CANSPY

A Platform for Auditing CAN Devices

Arnaud Lebrun

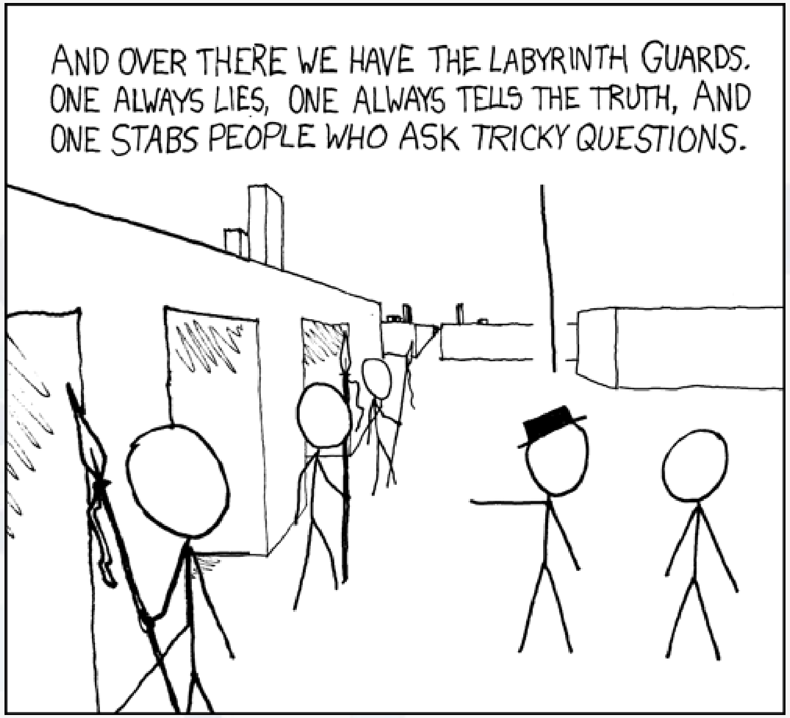
Jonathan-Christofer Demay

Auditing conventional IT systems

- **Penetration testing**
  - A form of security audit
  - Assess the risks of intrusion
  - Actual tests instead of a review process
  - The point of view of a real attacker (the “black-box” approach)
  - Relevant evaluation of impact and exploitability
- **Limitations**
  - Less time
  - Less resources
  - More ethics
- **Counter-measure: the “grey-box” approach**

The CISO’s dilemma

- **The hand they are dealt with**
  - Huge scope of responsibility
  - Continuous changes
  - Major security threats
  - Risk of substantial damages
  - Limited budget
- **Their response**
  - They rely on penetration testing
  - They welcome the “gray-box” approach
  - They rely on risk analysis first and foremost
  - They divide perimeters accordingly



What about car manufacturer ?

- **They are starting to include cyber-security along with conventional safety**

What about car manufacturer ?

- **They are starting to include cyber-security along with conventional safety**

www.heise.de/ct/artikel/Beemer-Open-Thyself-Security-vulnerabilities-in-BMW-s-ConnectedDrive-2540957.html

Beemer, Open Thyself! – Security vulnerabilities in BMW's ConnectedDrive

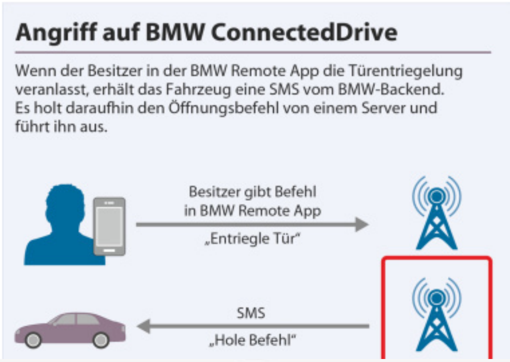
INFOS ZUM ARTIKEL

WISSEN | HINTERGRUND

Translation: Fabian A. Scherschel, Dieter Spaar 05.02.2015

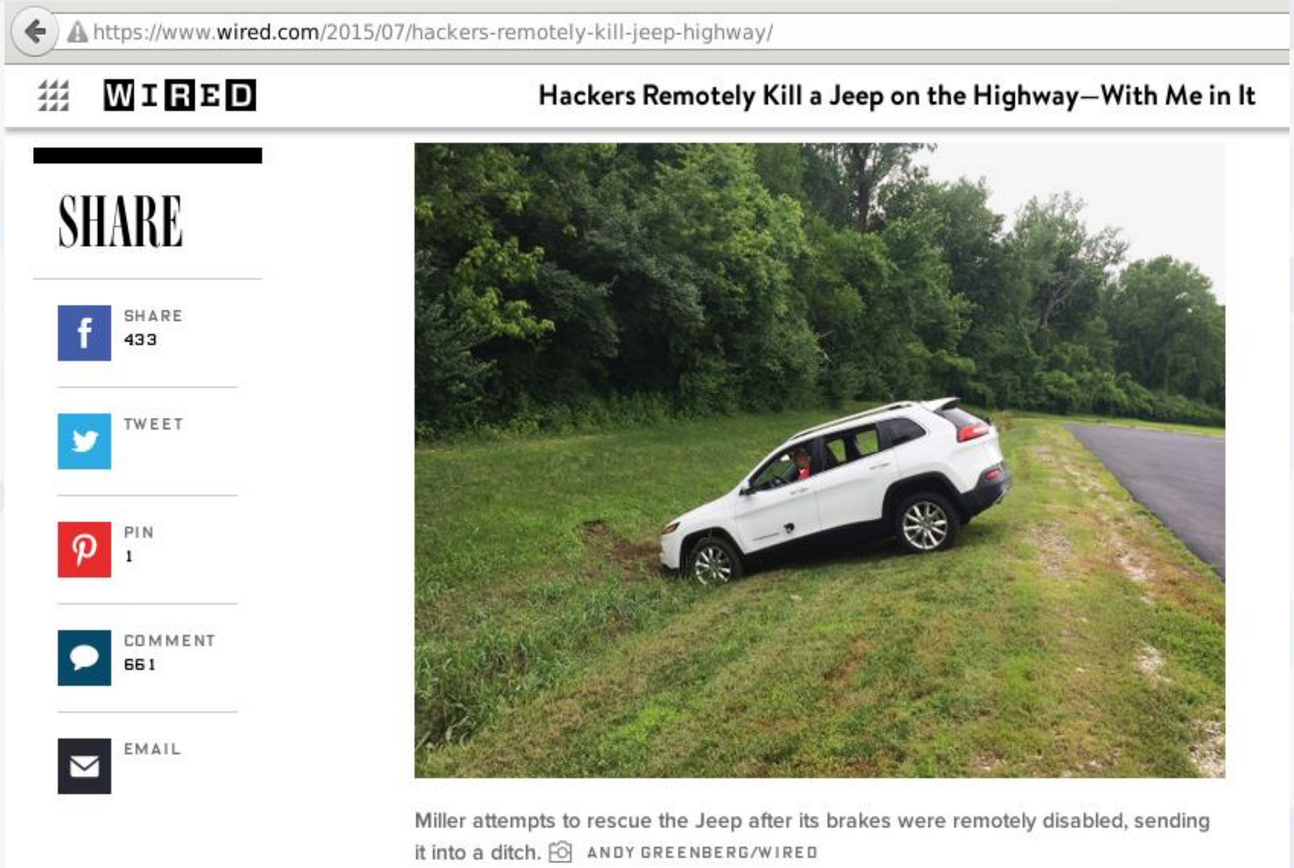
BMW, ConnectedDrive, Security, Sicherheitslücke

Cars with built-in modems are sending data to their manufacturers - German motorist's club ADAC wanted to know what exactly gets sent. c't connected ADAC with a specialist who analysed the data transmissions, using the example of BMW's ConnectedDrive technology. He discovered security vulnerabilities that even allow unauthorised attackers to open the vehicles.



What about car manufacturer ?

- They are starting to include cyber-security along with conventional safety



What about car manufacturer ?

- **They are starting to include cyber-security along with conventional safety**

https://www.theguardian.com/technology/2016/jun/06/mitsubishi-outlander-car-hacked-security

UK world sport football opinion culture business lifestyle fashion environment tech travel

Self-driving cars

Yet another car can be hacked - this time it's the Mitsubishi Outlander hybrid

Mitsubishi urged to recall at least 100,000 cars after hackers remotely turned off the alarm system, controlled the lights and drained the battery

Nellie Bowles in San Francisco

@NellieBowles

Tuesday 7 June 2016 02.55 BST

Shares 102 | Comments 17

Save for later

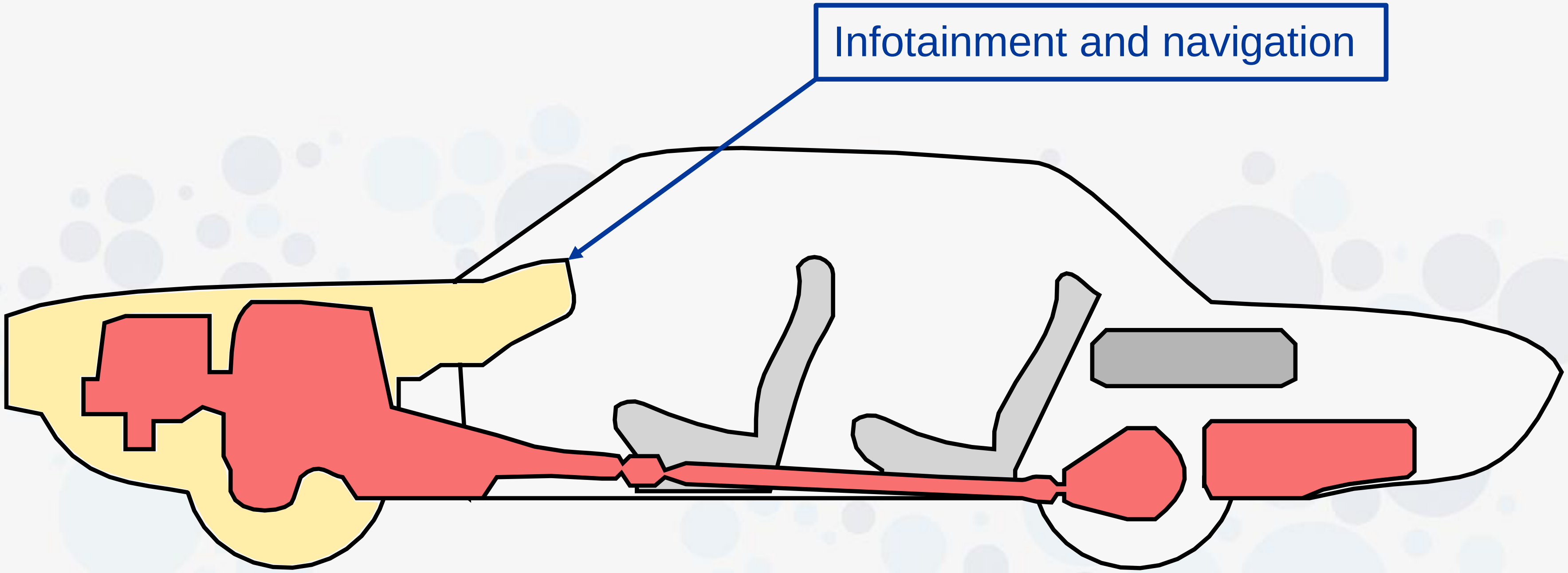
Mitsubishi joins Jeep, Nissan and Tesla on the list of cars that have had vulnerabilities highlighted. Photograph: Simon Stuart Miller (commissioned)

What about car manufacturer ?

- **They are starting to include cyber-security along with conventional safety**
- **The same approach can be applied**
  - For each vehicle
    - Conduct risk analysis
    - Prioritize ECUs
    - Conduct penetration tests accordingly
    - Carry out corrective actions
  - End for

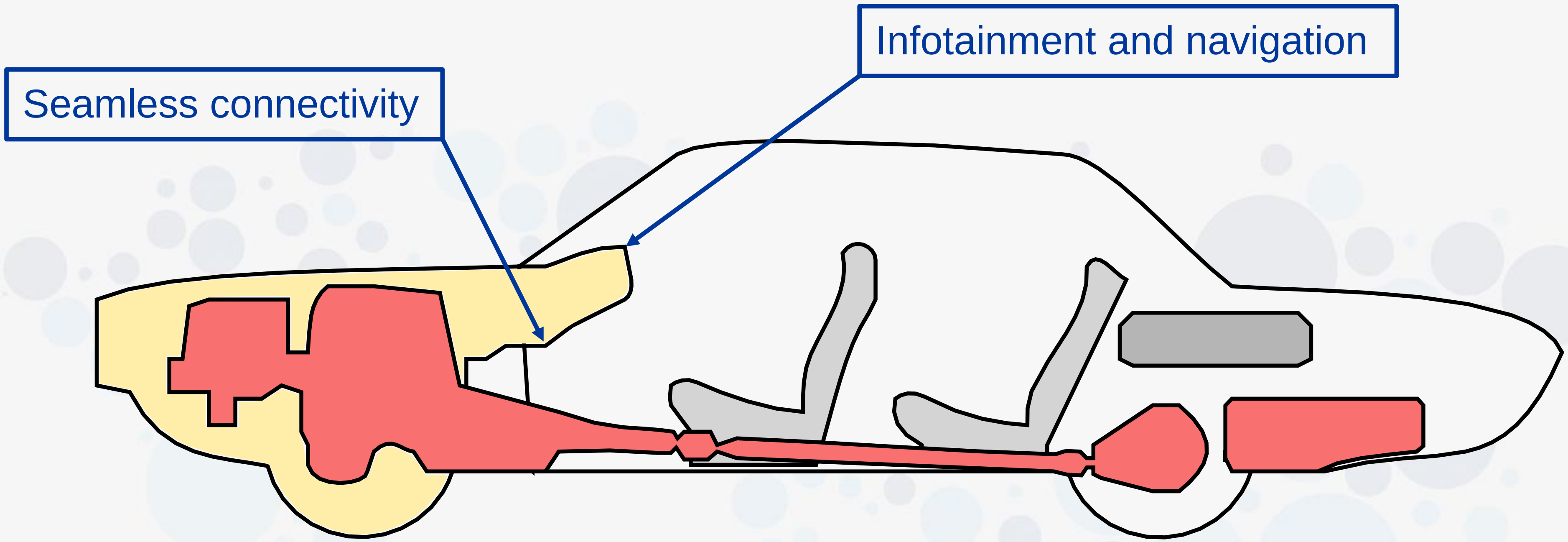
- **Some ECUs can be common to several vehicles**
- **Corrective actions may be difficult to carry out**

It always begins with…



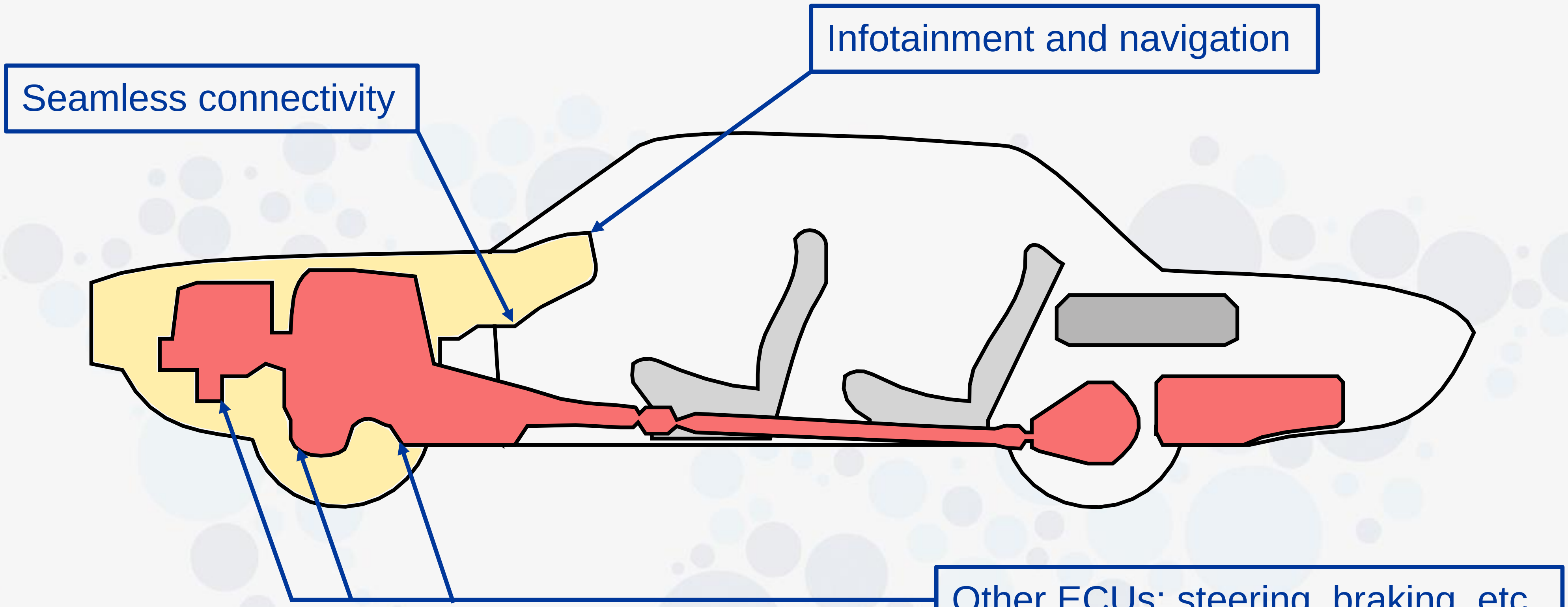
- **Consumer-grade connectivity**
  - Wi-Fi, Bluetooth and USB
  - Nothing new here

It always begins with…



- **Mobile broadband connectivity**
  - Conventional protocols (TCP, HTTP, …)
  - Setting up an IMSI catcher
  - Then again, nothing new here

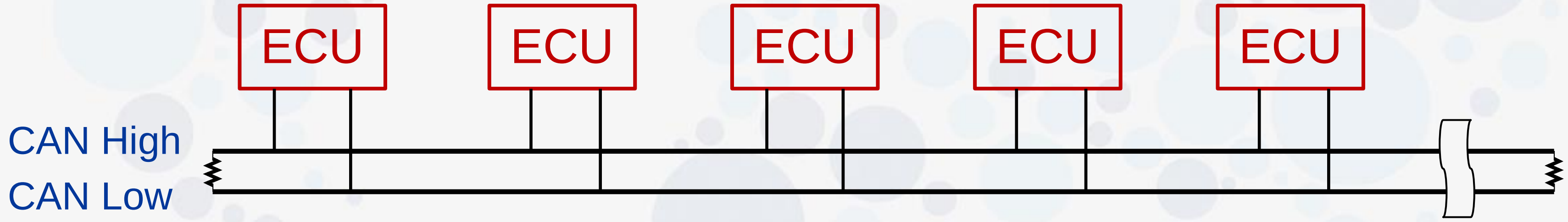
It always begins with…



- **CAN attacks**
  - Bypass CAN bus segmentation (architecture-dependant)
  - Reverse-engineer higher-layer protocols
  - Break the Security Access challenge (ISO 14229)

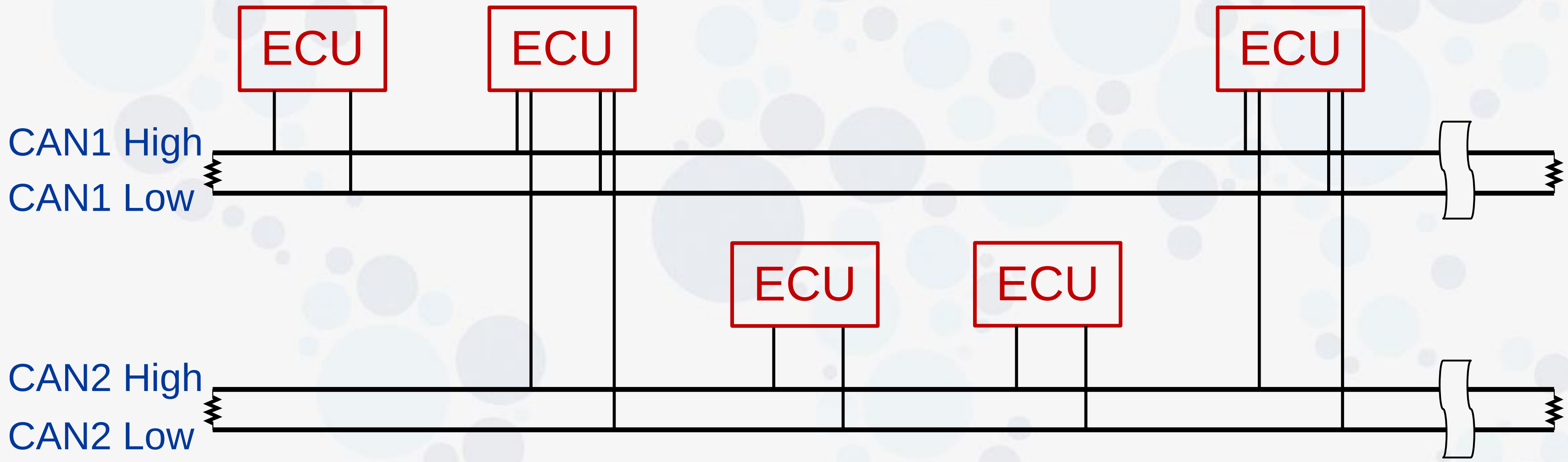
CAN architectures

- **One bus (to rule them all)**
  - Less common nowadays
  - Congestion issues



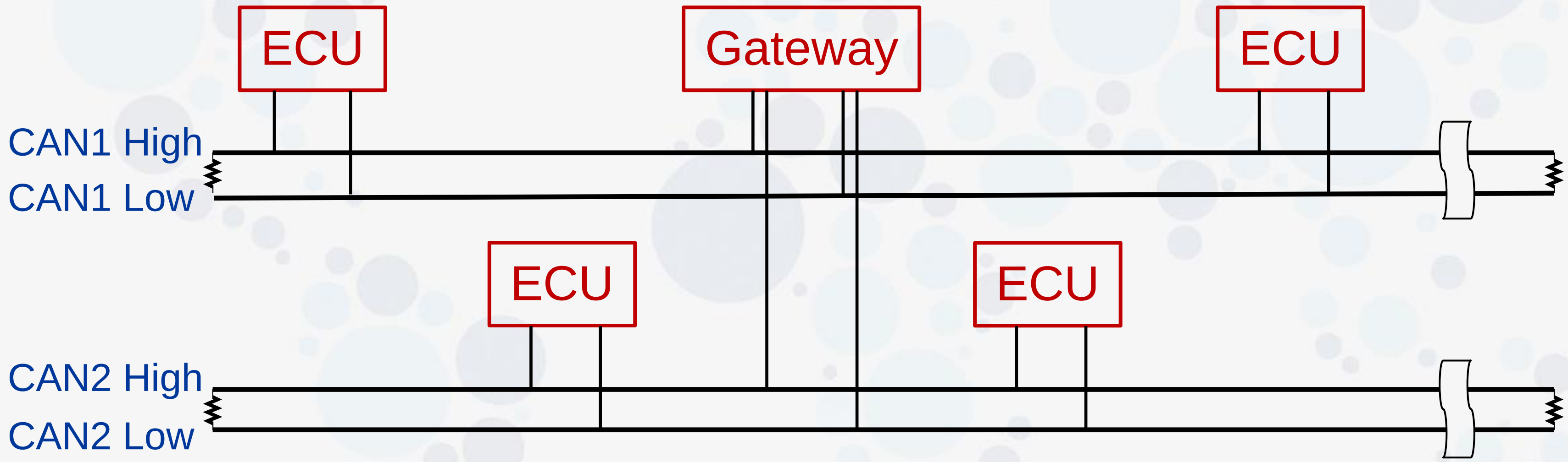
CAN architectures

- **Multiple separate buses**
  - Some ECUs have to be connected to multiple buses
  - They can be used to bypass the segmentation



CAN architectures

- **Multiple interconnected buses**
  - A gateway is routing frames between CAN buses
  - It may take into account the state of the vehicle
  - Both safety and cyber-security are considered



Crafting CAN attacks

- **Several attack vectors**
  - Misuse of intrinsic capabilities (e.g., remote diagnostic tool)
  - Exploit a higher-level parsing vulnerability
  - Break the Security Access challenge
  - Etc.
- **This will imply a substantial amount of work**
  - Unsolder EEPROM or identify on-chip debug (JTAG/BDM) and conventional debug (UART/WDBRPC) interfaces
  - Extract the firmware
  - Reverse-engineer the aforementioned items
  - Craft actual attacks

The Man in the Middle

- **Taking advantage of the client-server model**
  - Insert yourself in-between them
  - Do not alter traffic until you see something interesting
  - Then start to drop/alter/replay/…
  - Finalize with targeted reverse-engineering
- **In theory, this is transposable to the CAN bus**
  - We are auditing one device

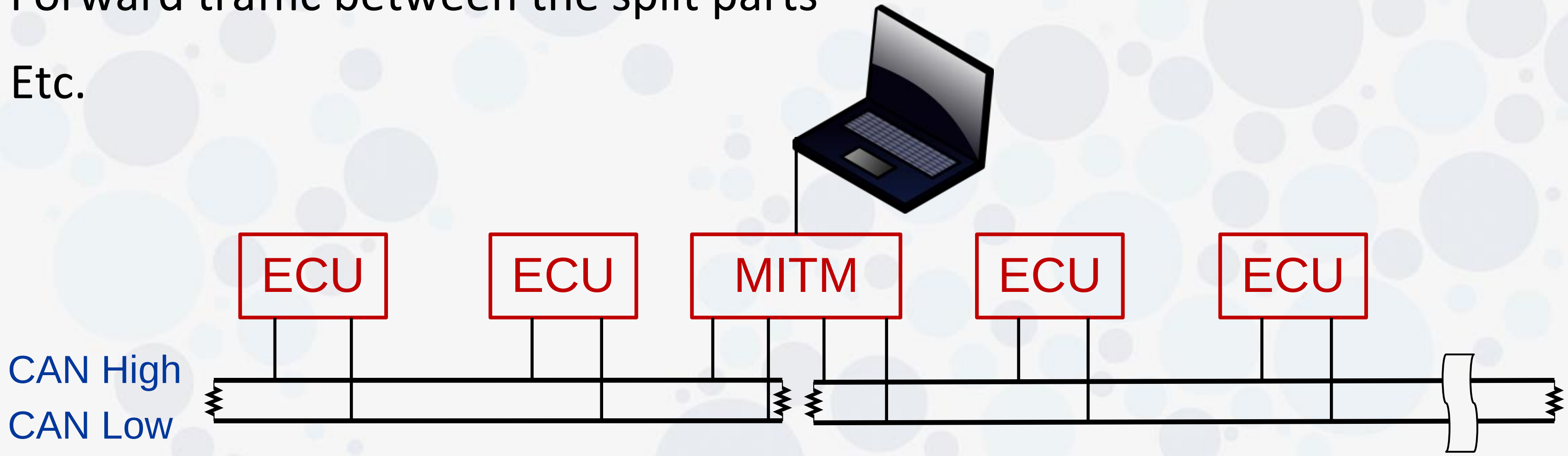
    → We could proxy the traffic from and to that device
  - We are working with the car manufacturer

    → We can ask for a restricted devices (e.g., a remote diagnostic tool)

    →This is limited by third-parties intellectual properties

However, in practice…

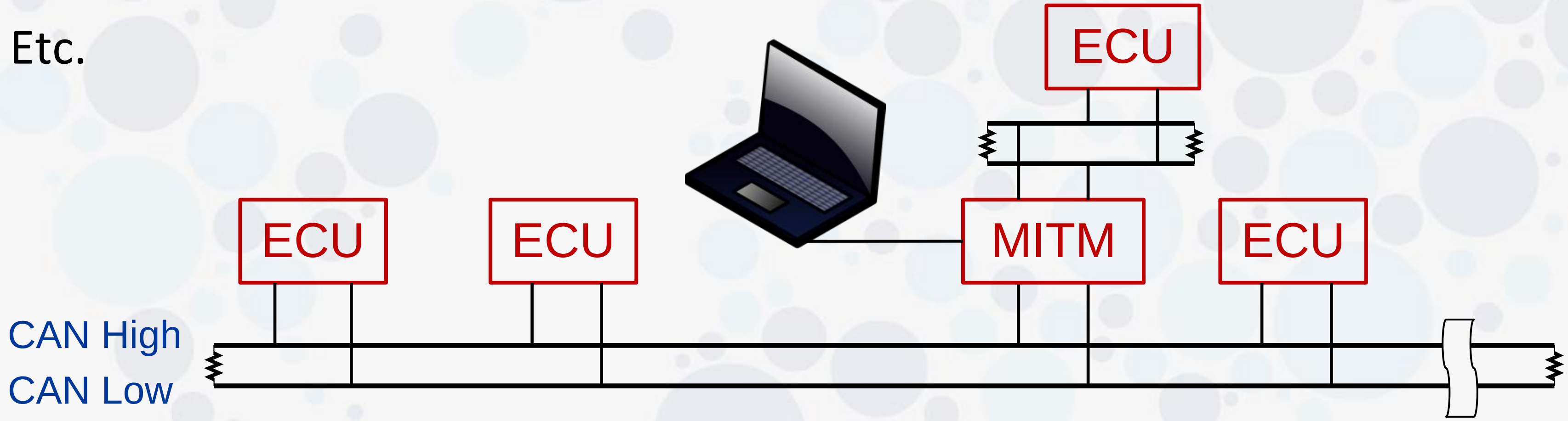
- **CAN is a multi-master serial bus**
  - Physically cut the bus and insert yourself in-between
  - Forward traffic between the split parts
  - Etc.



- **2 possible options (other than deep diving into the car)**
  - Emulate the car from the point of view of the audited device
  - Use an integration bench provided by the car manufacturer

However, in practice…

- **CAN is a multi-master serial bus**
  - Physically cut the bus and insert yourself in-between
  - Forward traffic between the split parts
  - Etc.



- **2 possible options (other than deep diving into the car)**
  - Emulate the car from the point of view of the audited device
  - Use an integration bench provided by the car manufacturer

What about existing tools ?

- **Only one interface to connect to CAN buses**
  - Bridging two devices could add a high latency
  - CAN was designed to meet deterministic timing constraints

What about existing tools ?

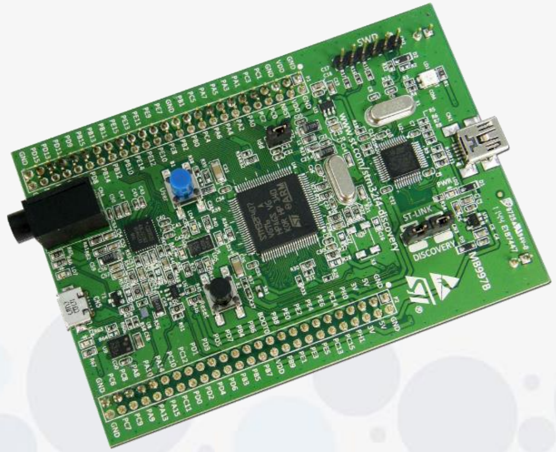
- **Only one interface to connect to CAN buses**
  - Bridging two devices could add a high latency
  - CAN was designed to meet deterministic timing constraints
- **Low-end FTDI chip to connect to a computer**
  - This is UART over USB at 115 200 bauds
  - CAN buses can go as far as 1Mbit/s
  - OBD-II is 250 or 500 kbit/s

What about existing tools ?

- **Only one interface to connect to CAN buses**
  - Bridging two devices will add a high latency
  - CAN was designed to meet deterministic timing constraints
- **Low-end FTDI chip to connect to a computer**
  - This is UART over USB at 115 200 bauds
  - CAN buses can go as far as 1Mbit/s
  - OBD-II is 250 or 500 kbit/s
- **Lack of a mature and powerful framework**
  - We get frustrated when we cannot use Scapy
  - Federate higher-layers reverse-engineering efforts

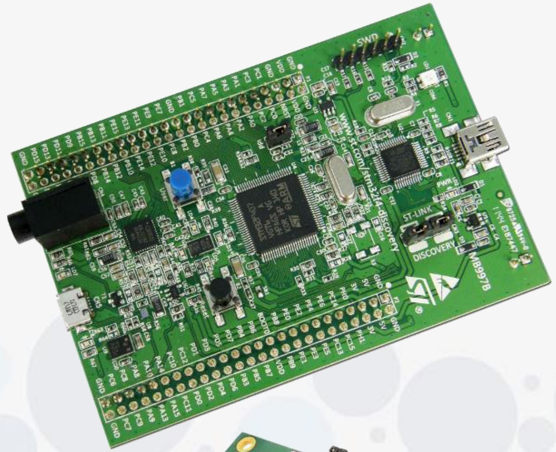
CANSPY hardware

- **STM32F4DISCOVERY board**
  - 168 MHz 32bit ARM Cortex M4
  - COTS ($20)



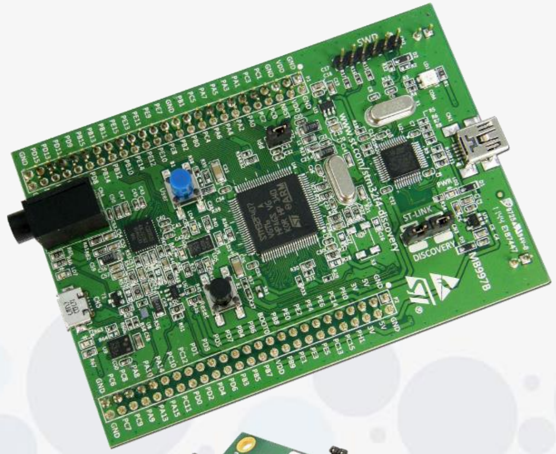
CANSPY hardware

- **STM32F4DISCOVERY board**
  - 168 MHz 32bit ARM Cortex M4
  - COTS ($20)
- **STM32F4DIS-BB extension board**
  - 1 RS232 interface
  - 1 Ethernet port
  - 1 SD card drive
  - COTS ($40)

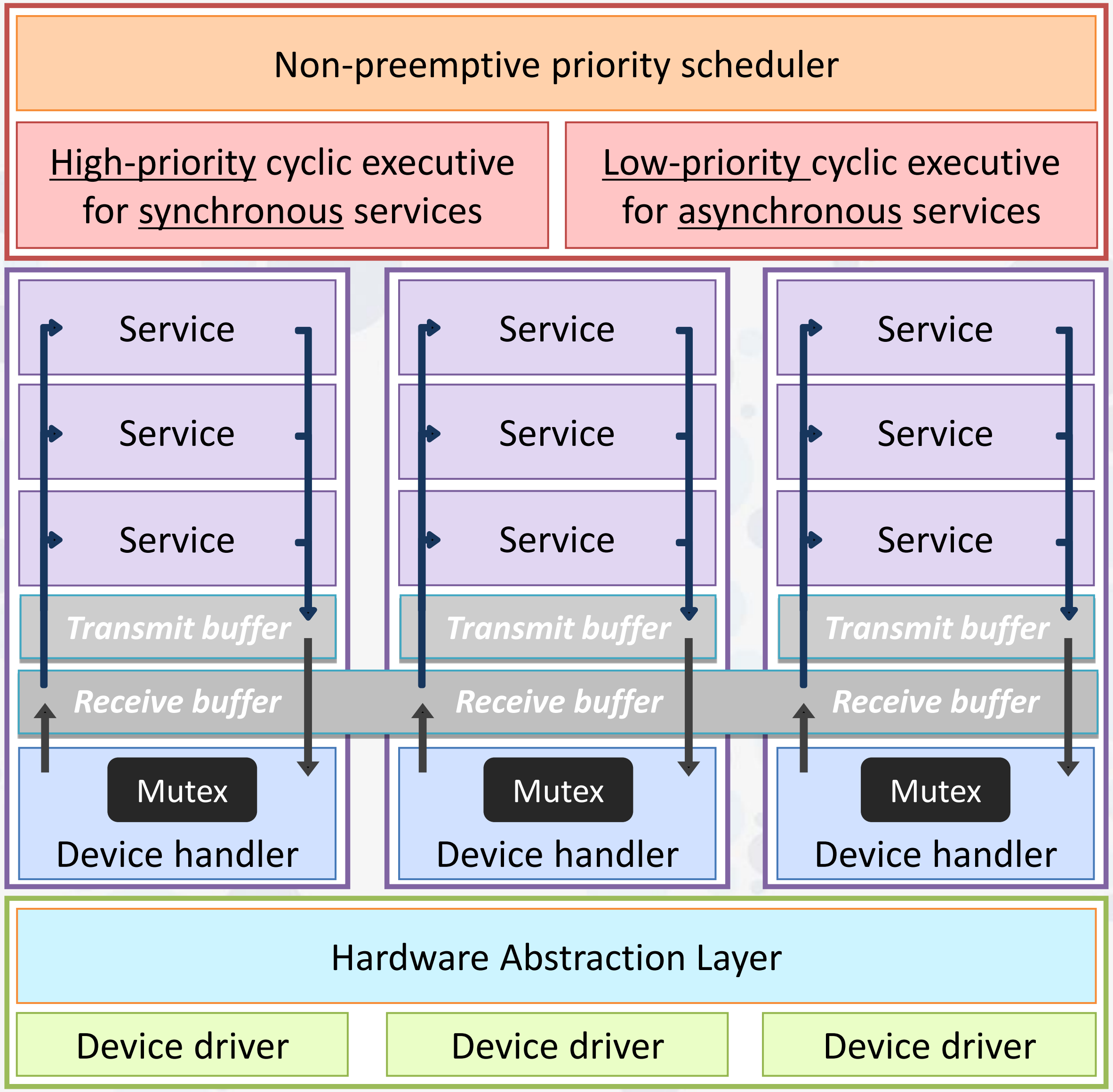


CANSPY hardware

- **STM32F4DISCOVERY board**
  - 168 MHz 32bit ARM Cortex M4
  - COTS ($20)
- **STM32F4DIS-BB extension board**
  - 1 RS232 interface
  - 1 Ethernet port
  - 1 SD card drive
  - COTS ($40)
- **DUAL-CAN extension board**
  - Configurable resistors, power supplies and circuit grounds
  - 2 CAN interfaces
  - Custom-made ($30 worth of PCB and components)



CANSPY firmware



CANSPY firmware

- **Event-driven scheduler**
  - Asynchronous I/O operations
  - Low latency processing
- **1 functionality == 1 service**
  - Start only what you need
  - Read from all devices, write to only one
  - Inter-service communication
  - Mutual exclusion is possible
- **Autonomous mode**
  - In-built filtering/altering engine
  - SD card for read or write operations
  - Power supply from the car battery
- **Open source licensed**
- **Several services**
  - CAN: Forward/Filter/Inject
  - Ethernet: Wiretap/Bridge
  - SDCard: Capture/Logdump
  - UART: Monitor/Logview/Shell
- **CAN devices**
  - 2 distinct devices
  - Support all standard speeds
  - Throttling mechanisms
    - Dummy frame injection
    - Delaying acknowledgments

CAN over Ethernet

- The SocketCAN format
- Ethertype 0x88b5
- Different MAC addresses
- Acknowledgments

CAN over Ethernet

- The SocketCAN format
- Ethertype 0x88b5
- Different MAC addresses
- Acknowledgments

```
class SocketCAN(Packet):
  name = "SocketCAN"
  fields_desc = [
    BitEnumField("EFF", 0, 1, {0:"Disabled", 1:"Enabled"}),
    BitEnumField("RTR", 0, 1, {0:"Disabled", 1:"Enabled"}),
    BitEnumField("ERR", 0, 1, {0:"Disabled", 1:"Enabled"}),
    XBitField("id", 1, 29),
    FieldLenField("dlc", None, length_of="data", fmt="B"),
    ByteField("__pad", 0),
    ByteField("__res0", 0),
    ByteField("__res1", 0),
    StrLenField("data", "", length_from = lambda pkt: pkt.dlc),
  ]
  def extract_padding(self, p):
    return "",p

bind_layers(Ether, SocketCAN, type=0x88b5)
```

CAN over Ethernet

- The SocketCAN format
- Ethertype 0x88b5
- Different MAC addresses
- Acknowledgments

```
#wireshark -X lua_script:ethcan.lua

local sll_tab =
DissectorTable.get("sll.ltype")
local can_hdl =
sll_tab:get_dissector(0x000C)
local eth_tab =
DissectorTable.get("ethertype")
eth_tab:add(0x88b5, can_hdl)
```

```
class SocketCAN(Packet):
  name = "SocketCAN"
  fields_desc = [
    BitEnumField("EFF", 0, 1, {0:"Disabled", 1:"Enabled"}),
    BitEnumField("RTR", 0, 1, {0:"Disabled", 1:"Enabled"}),
    BitEnumField("ERR", 0, 1, {0:"Disabled", 1:"Enabled"}),
    XBitField("id", 1, 29),
    FieldLenField("dlc", None, length_of="data", fmt="B"),
    ByteField("__pad", 0),
    ByteField("__res0", 0),
    ByteField("__res1", 0),
    StrLenField("data", "", length_from = lambda pkt: pkt.dlc),
  ]
  def extract_padding(self, p):
    return "",p

bind_layers(Ether, SocketCAN, type=0x88b5)
```

The OBD-II use case

- **No need to physically cut anything**
  - Buy a Goodthopter-compatible OBDII-to-DB9 cable
  - Build its female counterpart ($10 worth of components)
  - Setup the DUAL-CAN extension properly
  - Have fun
- **Several interesting cases**
  - Professional/consumer car diagnostic tools
  - Usage-based policies from insurance companies
  - Air-pollution control from law enforcement
- **They expose sensitive networks/hosts**
- **Demonstration**

Thank you for your attention