2018 车联网安全论坛
暨第三届ASC移动物联网安全高峰
推动车联网行业应用系统安全落地 贯通中国车联网产业安全服务链
PROMOTE THE IMPLEMENTATION OF APPLICATION SYSTEM SECURITY, THROUGHOUT THE IOV SECURITY SERVICE CHAIN
INTERNET OF VEHICLES

智能网联汽车安全防护体系

北京娜迦信息科技发展有限公司

CONTENTS

1 PART

智能网联汽车构成

汽车安全威胁划分

NAGA · IN

端：车辆
（智能移动终端）

管：局域/广域
（数据交互通信）

云：云控平台
（数据应用管理）

- 建立流程清晰、高效顺畅的智能汽车信息安全管理联动机制，从“端、管、云”三方面入手，加强关键芯片、传感设备、基础软件、核心算法、通信协议和系统应用等环节安全防控。
- 建立覆盖“中心平台-运营企业-只能车辆”的应急响应机制和智能汽车信息安全漏洞库，设计适应不同安全等级的响应和恢复策略，防范各种非法入侵攻击和信息安全事件。

2 PART

汽车安全威胁划分

汽车安全威胁划分

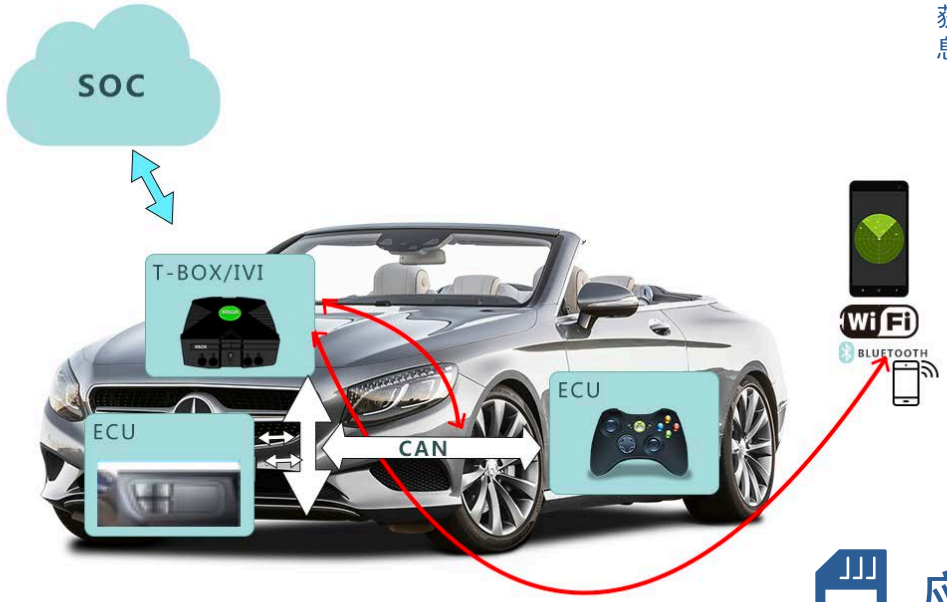
NAGA · IN

网络攻击

网络攻击，在网络传输层进行消息篡改、伪造服务、窃听等攻击，比如重放攻击、中间人攻击、TCP注入攻击

数据威胁

对车载终端系统进行分析，从而获取用户个人敏感信息；系统信息，分析出系统存在的漏洞



系统漏洞

对车载终端系统进行恶意修改，对系统进行攻击。比如植入木马、后门、二次打包应用等。

应用漏洞

通过利用车载终端软件系统安全漏洞，实施对车载设备的攻击威胁

3 PART

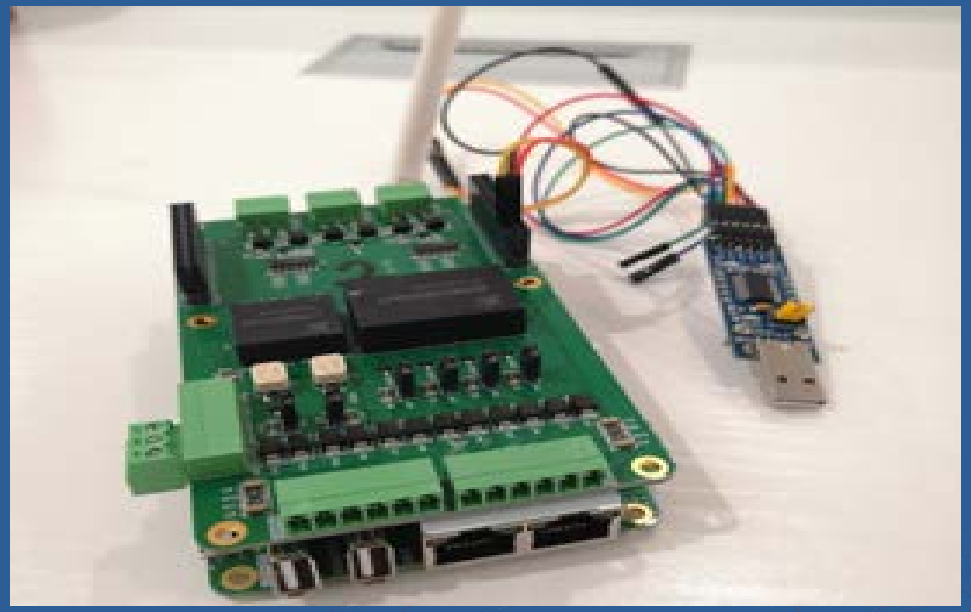
实现形式

实现形式

NAGA · IN


大鲲网联智慧管理云平台
车辆区域图
数据流向图
安全态势图
统计概览
实时累计
攻击类型排名
实时攻击趋势
被攻击车辆排名

+

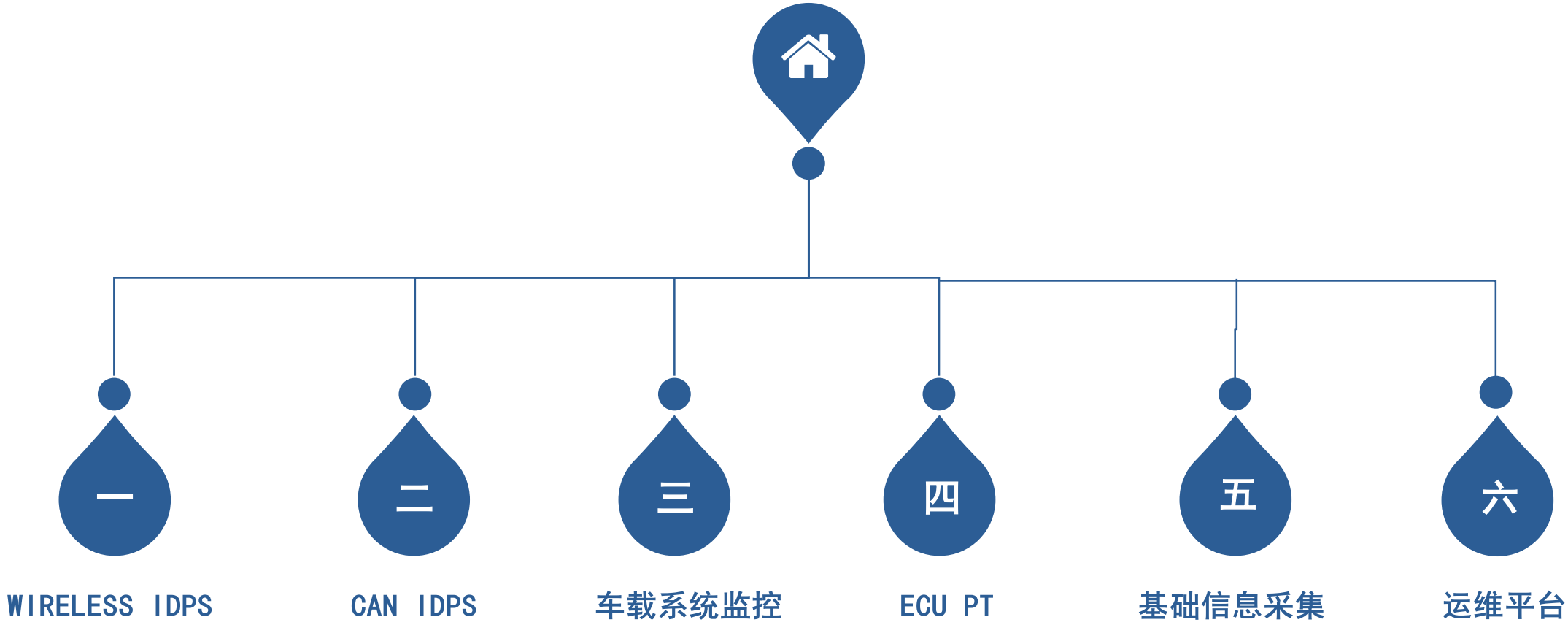


4 PART

六大功能点

六大功能点

NAGA · IN



一、WIRELESS IDPS

一 虚假wifi过滤

采用两种方式对WIFI安全进行检测、防护：

1. 采用对wpa_supplicant库进行移植，加入安全模块；
2. 采用shell封装调用的方式，对wifi安全进行处理。

二 可疑蓝牙过滤

采用对hcitool库/工具，进行二次封装，提供安全接口，以实现对bt的安全扫描、安全检测、安全阻断等功能。

三 伪造蜂窝网络过滤

从联通性和流量监控两个维度进行阻断：

1. 联通性：监测无线IDPS对外网的连接IP，进行规则过滤链接；
2. 流量监控：监控无线IDPS对外网的流量，根据规则进行监控、阻断。

二、CAN IDPS

NAGA · IN

1

白名单过滤

依据CAN报文集合，在运维平台配置“受信集合”。对广播在总线上的命令进行过滤，及时拦截第三方伪造的非法命令。

2

动作状态匹配

对CAN总线所有命令关联的动作进行整合，建立运行时状态机。对非法动作序列进行预警拦截。

3

指令阀设置

根据不同汽车各字段对应的物理含义进行预置，规范数值的合理范围，避免极端动作的发生。

4

流量监控

对非正常业务，以及ddos攻击进行检测，并根据旁路或串行部署模式实现相应的报警和阻断功能。

5

情景分析

依据长期的驾驶行为深度学习，建立驾驶行为画像，对不合理情景进行智能规避。

三、车载系统监控

NAGA · IN

端口扫描

01

采用本地语言编程，对系统服务的说有端口进行SOCKET通讯，以探测系统开放的服务端口是否符合安全规定，同时实时进行监测，对发生异常的服务端口进行预警

恶意程序扫描

02

采用本地语言编写，对系统中所有process进行内存hash处理，对比服务器提前存储的白名单，同时发现异常的hash后进行预警处理

本地提权

03

采用本地语言，对系统中的程序提权行为进行检测，同时动态的把相关数据上报服务器，服务器对系统中异常的行为进行预警处理

三、车载监控系统

NAGA · IN

系统库篡改

04

采用本地语言，对系统中的关键核心库进行特征校验和服务器对比，校验的方式为对核心库ELF格式进行二进制解析后，展开到内存中，对程序段中的函数指令进行hash处理，同时把hash后的结果与服务器中保存的标准值进行比对，以确保核心库的使用安全

登录用户过滤

05

采用本地语言编程，对登录BSRF的用户进行监测，同时把监测后的数据上传至服务器，服务器对登录的用户进行授权验证，查验是否是合法用户登录。

SYN Flood攻击

06

一种利用TCP协议缺陷，发送大量伪造的TCP连接请求，从而使得被攻击方资源耗尽（CPU满负荷或内存不足）的攻击方式

四、ECU PT

函数调用树监控

根据出厂设置，识别并且映射ECU的逻辑，生成函数调用树，在运行时匹配调用函数，防止注入行为的发生。

二进制代码完整性校验

对出厂时的各ECU核心二进制代码进行校验标记，生成专属的校验码，用于实时离线匹配，拦截针对ECU代码的篡改行为。

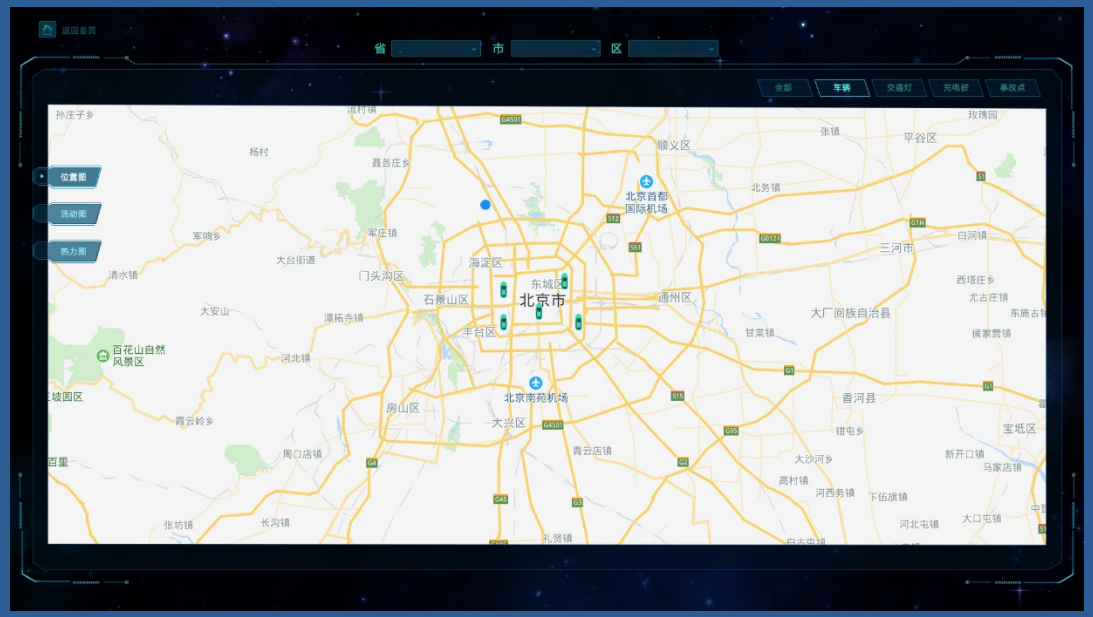
五、基础数据采集（国标80项）

NAGA · IN

整车数据	驱动电机数据	燃料电池数据	发动机数据	车辆位置数据	电池数据	报警数据
车辆状态	驱动电机个数	燃料电池电压	发动机状态	定位状态	最高电压电池子系统号	最高报警等级
充电状态	驱动电机总成信息	燃料电池电流	曲轴转速	经度	最高电压电池单体代号	通用报警标志
运行模式	驱动电机序号	燃料消耗率	燃料消耗率	纬度	电池单体电压最高值	可充电储能装置故障总数N1
车速	驱动电机状态	燃料电池温度探针总数			最低电压电池子系统号	可充电储能装置故障代码列表
里程	驱动电机控制器温度	探针温度值			最低电压电池单体代号	驱动电机故障总数N2
总电压	驱动机转速	氢系统中最高温度			电池单体电压最低值	驱动电机故障代码列表
总电流	驱动机转矩	氢系统中最高温度探针代号			最高温度子系统号	发动机故障总数N3
SOC	驱动机温度	氢气最高浓度			最高温度探针序号	发动机故障列表
DC-DC状态	电机控制器输入电压	氢气最高浓度传感器代号			最高温度值	其他故障总数N4
挡位	电机控制器直线母线电流	氢气最高压力			最低温度子系统号	其他故障代码列表
绝缘电阻		氢气最高压力传感器代号			最低温度探针序号	
		高压DC-DC状态			最低温度值	

六、运维平台

NAGA · IN



可视化展示

可针对CAN IDPS、WIRELESS IDPS、ECU PT、车载系统监控等安全防御模块进行参数配置，生成规则包，本地终端定期远程拉取，进行更新应用。

六、运维平台

NAGA · IN

运维管理

运维平台提供车辆实时位置跟踪，报警信息上报，车辆轨迹回放，车队管理，单车动静态数据查询，远程调度，安全事件响应等功能

防御规则配置

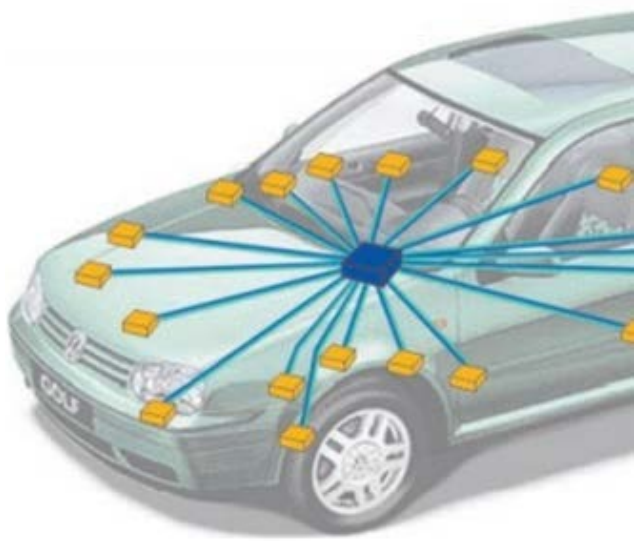
可针对CAN IDPS、WIRELESS IDPS、ECU PT、车载系统监控等安全防御模块进行参数配置，生成规则包，本地终端定期远程拉取，进行更新应用。

5 PART

CAN总线分析

CAN总线分析

NAGA · IN



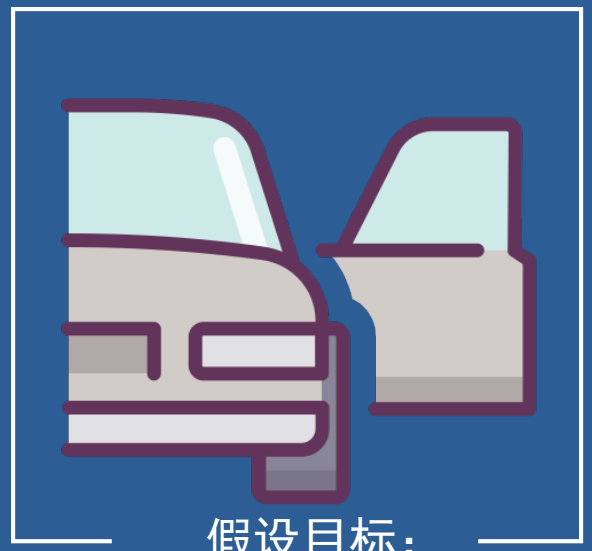
CAN总线安全

◆在现阶段的汽车行驶过程中，CAN总线缺乏加密和访问控制机制可被攻击者逆向总线通信协议，分析汽车控制指令，用于攻击指令伪造

◆缺乏认证及消息认证机制，不能对攻击者伪造篡改的异常消息进行识别和预警。鉴于CAN总线的特性，攻击者可通过物理侵入或远程侵入的方式实施消息伪造、拒绝服务、重放等攻击，需要通过安全隔离来确保智能网联汽车内部CAN网络不被非法入侵。

CAN总线分析

NAGA · IN



假设目标：
驾驶员的车门是打开还是关闭

使用can_unique.py对单个字段进行逆向（以下过程适用于简单的单比特消息）：

首先记录几分钟CAN消息，关闭所有门并将其保存在background.csv中

```
./can_logger.py
mv output.csv background.csv
```

然后在执行您感兴趣的操作时运can_logger.py几秒钟，例如打开然后关闭左前门并保存为门-fl-1.csv
重复此过程并将其保存为door-f1-2.csv，以便轻松确认任何疑问。

CAN总线分析

NAGA · IN

使用can_unique.py查找↓↓↓

```
$ ./can_unique.py door-fl-1.csv background*
id 820 new one  at byte 2 bitmask 2
id 520 new one  at byte 3 bitmask 7
id 520 new zero at byte 3 bitmask 8
id 520 new one  at byte 5 bitmask 6
id 520 new zero at byte 5 bitmask 9
id 559 new zero at byte 6 bitmask 4
id 804 new one  at byte 5 bitmask 2
id 804 new zero at byte 5 bitmask 1
```

```
$ ./can_unique.py door-fl-2.csv background*
id 672 new one  at byte 3 bitmask 3
id 820 new one  at byte 2 bitmask 2
id 520 new one  at byte 3 bitmask 7
id 520 new zero at byte 3 bitmask 8
id 520 new one  at byte 5 bitmask 6
id 520 new zero at byte 5 bitmask 9
id 559 new zero at byte 6 bitmask 4
```

其中一个有可能表明驾驶员的车门打开了。

再浏览每个消息ID，确定哪一个是正确的。在两次运行中都能改变任何正确的位。

可以排除804，因为它只发生在第一次运行中。可以排除672，因为它只发生在第二次运行中。

剩下这些消息ID：820,520,559。

再仔细看看每一个 ↓↓↓

```
$ fgrep，559，door-fl-1.csv | head
0,559,00ff0000000024f0
0,559,00ff000000004464
0,559,00ff0000000054a9
0,559,00ff0000000064e3
0,559,00ff00000000742e
0,559,00ff000000008451
0,559,00ff00000000949c
0,559,00ff00000000a4d6
0,559,00ff00000000b41b
0,559,00ff00000000c442
```

ID 559看起来像一个递增值，所以不是我们正在找的。

CAN总线分析

NAGA · IN

再仔细看看每一个 ↓↓↓↓

```
$ fgrep， 520， door-fl-2.csv
0,520,26ff00f8a1890000
0,520,26ff00f8a2890000
0,520,26ff00f8a2890000
0,520,26ff00f8a1890000
0,520,26ff00f8a2890000
0,520,26ff00f8a1890000
0,520,26ff00f8a2890000
0,520,26ff00f8a1890000
0,520,26ff00f8a2890000
0,520,26ff00f8a1890000
0,520,26ff00f8a2890000
0,520,26ff00f8a1890000
```

ID 520在两个值之间振荡。
但是我只打开并关闭了一次门，
所以可能不是它。

```
$ fgrep， 820， door-fl-1.csv
0,820,44000100a500c802
0,820,44000100a500c803
0,820,44000300a500c803
0,820,44000300a500c802
0,820,44000300a500c802
0,820,44000300a500c802
0,820,44000100a500c802
0,820,44000100a500c802
0,820,44000100a500c802
```

ID 820看起来很有可能是 ——
当门关闭时，
它从44000100a500c802开始。
门打开后转到44000300a500c802。
然后当门再次关闭时，
它会返回到44000100a500c802。

查看其他运行数据↓↓↓↓

```
$ fgrep， 820， door-fl-2.csv
0,820,44000100a500c802
0,820,44000300a500c802
0,820,44000100a500c802
```

如果驱动程序的门打开，
则设置字节2位掩码2处的消息id 820。
如果我们用前排乘客的门重复这个过程，
然后我们将发现如果右前门打开，
则设置字节2位掩码4处的消息id 820。
类似的信号彼此接近很常见。

CAN总线分析

NAGA · IN

使用can_bit_transition.py对单个字段进行逆向：

通过以下过程，可以快速找到一段时间内始终为0的位（没有踩刹车时），在不同的时间段内总是1的位（踩刹车时）

时间戳50.0到65.0之间没有按制动器，在时间戳69.0到79.0之间按制动器。
确定时间戳，鼠标悬停时的工具提示格式为：timestamp：value

下载日志并运行时间戳传递的脚本
./can_bit_transition.py ./honda_crv_ex_2017_can-1520354796875.csv 50.0-65.0 69.0-79.0

该脚本将输出在第一个时间范围内始终为低的位，在第二个时间范围内始终为高位（反之亦然）
id 17c 0 - > 1，字节4位掩码1
id 17c 0 - > 1，字节6位掩码32
id 221 1 - > 0，字节0位掩码4
id 1be 0 - > 1，字节0位掩码16

并通过id搜索消息，双击相应的位，然后来绘制上述位。
消息ID 0x17c是用户制动和自适应巡航控制制动相结合，所以沿着侧面0x221和0x1be绘制其中一个信号。
通过重放驱动器，可以看到0x221不是制动信号（当随机时间高时与制动不对应）。
接下来看0x1be，这是制动踏板信号（每当按下制动踏板时会变高，当自适应巡航控制制动时没有变高）。

假设目标：
找到制动踏板信号

谢谢观看

北京娜迦信息科技发展有限公司