SYNACKTIV
DIGITAL SECURITY

V2G Injector

Whispering to cars and charging units through the Power-Line

By Sébastien Dudek

SSTIC

June 7th 2019

Working team on the subject

@Fist0urs, @Karion_, and me

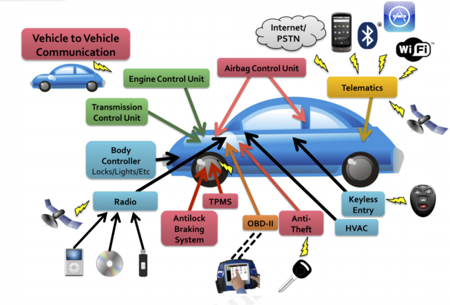
About me

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

* FR Offices in Paris, Toulouse, Lyon and now → Rennes!

Introduction

- Current cars → Controller Area Network (CAN) bus
- Engine Control Units (ECUs) → targeted via On-Board Diagnostics (OBD) port
- And plenty other surfaces to investigate:
  - Wi-Fi
  - GPRS, 3G and 4G*
  - etc.



source: thetruthaboutcars.com

*https://www.synacktiv.com/ressources/Troopers_NGI_2019-Modmobtools_and_tricks.pdf

Our interest: the charging connector

- Is it only used for charging?

Warning

Tons of abbreviations!

Let’s inspect this mysterious thing...

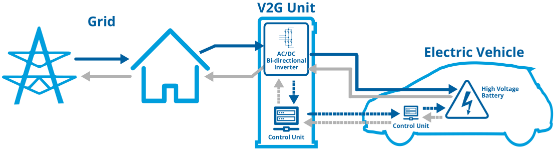
Long story short: renewable energy

- Renewable energy production → variable and difficult to predict (solar, wind, user consumption, etc.) → Smart Grids
- People had to think about ways to store it
- First energy storage system → Battery-to-Grid (B2G)

→ Why not use car's battery for energy storage too?

The rise of V2G

- V2G: Vehicle-to-Grid
- Use Electric Vehicles (EVs) to store energy
- In bidirectional charging/discharging systems → pay for charging or get paid → compensate battery deterioration



source: automobile-propre.com

Looking at specs → V2G systems communicate with a protocol

Standards for interoperability

V2G uses several standards to communicate:

- ISO/IEC 15118: Vehicle-to-Grid (V2G) communication
- IEC 61851: conductive charging system
- IEC 61850-90-8: communication networks for EVs
- and so on.

Publications

Very few of them tackle the security issues and improvements on V2G:

- Peng Wang Zhigang Ji Wenpeng Luan, Gen Li. Security of V2G Networks: A Review. Boletín Técnico, Vol.55, Issue 17, 2017
- Yan Zhang and Stein Gjessing. Securing Vehicle-to-Grid Communications in the Smart Grid. IEEE Wireless Communications, 2013.

Uses Power-Line → we published a critical vulnerability concerning DAK key generation on most HomePlug AV devices[1]

[1] http://www.nosuchcon.org/talks/2014/D1_03_Sebastien_Dudek_HomePlugAV_PLC.pdf

1 V2G communication

2 HomePlug Green PHY

3 Preliminaries

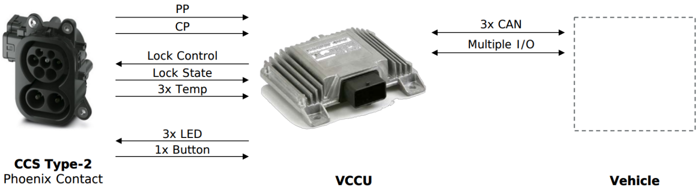
4 Intruding a V2G network

5 V2G Injector

6 Conclusion

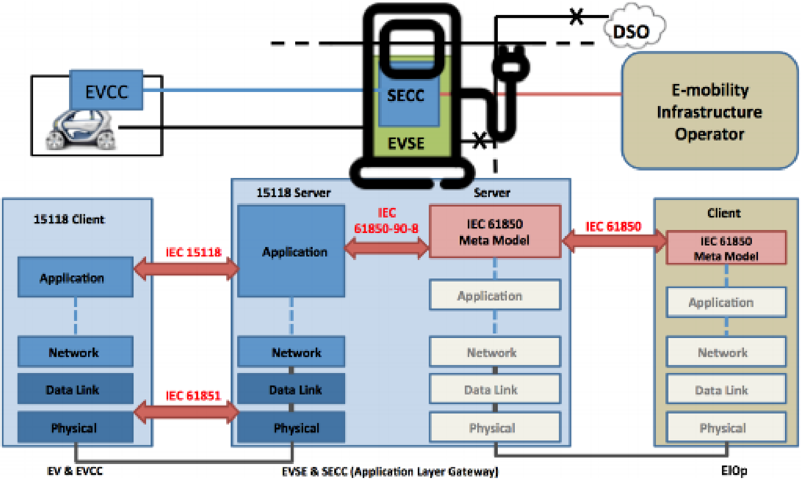
V2G ECU

- Known as Vehicle Charging Control Unit (VCCU)
- Interfaced with a Combined Charging System (CCS)
- ECU is used for: vehicle state management, communication with the backend, coordination, etc.



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

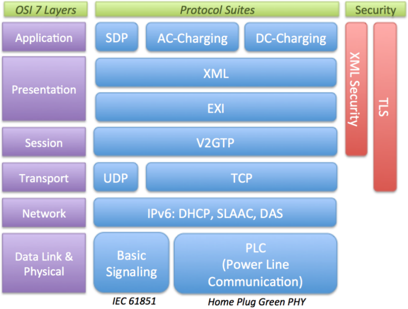
Architecture



source: https://res.mdpi.com/applsci/applsci-06- 00165/article_deploy/applsci-06-00165.pdf

V2G layers

- L1: PHY communication via a Power-Line Communication Device
- L2: Management Message Entries (MME)
- L3: Supply Equipment Communication Controller (SECC) on → EV Supply Equipment (EVSE) host and port
- L4: V2GTP transports V2G data
- ...



OSI 7 Layers | *Protocol Suites* | Security

Application | SDP | AC-Charging | DC-Charging

Presentation | XML | EXI

Session | V2GTP

Transport | UDP | TCP

Network | IPv6: DHCP, SLAAC, DAS

Data Link & Physical | Basic Signaling | PLC (Power Line Communication)

IEC 61851 | *Home Plug Green PHY*

XML Security | TLS

source: https://res.mdpi.com/applsci/applsci-06-00165/article_deploy/applsci-06-00165.pdf

TLS with V2G data

- TLS can be enabled → usually asked by EV Communication Controller (EVCC, client part)
- Must have two distinct private keys and certificates → ensure encryption and authenticity
- Needs a Certificate Authority (CA) to check Supply Equipment Communication Controller (SECC, server part)

Interesting to test to confront specs ↔ targeted implementation

TLS with V2G data

- TLS can be enabled → usually asked by EV Communication Controller (EVCC, client part)
- Must have two distinct private keys and certificates → ensure encryption and authenticity
- Needs a Certificate Authority (CA) to check Supply Equipment Communication Controller (SECC, server part)

Interesting to test to confront specs ↔ targeted implementation

Reality in heterogeneous envs

Complicated to put in the chain → how vendors are dealing with it? ... ;)

1 V2G communication

2 HomePlug Green PHY

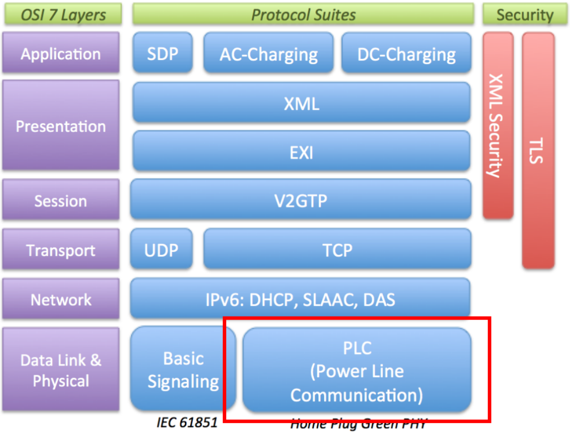
3 Preliminaries

4 Intruding a V2G network

5 V2G Injector

6 Conclusion

HomePlug Green PHY



OSI 7 Layers | Protocol Suites | Security

Application | SDP | AC-Charging | DC-Charging

Presentation | XML | EXI

Session | V2GTP

Transport | UDP | TCP

Network | IPv6: DHCP, SLAAC, DAS

Data Link & Physical | Basic Signaling | PLC (Power Line Communication)

XML Security | TLS

IEC 61851

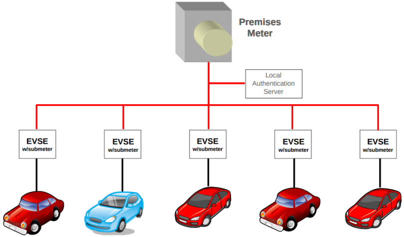
Home Plug Green PHY

HomePlug AV and Green PHY

- HomePlug Green PHY (HPGP) → subset of HomePlug AV
- HomePlug AV used to extend domestic local network
- HPGP Intented to be used for ”smart” grid or other automation systems
- HomePlug AV higher peak rate than HomePlug Green PHY
- Keys:
  - Network Membership Key (NMK): to encrypt the communication using 128-bit AES CBC
  - Direct Access Key (DAK): to remotely configure the NMK of a argeted PLC device over the Power-Line interface

Plug-in Electrical Vehicle (PEV) Association

- PLC packets are broadcasted in the Power-Line
- So after plugging → PEV does not know on which station it is connected

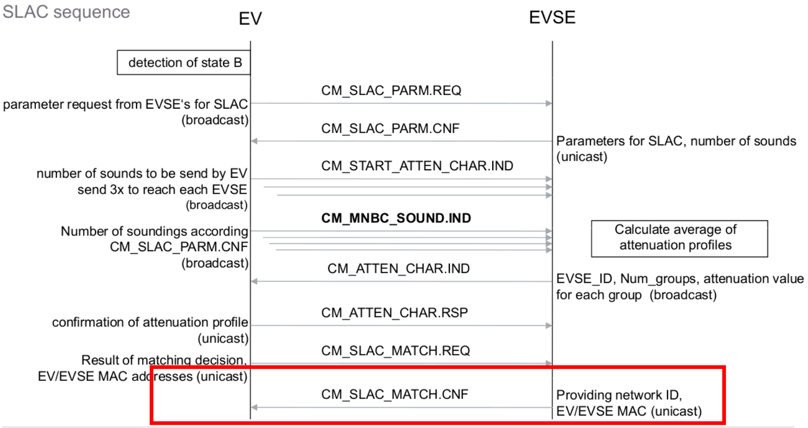


source: HomePlug Green PHY whitepaper

How to prevent from billing errors?

SLAC procedure

SLAC: Signal Level Attenuation Characterization



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

1 V2G communication

2 HomePlug Green PHY

3 Preliminaries

4 Intruding a V2G network

5 V2G Injector

6 Conclusion

Tools and specifications

- No free specifications
- Some monitoring tools like “V2G Viewer pro” exist, but expensive
- Free and useful stacks to understand V2G:
    - RISE-V2G
    - Open V2G
- Even HPGP dissectors are publicly missing for Wireshark, Scapy, etc.

Our contribution

- Made SECC, V2GTP and HomePlug GP Scapy layers
- Developed a V2G data encoder/decoder, based on RISE-V2G shared library
- Found a new flaw in HPGP SLAC procedure
- Combined all these tools to make a tool to monitor and inject crafted packets, called “V2G Injector”

Without reinventing the wheel!

1 V2G communication

2 HomePlug Green PHY

3 Preliminaries

4 **Intruding a V2G network**

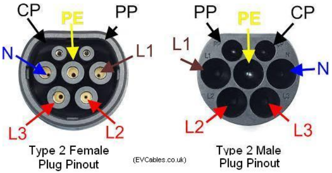
5 V2G Injector

6 Conclusion

Our interface: 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
- etc.



HGPG data multiplexed onto the Control Pilot and ground lines

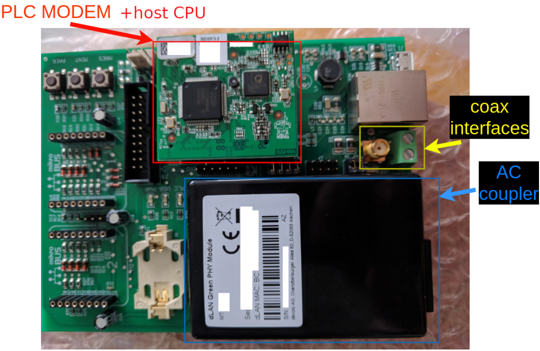
Data Propagation over Power-Line

As shown at NSC 2014 for HomePlug AV wallplugs:

- Data over Power-Line is superposed on the power supply
- Any information can propagate through many installations depending on signal strength
- If charging station charges shared the electrical network as a resident → a resident can see and contact charging station's PLC

Required hardware

- PLC with a QCA7k modem
- Tested with:
  - PLC Stamp Micro 2 Ev. Board (300€)
  - Devolo 1200+ (50€) → to rework if you want to bind it to CP lines
  - dLAN Green PHY ev. board EU II (150€):



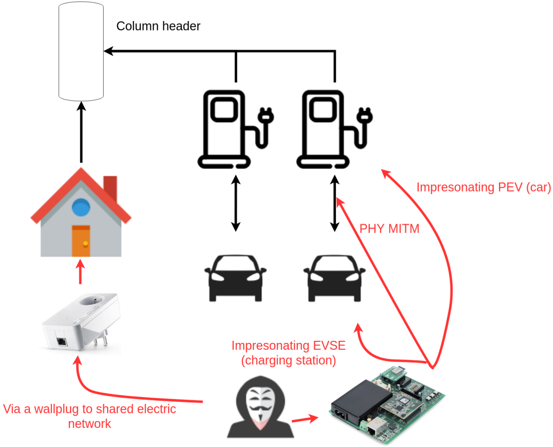
Text

Cheapest way: the wallplug

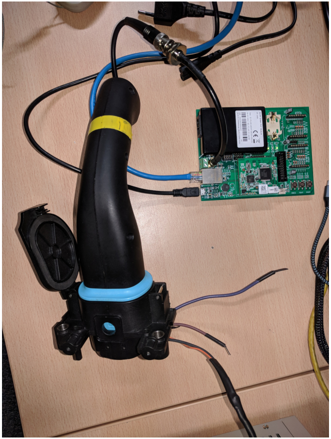
- Devolo 1200+ works like a charm
- No modification needed if charging stations share the same electrical network
- Otherwise some rework should be done on the coupler

We are actually working on some modular rework with this adaptor

How to interface

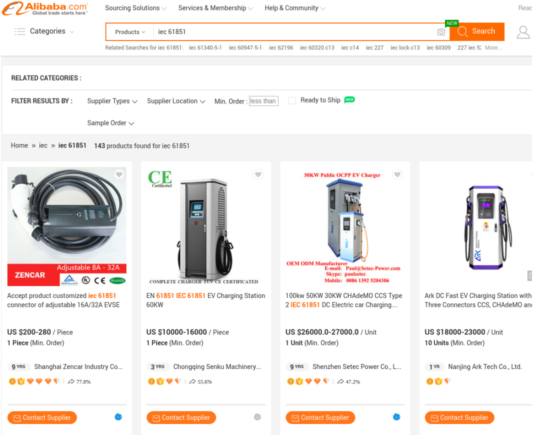


Impersonating a charging station (EVSE)



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

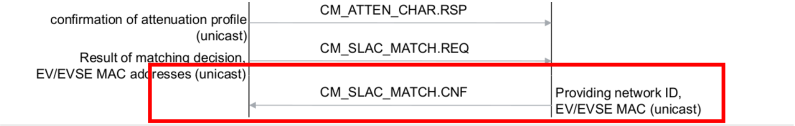
ZENCAR
Adjustable 8A - 32A
CE
RoHS
CE
Certificated
COMPLETE CHARGER TUV CE CERTIFICATED
50KW Public OCPP EV Charger
OEM ODM Manufacturer
E-mail: Paul@Setec-Power.com
Skype: paulsetec
Mobile: 0086 1392 5204306

Accept product customized iec 61851 connector of adjustable 16A/32A EVSE	EN 61851 IEC 61851 EV Charging Station 60KW	100kw 50KW 30KW CHAdeMO CCS Type 2 IEC 61851 DC Electric car Charging...	Ark DC Fast EV Charging Station with Three Connectors CCS, CHAdeMO an
US $200-280 / Piece	**US $10000-16000** / Piece	**US $26000.0-27000.0** / Unit	**US $18000-23000** / Unit
1 Piece (Min. Order)	**1 Piece** (Min. Order)	**1 Unit** (Min. Order)	**10 Units** (Min. Order)
9 YRS Shanghai Zencar Industry Co...	3 YRS Chongqing Senku Machinery...	9 YRS Shenzhen Setec Power Co., L...	1 YR Nanjing Ark Tech Co., Ltd.
77.8%	55.6%	47.2%	
Contact Supplier	Contact Supplier	Contact Supplier	Contact Supplier

HomePlug Green PHY modes

Can be set in 3 specific modes:

- Unconfigured
- EVSE (charging station): see HGPG specific packets from PEV
- **PEV (car): can see HPGP specific packets from EVSE → interesting one**



confirmation of attenuation profile (unicast)

CM_ATTEN_CHAR.RSP

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

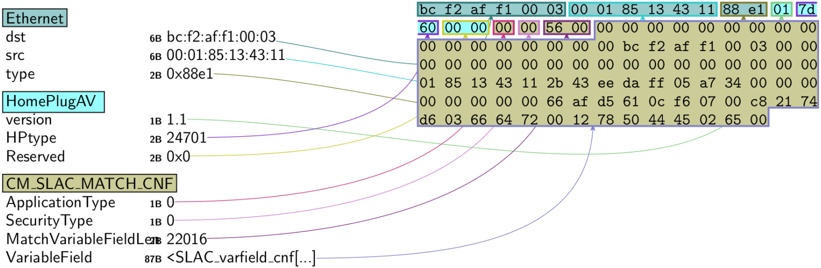
CM_SLAC_MATCH.REQ

CM_SLAC_MATCH.CNF

Providing network ID, EV/EVSE MAC (unicast)

Flaw SLAC procedure

When analysing the SLAC procedure → surprise!



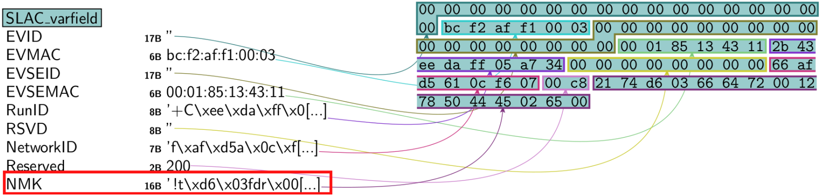
Ethernet		
dst	6B	bc:f2:af:f1:00:03
src	6B	00:01:85:13:43:11
type	2B	0x88e1
HomePlugAV		
version	1B	1.1
HPtype	2B	24701
Reserved	2B	0x0
CM_SLAC_MATCH_CNF		
ApplicationType	1B	0
SecurityType	1B	0
MatchVariableFieldLen	2B	22016
VariableField	87B	<SLAC_varfield_cnf[...]

```
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[2]

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

Into the logical PLC network (AVLN)

Conventional VCCU (car ECU):

1. Gets an IPv6 address
2. Looks for a V2G server → send a multicasted SECC query with required security level (encryption → *SecurityProtocol*)
3. Charging station answer giving corresponding host and port → SECC response
4. Car and charging station exchange data in V2G

Attacker

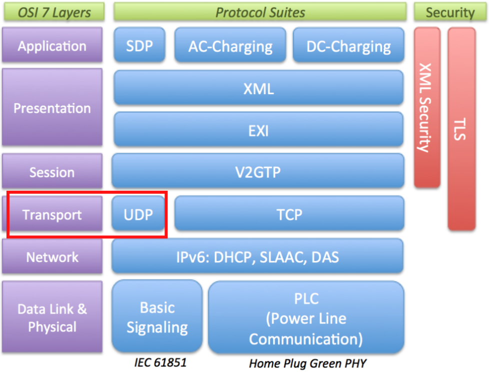
Can attack exposed services of devices and intercept communications

Intercepting communications

2 obvious ways:

- IPv6 neighbour spoofing attack
- Racing SECC procedure

SECC procedure



OSI 7 Layers	Protocol Suites	Security
Application	SDP AC-Charging DC-Charging	XML Security TLS
Presentation	XML EXI	
Session	V2GTP	
Transport	UDP TCP	
Network	IPv6: DHCP, SLAAC, DAS	
Data Link & Physical	Basic Signaling *IEC 61851* PLC (Power Line Communication) *Home Plug Green PHY*	

SECC procedure (2)

Clients (ECU) → SECC REQUEST in multicast:

```
###[ Ethernet ]###
[...]
###[ IPv6 ]###
[...]
###[ UDP ]###
        sport     = 60806
        dport     = 15118
        len       = 18
        chksum    = 0xc9c7
###[ SECC ]###
           Version    = 1
           Inversion  = 254
           SECCType   = SECC_RequestMessage
           PayloadLen= 2
###[ SECC_RequestMessage ]###
              SecurityProtocol= 16
              TransportProtocol= 0
```

SECC procedure (3)

A fake station can craft an answer with fake host address and port:

```
 [...]
###[ SECC ]###
           Version    = 1
           Inversion  = 254
           SECCType   = SECC_ResponseMessage
           PayloadLen= 20
###[ SECC_ResponseMessage ]###
              TargetAddress= fe80::201:85ff:fe13:4311
              TargetPort= 56330
              SecurityProtocol= 16
              TransportProtocol= 0
```

More stable than IPv6 neighbour spoofing attack

SECC procedure (3)

A fake station can craft an answer with fake host address and port:

```
 [...]
###[ SECC ]###
           Version    = 1
           Inversion  = 254
           SECCType   = SECC_ResponseMessage
           PayloadLen= 20
###[ SECC_ResponseMessage ]###
              TargetAddress= fe80::201:85ff:fe13:4311
              TargetPort= 56330
              SecurityProtocol= 16
              TransportProtocol= 0
```

More stable than IPv6 neighbour spoofing attack

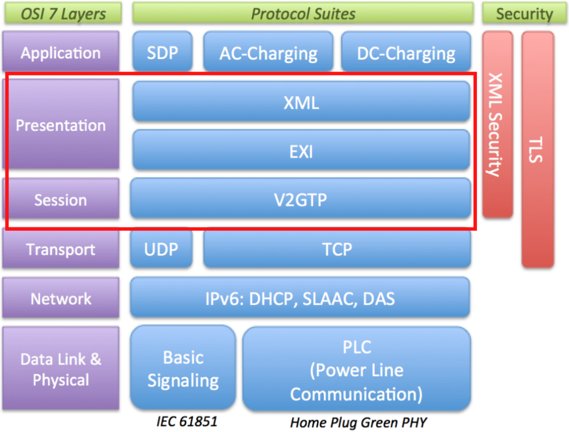
Need to be fast

Be fast to impersonate legit SECC servers Otherwise → IPv6 neighbour spoofing

SECC: other vectors

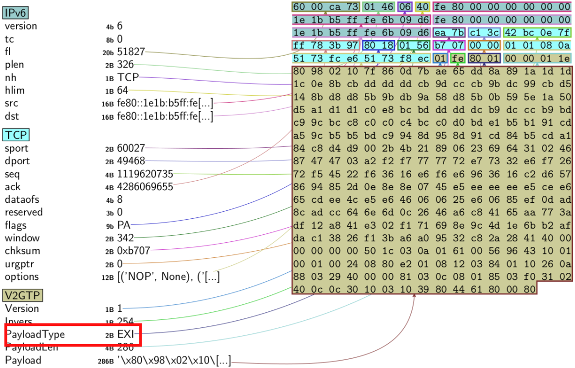
- *SecurityProtocol* is “16” by default → for clear-text and “0” when TLS is enabled
- This field can be tricked to force the client to talk in clear-text by crafting a *SECC_ResponseMessage* with a *SecurityProtocol*=16
- Interesting to test in different implementations

V2G interception



V2GTP packet

After decoding the V2GTP header:



```
IPv6
version                 4b 6
tc                      8b 0
fl                     20b 51827
plen                    2B 326
nh                      1B TCP
hlim                    1B 64
src                    16B fe80::1e1b:b5ff:fe[...]
dst                    16B fe80::1e1b:b5ff:fe[...]
TCP
sport                   2B 60027
dport                   2B 49468
seq                     4B 1119620735
ack                     4B 4286069655
dataofs                 4b 8
reserved                3b 0
flags                   9b PA
window                  2B 342
chksum                  2B 0xb707
urgptr                  2B 0
options                12B [('NOP', None), ('[...]
V2GTP
Version                 1B 1
Invers                  1B 254
PayloadType             2B EXI
PayloadLen              4B 286
Payload               286B '\x80\x98\x02\x10\[...]
```

```
60 00 ca 73 01 46 06 40 fe 80 00 00 00 00 00 00
1e 1b b5 ff fe 6b 09 d6 fe 80 00 00 00 00 00 00
1e 1b b5 ff fe 6b 09 d6 ea 7b c1 3c 42 bc 0e 7f
ff 78 3b 97 80 18 01 56 b7 07 00 00 01 01 08 0a
51 73 fc e6 51 73 f8 ec 01 fe 80 01 00 00 01 1e
80 98 02 10 7f 86 0d 7b ae 65 dd 8a 89 1a 1d 1d
1c 0e 8b cb dd dd dd cb 9d cc cb 9b dc 99 cb d5
14 8b d8 d8 5b 9b db 9a 58 d8 5b 0b 59 5e 1a 50
d5 a1 d1 d1 c0 e8 bc bd dd dd dc b9 dc cc b9 bd
c9 9c bc c8 c0 c0 c4 bc c0 d0 bd e1 b5 b1 91 cd
a5 9c b5 b5 bd c9 94 8d 95 8d 91 cd 84 b5 cd a1
84 c8 d4 d9 00 2b 4b 21 89 06 23 69 64 31 02 46
87 47 47 03 a2 f2 f7 77 77 72 e7 73 32 e6 f7 26
72 f5 45 22 f6 36 16 e6 f6 e6 96 36 16 c2 d6 57
86 94 85 2d 0e 8e 8e 07 45 e5 ee ee ee e5 ce e6
65 cd ee 4c e5 e6 46 06 06 25 e6 06 85 ef 0d ad
8c ad cc 64 6e 6d 0c 26 46 a6 c8 41 65 aa 77 3a
df 12 a8 41 e3 02 f1 71 69 8e 9c 4d 1e 6b b2 af
da c1 38 26 f1 3b a6 a0 95 32 c8 2a 28 41 40 00
00 00 00 00 50 1c 03 0a 01 61 00 56 96 43 10 01
00 01 00 24 08 80 e2 01 08 12 03 84 01 10 26 0a
88 03 29 40 00 00 81 03 0c 08 01 85 03 f0 31 02
40 0c 0c 30 10 03 10 39 80 44 61 80 00 80
```

There is still unknown data in the V2GTP payload

The EXI format

- Refering IEC/ISO 15118 → data in V2G is EXI compressed
- To compress as much data → use of specific grammar → XSD schemas specific to V2G
- EXI: Efficient XML Interchange
- Aims to encode:
  - **XML (and formats using XML syntax, e.g., SVG, RSS, MathML, GraphML, ...)**
  - HTML
  - JSON
  - CSS
  - JavaScript

Contexts

- Each context as a XSD file, as probided in RISE V2G:
  - V2G_CI_AppProtocol.xsd
  - V2G_CI_MsgDef.xsd
  - V2G_CI_MsgHeader.xsd
  - V2G_CI_MsgBody.xsd
  - V2G_CI_MsgDataTypes.xsd
- EXI data does not provide any context

To decode EXI → RISE V2G uses state machines to select corresponding grammar → complicated in our case

Contexts

- Each context as a XSD file, as probided in RISE V2G:
    - V2G_CI_AppProtocol.xsd
    - V2G_CI_MsgDef.xsd
    - V2G_CI_MsgHeader.xsd
    - V2G_CI_MsgBody.xsd
    - V2G_CI_MsgDataTypes.xsd
- EXI data does not provide any context

To decode EXI → RISE V2G uses state machines to select corresponding grammar → complicated in our case

Circumvent: DFA

Exactly! Let's try DFA!

DFA method != Differential Fault Analysis

D for Dirty, **F** for fuzzy and **A** for Approach:

```
public static String fuzzyExiDecoder(String strinput, decodeMode dmode)
{
    String grammar = null;
    String result = null;

    grammar = GlobalValues.SCHEMA_PATH_MSG_BODY.toString();
    try {
        result = Exi2Xml(strinput, dmode, grammar);
    } catch (EXIException e1) {
        try {
            grammar = GlobalValues.SCHEMA_PATH_APP_PROTOCOL.toString();
            result = Exi2Xml(strinput, dmode, grammar);
        } catch (EXIException e2) {
            grammar = GlobalValues.SCHEMA_PATH_XMLDSIG.toString();
            try {
                result = Exi2Xml(strinput, dmode, grammar);
            } catch (EXIException e3) {
                // do nothing
            } catch (Exception b3) {
                b3.printStackTrace();
            }
[...]
```

in a failing order of course :)!

V2Gdecoder: decode and encode

Decode EXI:

```
$ java —jar V2Gdecoder.jar —e —s 809802107f860d7bae....
<?xml version=”1.0” encoding=”UTF—8”?><ns7:V2G_Message ...
```

Encode XML:

```
$ java —jar V2Gdecoder.jar —x —s ’<?xml version=”1.0”
  encoding=”UTF—8”?><ns4:supportedAppProtocolReq

8000DBAB9371D3234B71D1B981899189D191818991D26B...
```

Available: https://github.com/FlUxIuS/V2Gdecoder

Issues with old protocols

- We are able to decode first V2G packet from the car
- Contains supported application protocols including *urn:iso:15118:2:2010* → not supported in RISE V2G OSS stack → remove the XML node during a MITM

```
<?xml version=”1.0” encoding=”UTF−8”?>
<ns4:supportedAppProtocolReq xmlns:ns4=”urn:iso:15118:2:2010:AppProtocol” ...>
    <AppProtocol>
        <ProtocolNamespace>urn:din:70121:2012:MsgDef</ProtocolNamespace>
        <VersionNumberMajor>2</VersionNumberMajor>
        <VersionNumberMinor>0</VersionNumberMinor>
        <SchemaID>0</SchemaID>
        <Priority>1</Priority>
    </AppProtocol>
    <AppProtocol>
        <ProtocolNamespace>urn:iso:15118:2:2013:MsgDef</ProtocolNamespace>
        <VersionNumberMajor>2</VersionNumberMajor><
        VersionNumberMinor>0</VersionNumberMinor>
        <SchemaID>1</SchemaID>
        <Priority>2</Priority>
    </AppProtocol>
</ns4:supportedAppProtocolReq>
```

Support for DIN 70121

- We have adapted schemas
- Based on C++ implementation in OpenV2G
- Available: https://github.com/FlUxIuS/V2Gdecoder/tree/master/schemas_din

1 V2G communication

2 HomePlug Green PHY

3 Preliminaries

4 Intruding a V2G network

5 **V2G Injector**

6 Conclusion

Rise of the HPGPhoenix

Available: https://github.com/FlUxIuS/V2GInjector

HPGP keys

Automatically done:

```
~>>> n=Network()
~>>> n.sniff(iface="eth0")
 [...]
 [New HPGP network spotted!]
— EVSEID: '\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
— NetID: '\xae\x20\x00\xff\x82\x02\x00'
— NMK: '\x43F\xc8\xaeT\xbf\xefs\x01\x84\x94\xf8\xc3\x17'
— EVID: '\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff'
— RunID: '\xef\x34C\xf5E\xe0\xa6\x01'
```

Generate V2G packets

Use the dedicated Scapy layers:

```
~>>> ether = Ether()
~>>> ip = IPv6(dst=”fe80::3e2a:b4ff:3e5f:1a4”)
~>>> tcp = TCP(sport=6666, dport=54054, flags=24)
~>>> v2g=V2GTP()
~>>> packet = ether/ip/tcp/v2g
~>>> packet
<Ether  type=0x86dd |<IPv6  nh=TCP dst=fe80::3e2a:b4ff:3e5f:1a4 |
<TCP  sport=6666 dport=54054 flags=PA |<V2GTP  |>>>>
```

XML → compressed in EXI → included in the V2GTP payload:

```
~>>> xml = ’<?xml version=”1.0” encoding=”UTF−8”?><ns7:V2G_Message ....
</ns7:V2G_Message>’
~>>> encoded_xml=encodeEXI(xml)
~>>> encoded_xml
u’809802000000000000000011D018706ED5AC275800’
~>>> packet.Payload=encoded_xml
~>>> packet
<Ether  type=0x86dd |<IPv6  nh=TCP dst=fe80::3e2a:b4ff:3e5f:1a4 |
<TCP  sport=6666 dport=54054 flags=PA |
<V2GTP  Payload=’809802000000000000000011D018706ED5AC275800’ |>>>>
```

Then send it using *sendp()* function.

1 V2G communication

2 HomePlug Green PHY

3 Preliminaries

4 Intruding a V2G network

5 V2G Injector

6 Conclusion

Conclusion

- V2G opens new interesting surfaces
- We have developed a tool to play with it → V2G Injector
- The project is free to use and also to contribute ;)
- ECU are less featured than charging stations
- Intruding charging station could lead to interesting pivots
- Further work:
  - Add a complete simulator
  - more EXI grammars
  - Add attacks and fuzzing wrappers for SECC, V2GTP, EXI and HomePlug GP

Other areas of research

- EXI format fuzzing [3]:
  - Fuzzing from XML → difficult as XML are parsed and processed against XSD
  - Better chances with the compressed data against C/C++ implementations → AFL for the road
  - Real ECUs' firmware use proprietary a proprietary EXI decoders
  - But public EXI libraries could be interesting to attack charging stations

[3] Suggested also by @agarri_fr :)

Few words on public charging stations

- Runs a complex OS (Linux generally)
- Some available services:
  - V2G webservice
  - SSH
  - Web console/management/log interface
  - Sometimes: Telnet and more...
- Connected to an operator
- If attacked → used as pivot

ANY QUESTIONS?

THANK YOU FOR YOUR ATTENTION,

SYNACKTIV
DIGITAL SECURITY