HomePlugAV PLC: practical attacks and backdooring

Sébastien Dudek - sebastien.dudek@synacktiv.com

August 2015 - Additional information after the presentation made at NoSuchCon 2014

Abstract

Domestic Powerline Communication (PLC) devices are used to extend a LAN network in the same way as Wi-Fi does, but using the powerline support. Designed under the HomePlugAV specification, latest devices available on the market allow a speed rate up to 200 Mb/s and 600 Mb/s for HomePlug AV and HomePlug AV2 respectively. Even if PLC have a bad reputation in the past (security issues, low speed, not stable because of perturbations, ...), this technology grown up and offers now a better connection with an encrypted conversation between two PLC devices. Someone who wants to extend his network without additionnal wires or without spending too much money and time using wireless repeaters will use PLC. Moreover, Internet Service Providers in France usually provide a HomePlugAV embedded in the power supply of their routers and set-top-boxes. As HomePlugAV is implemented on a lot of devices, it is interesting to study its security and its weaknesses.

In this paper, we will describe the PLC technology from the start, introducing its physical aspects, and evolution to understand its behaviour in the electric line, the impact of this technology in the domestic but also the industrial context. Then we will present our network analysis done with existing tools and our tools. After this analysis, we will show a few practical local and remote attacks to penetrate and backdoor a private LAN. To finish with, we will talk about PLC remote memory access that could be done once we got the keys to be part of the network, and some future work around this subject (firmware disassembling, backdooring stuff, authentication messages fuzzing).

Acknowledgement

I would like to thank my previous employer, Sogeti R&D Lab, and my current one, Synacktiv, for giving me time to study this subject and many other cool ones, as well as my teammates for their time reviewing this paper and giving me advices and feedbacks. I also would like to thank the NoSuchCon team that has given me the opportunity to present this subject.

For people interested in this area, I hope this paper will help you to understand PLCs and attacks that could be performed on HomePlug PLC devices. Your feedback would also be appreciated on this interesting subject. There are still a lot of interesting work to do and HomePlug PLC are not the only systems where the PowerLine-Communication transmission is implemented ("smart" meters, cars, and so on).

Contents

1 Introduction

1.1 Context

A Power Line Carrier/Communication (PLC) device is an electric adapter that carries information through an electric power line. These products appeared in the early 2000s and are increasingly efficient. Indeed, the new PLCs are faster and more usable for a public who wants to extend its network without using wires and enjoy services provided by the ISP like IPTV (Internet Protocol television), SIP (Session Initiation Protocol), and so on.

Also, from a security point of view, these products tend to be stronger and more mature than previous versions that broadcast your information over the electric power, regardless of your need for confidentiality. In fact, unlike the old version that uses no encryption or DES, new HomePlug AV standards use 128-bit AES CBC to communicate through an isolated cryptographic network called AVLN (AV Logical Network) with a 128-bit key between two PLC devices.

1.2 History

In 1836, in England, Edward Davy, proposed a solution to remotely measure the battery levels of a site far from the telegraph system between Liverpool and London. He published the first patent (British patent no. 24833) that describes a technique for the remote measure of electrical network meters communicating over electrical distribution wiring [1] [2].

In 1950, the first PLC system is released, known as "Ripple Control", in order to remotely switch on and off public lights or tarif changes. This system was designed and deployed over medium and low-voltage, using a carrier frequency between 100Hz and 1kHz to establish bi-directional communications.

During World War II, radio amateurs experimented with power line communication. During the war, a lot of telephone lines were destroyed, so power line communication was used to communicate using the techniques based on meter measurement.

The first industrial systems appeared in France in 1960 and was named Pulsadis. It was approximately a hundred kilovolt-amperes (kVA). Then, the first PLC arrived (called Cenélec), extending from 3 to 148.5 kHz, and allowing bidirectional communications over low voltage. This system allows meter readings and other applications in home automation field like: intruder alarm, fire detection, gas leak detection, and so on. The injected power was much smaller and reduced the level of approximately a hundred milliwatts [3].

1.3 PLC at home

PLCs are used to extend a domestic network as shown in the following picture (figure 1).

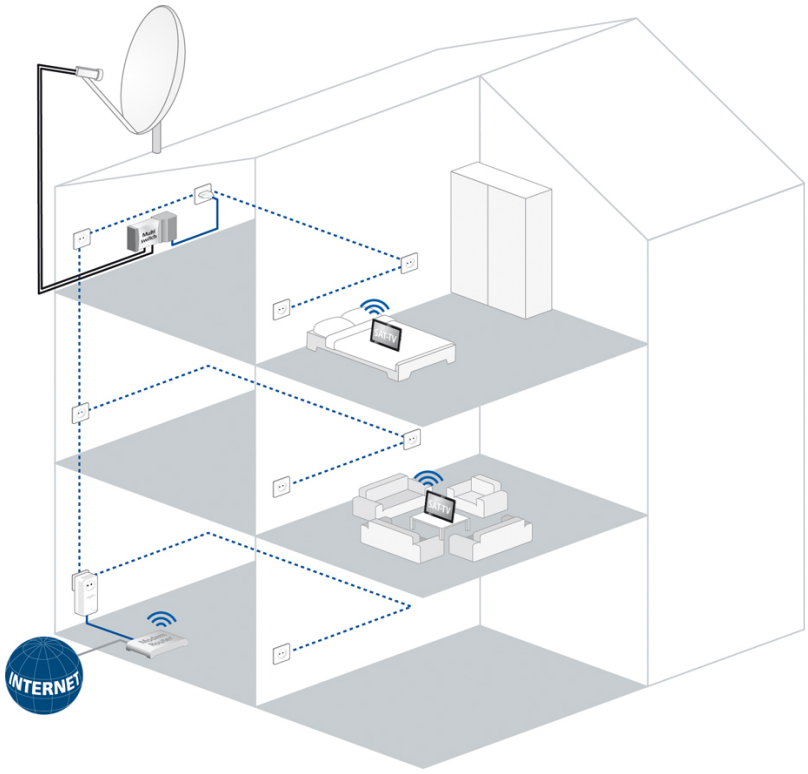

Figure 1: *A PLC-equiped house (source: http://www.devolo.com)*

2 Powerline data transport

2.1 Exploitation of the power supply

The PLC technology is used to transfer data over the power line. The cable can be compared to the first layer (Physical) of the OSI model but, unlike Ethernet, or optical fiber, this support was just meant to supply electricity to devices such as our Television, computer, fridge, and so on.

To be transfered, data has to be added to the energy supply. It is 200V/50Hz in Europe, and 100V/60Hz in United States and Japan. Figure 2 shows the different levels of electric current used in France.

The distribution network of energy supply is a similar to the telephony network RTC in France. It is composed of an electricity central and a route network. The transformer Medium Voltage/Low (MV/LV) links the MV network and the route network that supplies around 200 electric meters (figure 3).

The meaning of a 50 Hz AC voltage is that the signal does 50 cycles per second. The power supply is represented with the following formula: $Ps = A\sqrt{2}\sin(2\pi ft)$ where A is the voltage (220 V in Europe, 100 V in US and Japan) and f the number of cycles per second the signal does (50 Hz in Europe, 60 US and Japan).

Appelation (=in FR)	Old appelation	Level of Voltage in France
HVB (HTB)	Very High-Voltage	400 000 V
		225 000 V
	High-Voltage	90 000 V
		65 000 V
HVA (HTA)	Medium-Voltage	20 000 V
LV (BT)	Low-Voltage	380 V (three-phase)
		220 V (single-phased)

Figure 2: *Actual current used in France*

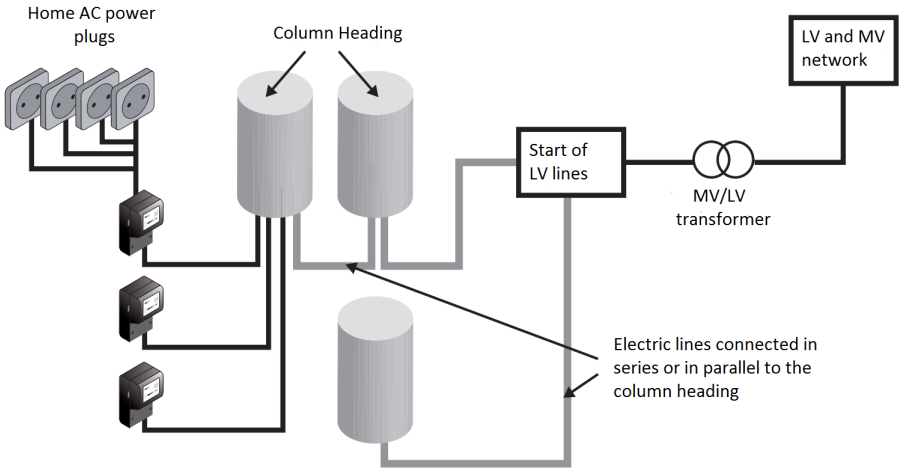

Figure 3: *Simplified architecture of a power supply network (modified from source: [4])*

The implementation of a short Python script that helps us represent the power supply signal can be seen in figure 4, and its resulting plot in figure 5 where we can count 25 impulsions for a time period of 1/2 seconds.

```
import matplotlib.pyplot as plt
import numpy as np
t=np.linspace(0,0.5, 10000) # 1/2 seconds, 10000 samples
plot = plt.plot(t,220*np.sqrt(2)*np.sin(2*np.pi*50*t))
plt.show()
```

Figure 4: *Power supply signal represented in Python*

Let us suppose the carrier is 60 kHz and is represented with the formula: $Ca = 2\sqrt{2}\sin(2\pi 60000t)$, the plot will look like figure 6.

If we want to transport the information, we need to superpose/sum the signal for the power supply with the carrier: $Ps + Ca = 220\sqrt{2}\sin(2\pi 50t) + 2\sqrt{2}\sin(2\pi 60000t)$.

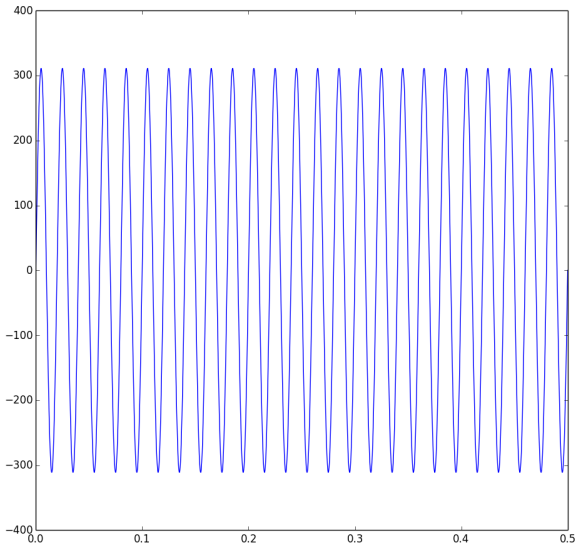

Figure 5: *50 Hz signal plot in blue*

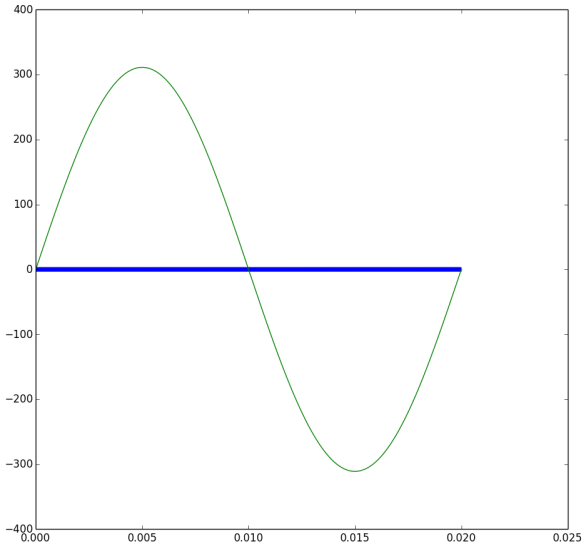

Figure 6: *Power supply signal (green) against the carrier (blue)*

The Python script that superposes these two signals is shown in figure 7, and its resulting plot (zoomed at 1ms) in figure 8.

```
[...]
t=np.linspace(0,0.001, 10000) # we zoom at 1ms, 10000 samples
plot = plt.plot(t,220*np.sqrt(2)*np.sin(2*np.pi*50*t)+2*np.sqrt(2)*np.sin(2*np.pi*60000*t)
plt.show()
```

Figure 7: *Power supply signal represented in Python*

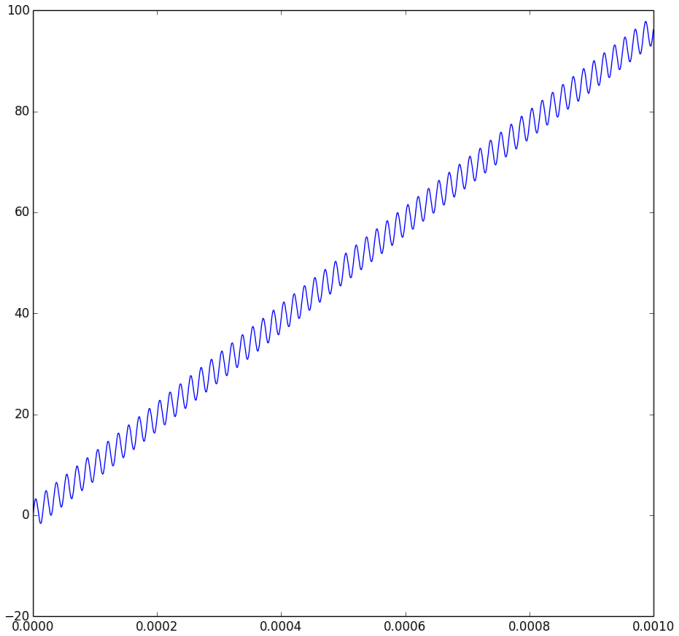

Figure 8: *Carrier superposed with power supply signal (zoomed at 1ms)*

We saw how data is transported over the powerline, but it is not enough to guarantee a good transmission of data. Of course, we need at least error detection, code mapping and multi-carrier modulation.

To avoid a succession of "1" and "0" bits we need to scramble the data by using $\oplus$ operator between the data and a pseudo-noise sequence generated with a linear feedback shift register (LFSR) defined by the polynomial $P(x) = x^{10} + x^3 + 1$ (specified by HomePlug). Then the Turbo Encoding allows an error rate of 105 against a SNR (Signal-to-Noise Ratio) equal to 0,7 dB. The output of Turbo-code are then mixed by an interleaver before the modulation. This allows to disperse bits with the same informations on many OFDM symbols, so transmission errors would be dispersed. Control and data frame are modulated by constellation. Control frame are modulated with QPSK (Quadrature Phase Shift Key) and data frame can be modulated with QPSK, BPSK (Binary Phase Shift Keying), 8-QAM (Quadrature Amplitude Modulation), 16-QAM, 64-QAM, 256-QUAM and 1024-QAM. The constellation allows to code the symbols that will be grouped to

form an OFDM symbol. Then these symbols are transmitted to multiple carriers. This procedure consists of adding many signals of different frequencies and narrow band to form a large signal to transmit all symbols in parallel [6] [7].

Due to multi-path scattering, an OFDM symbol could be perturbed by the previous symbol if it arrives too late. To avoid this phenomenon known as ISI (Inter-Symbol Interference), we add a GI (Guard Interval) which consists to copy the end of the previous symbol at the beginning of the next one. Then the windowing is performed on OFDM symbols to reduce the out-of-band emission and reduce the spectral side lobe. To finish, the symbols are converted and transmitted by the electrical coupler.

To receive the data, the inverse of this process should be done (see figure 9).

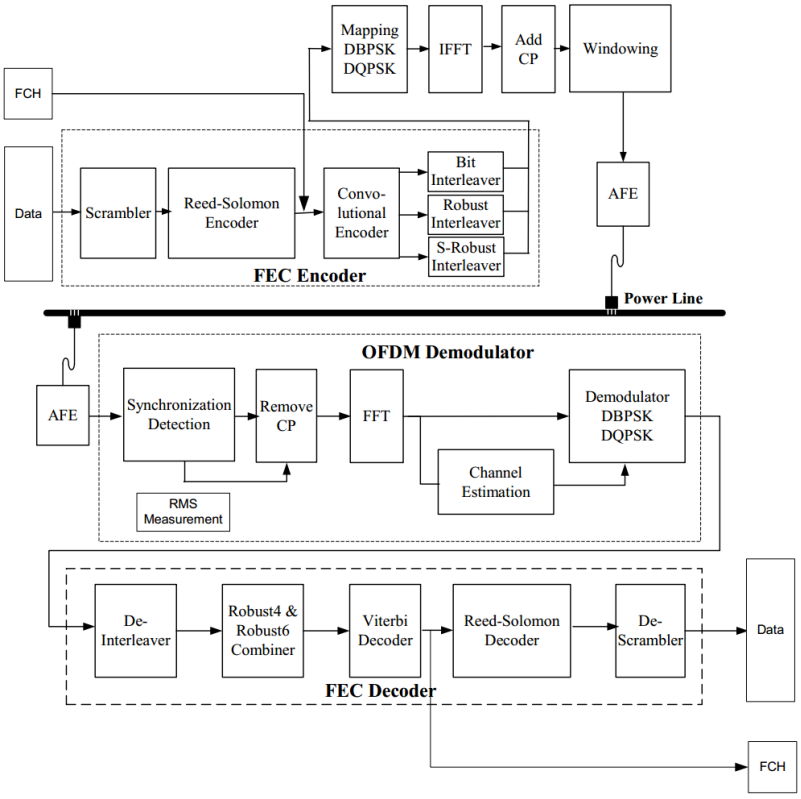

Figure 9: *OFDM modulation/demodulation [6]*

2.2 PLC layers

A PLC uses 2 layers to communicate: Physical and Data link (precisely the MAC). Endpoints offer an Ethernet IEEE 802.3 interface, so a user could easily plug its cable (figure 10).

To avoid collision, as the power line is not normally supposed to transport data, PLC uses the CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) protocol with the back-off

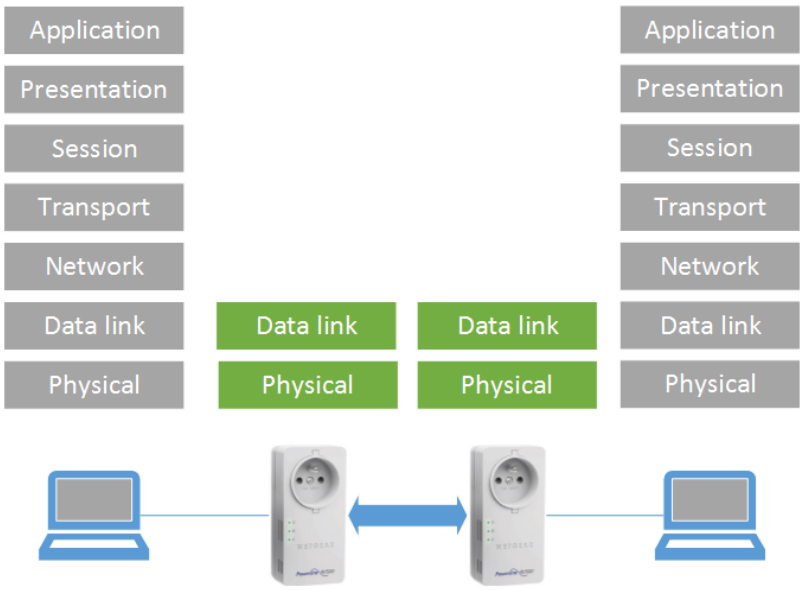

Figure 10: *OSI model applied to PLC*

algorithm to wait until the station is able to send data.

In HomePlug AV, a TDMA timeslot (Time Division Multiple Access) is used to allocate a period of time for a transmission for each station. That allows to manage the QoS from the CCo (Central Coordinator). Generally, 1 TDMA slot is reserved for CSMA/CA frames for services that do not need QoS, and the other slots are used for VoIP, TV, streaming, etc.

2.3 Evolution

The figure 11 represents the evolution of PLC speed rate over the time. Developed in 1975 by Pico Electronics, the X10 was designed to compete with other transmission systems communicating by radio waves or bus wires.

From our point of view, only the specification of HomePlug 1.0 and 1.1 are finalized, implemented and used by many products in the market as we will see in the next sections.

Also the last generation of PLC uses the best mechanisms of coding, modulation, and error correction to offer a good speed of transmission. Latest HomePlug AV also offers a better encryption between devices.

The following table shows a comparison between different HomePlug systems, WiFi and Ethernet accesses:

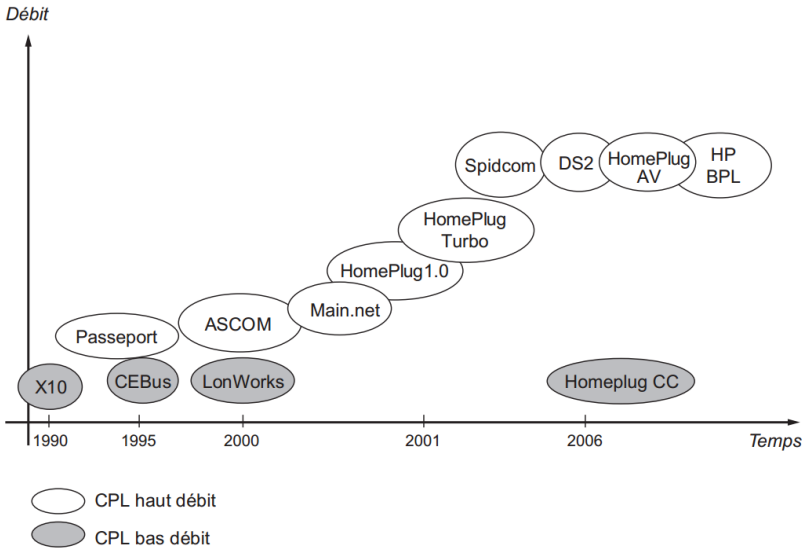

Figure 11: *Evolution of PLC speed rate over the time. Source: [4]*

Device	Speed in Mbits/s	Encryption
HomePlug 1.0	14 Mbits/s	56-bits DES
HomePlug Turbo	85 Mbits/s	56-bits DES
HomePlug AV 1.0	200 Mbits/s	128-bits AES CBC
HomePlug AV 1.1	500 Mbits/s	128-bits AES CBC
HomePlug AV 2	600 Mbits/s	128-bits AES CBC
Ethernet IEEE 802.3ba	40-100 Gbits/s	
WiFi 802.11n	300 Mbits/s	
WiFi 802.11ac	> 800 Mbits/s	

As we can see, the PLC begins to compete with Wi-Fi exceeding 802.11n, but slower than 802.11ac and Ethernet. These different systems are interoperable with each other.

2.4 Interoperability

Generally, HomePlug AV and AV2 (as GP) are fully interoperable, and can also interoperate with IEEE 1901 devices [5]. The different specifications of HomePlug devices are aimed to be interoperable and scalable but, other devices and HomePlug CC are not compatible with HomePlug products (figure 12).

2.5 Perturbations and attenuations

2.5.1 Perturbations

The electrical wiring is not aimed to allow data transfert, and a communication between two PLCs is often disturbed by other elements in this line. In sinusoidal steady state, an electronic circuit connected to the power supply is characterised by its impedance: $Z = R + jL2\pi f + \frac{1}{C2\pi f}$ Ohm, by and every connected circuits make a network of impedances (in series/parallel). The

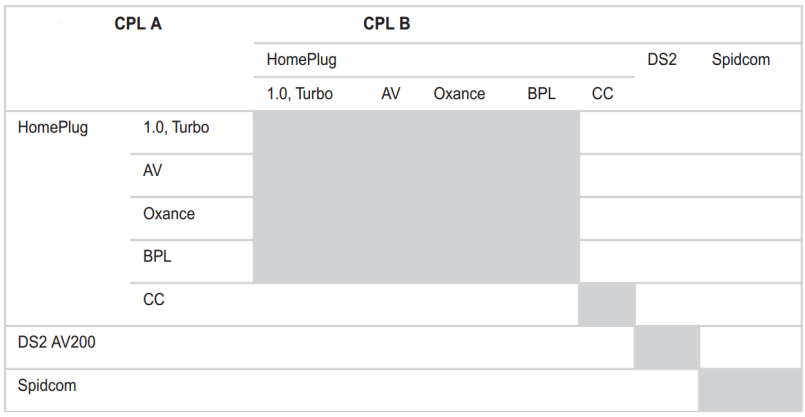

Figure 12: *Evolution of PLC speed over the time. Source: [4]. Legend: - Blank => not compatible; - Gray => compatible*

induction ($L = \frac{U}{2\pi f I}$ henry) and the capacity ($C = \frac{I}{2\pi f I}$ farad) parts of a circuit modify the transmission over the power supply. Moreover, circuits aim to be plugged and unplugged all the time, so good techniques of transmission are needed, but also standards to measure the EMC (Electro-Magnetic Compatibility) of each circuit. In Europe, when a product is marked as CE, it means it satisfies EMC's required level/standart. We will explain further this part, but it should be said that committees like Cenélec, CEN and ETSI (in Europe) are working hard to harmonize Power-Line Comunications.

2.5.2 Attenuations

Like many signals (Ethernet, radio, ...), the power supply signal suffers some attenuation, depending on the distance that crossed this signal on the line. Also when the signal crosses equipments like meters, circuit breaker, multi-sockets, and so on, the attenuation (measured in dB/Km) varies depending on the quality and age of the equipments. Moreover, multi-sockets are known to be an important source of attenuation that makes harder the communication between two PLC.

For security reasons, and to avoid perturbation on the public network, electricity providers use a sort of filter on the meters. But as we will see on the next sections, these filters are not always efficient, or even not implemented at all on many installations.

3 Our targets

3.1 Our devices

To perform our tests, we used a Netgear XWNB5602 kit with two different PLCs:

Model	Max Speed	Chipset	Extra features
XAV5401	500 Mb/s	Qualcomm Atheros 7420	
XWN5001	500 Mb/s	Qualcomm Atheros 7420	Smart Plug + WiFi N300

We also acquired other PLCs to test different versions of the HomePlugAV protocols:

Model	Max Speed	Chipset
TL-PA6030	600 Mb/s	Qualcomm Atheros 7450
FreeplugV1	200 Mb/s	INT6300
FreeplugV2	200 Mb/s	INT6400

In France, the PLC FreeplugV1 was sold with the Freebox v5 and the FreeplugV2 with Freebox v6. Freebox v5 and v6 are distributed by the ISP Free.

3.2 Public and private networks: myths and reality

3.2.1 Public and private network: the concept

The concept of public and private networks is very simple. If the PLC signal is broadcasted behind the electrical counter, it should be restricted to the apartment. On the other hand, if the signal is broadcasted before the electrical counter, then any device plugged in the building hall can intercept it and so it becomes a public network as represented in figure 13.

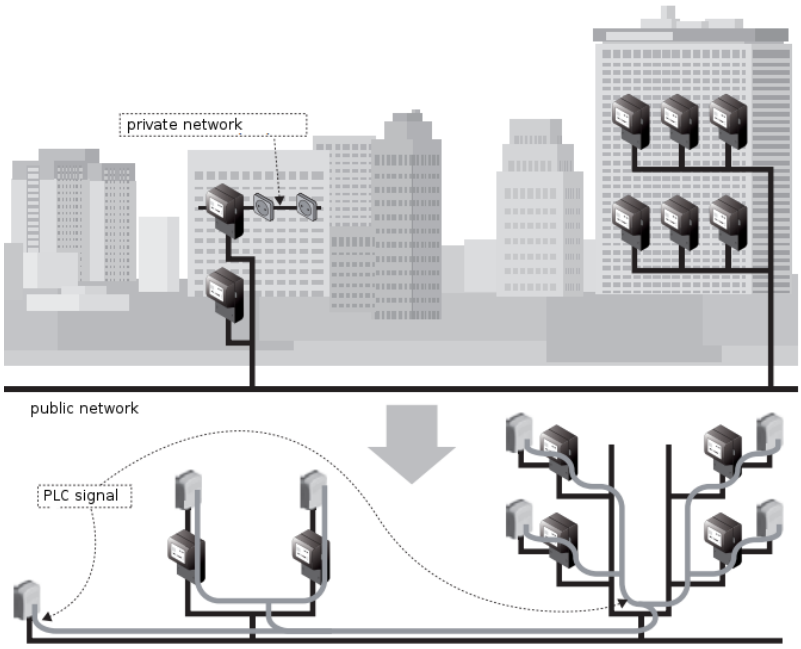

Figure 13: *Private and public network (source: [4])*

However, in reality, if the electrical counter does not have an appropriate choke coil, the frequencies used by the PLC can easily pass it.

This choke coil behaves as a resistance that changes depending on the frequency. The higher the frequency is, the more impedance of the choke coil will have. But even if the electrical counter

is equipped with a choke coil, sometimes, some frequencies as PLC might pass as the filters are not perfect.

3.2.2 Our tests: the reality in France

We tested the HomePlugAV by Netgear (the XWN5001 and XAV5602) to see if the concept of private network is realistic. We also tested HomePlugAV2.

The tests have been realized in the 15th district of Paris and in two apartments in the suburb of Paris. One PLC was plugged at home and the second in the bottom landing. Our results show us that it is possible to communicate between apartments without problems. We did not test the communication between two houses or two buildings.

3.3 Internet providers: the case of Freebox ADSL (in France)

3.3.1 Integrated PLC in Freebox power adapters

Free provides two power supplies to connect the Set-top-Box and the router. If we take a better look at the power supply cable, there is also an Ethernet cable joined with the power supply cable. Normally, we suspect that an unsuspecting user will connect both, just to be sure that everything will work as expected (figure 14).

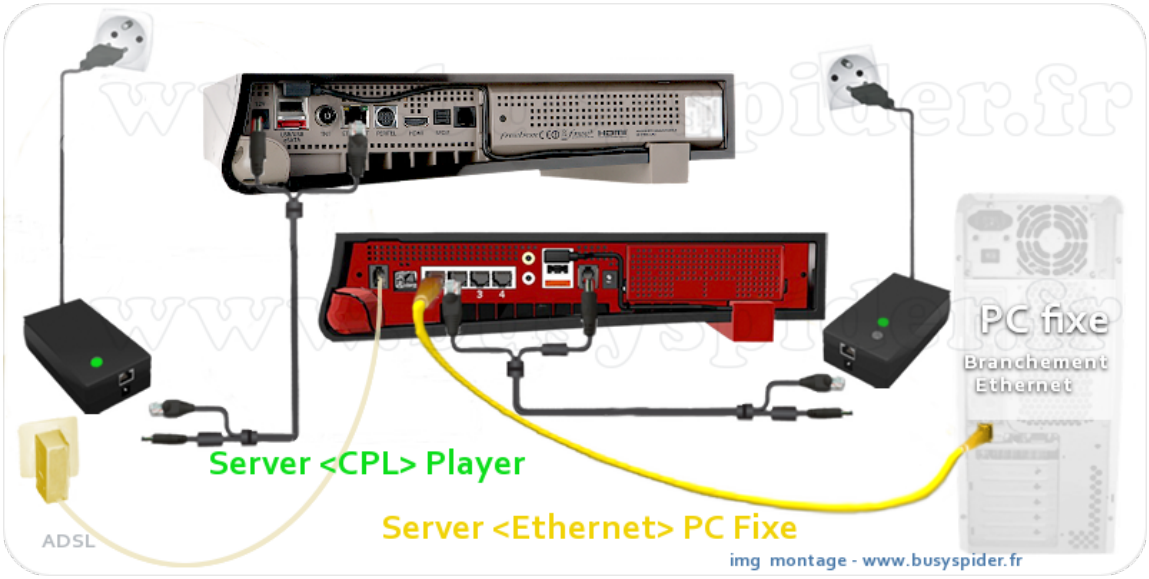

Figure 14: *Freeplugs for the STB and the Freebox (source: http://www.busyspider.fr)*

However, Freeplug is not our only target as many other ISP provide PLC to their customers.

3.3.2 Theoretical covering

Theoretically, Free counts 5 702 000 users (figure 15). Starting with the Freebox v5, "Free.fr" gave PLC for every subscribers. The impact of a vulnerability could be very interesting for an

attacker.

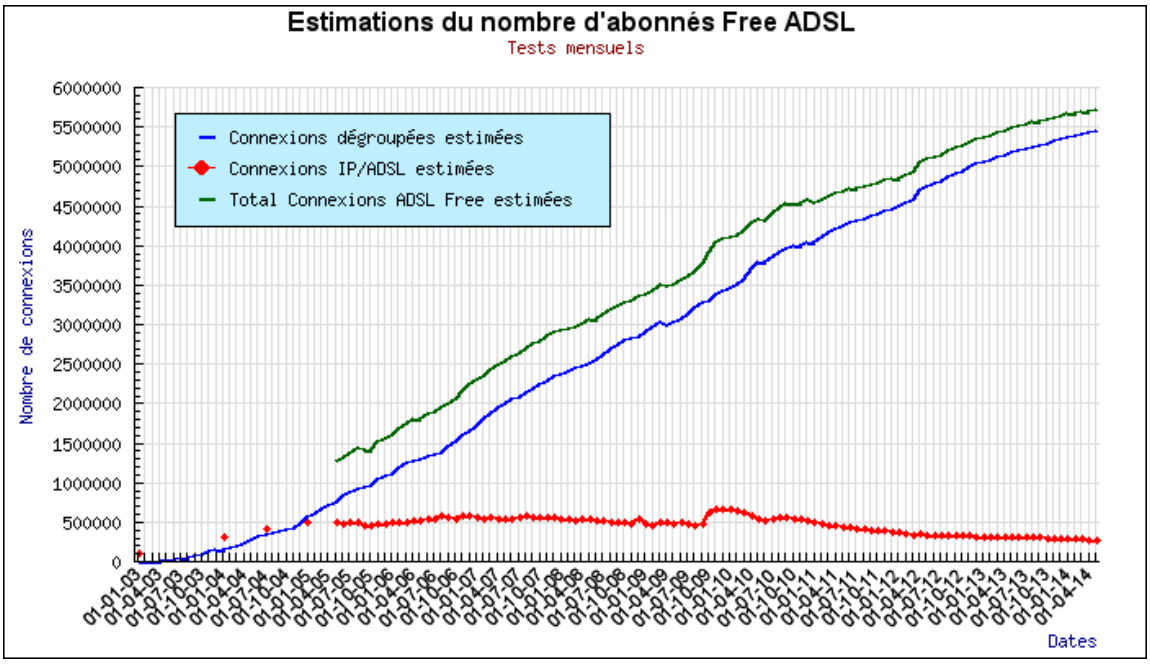

Figure 15: *Estimation of Free.fr ADSL subscribers. Source:* `http://francois04.free.fr`

4 State Of the Art

4.1 Publications

Unfortunately, there are not many publications on the subject. Xavier Carcelle has published a book that explains how PLCs work, the network concepts, few security modes and mechanisms, packet formats, etc.[4] This book is definitely a must read!

Also, a great overview of HomePlug AV Security Mechanisms has been published in 2007 by Richard Newman, Larry Younge, Sherman Gavette, and Ross Anderson [8]. This paper gives some details about low level pairing between two or more PLC.

Xavier Carcelle also contributed for an article, published in MISC #37 (2008), showing the security modes and mechanisms and highlighting the strength of the ciphering on HomePlug AV PLC. However, none of those books or papers focus on different possible attacks.

Only one research in 2009 shows an interesting attack by Axel Puppe and Jeroen Vanderauwera, that consists of bruteforcing the NEK key to access on a network with old HomePlugs [9]. However the choice of the NEK key was limited to a specific dictionary.

Another paper that we discovered later is “The security mechanisms in PLC technologies” (Bezpenostní mechanismy v PLC technologii) [12]. This paper published in 2011 describes a few cryptographic mechanisms and exchanges in the network, but does not explains possible attacks.

4.2 Tools

To manage or study a PLC network, some tools exist in addition to vendor tools like "plconfig" [11]. At 25C3, Xavier Carcelle also presented an open PLC tool named FAIFA [10] to support the HomePlug and HomePlugAV standard.

To study HomePlugAV packets over the network, we can use the existing Wireshark dissector. But no Scapy layer exists for the moment to forge special packets and to find bugs in the protocol.

4.3 Our researches

In 2014, we dug up this subject and upgraded it with old and new angles of attacks.

We present here some new attacks on PLC. In this work we focus on possible attacks that could be performed on a local Ethernet network context, but also when remotely targeting a private network. Also, we try to understand how these PLC devices work to get sufficient knowledge to backdoor them. Also, we show a Scapy layer and additional tools that can be used to attack the HomePlugAV protocol.

5 Network analysis

There are 3 different ways to configure the network with two PLC:
- default configuration (open network);
- pairing button;
- or using with a vendor utility software.

As the software is made to configure a PLC, we use the third method to capture the communication between the software and the device. For this analysis, we will use the “Netgear Powerline Utility” acquired with the Netgear PLC.

5.1 Vendors utility

The utility (in figure 16) provides many information about PLC configuration like the MAC address, PLC name, firmware version, and other information depending on the PLC type. As there is no Scapy layer for HomePlugAV, we will create and try to reproduce what the software does during the first discovery.

5.2 Analysis of PLC packets

5.2.1 Get a list of connected devices and information

In order to discover the list of connected devices the software use a “Get Version Request” packet as shown in figure 17. The destination MAC address corresponds to the 3 bytes “OUI” relative to Atheros and the last “NIC” byte is set to “1”.

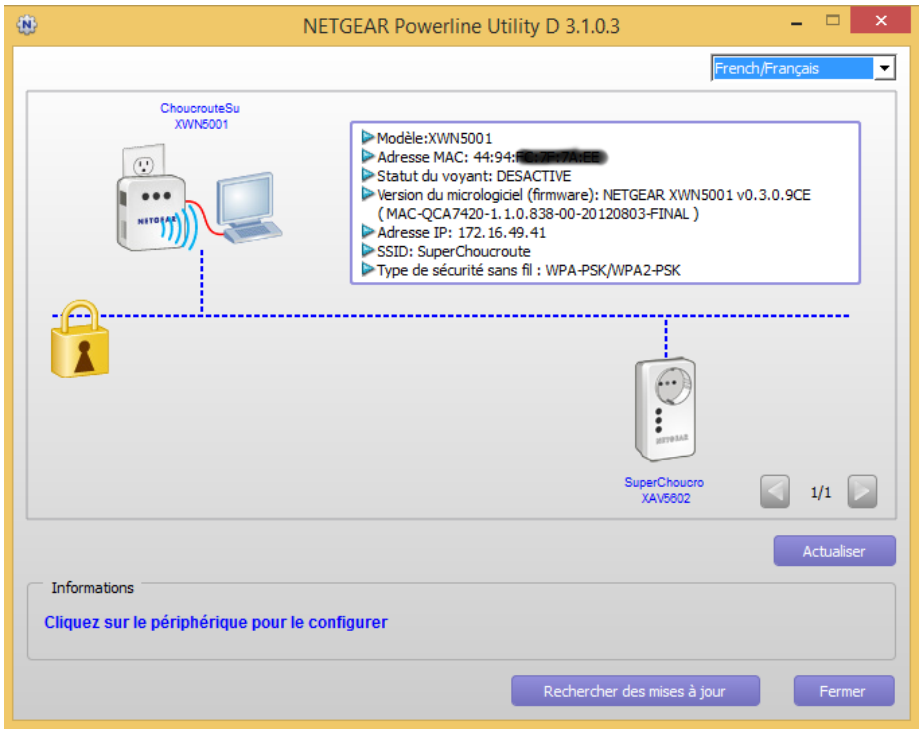

Figure 16: *Netgear Powerline Utility*

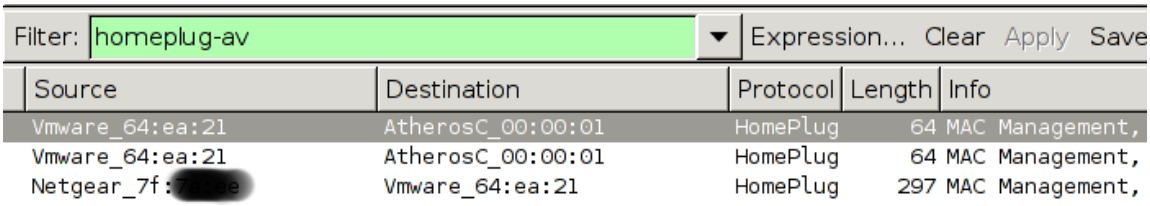

Figure 17: *Analysis with Wireshark*

The implementation of the first packet "Get Version Request" sent by the software is quite simple, and it is encapsulated in the MAC layer as shown in figure 18. Usually, the vendor software is using Atheros MAC address to contact PLCs. This address is like a broadcast for every PLC, and as Qualcomm acquired INTELLON and Atheros, all HomePlug AV should reply to this MAC address. However, as we want to cover as much devices as possible, we use the full LAN segment broadcast address `ff:ff:ff:ff:ff:ff`.

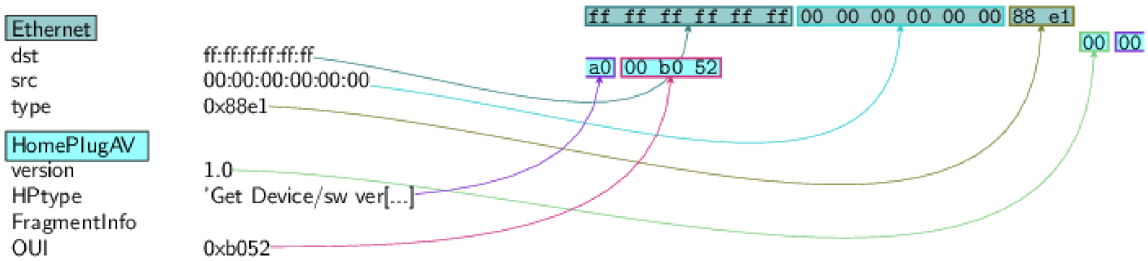

Figure 18: *Get Version Packet*

The first byte in this layer indicates the type of the request which is "Get Version Packet type". When the packet reaches a device it responds with a "Get Version Confirmation" as show in figure 19.

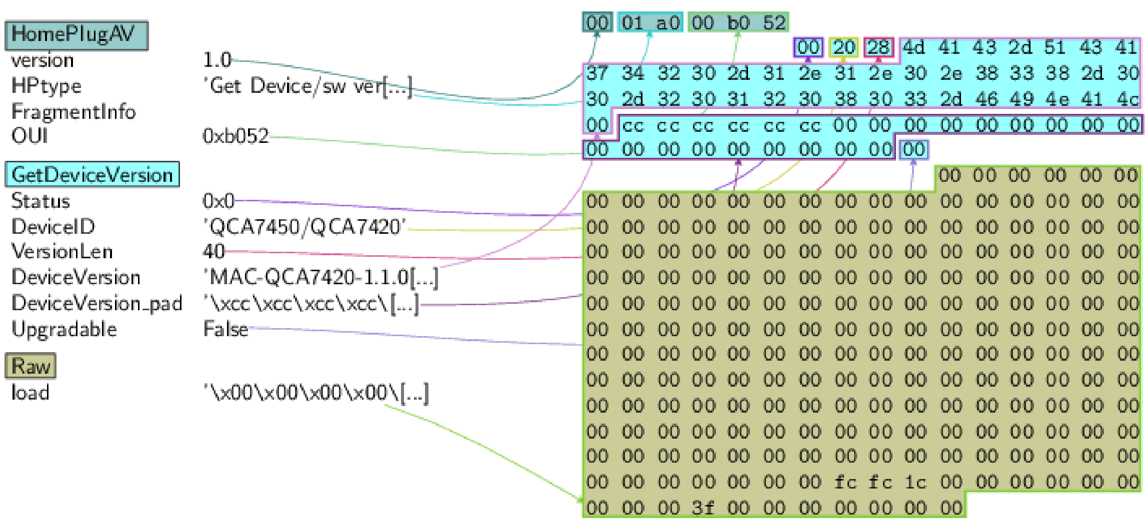

Figure 19: *Get Version Confirmation*

The response contains the device ID and firmware version. The information about the network is requested by the software in the form of a "Network Information Request" (figure 20) packet and the local PLC device replies with a "Network Information Confirmation" packet (figure 21).

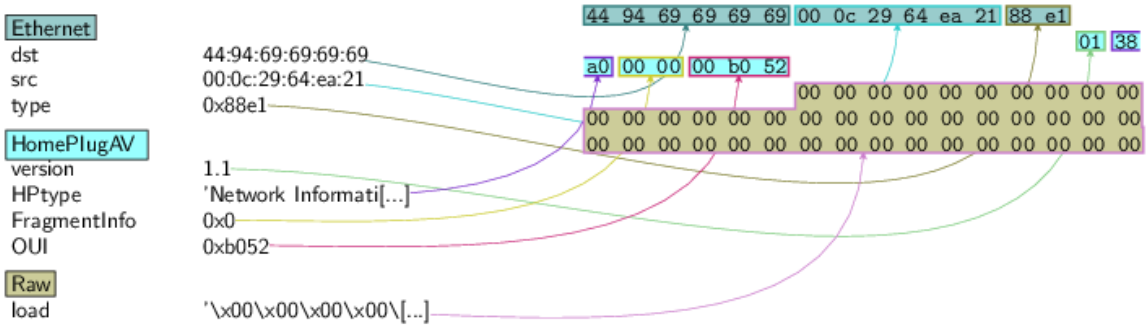

Figure 20: *Network Information Request*

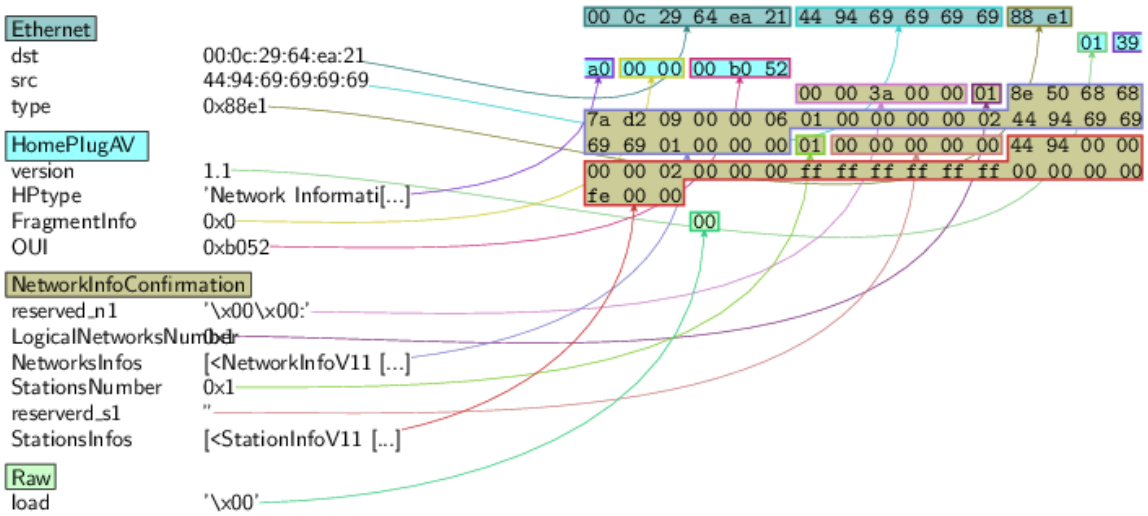

Figure 21: *Network Information Confirmation*

The stations information ("StationsInfos") field contains a list of stations connected to the same network as shown in figure 22.

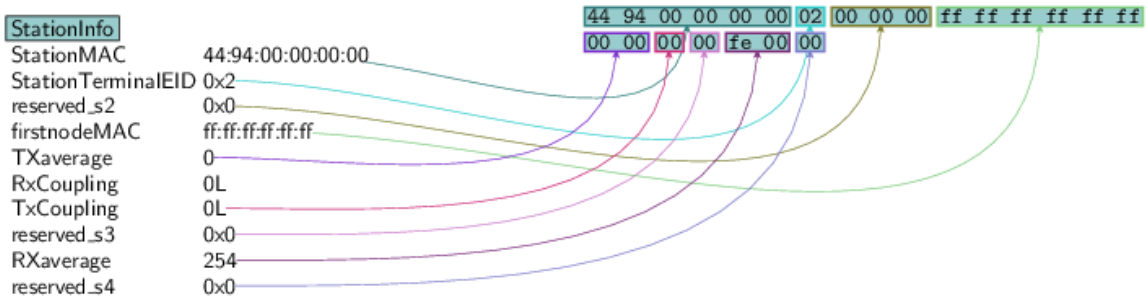

Figure 22: *Stations information field*

The software thus knows which MAC address it has to contact in order to get information about other devices. PLC devices are managed by a coordinator called CCo (Central Coordinator).

5.2.2 Central Coordinator

The "Networks Information" field contains a list of networks the device belongs to, and each network is identified with a "NetworkID" as shown in figure 23.

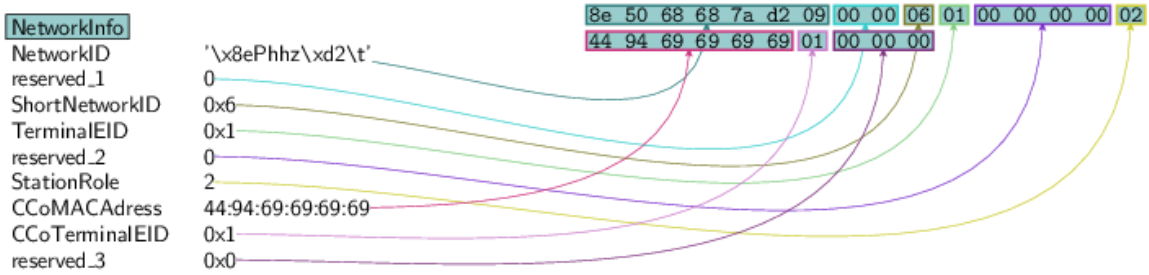

Figure 23: *Networks Information*

Two stations using the same NMK (Network Membership Key) are in the same network, more precisely an AVLN (AV Logical Network). The central coordinator manages contention-free streams time allocation, period for CSMA access and defines a AVLN node. Inside this AVLN node, the information that passes the electrical signal is encrypted using AES-CBC.

5.2.3 Association with a custom passphrase

Back to Netgear Powerline Utility software, we change the NMK of our PLC (that was our CCo) (figure 24).

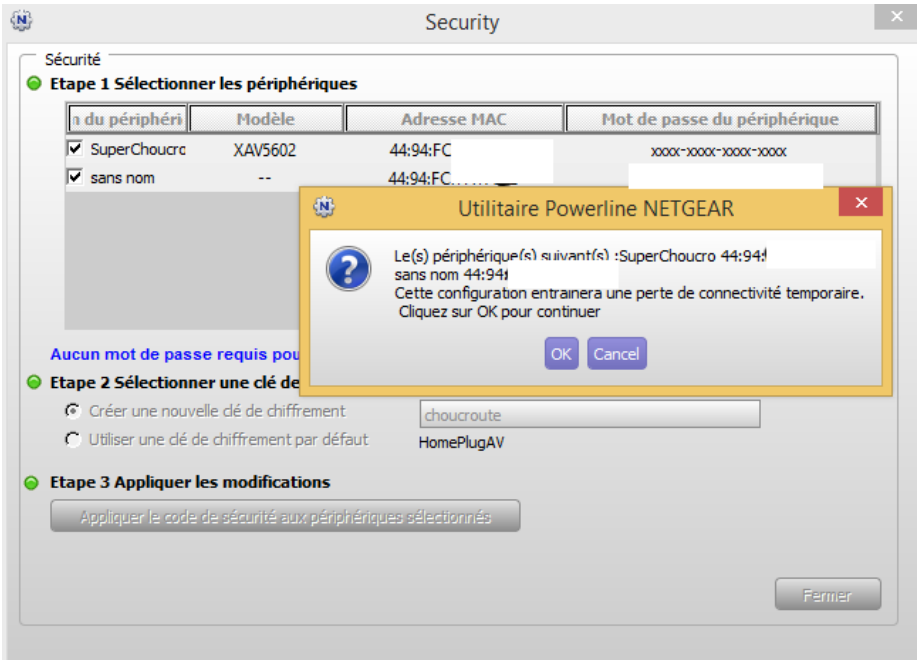

Figure 24: *Custom password setting*

With Wireshark we capture the packets sent by the software as shown in figure 25 and observe the PLC response that acknowledges the setup, with a confirmation packet.

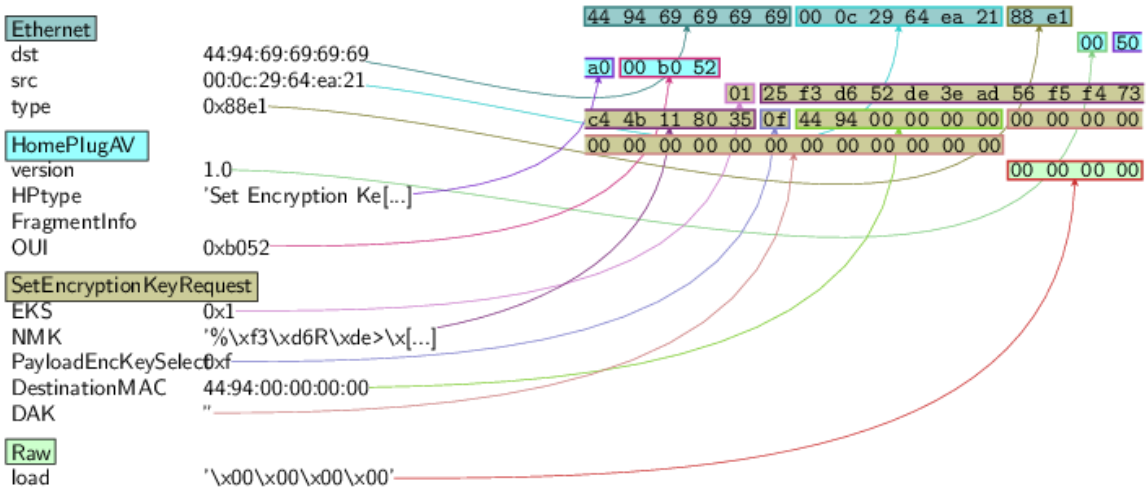

Figure 25: *Set Encryption Key Request packet*

This process is the same for a remote device, but need a DAK (Direct Access Key). The DAK is the key used to configure a remote PLC. This key, written on the PLC, is static and cannot be changed.

If we send a "Set Encryption Key Request" packet from a local Ethernet network, the Encryption Key of every PLC connected to this network with the Ethernet interface could be changed without the DAK key.

5.2.4 NMK and DAK generation

Captured packets contain the NMK and the DAK keys which are not sent in clear text. As defined by HomePlug specifications, NMK and DAK are generated by using the Password-Based Derivation Function 1 (PBKDF1): $NMK = PBKDF1(P, S, HF, c, dkLen)$, where P is the passphrase, S the salt, HF the hash function, c the number of iterations, $dkLen$ the digest key length.

The salts used for DAK key is `0x08856DAF7CF58185` and for NMK `0x08856DAF7CF58186` [13]. The hash algorithm used is SHA-256, the number of iterations is 1000 and size of the desired digest key is 16 bytes length. We can now focus on possible attacks that could be performed against the NMK.

5.3 Attacks on NMK

5.3.1 Interception

A NMK sent over the local network is not encrypted, so if any eavesdropper intercept it, he or she can use it to configure his or her own PLC device to access the network.

5.3.2 The Ethernet attack

A PLC in local or connected to the router/modem can be configured without any DAK key. So in this scenario, the attacker can change the NMK of the CCo (figure 26) and then use his own NMK. But we will see later that it is possible to recover the NMK key from HomePlugAV memory when we can communicate with these devices.

5.3.3 Bruteforcing the NMK

Inspired from "HomePlug AV Security Mechanisms" paper[9], the difficulty of this attack depends on user password policy. Unfortunately, many users do not configure their PLC and a default one ("HomePlug") will be used. The bruteforce consists in changing the NMK of the local device at every iteration, and trying to discover other devices with a "Get Version Packet". However, this attack is time consuming and varies by passphrase strength, so we switched to another more efficient attack.

6 The K.O.DAK attack

We observed that, when changing the NMK remotely, our devices needed a DAK key. If we use that key to change the NMK with our key, we could penetrate neighbours LAN.

6.1 Market Researches

To get a better overview of every possible DAK passhrase patterns, we need as much samples possible. For such need, we go hunting in a few market places.

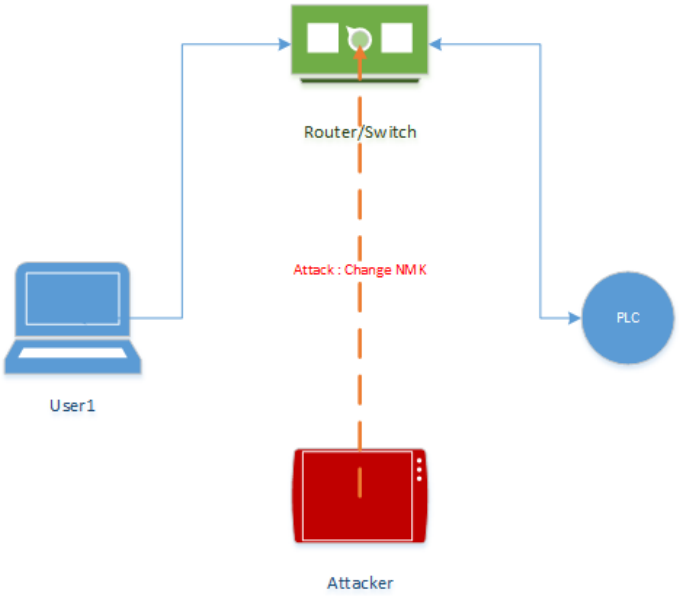

Figure 26: *Ethernet attack architecture*

6.1.1 Sample hunting in market places

In market places we took many picture of PLC's back as shown in figure 27 and 28.

Figure 27: *Captured DAK passphrase 1*

To continue our investigations we tried also online market place such as "leboncoin.fr" or "ebay", where we could find pictures of used DAK passphrase from many vendors.

Figure 28: *Captured DAK passphrase 2*

6.2 Our conclusion on the used pattern

After our researches, we reached the conclusion that a DAK passphrase is generally composed of 4 groups of 4 upper letters separated by "-" delimiters. Unlike NMK passphrase, the DAK passphrase is static and has a precise pattern. So we could use it as an advantage to perform a remote attack on PLC devices.

6.3 Attacking the neighbours

6.3.1 The dummy technique

With the previous pattern we can start a first algorithm to bruteforce keys implemented in figure 29.

```
import itertools
chars = [chr(65+i) for i in range(26)]
f = open("creepyDict.txt", "w+")
for x in itertools.product(chars, repeat=16)
    f.write("".join(chars[:4])+"-"+"".join(chars[4:8])+"-"+"".join(chars[8:12])+
        " -"+"".join(chars[12:16]))
f.close()
```

Figure 29: *Dummy DAK passphrase bruteforce*

But we have to remember that this technique consumes time but also disk space if we want to store all hashes. Also, we are limited by the rate of transfer over the network. It would therefore be interesting to understand how these DAK passphrases are generated (as Qualcomm Atheros has a strong presence on this market).

6.3.2 PWD brute-force reduction/optimisation

Looking at other PLC devices, we have found that TP-Link's software retrieves passphrases used for each DAK key in clear, even for Netgear devices (figure 30).

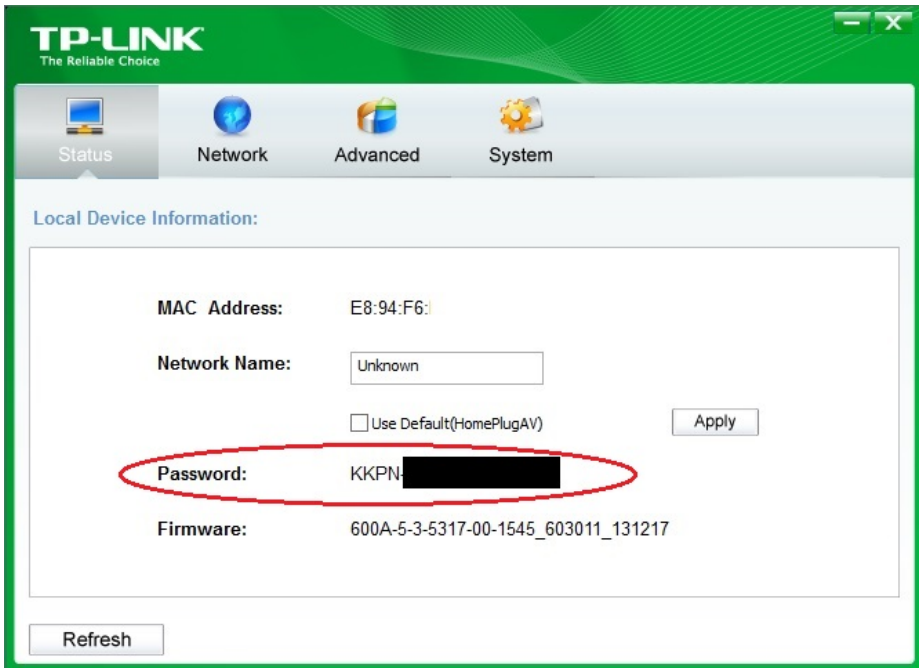

Figure 30: *TP-Link soft showing the passphrase used for DAK key*

Investingating HomePlugAV packets, some data seem to be sent to vendor software as shown in figure 31.

```
>>> hexdump(pkt.ModuleData)
0020   14 D1 00 00 41 74 68 65  72 6F 73 20 48 6F 6D 65   ....Atheros Home
0030   50 6C 75 67 20 41 56 20  44 65 76 69 63 65 00 00   Plug AV Device..
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   ................
0060   00 00 00 00 50 D3 E4 93  3F 85 5B 70 40 78 4D F8   ....P...?.[p@xM.
0070   15 AA 8D B7 74 70 76 65  72 5F 36 30 33 30 31 31   ....tpver_603011
0080   5F 31 33 31 32 31 37 5F  30 30 32 00 00 00 00 00   _131217_002.....
0090   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00a0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00b0   00 00 00 00 41 74 68 65  72 6F 73 20 45 6E 61 62   ....Atheros Enab
00c0   6C 65 64 20 4E 65 74 77  6F 72 6B 00 00 00 00 00   led Network.....
```

Figure 31: *Hexdump of a captured Read Module Data Confirmation Message*

To get these data, the software send a "Read Module Data Request" at offset `0x0` with a length of `0x400`. As we can see, the default passphrase is HomePlug and corresponds to `0x50d...b7` digest that we can quickly identify at address `0x064`. The DAK key is shown at the address `0x008` (that is hidden in this example).

Unfortunately, by observing every "Read Module Data Confirmation" packet, we didn't find any trace of the DAK passphrase. So, at this point, we think that the software computes the DAK passphrase. To be sure we look at `PLCOperApi.dll` file, providen by TP-Link, with a disassembler,

and can observe, in figure 32, some interesting strings like %02X%02X%02X%02X%02X%02X one in the .rdata section. As we can see the sub_10001190 address is called by a function called GetLocalDevInfo that retrieves information sending a "Read Module Data Request" to PIB. By disassembling this function, we see that the MAC address is divided in two parts: vendor part (more significant bytes) and device manufactured part (least significant bytes). Then two numbers will be generated for each part as described in figure 33.

```
.text:100011C5 MACProcess:                             ; CODE XREF: sub_10001190+1Fj
.text:100011C5                 movzx   ecx, byte ptr [eax+5]
.text:100011C9                 movzx   edx, byte ptr [eax+4]
.text:100011CD                 push    ecx
.text:100011CE                 movzx   ecx, byte ptr [eax+3]
.text:100011D2                 push    edx
.text:100011D3                 movzx   edx, byte ptr [eax+2]
.text:100011D7                 push    ecx
.text:100011D8                 movzx   ecx, byte ptr [eax+1]
.text:100011DC                 push    edx
.text:100011DD                 movzx   edx, byte ptr [eax]
.text:100011E0                 push    ecx
.text:100011E1                 push    edx
.text:100011E2                 lea     eax, [esp+38h+var_14]
.text:100011E6                 push    offset a02x02x02x02x02 ; "%02X%02X%02X%02X%02X%02X"
    // 6 bytes, sounds like MAC addr
.text:100011EB                 push    eax             ; char *
.text:100011EC                 call    _sprintf
```

Figure 32: *MAC bytes pushed on the stack*

```
integer32 count <- 0
integer32 vendornumber <- 0
For count < 6 do
    vendornumber <- vendornumber * 0x10
    vendornumber <- vendornumber + MACvendor[count]
    count <- count+1
// simplified thanks to open-plc-utils sources
// The same algorithm is used for device part
```

Figure 33: *Seed generation: vendor part*

When these two numbers are generated with the last algorithm, the vendor part will be computed by a function that will store the resulting bytes in an array (in figure 34) and like in pseudo-algorithm written in figure 35. The generation of the charset table is done in while until it stores 256 upper characters (figure 36).

Then comes the part of choosing these characters to build our DAK passphrase (figure 37). The last part is the selection of chars in the array. To do that, the algorithm uses the same steps as CharsetTableInit to generate the array of integers, but modulate the index with the size of the array to select a line of this array as described in figure 37.

```
.text:1000109B CharsetTableInit:                       ; CODE XREF: DAKprocess+2Ej
.text:1000109B                 push    esi
.text:1000109C                 push    41C64E6Dh
.text:100010A1                 push    eax
.text:100010A2                 push    ecx
.text:100010A3                 call    __allmul
.text:100010A8                 add     eax, 3029h
.text:100010AD                 adc     edx, esi
.text:100010AF                 mov     dword_100360B8, eax
.text:100010B4                 shrd    eax, edx, 10h
.text:100010B8                 mov     bl, al
.text:100010BA                 and     bl, 7Fh
.text:100010BD                 movzx   ecx, bl
.text:100010C0                 mov     dword_100360BC, edx
.text:100010C6                 push    ecx             ; int
.text:100010C7                 shr     edx, 10h
```

Figure 34: *Charset table generation with MAC vendor bytes*

```
integer32 vendornumber;
vendornumber <- vendonumber * 0x41C64E6D;
vendornumber <- vendornumber + 0x3029,
vendornumber <- shift_right(vendornumber, 0x10) & 0x7FFFFFFF // simplified
```

Figure 35: *Charset generation pseudo-algorithm*

```
.text:100010D6                 mov     [esp+ebp+120h+var_104], bl
.text:100010DA                 inc     ebp
.text:100010DB
.text:100010DB loc_100010DB:                           ; CODE XREF: DAKprocess+74j
.text:100010DB                 cmp     ebp, 100h
.text:100010E1                 jb      short loc_10001090
```

Figure 36: *Storing the characters*

```
integer32 i <- 0;
char DAK[0x10];
For i > 16 do
    DAK[i] <- toChar(CharsetTable[CharsetTableInit_begining % 256]); // another simplification
```

Figure 37: *Selecting characters*

After this analysis, we understand that this library helps the software to retrieve DAK device key, and if we implement it in Python for exemple using "-" separators and `f0:de:f1:c0:ff:ee` MAC address, we get something similar to a DAK pattern as shown in figure 38.

```
% python2 genDAK.py f0:de:f1:c0:ff:ee
QFLX-EFRE-QTGC-SZB
```

Figure 38: *Python implementation of DAK passhrase generation*

Here is a summary table of bruteforcing techniques difficulties:

Bruteforce technique	Possibilities
DAK passphrase	26^{16}
K.O.DAK classic	$256^6 \ll$ DAK brute-force
K.O.DAK with vendor bytes	256^3

Looking for a possible SDK, we have found with some keys like PLC, the strange value 0x41C64E6D and DAK a very interesting open source toolkit [14], commited by some developpers at Qualcomm Atheros. We suppose this kit is the base of the SDK provided for all vendors to help them generate a unique DAK passhrase. That means that our attack could work for every PLC manufactured by Qualcomm Atheros.

Note: In a perfect environment, this attack could be broadcasted in the entire building. And also PLC reply with a confirmation message, so the only thing that matters is the MAC address of the replying PLC.

6.4 Our results on different PLC

With different vendors, the K.O.DAK attack seems to work with all PLC provided by Qualcomm Atheros. On the other hand, Freeplug devices do not use the same Qualcomm Atheros algorithm for DAK passphrase generation. As "Free.fr" does not provide any software or clues, we could not retrieve it.

Here is a summary table of possible attacks on different PLC:

PLC Providers	Ethernet	NMK bruteforce	DAK bruteforce	K.O.DAK Attack
Qualcomm Atheros PLC	YES	YES	YES	YES
INTELLON	YES	YES	YES	MAYBE
ISP PLC	YES	YES	YES	NOT ALL Devices

Once we changed CCo's NMK, we can use neighbours internet access, but some data sent over the network caught our attention. So we will go deeper to understand how vendor and PLC part communicate.

7 Inside the PLC

7.1 Disassembling the hardware

When disassembling the device, we could distinguish 2 parts:
- vendor part (for example: Netgear);
- and the PLC part (for example: Qualcomm Atheros);

As we can see in figure 39, the vendor contains the Ethernet controller used to communicate through RJ45 line.

In figure 40, we have encircled some pins that are connected to the PLC part used to communicate in the Powerline.

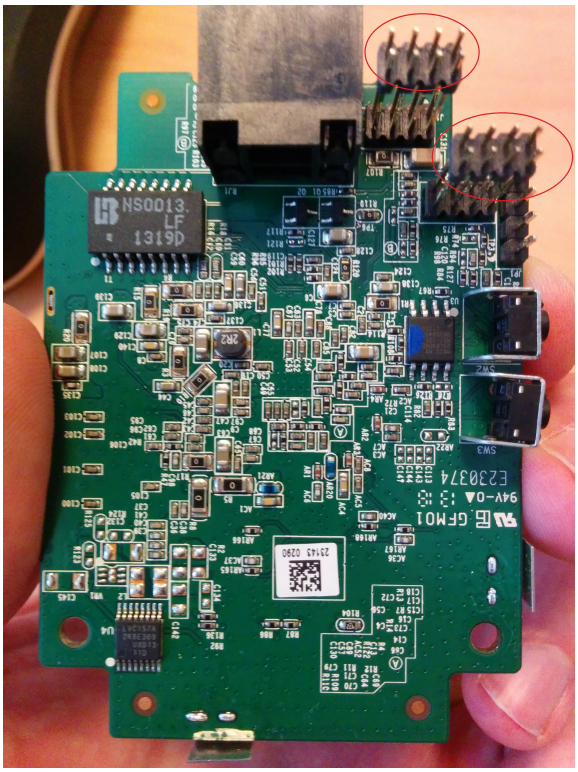

Figure 39: *Vendor part of a HomePlugAV*

These two previous pins correspond to MII (Media Independent Interface), or GPSI (General Purpose Serial Interface), that connects the PLC MAC/PHY transceiver to IEEE802.3 Ethernet MAC controllers. Figure 41 shows a simplified schema of the MAC/PHY transceiver.

7.2 Memory accesses with Ethernet interface

As we could see in the K.O.DAK attack part of this paper, the network interface has an access to the PIB (Parameter Information Block) or/and image of the memory, but we have also a write access to the PIB and IMG parts into the NVM. To perform an access, new PLC perform a "Read Data Module Request" command with 3 parameters:

- part of the memory : "MAC Soft-Loader Image" (0x0), "MAC Software Image" (0x01), "PIB" (0x02);
- offset;
- length.

7.2.1 Arbitrary Read and Write in memory

To read or write in memory, we choose what module we want to read from, then the offset and the size as in figure 42.

To write in a module, the fields are similar `Read Module Data Confirmation packet` without the "Status" field (figure 43).

Figure 40: *Qualcomm Atheros part of the HomePlugAV*

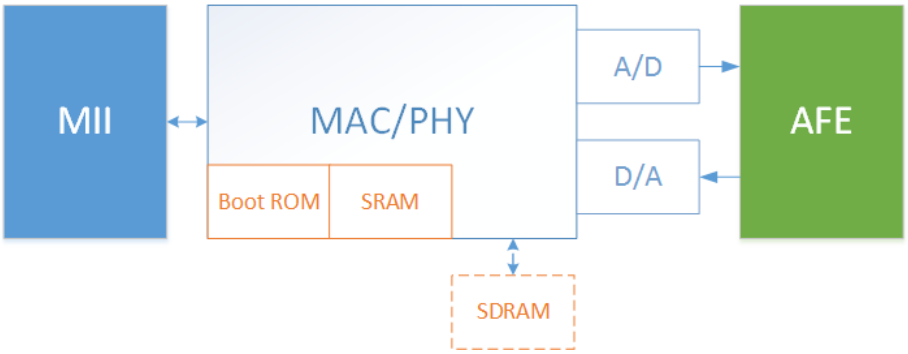

Figure 41: *MAC/PHY transceiver (simplified)*

Note: It is possible to patch the PIB to change the DAK key and avoid the K.O.DAK attack. Also, if you have a PLC Qualcomm Atheros 7420, the command may differ from QCA 7450 (0xA0B0 to read module data for example) and INTELLON PLC, but the principles are the same.

```
###[ HomePlugAV  ]###
    version   = 1.0
    HPtype    = 'Read Module Data Request'
    OUI       = 0xb052
    ###[ ReadModuleData ]###
        ModuleID  = PIB
        reserved  = 0x0
        Length    = 1024
        Offset    = 5120
```

Figure 42: *Selecting characters*

```
###[ HomePlugAV  ]###
    version   = 1.0
    HPtype    = 'Write Module Data Request'
    OUI       = 0xb052
    ###[ WriteModuleDataRequest ]###
        ModuleID  = PIB
        reserved  = 0x0
        DataLen   = 1024
        Offset    = 0
        checksum  = 975459083
        ModuleData= '\x05\x07\x00\x008@\x00\x00\xb1\x15)#
        [...]
```

Figure 43: *Selecting characters*

7.2.2 Some other functionalities

PLC have many functionalities a user can play with. One intersting functionality is the "Sniffer Indicate" as shown in figure 44.

140	158.140775000	WistronI_b	Broadcast	HomePlug	21	MAC Management, Sniffer Request
141	158.141081000	Tp-LinkT_e	WistronI_b3	HomePlug	60	MAC Management, Sniffer Confirmation
142	158.141474000	Tp-LinkT_e	WistronI_b3	HomePlug	186	MAC Management, Sniffer Indicate
143	158.153746000	Tp-LinkT_e	WistronI_b3	HomePlug	186	MAC Management, Sniffer Indicate
144	158.193671000	Tp-LinkT_e	WistronI_b3	HomePlug	186	MAC Management, Sniffer Indicate
145	158.233831000	Tp-LinkT_e	WistronI_b3	HomePlug	186	MAC Management, Sniffer Indicate
146	158.273699000	Tp-LinkT_e	WistronI_b3	HomePlug	186	MAC Management, Sniffer Indicate
147	158.313759000	Tp-LinkT_e	WistronI_b3	HomePlug	186	MAC Management, Sniffer Indicate

Figure 44: *Sniffer Indicate packets with Wireshark*

These packets give frame control and beacon status details, but also other surprising data that we will discuss in the following subsection.

Other commands could be interesting to discover like `VS_WRITE_AND_EXECUTE_APPLET` or `VS_MICROCONTROLLER_DIAG`. We will dig a little more to know if we can execute any other applet or try to communicate with the microcontroller. This way, we could avoid firmware updating and maybe stay persistent at the same time.

7.3 Avoid bruteforce with `Sniffer Indicate` packets

Later we discovered that someone independently found the same flaw as ours on Qualcomm Atheros vendors DAK key generation, but also saw that "Sniffer Indicate" packets contain MAC addresses of some CCo outside our AVLN [15].

Indeed, when we setup two PLC with Qualcomm chipset with a same NMK, and take another Qualcomm PLC that is not part of that network, change its `pkt.SnifferControl` parameter to "1", we get a few interesting "Sniffer Indicate" packets as shown in figure 45.

```
###[ HomePlugAV  ]###
    version   = 1.0
    HPtype    = 'Sniffer Indicate'
    OUI       = 0xb052
    ###[ SnifferIndicate ]###
        SnifferType= Regular
        Direction = Tx
        SystemTime= 399103809
        BeaconTime= 43033
        ShortNetworkID= 0x80
        [..]
        BeaconTimestamp= 2820139316
        BeaconTransOffset_0= 0x778b
        BeaconTransOffset_1= 0x108
        BeaconTransOffset_2= 0x100
        BeaconTransOffset_3= 0x205
        FrameContrchkSeq= 0x10600
        ###[ Raw ]###
            load      = '\x01\xfd40[...]
```

Figure 45: *Indicate Packet*

If we take the "Raw" part of that packet and search for bytes containing any of our PLC addresses, we can see at offset `0x0e` that the MAC address of one CCo is present in one of these packets beginning with the following vendor bytes: `E8 94 F6` (figure 46).

```
>>> hexdump(pkt.load)
0000   XX XX XX XX XX XX XX XX  XX XX XX XX XX XX E8 94   XXXXXXXXXXXXXX..
0010   F6 XX XX XX XX XX XX XX  XX XX XX XX XX XX XX XX   .XXXXXXXXXXXXXX
0020   BC 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
[...]
0070   00 00 00 00 CC CC CC CC
```

Figure 46: *Sniffer Indicate Raw payload*

That means we are able to get the MAC address of some CCo that are not part of our network. So we will not need to do any bruteforce to get an Internet access around us with that technique.

Note: If we change the key of the CCo, it will be better to change the key of the entire network, just to be sure that no one suspects any change. Moreover, if we consider the PLC were bought in the same KIT, we could combine the Sniffer technique with the bruteforce part of K.O.DAK attack, as only two bytes change generally between two PLC bought in a KIT.

8 Conclusion

Many vendors in the market are affected by the K.O.DAK attack that helps to retrieve many DAK passphrases. Freeplugs from "Free.fr" are not affected by this attack, but many other scenario could be performed to get and keep a permanent access like the Ethernet attack. A device connected to the Ethernet network is as accessible as a local device, giving accesses also to the entire memory of the attacked the device.

At least, to avoid the K.O.DAK attack it is recommended to patch the PIB itself to increase the bruteforcing difficulty.

Tools have been release at the follwing URL: https://github.com/sogeti-esec-lab/HomePlugPWN

References

[1] The Internet of Things: Connecting Objects by Hakima Chaouchi

[2] Sensors and the Internet of Things - ECS722 by Dr. Manik Gupta and Dr. Stefan Poslad

[3] http://classof1.com/homework_answers/electrical_engineering/power_line_communication

[4] Réseaux CPL par la pratique by Xavier Carcelle

[5] IEEE 1901 - Wikipedia: http://en.wikipedia.org/wiki/IEEE_1901

[6] PLC G3 Physical Layer Specification: http://www.maximintegrated.com/products/powerline/pdfs/G3-PLC-Physical-Layer-Specification.pdf

[7] An Overview of OFDM Based Narrowb and Power Line Communication Standards for Smart Grid Applications: http://www.idosi.org/wasj/wasj20(9)12/6.pdf

[8] HomePlug AV Security Mechanism by Richard Newman, Larry Younge, Sherman Gavette, and Ross Anderson

[9] Research Project: HomePlug Security by Axel Puppe and Jeroen Vanderauwera - http://www.delaat.net/rp/2009-2010/p19/report.pdf

[10] FAIFA: A first open source PLC tool by Xavier Carcelle and Florian - http://events.ccc.de/congress/2008/Fahrplan/events/2901.en.html

[11] https://neon1.net/prog/plconfig.html

[12] Bezpenostní mechanismy v PLC technologii - http://www.elektrorevue.cz/file.php?id=200000771-3e0553eff5

[13] HomePlug Specification Chapter 7 - http://www.cise.ufl.edu/~nemo/plc/refs/HomePlug%20GP_Specification_Ch7_CCo_v1.1-Jan_23_2012%20DRAFT.docx

[14] Qualcomm PLC toolkit - https://github.com/qca/open-plc-utils

[15] Vulnerability: Infiltrating a network via Powerline (HomePlugAV) adapters by Ben Tasker - https://www.bentasker.co.uk/documentation/security/282-infiltrating-a-network-via-powerline-homeplugav-adapters