- Used to store PLC's configuration
- Enables/Disables certains modes (WireTap, Sniffing, SLAC, etc.)
- A lot of non-documented blocks
- Many features could be discovered by digging this way

A lot of blocks have been retrieved and implemented in *ModulePIB*[8] of the *HomePlugAV.py* Scapy layer → still needs more work to decode all of them

Dump PIB

2 tools:

- *PIBdump.py* of *HomePlugPWN*
- *plctool* of *open-plc-utils* → support more PLC chipsets

```
./plctool -f -i enp0s31f6 -p /tmp/plc.pib local
enp0s31f6 00:B0:52:00:00:01 Fetch NVRAM Configuration
enp0s31f6 F4:06:8D:CE:00:7D TYPE=0x15 (M25P32_ES) PAGE=0x0100 (256) BLOCK=0x10000
 (65536) SIZE=0x400000 (4194304)
enp0s31f6 00:B0:52:00:00:01 Read Module from Memory
```

Analyse PIB

The tool *chkpib* of *open-plc-utils* allows to extract informations:

- *PIBdump.py* of *HomePlugPWN*
- *plctool* of *open-plc-utils* → support more PLC chipsets

```
 ./chkpib –mv /tmp/plc.pib
——————— /tmp/plc.pib (0) ———————
        [...]
——————— /tmp/plc.pib ———————
        PIB 0–0 19928 bytes
        MAC F4:06:8D:CE:00:7D
        DAK A7:6B:****************************************
        NMK 36:34:C5:DF:2E:6E:4F:7D:72:05:F5:8D:39:29:53:C0
        NID 96:46:60:59:BF:F8:05
        Security level 0
        NET
        MFG Delta Electronics Mon 27 May 2019 06:05:29 PM CEST
        USR Qualcomm Atheros Enabled PEV
        CCo Never
        MDU N/A
```

Analyse PIB (2)

- A lot of undocumented blocks → implemented in *ModulePIB*[9]
- Still needs more work to decode all of them

Hidden commands

- Our tools (FAIFA[10] and HomePlugPWN) implent usefull commands to test and intrude network
- A lot of commands are to be discovered + probably more logical vulnerabilities
- A lot to be documented and implemented → as shown in “Homeplug AV and IEEE 1901”
- Call for contributors!

1. Introduction
2. State of the art
3. Current attacks
4. Attacking HomePlug GP
5. Smart Grids
6. Non documented things
7. **Work in progress**
8. Conclusion

Mysteries

Some assumptions:

- Under MAC Layer 2 messages, interesting exchanged could also be observed
- But there is no tool to observe that
- Hard to implement in Software-Defined Radio + need an hardware managing the bandwidth
- Better chances looking at closed firmwares and hardware

Dump memory

2 methods:

- From the exposed flash memory
- For some vendors → with HomePlug AV specific commands (supported in *HomePlugPWN* (QCA < 7k for the moment) and *open-plc-utils*)

```
$ ./plctool —i enp0s31f6 —n image.nvm local
enp0s31f6 00:B0:52:00:00:01 Read Module from Memory
[...]
```

Getting NVM

- Non-Volatile Memory
- Getting the NVM from SDRAM:

```
$ ./plctool –i enp0s31f6 –n image.nvm local
enp0s31f6 00:B0:52:00:00:01 Read Module from Memory
[...]
```

If the device denies the command, some vendors release complet firmware.

NVM structure

Could be obtained with *open-plc-utils-master/nvm*:

```
typedef struct __packed nvm_header2
{
        uint16_t MajorVersion;
        uint16_t MinorVersion;
        uint32_t ExecuteMask;
        uint32_t ImageNvmAddress;
        uint32_t ImageAddress;
        uint32_t ImageLength;
        uint32_t ImageChecksum;
        uint32_t EntryPoint;
        uint32_t NextHeader;
        uint32_t PrevHeader;
        uint32_t ImageType;
        uint16_t ModuleID;
        uint16_t ModuleSubID;
        uint16_t AppletEntryVersion;
        uint16_t Reserved0;
    [...]
        uint32_t Reserved11;
        uint32_t HeaderChecksum;
}
```

NVM structure (2)

```
$ ./chknvm –s –v plc.nvm
———————— plc.nvm (0) ————————
         Header Version = 0x0001–0x0001
         Header Checksum = 0xA7A78802
         Header Next = 0x00000360
         Header Prev = 0xFFFFFFFF
         Flash Address = 0x00000060
         Image Address = 0x00000000
         Entry Address = 0xFFFFFFFF
         Entry Version = 0x0000
     [...]
———————— plc.nvm (1) ————————
     [...]
———————— plc.nvm (2) ————————
     [...]
         Image Type = Custom Module Update Applet
         Image Exec = INT6000|INT6300
———————— plc.nvm (3) ————————
     [...]
         Image Type = Power Management Applet
         Image Exec = INT6000|INT6300
———————— plc.nvm (4) ————————
     [...]
         Image Type = Generic Image
         Image Exec = INT6000|INT6300
———————— plc.nvm (5) ————————
     [...]
         Image Type = Runtime Firmware
         Image Exec = INT6000|INT6300
```

Split the NVM

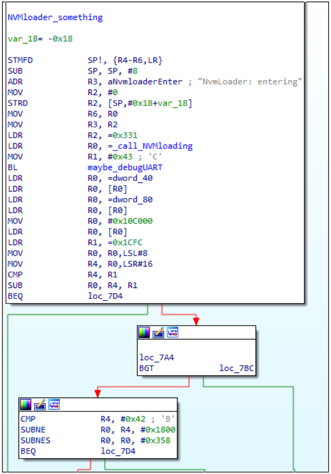
NVM Could be split by type of block easily with qca utilities:

```
$ ./nvmsplit plc.nvm
$ ls plc−∗
 plc−01.nvm
 [...]
 plc−05.nvm #<−− let's look at each image
```

Let’s now look each block

Disassembling the firmware

From specs the QCA7420 → AR7420 → ARM processor.



```
NVMloader_something

var_18= -0x18

STMFD           SP!, {R4-R6,LR}
SUB             SP, SP, #8
ADR             R3, aNvmloaderEnter ; "NvmLoader: entering"
MOV             R2, #0
STRD            R2, [SP,#0x18+var_18]
MOV             R6, R0
MOV             R3, R2
LDR             R2, =0x331
LDR             R0, =_call_NVMloading
MOV             R1, #0x43 ; 'C'
BL              maybe_debugUART
LDR             R0, =dword_40
LDR             R0, [R0]
LDR             R0, =dword_80
LDR             R0, [R0]
MOV             R0, #0x10C000
LDR             R0, [R0]
LDR             R1, =0x1CFC
MOV             R0, R0,LSL#8
MOV             R4, R0,LSR#16
CMP             R4, R1
SUB             R0, R4, R1
BEQ             loc_7D4
```

```
loc_7A4
BGT             loc_7BC
```

```
CMP             R4, #0x42 ; 'B'
SUBNE           R0, R4, #0x1800
SUBNES          R0, R0, #0x358
BEQ             loc_7D4
```

→ 4th block

Disassembling the firmware (2)

- the code is minimal → not many strings but still helpful
- written in C++
- some time and coffee are needed
- fuzzy patching Applets takes time:
  1. patch
  2. merge blocks
  3. flash and see what happens...

Disassembling the firmware (2)

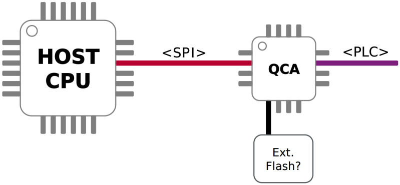
- the code is minimal → not many strings but still helpful
- written in C++
- some time and coffee are needed
- fuzzy patching Applets takes time:
  1. patch
  2. merge blocks
  3. flash and see what happens...

Warning

May brick your device :S → need something safer

SPI accesses

- Devkit exposes explicit SPI access to interface with the PLC modem:
  - 2 parts: host/app CPU and a PLC modem/baseband
- Possible to get Direct Memory Access + accesses to registers



source: Michael Epping. Vehicle Charging Control Unit. EMOB, 2017

More on that a bit later...

1 Introduction

2 State of the art

3 Current attacks

4 Attacking HomePlug GP

5 Smart Grids

6 Non documented things

7 Work in progress

8 Conclusion

Conclusion

- FAIFA and HomePlugPWN are back in the game
- Power-Line Communication is almost everywhere
- Logical vulnerability exist in specs and vendors configurations
- A lot of bugs under the Layer 2 MAC could be found → but PLC is not open enough (we're working on it)
- Finding bugs in the PLC baseband → difficult to debug for the moment, even with a devkit
- The work is not finished → interested people can contact us to advance these researches (we've been doing @home)

ANY QUESTIONS?

THANK YOU FOR YOUR ATTENTION,

SYNACKTIV
DIGITAL SECURITY