PLAXIDITYX

GO EVERYWHERE

Breaking the Circuit

ESCAR USA 2025

Shaked Delarea | Omer Ziv

Threats in the EV Charging Ecosystem

PLAXIDITYX
GO EVERYWHERE

Embedded Security Research Team

PLAXIDITYX
GO EVERYWHERE

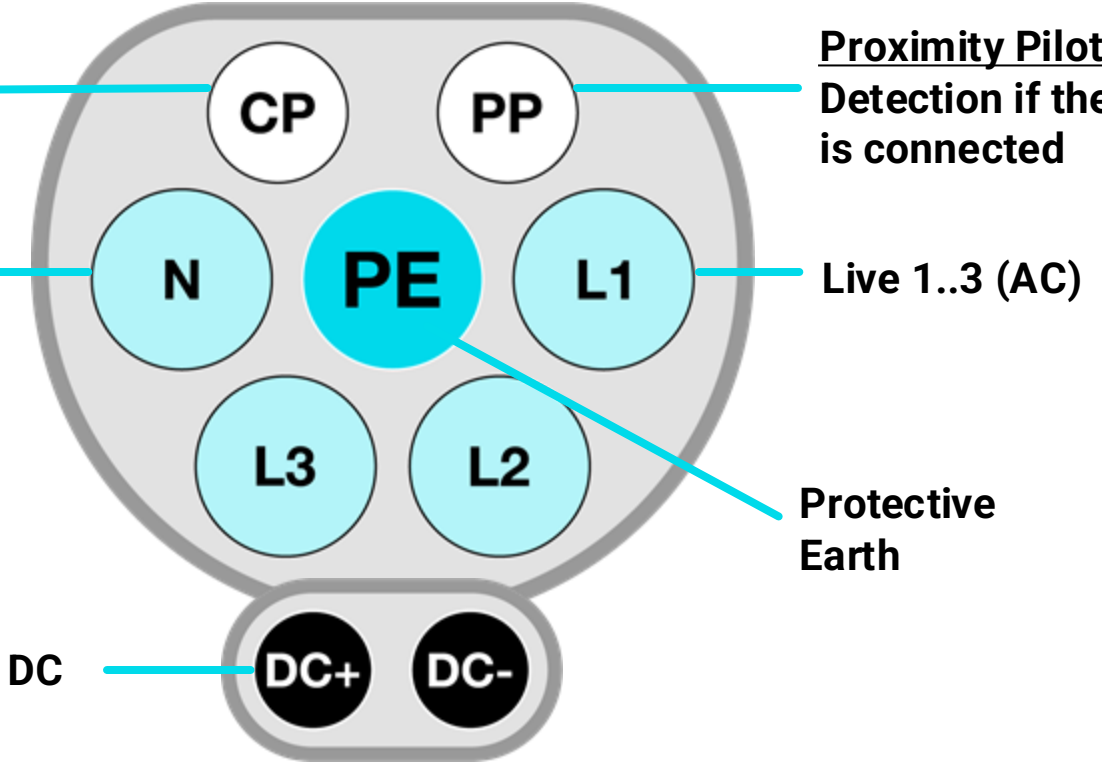
Home Charging Stations

PLAXIDITYX
GO EVERYWHERE

What happens when the charging station (EVSE) is plugged to an EV?

PLAXIDITYX
GO EVERYWHERE

What happens when the vehicle is plugged to an EV?



PLAXIDITYX
GO EVERYWHERE

Control Pilot (CP) & Proximity Pilot (PP)

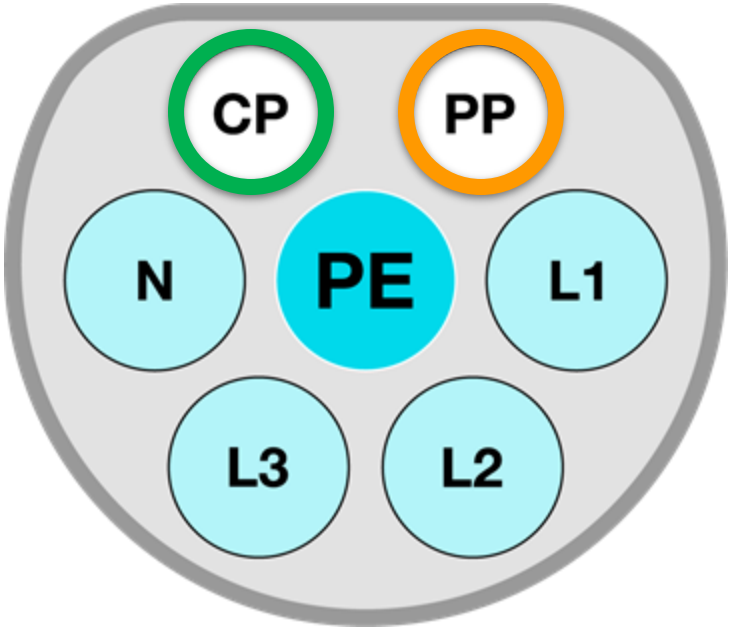
Both lines **are low-voltage** lines used for communication

As soon as the EV detects that the PP is connected, it is ready to begin a charging session

Communication between the EV and the EVSE is performed using the CP line.

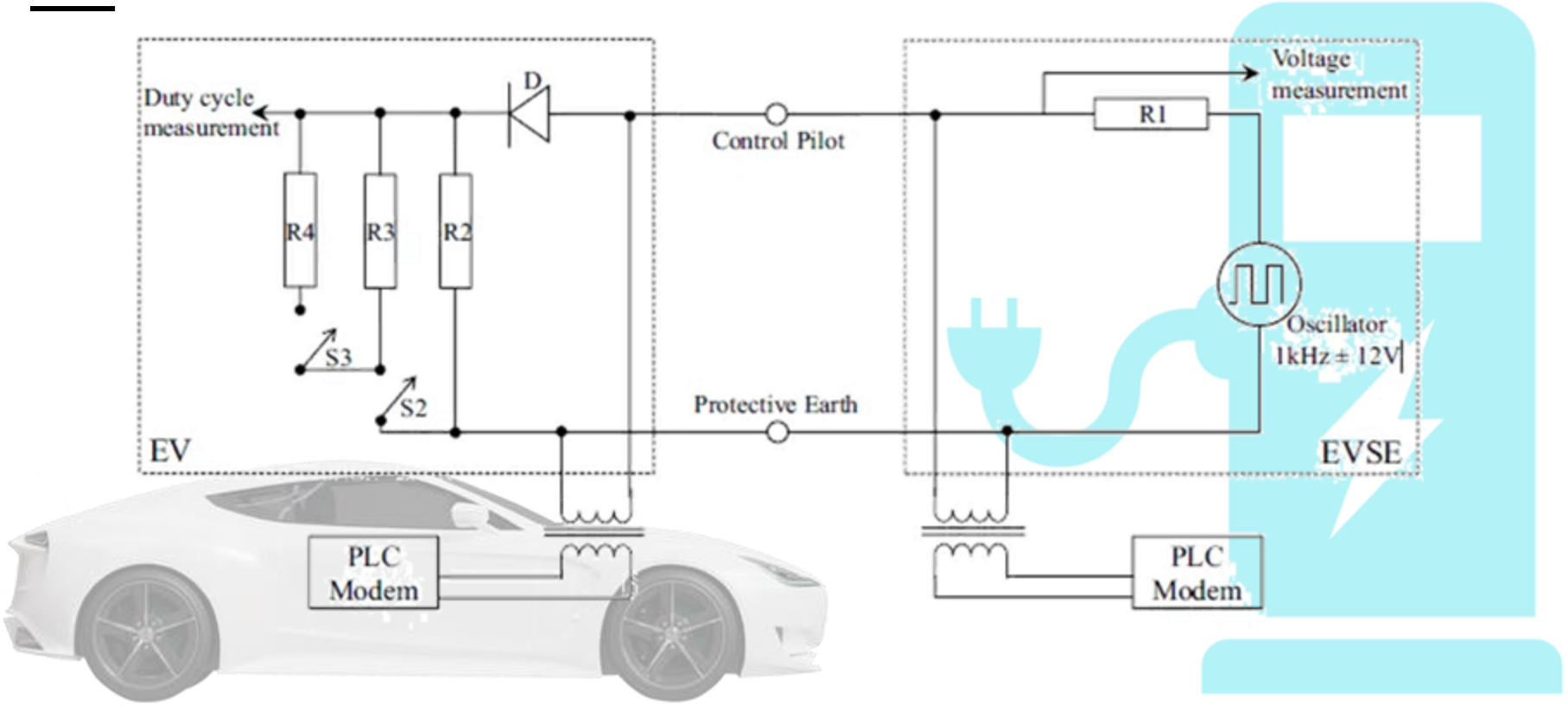
The voltage on the CP line indicates the current state:

- **9V - Connected**
- **6V - Request for charge / Charge**
- **3V - Charge with ventilation**
- **0V - EV Error**
- **-12V - EVSE error**



PLAXIDITYX
GO EVERYWHERE

Circuitry

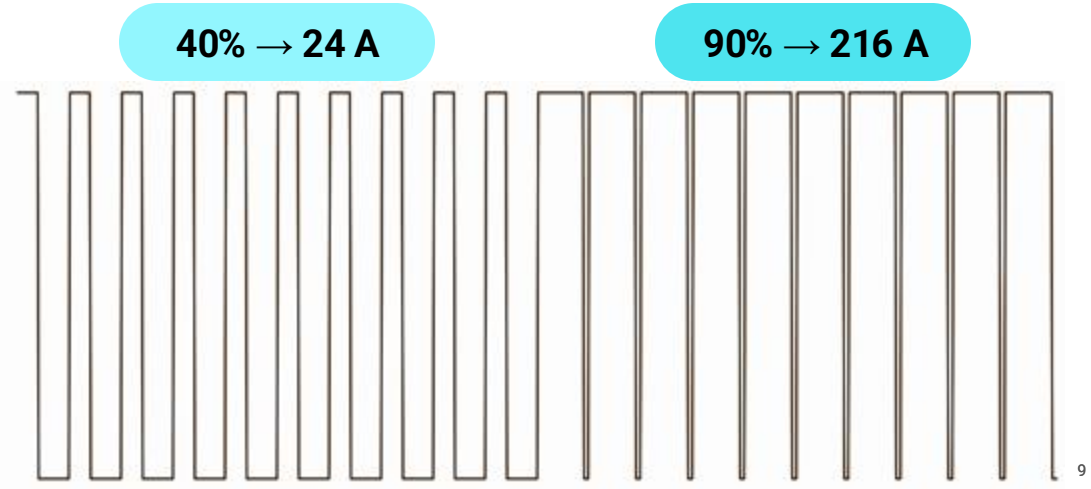


PLAXIDITYX
GO EVERYWHERE

Control Pilot during Charging

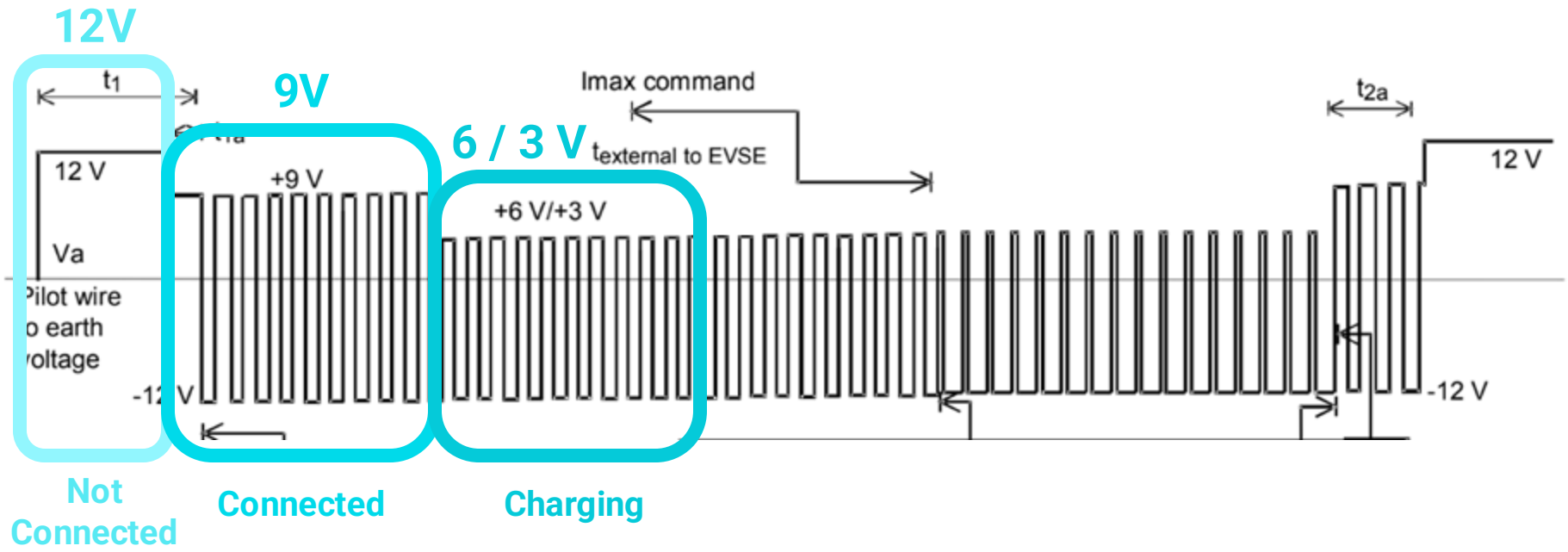
Duty Cycle	Available current
5%	High Level Communication (HLC)
10% to 85%	duty_cycle * 0.6 [A]
85% to 96%	duty_cycle * 2.5 [A]

PWM – Pulse-width modulation
Duty Cycle - % of “up” time



PLAXIDITYX
GO EVERYWHERE

TL;DR



Source: IEC-61851-1

Commercial Charging

PLAXIDITYX
GO EVERYWHERE

Open Charge Point Protocol (OCPP)

- OCPP is a standard **open protocol** for communication between an EVSE and a Central System to manage any charging technique
- Central system may manage multiple EVSEs
  - Handle billing
  - Load management
  - Monitoring
  - Firmware upgrade

PLAXIDITYX
GO EVERYWHERE

What threats could this be exposed to?

These OCPP servers might be reside in the same physical location of the grid they manage

Or… they can reside in a remote location, accessible to the internet

This server might offer different services on unrelated to OCPP such as HTTP/FTP

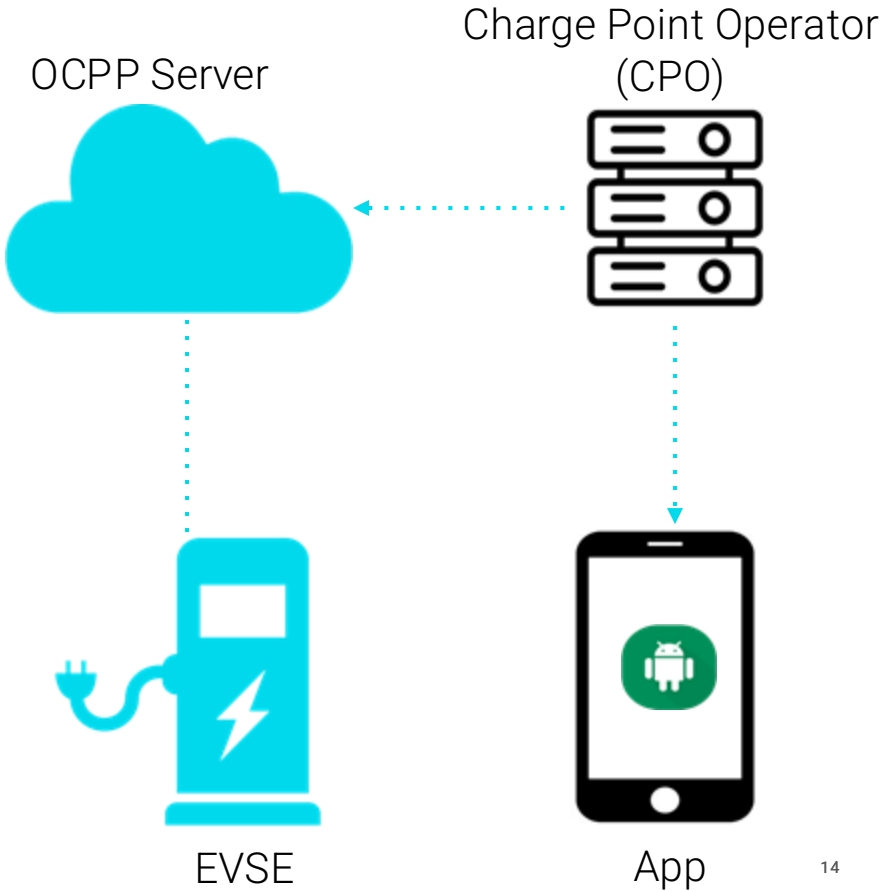
Let’s tell you a story about that

PLAXIDITYX
GO EVERYWHERE

Reversing the Android App

Main steps:

1. Get the App from the store
2. Reverse Engineer / Sniff internet traffic
3. Find the backend server
4. Hope it will somehow lead to interesting results



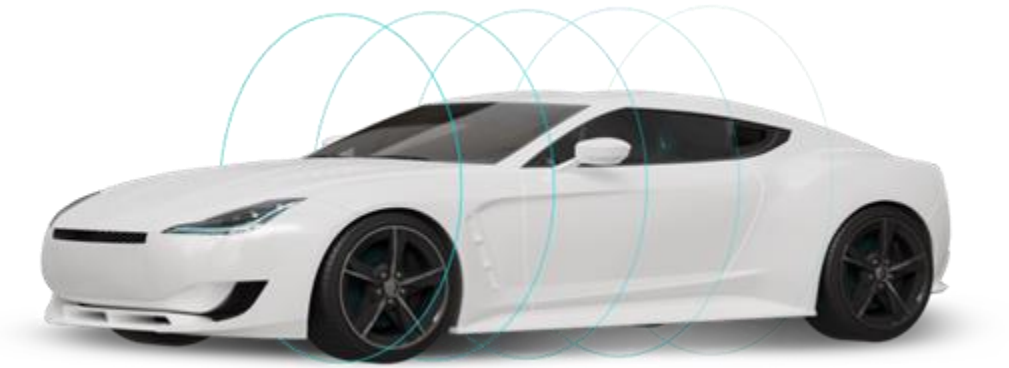
PLAXIDITYX
GO EVERYWHERE

CPO Server

Found the CPO server from the App

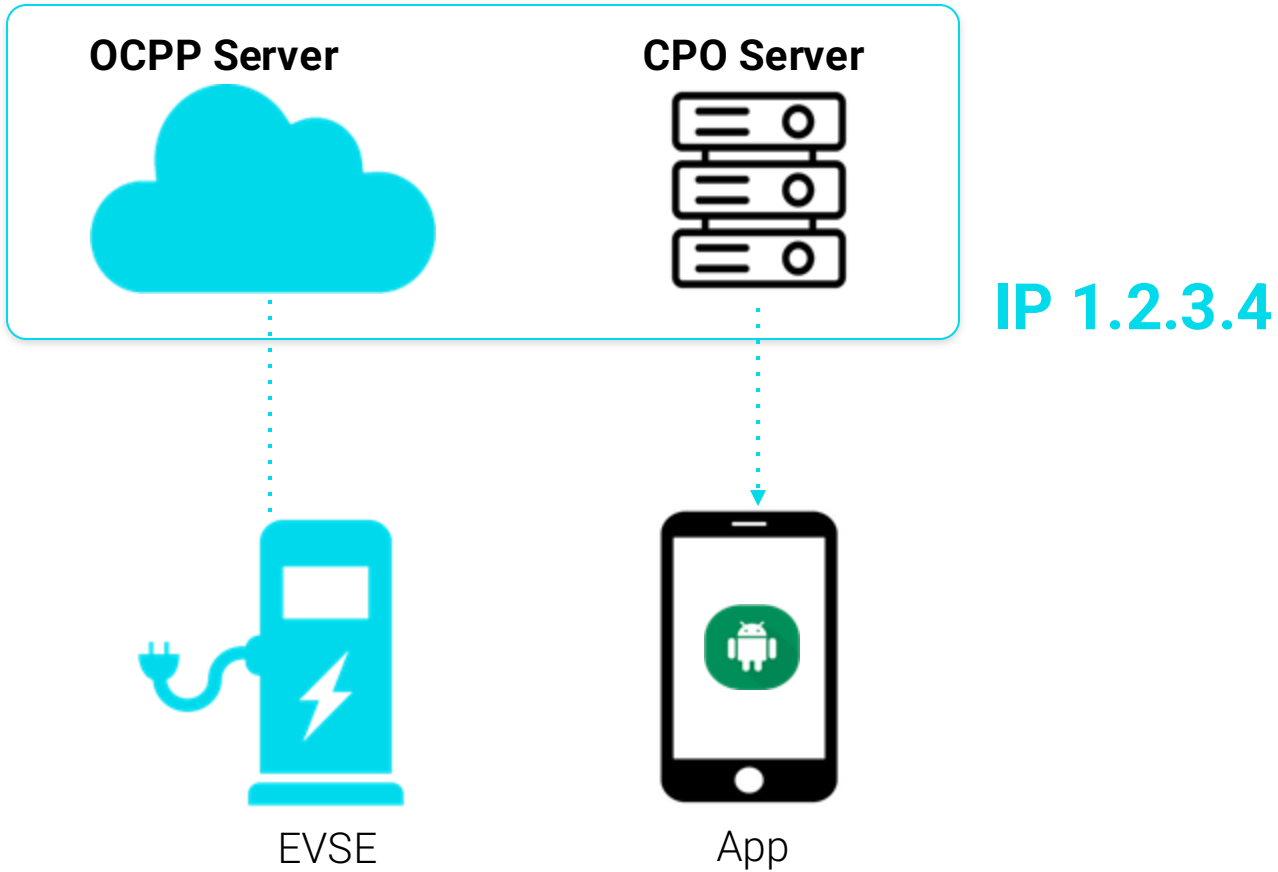
nmap’d it

We found something interesting



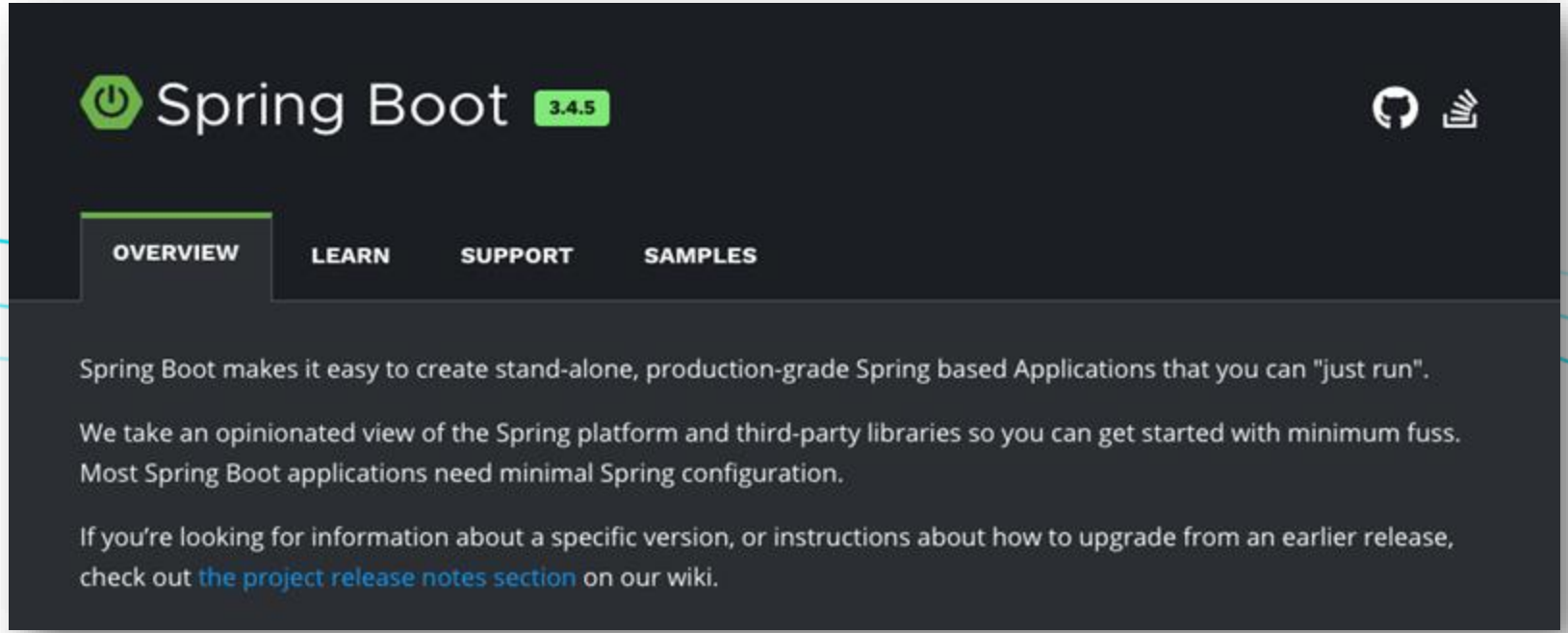
PLAXIDITYX
GO EVERYWHERE

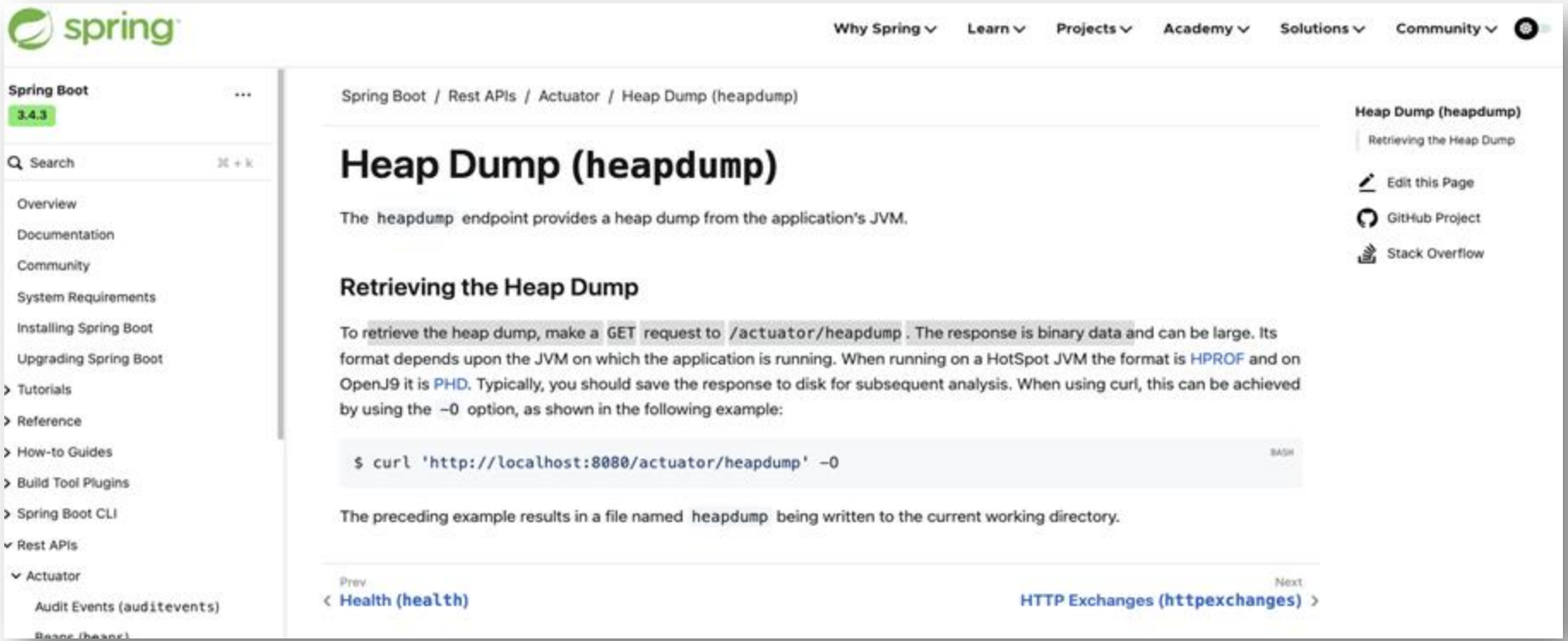
Surprise #1 - CPO == OCPP



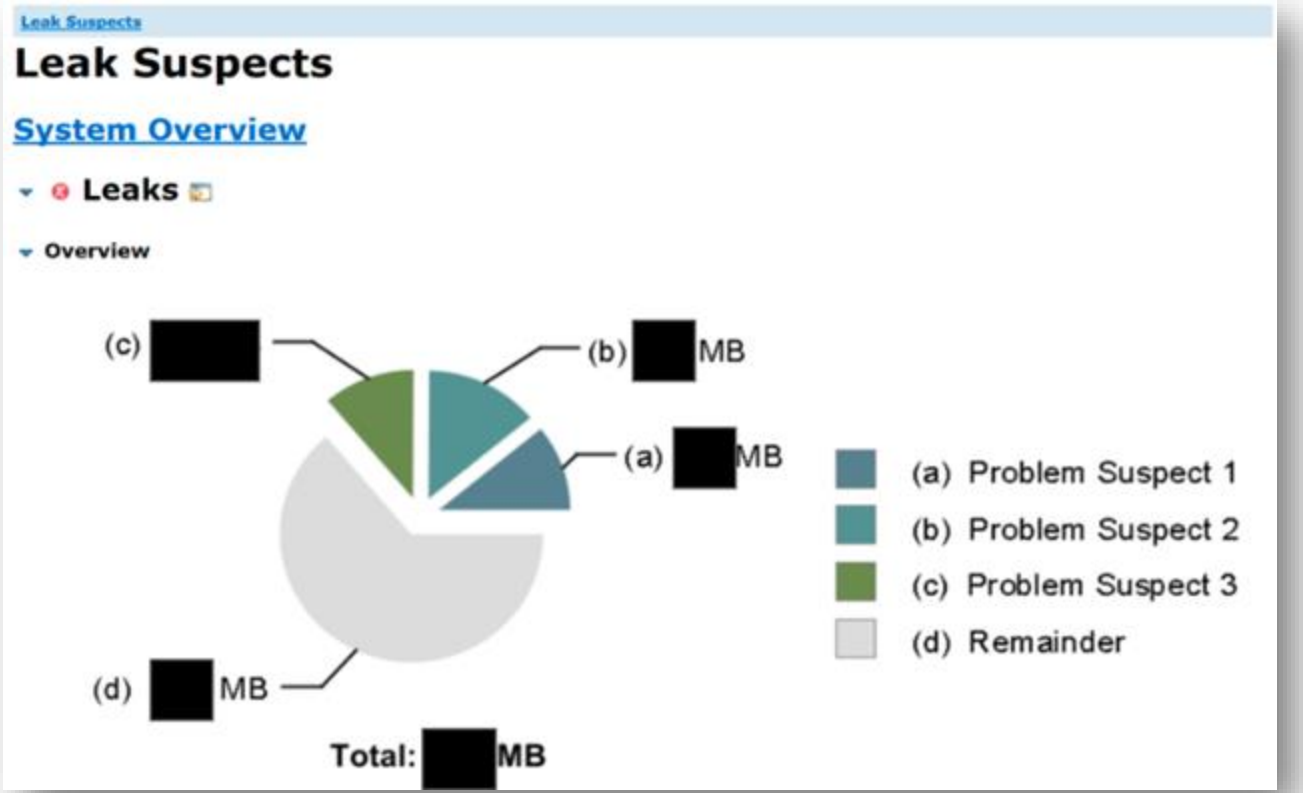
PLAXIDITYX
GO EVERYWHERE

Surprise #2 - Spring Boot



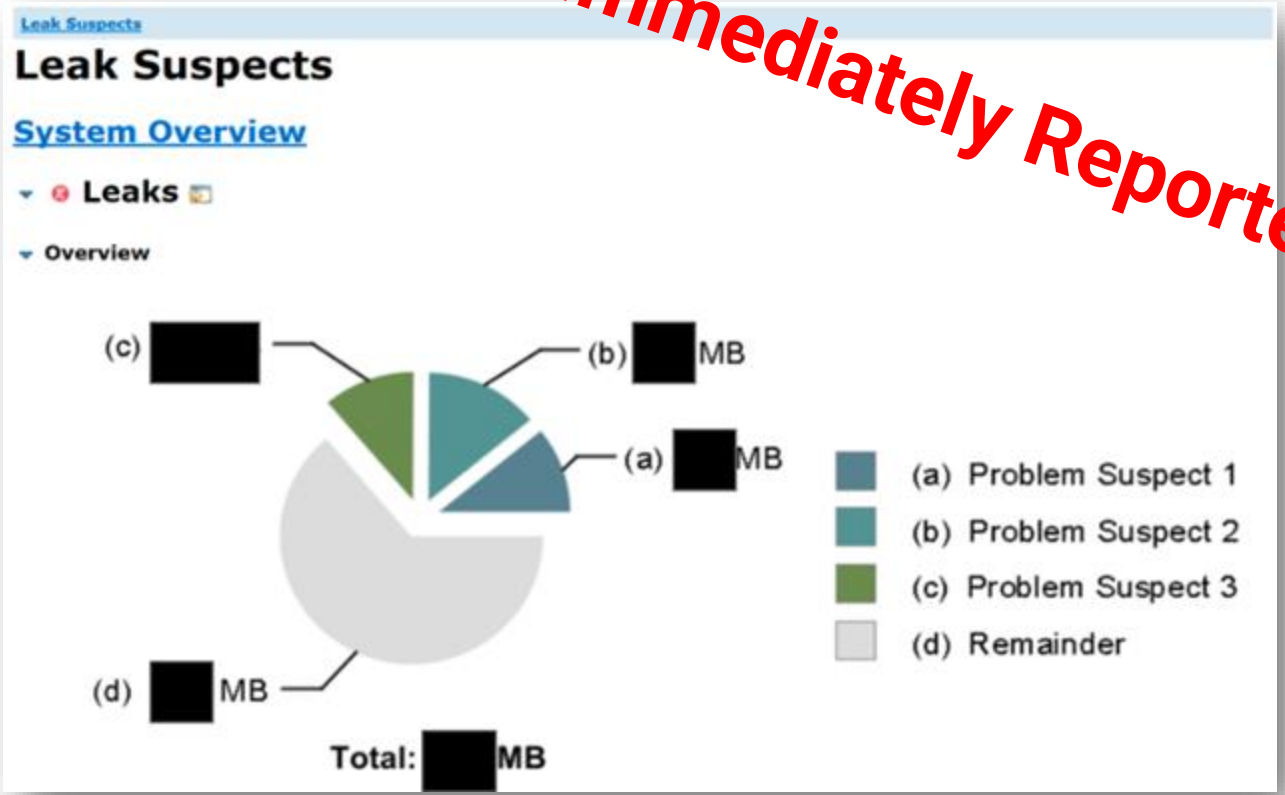

spring
Why Spring
Learn
Projects
Academy
Solutions
Community
Spring Boot
3.4.3
Search
Overview
Documentation
Community
System Requirements
Installing Spring Boot
Upgrading Spring Boot
Tutorials
Reference
How-to Guides
Build Tool Plugins
Spring Boot CLI
Rest APIs
Actuator
Audit Events (auditevents)
Spring Boot / Rest APIs / Actuator / Heap Dump (heapdump)
Heap Dump (heapdump)
The heapdump endpoint provides a heap dump from the application's JVM.
Retrieving the Heap Dump
To retrieve the heap dump, make a GET request to /actuator/heapdump. The response is binary data and can be large. Its format depends upon the JVM on which the application is running. When running on a HotSpot JVM the format is HPROF and on OpenJ9 it is PHD. Typically, you should save the response to disk for subsequent analysis. When using curl, this can be achieved by using the -O option, as shown in the following example:
$ curl 'http://localhost:8080/actuator/heapdump' -O
BASH
The preceding example results in a file named heapdump being written to the current working directory.
Prev
Health (health)
Next
HTTP Exchanges (httpexchanges)
Heap Dump (heapdump)
Retrieving the Heap Dump
Edit this Page
GitHub Project
Stack Overflow

Heap dump



PLAXIDITYX
GO EVERYWHERE

Heap dump

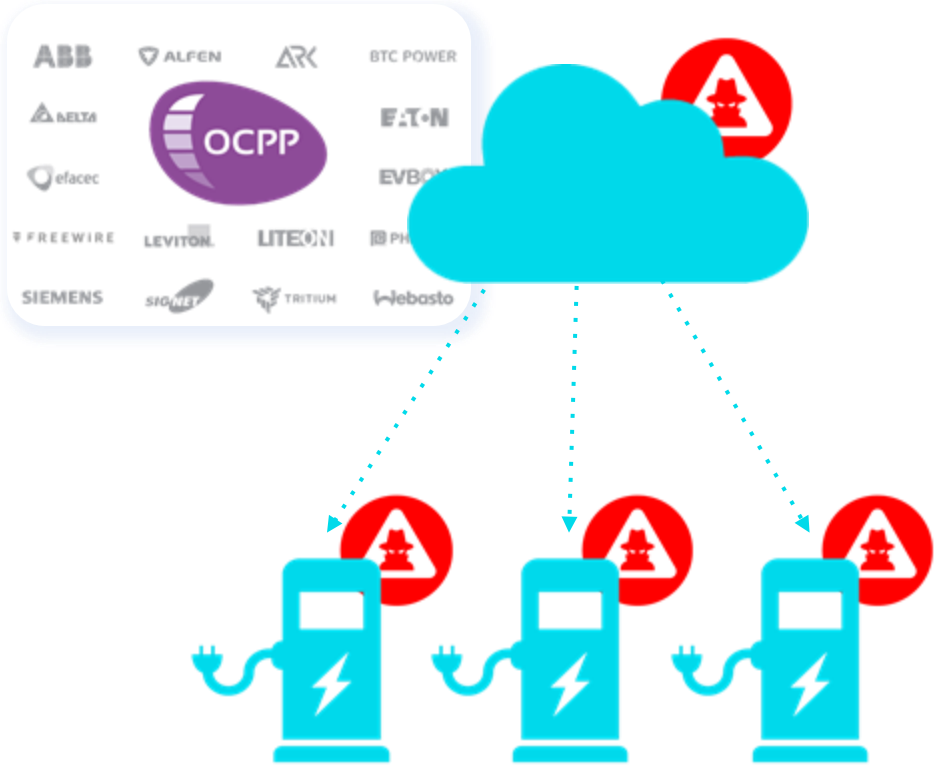


Immediately Reported

Impact

The Charge Point Operator backend (CPO) controls the Charging Stations it manages:

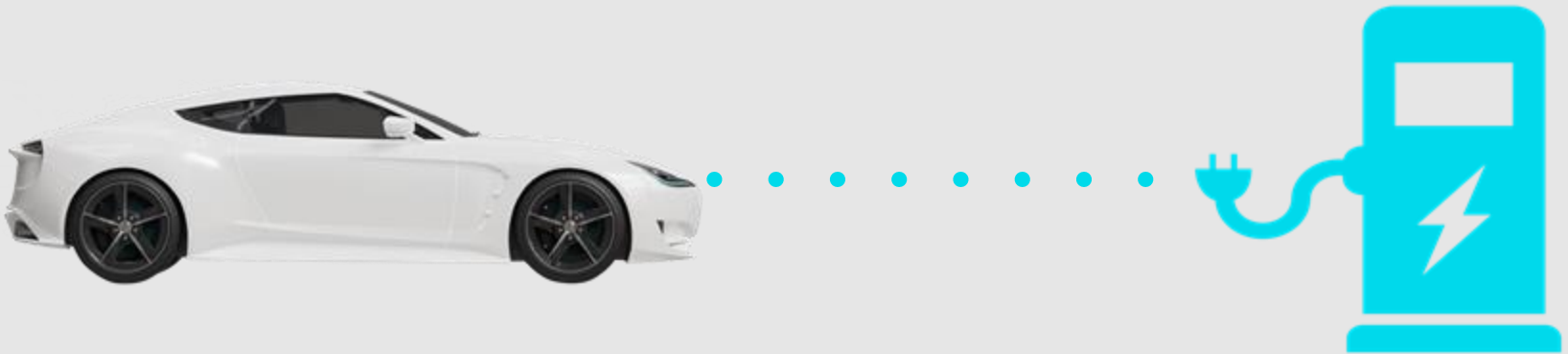
1. Firmware upgrade
2. Dictate power supply
3. Potentially allow free charging! $$



Recap – CPO ⟷ EVSE communication

PLAXIDITY X
GO EVERYWHERE

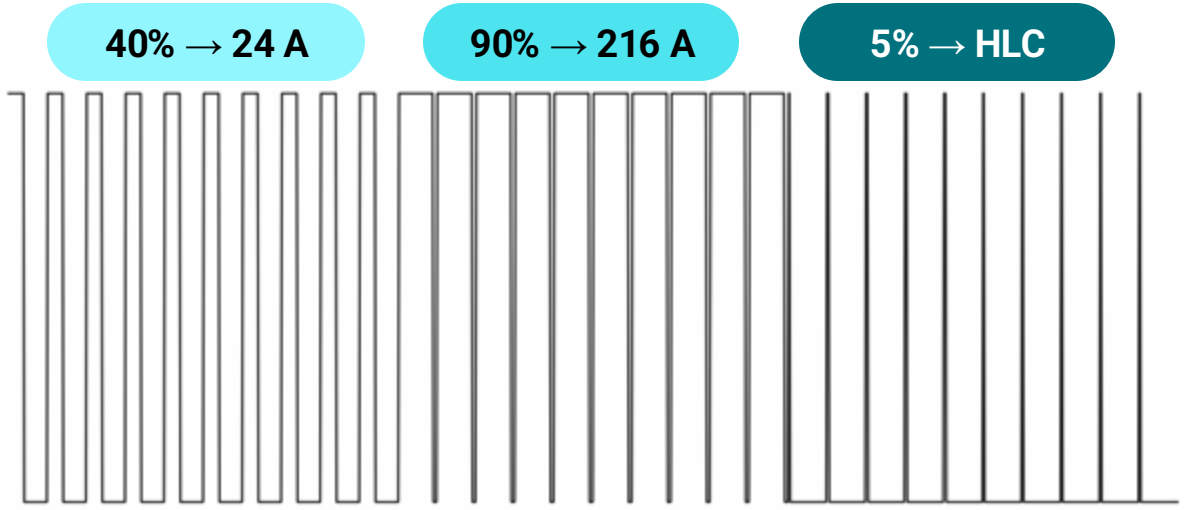
What happens between the vehicle and the charging station?



Main difference – more information requires

High Level Communication

Duty Cycle	Available current
5%	High Level Communication (HLC)
10% to 85%	duty_cycle * 0.6 [A]
85% to 96%	duty_cycle * 2.5 [A]



PLAXIDITYX
GO EVERYWHERE

Phase 1 - MAC Discovery

MAC Discovery

Powerline communication (PLC)

SLAC

PLC

HP-5103

Features Specifications Download Where to buy

AV500 Nano PowerLine Adapter

HP-5103

- Easy plug-n-play setup and 128 bit AES security
- Maximum Powerline speed up to 500Mbps
- Backward compatible with 200Mbps Powerline adapters
- Features energy saving mode to reduce power consumption
- Utilizes existing electrical wires to transmit network data
- Powerline transmission range up to 300 meters

PLAXIDITYX
GO EVERYWHERE

Back to the Control Pilot

SLAC

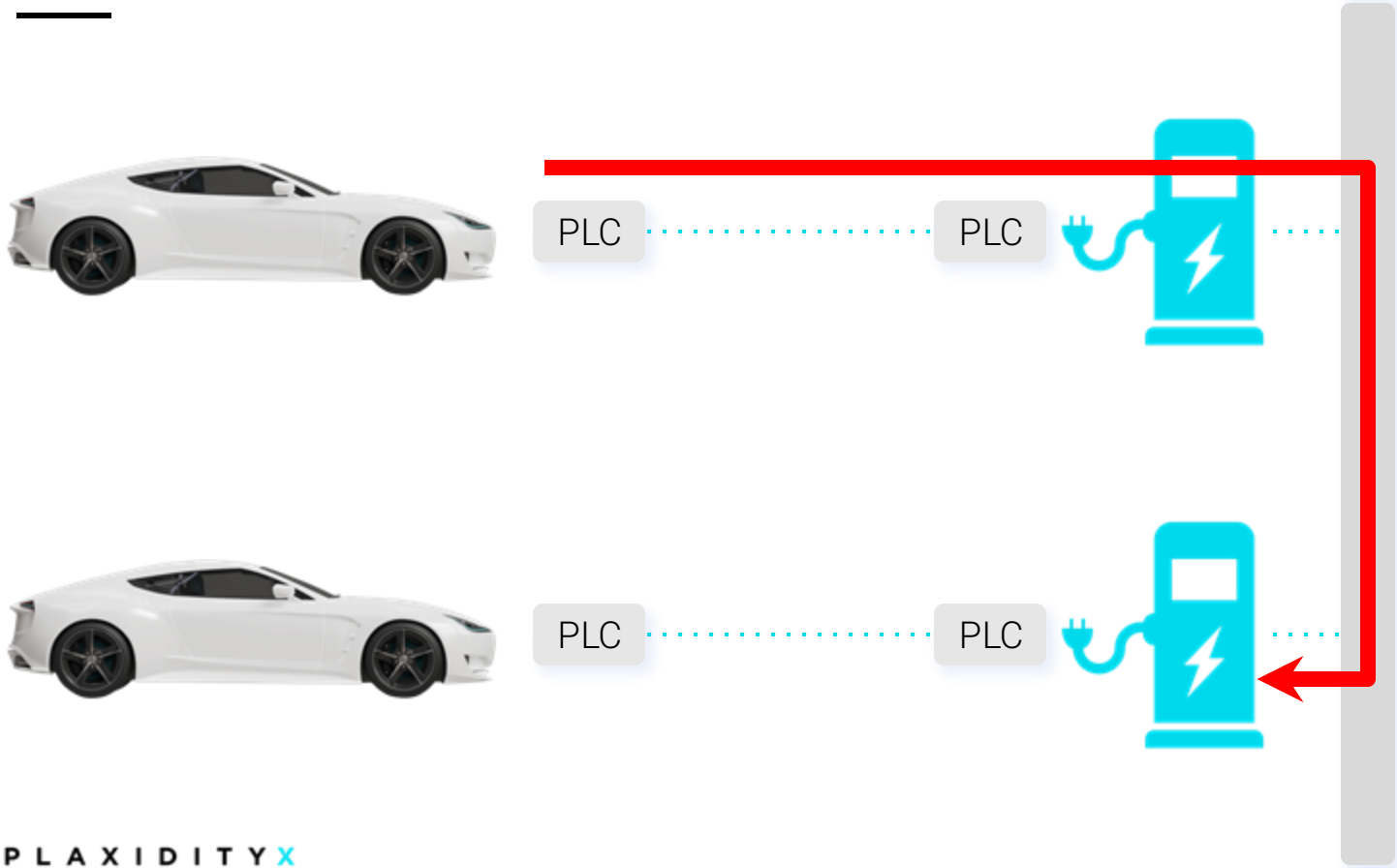
PLC

PLAXIDITYX
GO EVERYWHERE

Signal Level Attenuation Characterization (SLAC)

SLAC

PLC

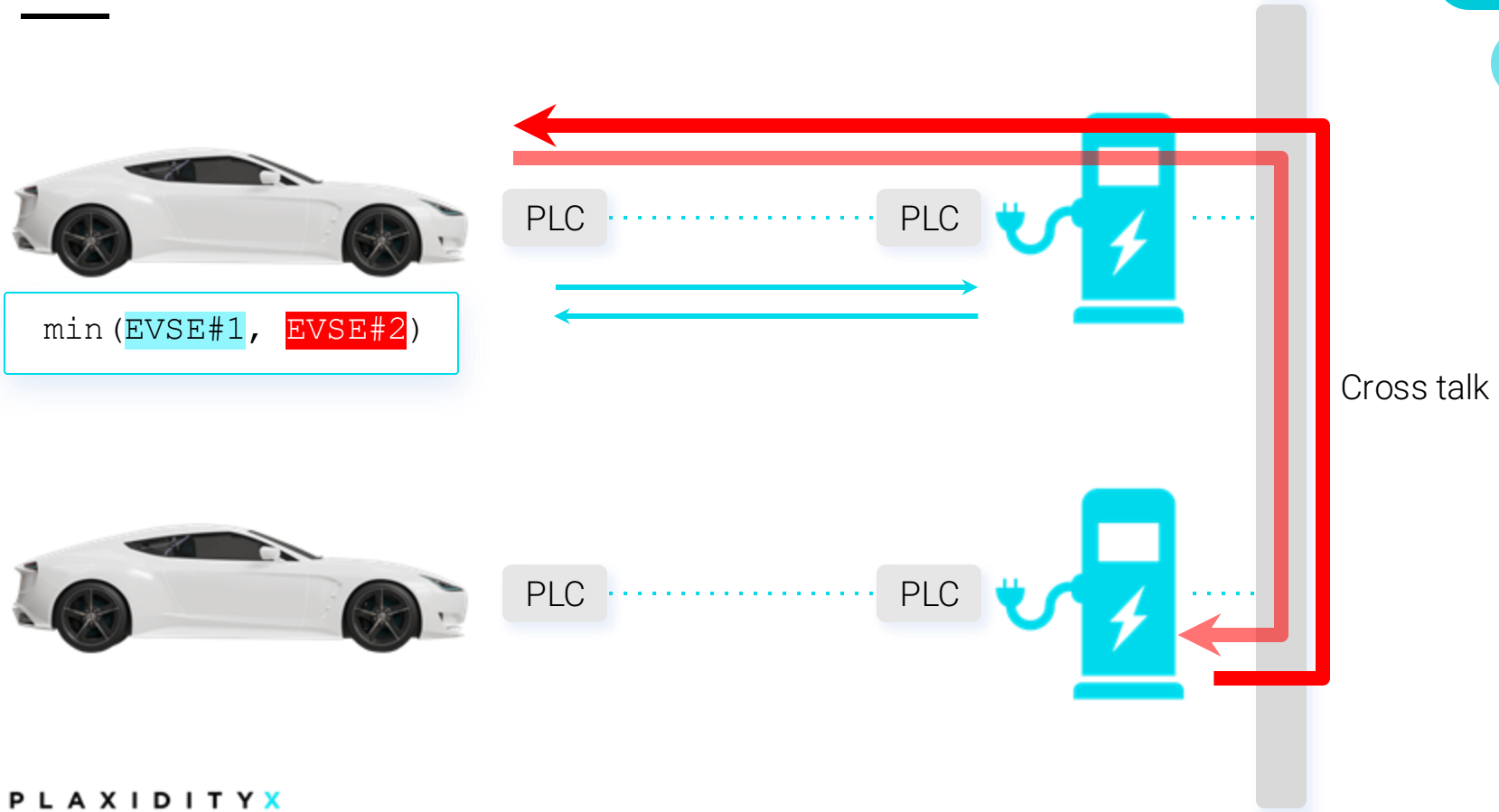


PLAXIDITYX
GO EVERYWHERE

Signal Level Attenuation Characterization (SLAC)

SLAC

PLC

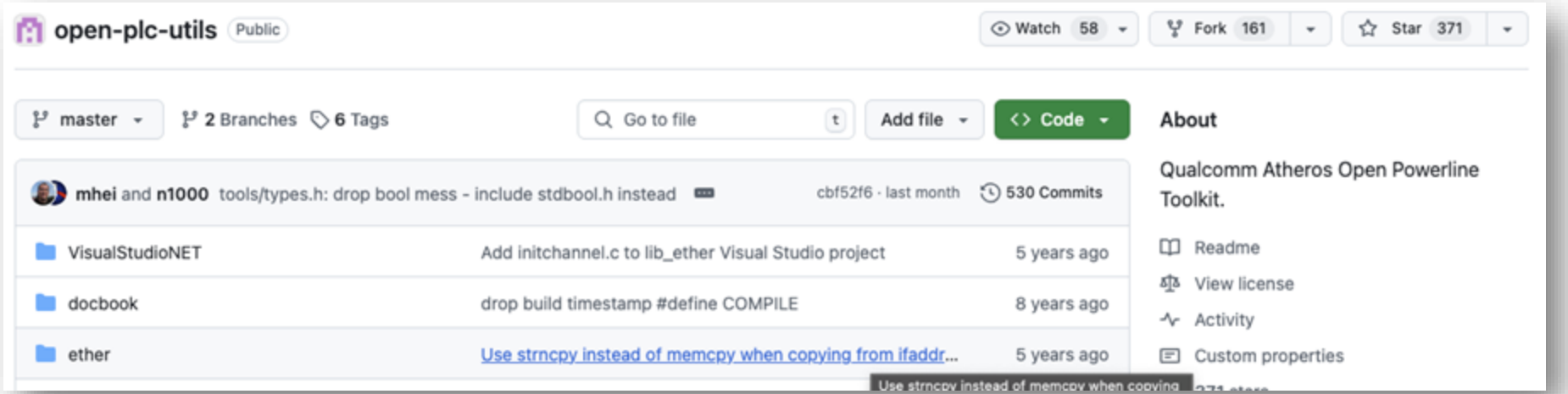


PLAXIDITY X
GO EVERYWHERE

Open PLC Utils

Qualcomm offers PLC testing tools for their customers

Their tools are based on an open-sourced driver software which still exists in GitHub and used in various other PLC applications not directly linked to Qualcomm

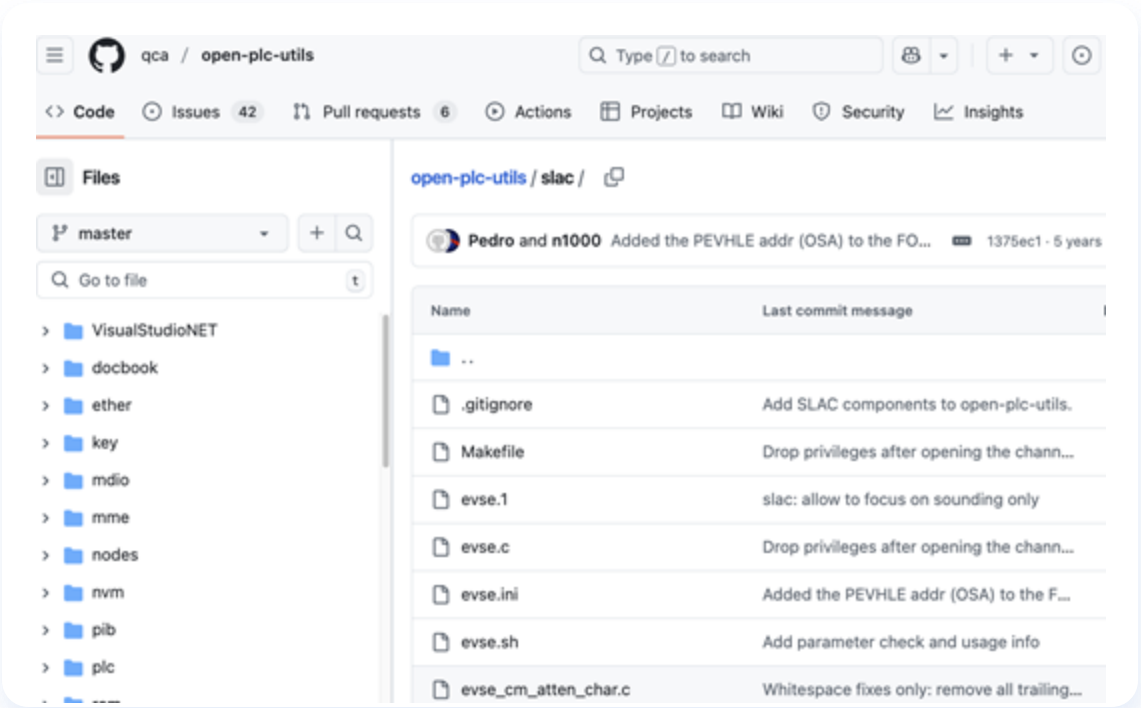


Open PLC Utils

SLAC

PLC

- The open source package also offers an implementation for the SLAC protocol



PLAXIDITYX
GO EVERYWHERE

Open PLC Utils

SLAC

PLC

- Which contained a trivial buffer overflow in the SLAC protocol

```
137         else if (LE16TOH (homeplug->homeplug.MMTYPE) == (CM_ATTEN_PROFILE | MMTYPE_IND))
138         {
139             struct cm_atten_profile_indicate * indicate = (struct cm_atten_profile_indicate *) (message);
140             if (! memcmp (session->PEV_MAC, indicate->PEV_MAC, sizeof (session->PEV_MAC)))
141             {
142                 slac_debug (session, 0, __func__, "<-- CM_ATTEN_PROFILE.IND (%d)", session->sounds);
143
144 > #if SLAC_DEBUG ⋯
```

EV MAC Address	Number Array Size	Array of values

PLAXIDITYX
GO EVERYWHERE

Open PLC Utils

SLAC

PLC

- Which contained a trivial buffer overflow in the SLAC protocol

```
137         else if (LE16TOH (homeplug->homeplug.MMTYPE) == (CM_ATTEN_PROFILE | MMTYPE_IND))
138         {
139             struct cm_atten_profile_indicate * indicate = (struct cm_atten_profile_indicate *) (message);
140             if (! memcmp (session->PEV_MAC, indicate->PEV_MAC, sizeof (session->PEV_MAC)))
141             {
142                 slac_debug (session, 0, __func__, "<-- CM_ATTEN_PROFILE.IND (%d)", session->sounds);
143
144 > #if SLAC_DEBUG ⋯
154 #endif
155
156                 for (session->NumGroups = 0; session->NumGroups < indicate->NumGroups; session->NumGroups++)
157                 {
158                     AAG [session->NumGroups] += indicate->AAG [session->NumGroups];
159                 }
160                 session->NumGroups = indicate->NumGroups;
161                 session->sounds++;
162             }
163         }
```

AAG -> Stack Based buffer of size 58

PLAXIDITYX
GO EVERYWHERE

Open PLC Utils - Summary

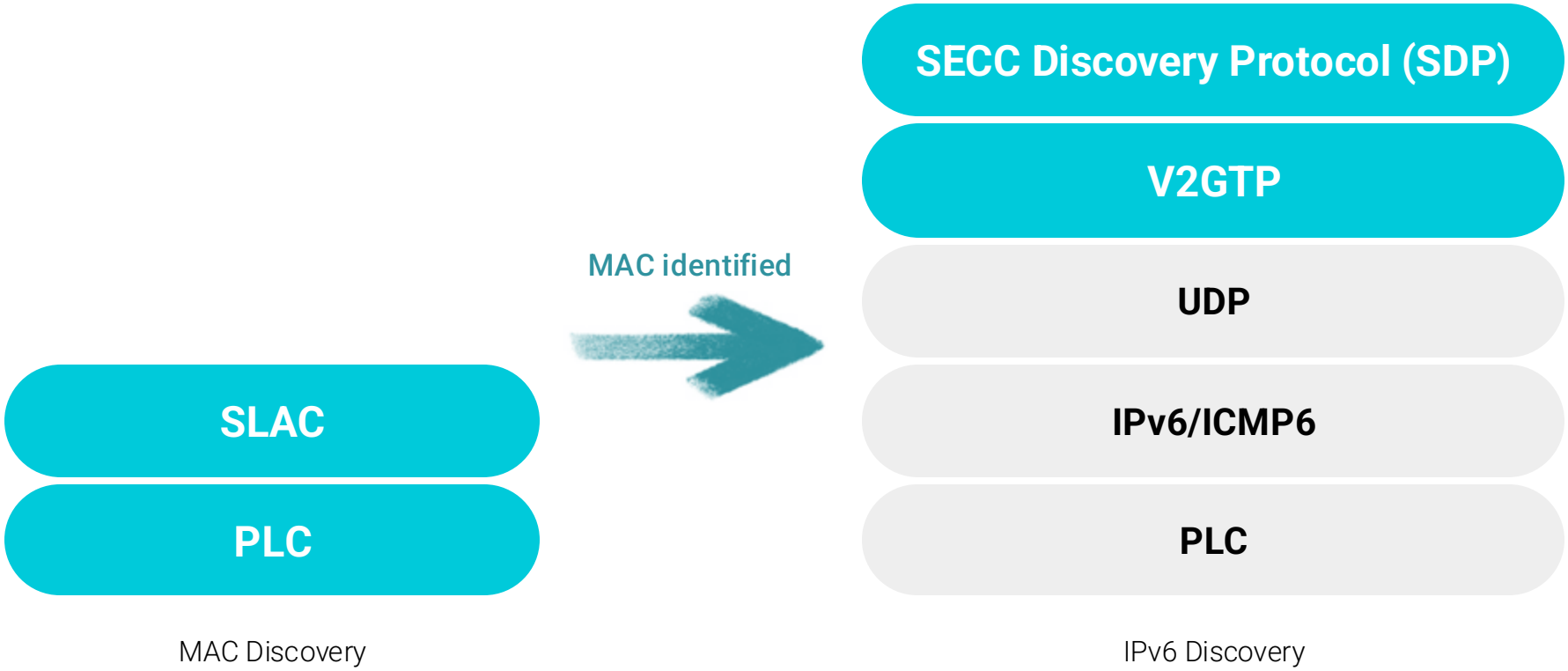
SLAC

PLC

- Mostly used by PLC testing tools
- Issue reported to Qualcomm and fixed on May on their commercial product
- CVE is issued CVE-2025-27071

PLAXIDITYX
GO EVERYWHERE

Phase 2 - IPv6 Discovery



PLAXIDITY X
GO EVERYWHERE

V2G Transport Protocol (V2GTP)

7.8.3.1 Structure

The V2GTP PDU consists of a header and a body section as shown in Figure 8.

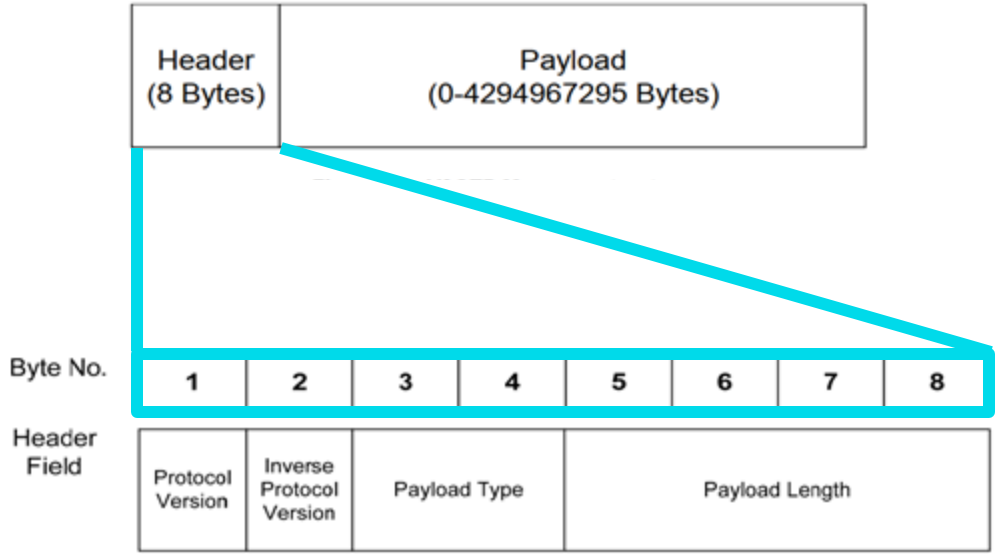
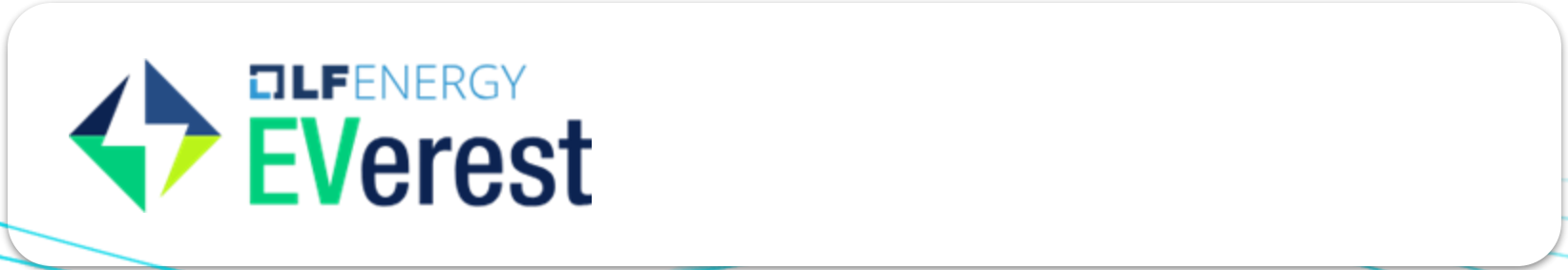


Figure 9 — V2GTP Message header structure

EVerest

EVerest is a Linux Foundation backed **open-source** modular framework for setting up a full stack environment **for EV charging**



EVerest

- A faulty boundary check allows an attacker to bypass the check
  and overflow the process memory with arbitrary data

  Leading to possible remote code execution from the charge port

everest-core / modules / EvseV2G / v2g_server.cpp

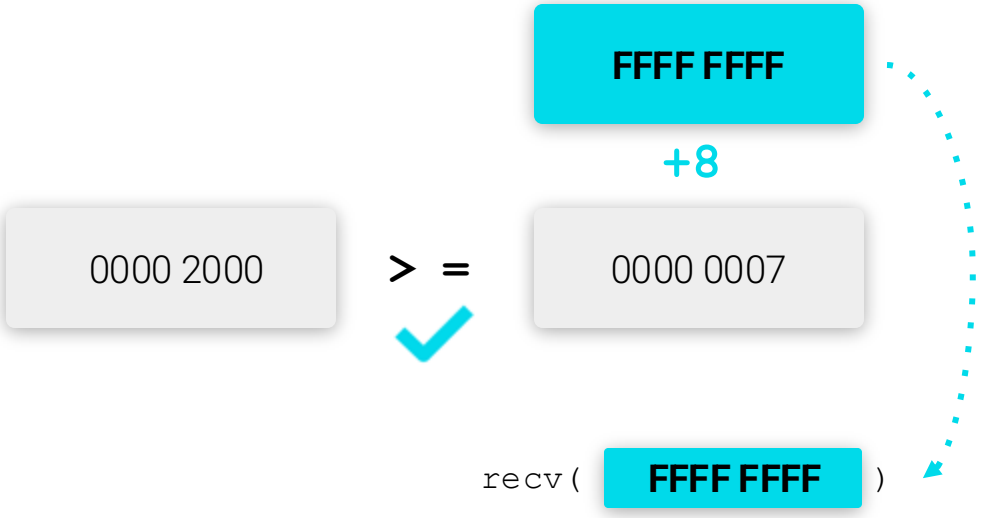
Code | Blame | 640 lines (561 loc) · 28.5 KB ·

```
141  static int v2g_incoming_v2gtp(struct v2g_connection* conn) {
161          dlog(DLOG_LEVEL_ERROR, "Invalid v2gtp header");
162          return -1;
163      }
164
165      if (conn->payload_len + V2GTP_HEADER_LENGTH > DEFAULT_BUFFER_SIZE) {
166          dlog(DLOG_LEVEL_ERROR, "payload too long: have %d, would need %d", DEFAULT_BUFFER_SIZE,
167              conn->payload_len + V2GTP_HEADER_LENGTH);
168
```

Received length Fixed 8

PLAXIDITY X
GO EVERYWHERE

CVE-2024-37310




```
Payload_length = FFFF FFFF

Integer overflow to 7

Check is bypassed


recv() unlimited bytes to the
heap buffer
```

EVerest

recv(conn, dst[8192], **0xFFFFFFFF**)

```
175         /* read request */
176         rv = connection_read(conn, &conn->buffer[V2GTP_HEADER_LENGTH], conn->payload_len);
177         if (rv < 0) {
178             dlog(DLOG_LEVEL_ERROR, "connection_read(payload) failed: %s",
179                  (rv == -1) ? strerror(errno) : "connection terminated");
180             return -1;
181         }
```

EVerest vulnerability

- Reported to EVerest maintainers which responded quickly
- Registered CVE-2024-37310 (Critical 9.0)

CVE-2024-37310 Detail

AWAITING ANALYSIS

This CVE record has been marked for NVD enrichment efforts.

Description

EVerest is an EV charging software stack. An integer overflow in the "v2g_incoming_v2gtp" function in the v2g_server.cpp implementation can allow a remote attacker to overflow the process' heap. This vulnerability is fixed in 2024.3.1 and 2024.6.0.

Metrics

CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

CVSS 3.x Severity and Vector Strings:

NIST: NVD	**Base Score:** N/A	NVD assessment not yet provided.
CNA: GitHub, Inc.	**Base Score:** 9.0 CRITICAL	**Vector:** CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H

PLAXIDITYX
GO EVERYWHERE

SECC Discover Protocol (SDP)

SDP
V2GTP
UDP
IPv6/ICMP6
PLC

TLS - Yes / No
Protocol - TCP / UDP

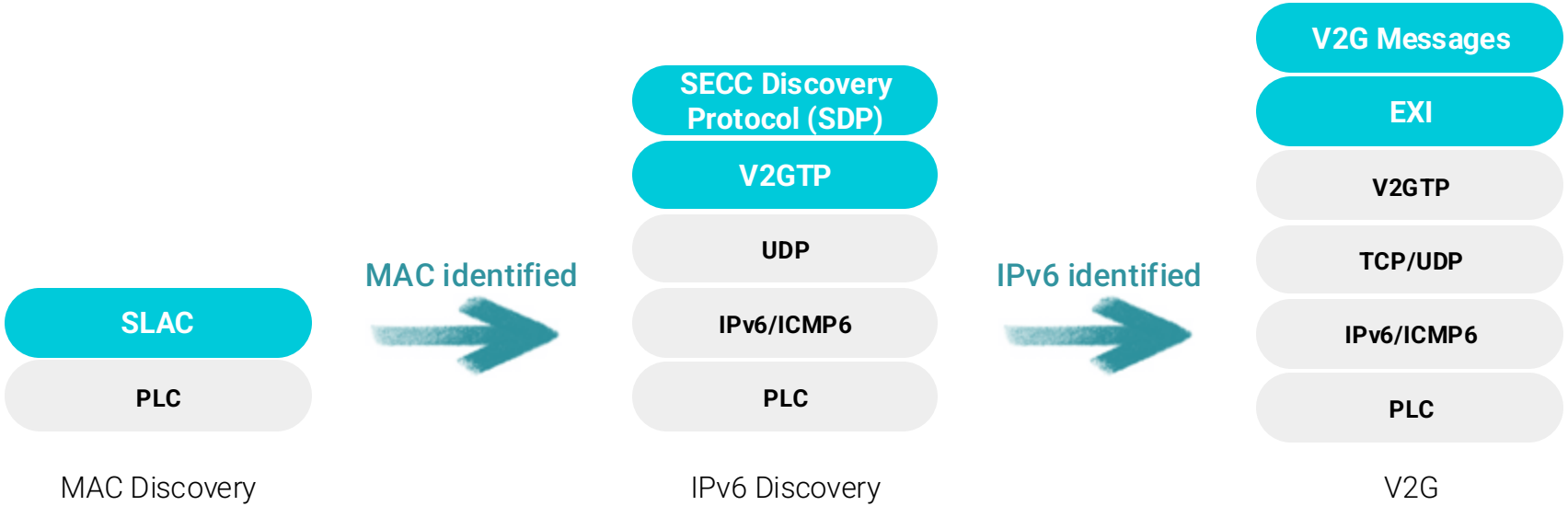
Service Discovery Request

Service Discovery Response

TLS Response
Protocol Response
IPv6 Address
Port

PLAXIDITYX
GO EVERYWHERE

Phase 3 - V2G

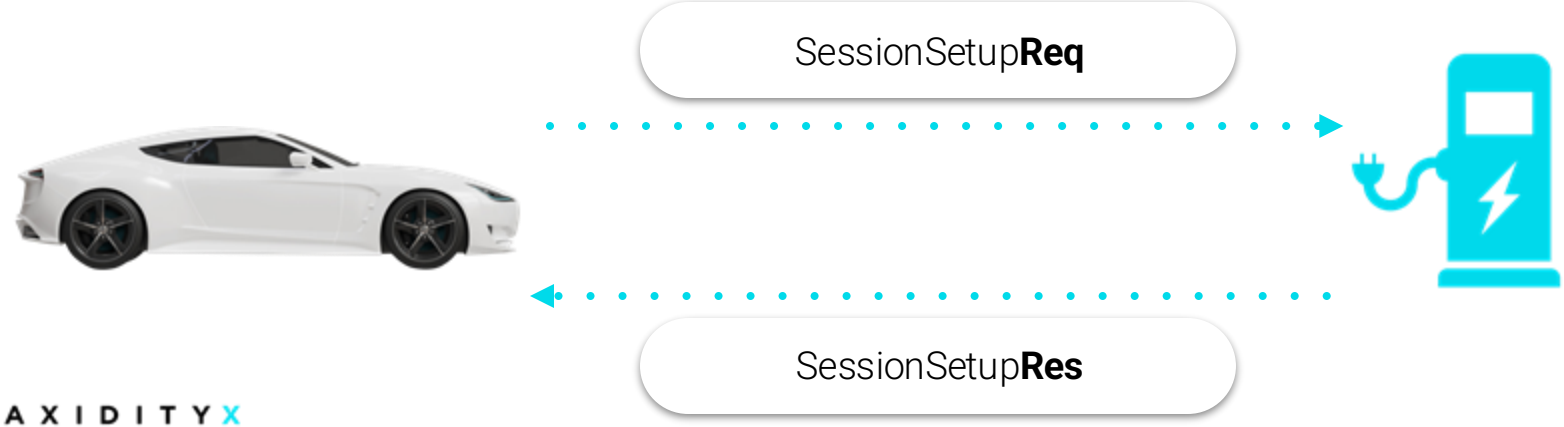

SLAC
PLC
MAC identified
SECC Discovery Protocol (SDP)
V2GTP
UDP
IPv6/ICMP6
PLC
IPv6 identified
V2G Messages
EXI
V2GTP
TCP/UDP
IPv6/ICMP6
PLC
MAC Discovery
IPv6 Discovery
V2G

Efficient XML Interchange (EXI)

- Binary serialization of XML documents between the two parties
  - Compact
  - Reduced payload on-the-wire
  - Schema based (XML) so decoders are efficient

V2G Messages

- The client (EVCC) initiates requests for the server (EVSE) which returns a corresponding message response message.
- All messages types and structures are defined in the XML schemas of ISO-15118 / DINSPEC-70121

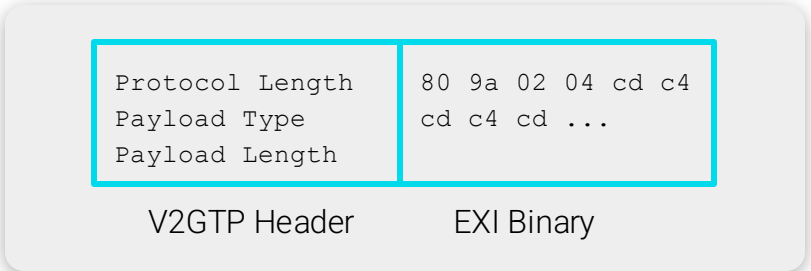


Protocol Stack

```
{
 "Header": {
  "SessionID": "1337133713371337"
 },
 "Body": {
  "SessionSetupRes": {
   "ResponseCode": 1,
   "EVSEID": "AAAAAAAAAA"
  }
 }
}
```

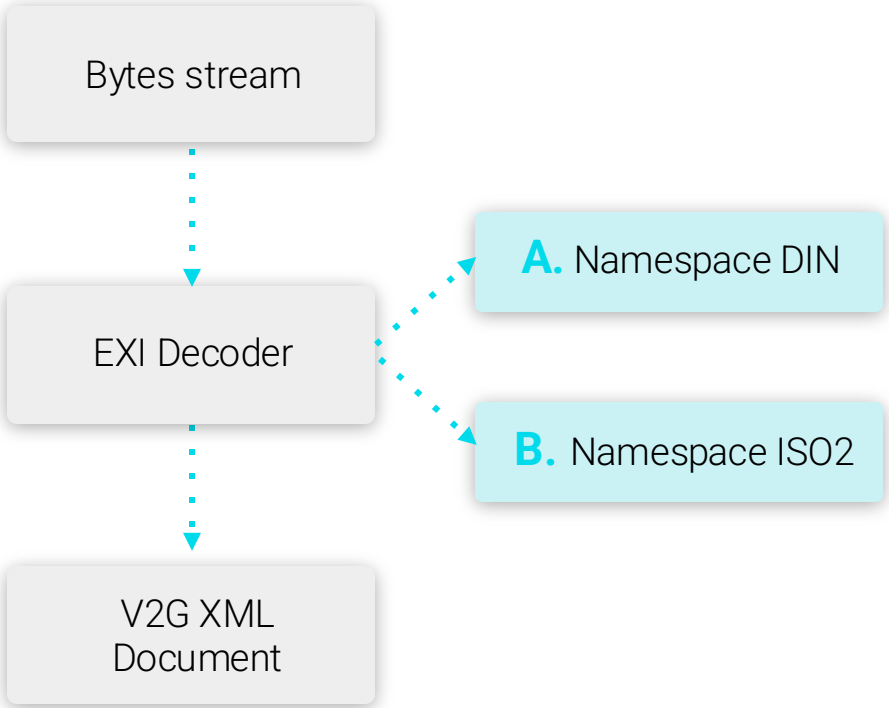
V2G Message

EXI Encoding



EXI & V2G

- EXI and V2G are tightly coupled
- V2G Messages are defined by an XML schema which is used when EXI decoding
- DIN and ISO define messages the same way, **but not all of them**



DIN

```
<xs:element name="ServiceDiscoveryRes" type="ServiceDiscoveryResType"
substitutionGroup="v2gci_d:BodyElement"/>
<xs:complexType name="ServiceDiscoveryResType">
<xs:complexContent>
  <xs:extension base="v2gci_d:BodyBaseType">
    <xs:sequence>
      <xs:element name="ResponseCode" type="v2gci_t:responseCodeType"/>
      <xs:element name="PaymentOptions" type="v2gci_t:PaymentOptionsType"/>
      <xs:element name="ChargeService" type="v2gci_t:ServiceChargeType"/>
      <xs:element name="ServiceList" type="v2gci_t:ServiceTagListType" minOccurs="0"/>
    </xs:sequence>
```

ISO

```
<xs:element name="ServiceDiscoveryRes" type="ServiceDiscoveryResType"
substitutionGroup="BodyElement"/>
<xs:complexType name="ServiceDiscoveryResType">
  <xs:complexContent>
    <xs:extension base="BodyBaseType">
      <xs:sequence>
        <xs:element name="ResponseCode" type="v2gci_t:responseCodeType"/>
        <xs:element name="PaymentOptionList" type="v2gci_t:PaymentOptionListType"/>
        <xs:element name="ChargeService" type="v2gci_t:ChargeServiceType"/>
        <xs:element name="ServiceList" type="v2gci_t:ServiceListType" minOccurs="0"/>
      </xs:sequence>
```

DIN

```
<xs:element name="ServiceDiscoveryRes" type="ServiceDiscoveryResType"
substitutionGroup="v2gci_d:BodyElement"/>
<xs:complexType name="ServiceDiscoveryResType">
<xs:complexContent>
  <xs:extension base="v2gci_d:BodyBaseType">
    <xs:sequence>
      <xs:element name="ResponseCode" type="v2gci_t:responseCodeType"/>
      <xs:element name="PaymentOptions" type="v2gci_t:PaymentOptionsType"/>
      <xs:element name="ChargeService" type="v2gci_t:ServiceChargeType"/>
      <xs:element name="ServiceList" type="v2gci_t:ServiceTagListType" minOccurs="0"/>
    </xs:sequence>
```

ISO

```
<xs:element name="ServiceDiscoveryRes" type="ServiceDiscoveryResType"
substitutionGroup="BodyElement"/>
<xs:complexType name="ServiceDiscoveryResType">
  <xs:complexContent>
    <xs:extension base="BodyBaseType">
      <xs:sequence>
        <xs:element name="ResponseCode" type="v2gci_t:responseCodeType"/>
        <xs:element name="PaymentOptionList" type="v2gci_t:PaymentOptionListType"/>
        <xs:element name="ChargeService" type="v2gci_t:ChargeServiceType"/>
        <xs:element name="ServiceList" type="v2gci_t:ServiceListType" minOccurs="0"/>
      </xs:sequence>
```

```
<xs:element name="ServiceList" type="v2gci_t:ServiceTagListType" minOccurs="0"/>
```

DIN

```
<xs:complexType name="ServiceTagListType">
<xs:sequence>
<xs:element name="Service" type="ServiceType"
maxOccurs="unbounded"/>
</xs:sequence>
```

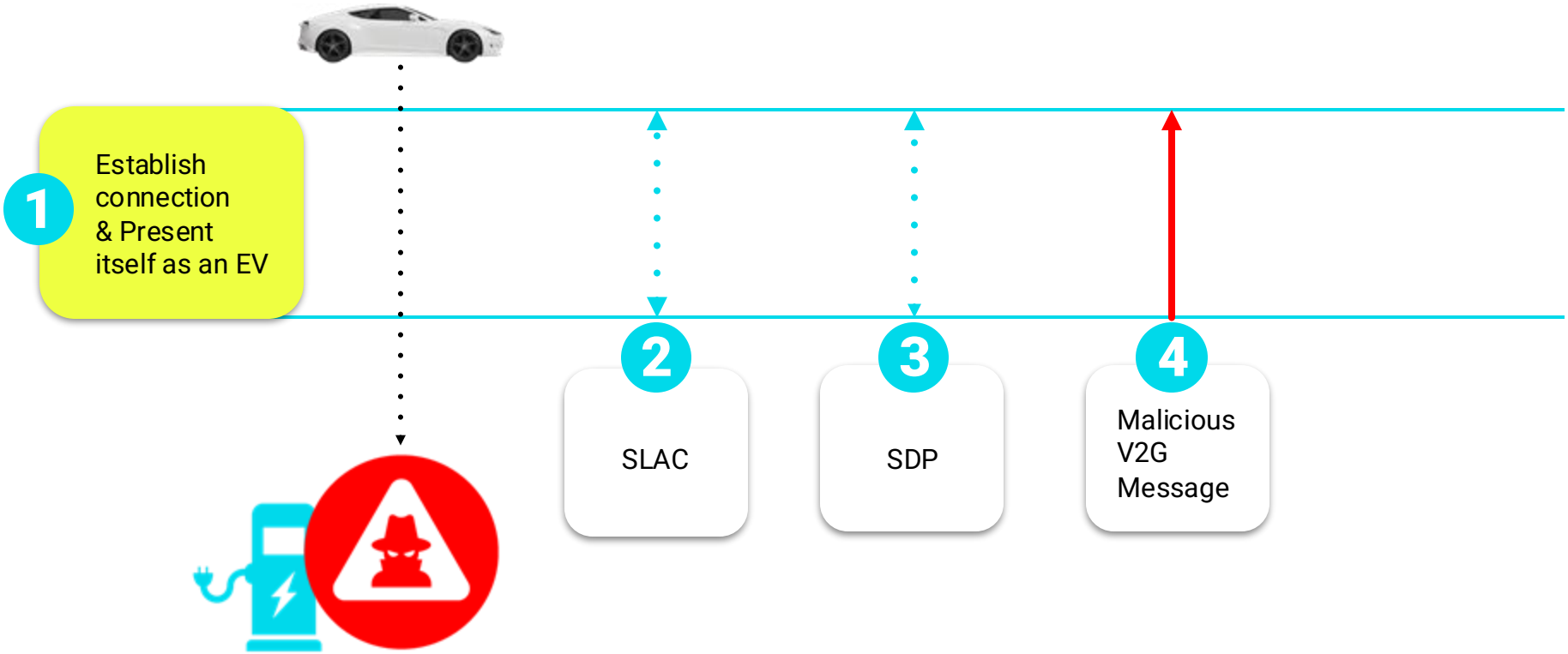
```
<xs:element name="ServiceList" type="v2gci_t:ServiceListType" minOccurs="0"/>
```

ISO

```
<xs:complexType name="ServiceListType">
          <xs:sequence>
                    <xs:element name="Service" type="ServiceType"
maxOccurs="8"/>
          </xs:sequence>
</xs:complexType>
```

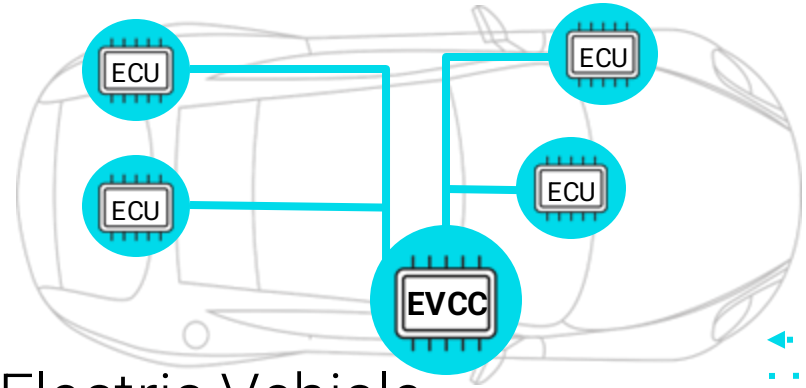
PLAXIDITYX
GO EVERYWHERE

Attack Illustration



PLAXIDITYX
GO EVERYWHERE

Charging Ecosystem



Electric Vehicle

CPO

OCPP

V2G

ISO-15118

DINSPEC-70121

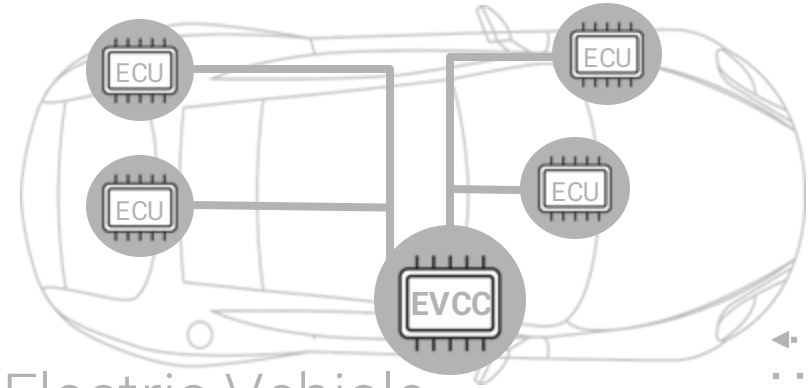
V2GTP

SLAC

Charging Stations

PLAXIDITYX
GO EVERYWHERE

Charging Ecosystem



Electric Vehicle

V2G
ISO-15118
DINSPEC-70121

V2GTP

SLAC

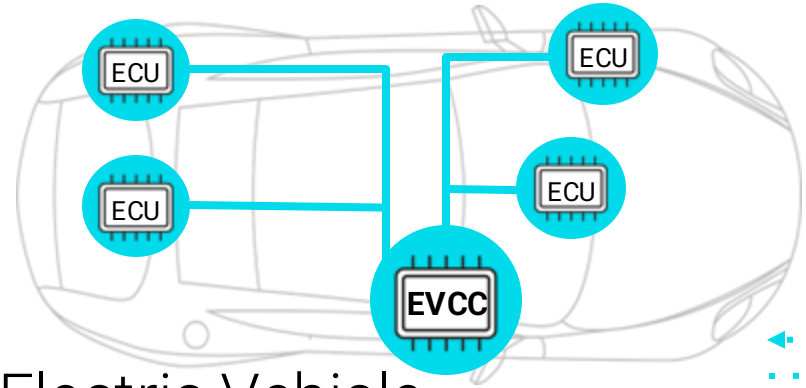
CPO Server

OCPP

Charging Stations

PLAXIDITY X
GO EVERYWHERE

Charging Ecosystem



Electric Vehicle

V2G

ISO-15118

DINSPEC-70121

V2GTP

SLAC

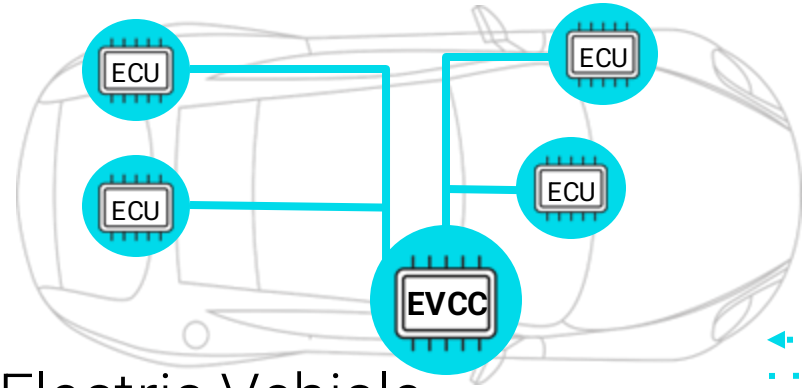
CPO

OCPP

Charging Stations

PLAXIDITY X
GO EVERYWHERE

Charging Ecosystem



Electric Vehicle

CPO

OCPP

V2G

ISO-15118

DINSPEC-70121

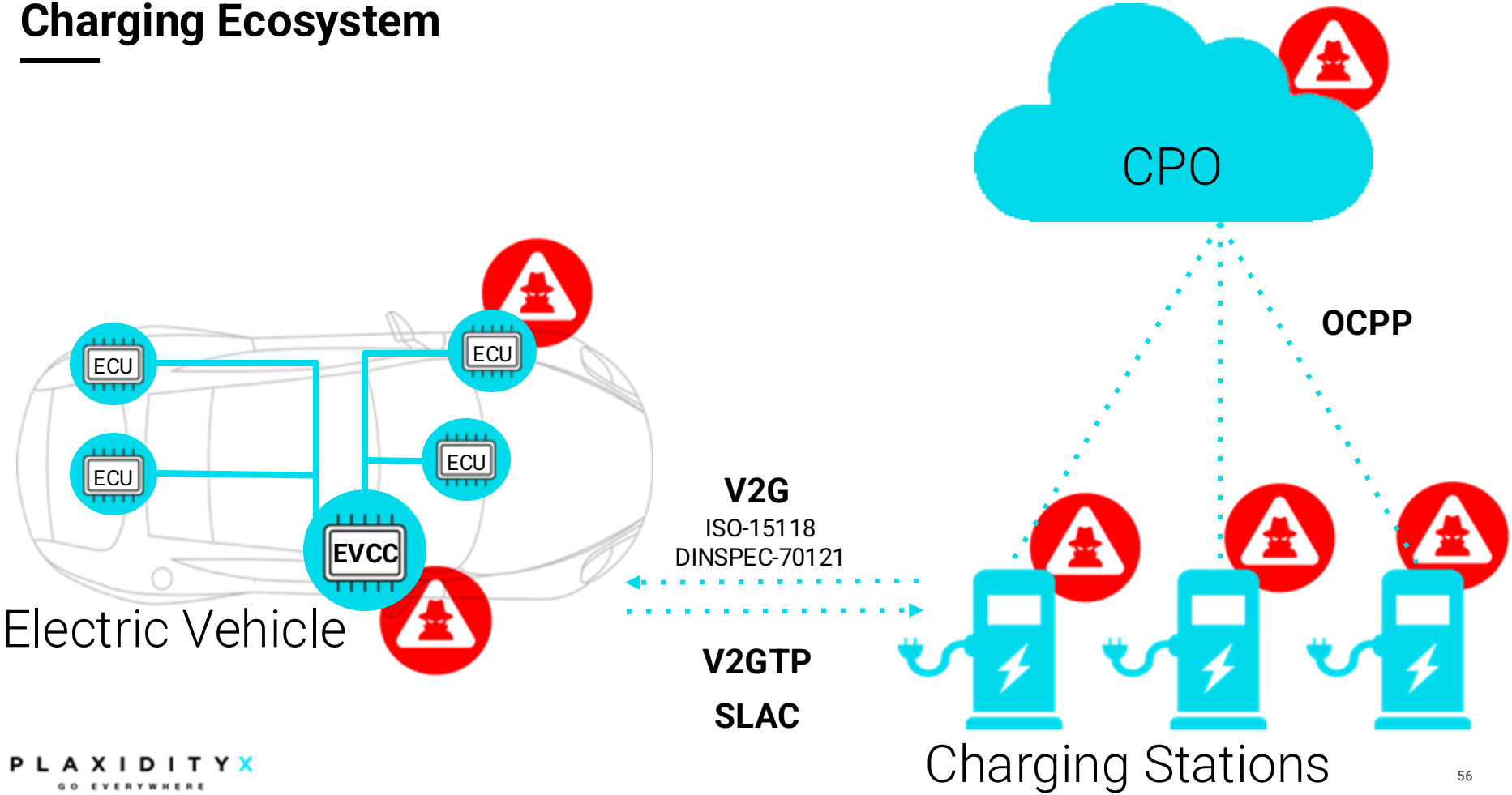
V2GTP

SLAC

Charging Stations

PLAXIDITYX
GO EVERYWHERE

Charging Ecosystem



PLAXIDITYX
GO EVERYWHERE

Conclusion

OCPP

Implementing CPOs should be done with security in mind

- Secure endpoints
- Limit public exposure to OCPP servers
- Least privilege principle

SLAC / V2GTP

Usage of 3rd party tools must be used with the necessary caution in mind

- Maintained and trusted projects
- Vulnerability management (CVE)

EXI

Security Controls on ECUs

- Find the match between the standard and your hardware capabilities
- Exploit mitigations in the embedded environment (Stack protection, MPU)

PLAXIDITYX
GO EVERYWHERE

Thank you

Meet us at booth 11

PLAXIDITYX
GO EVERYWHERE