Linux in the connected car platform

Background

- Long time desktop Linux user
- Designed several capes for the BeagleBone Black
- Currently an Embedded engineer for Dialexa

What is a connected car anyway?

- A physical device, either original equipment or aftermarket, that provides data about your car to a web service.
- Can also allow remote control of some aspects of your car, such as starting the car, controlling the locks, and more.

Connected cars now

- Connected cars now are where cell phones were back in the late 90s. Proprietary operating systems, archaic interfaces, and little cross platform standardization
- This is set to change rapidly as connected car hardware becomes commoditized, just as happened with cell phones

Challenges

- Security of OBDII bus
- Firmware updates
- Power consumption
- Reliability
- Boot time


DON'T WORRY
PROGRESSIVE
ITS SAFE
memegenerator.net

Security

- Automakers have historically relied on security through obscurity
- OBDII defines no means of securing access to the car’s network
- Devices already embedded in the car could be compromised

Security

- J1850 and older ISO protocols communicate only with the ECU
  - Still potentially an attack vector, but not as bad as CAN bus
- More recent cars are partitioning critical systems onto a separate bus, but there are still devices which bridge between the two networks

Security

- Legislators are beginning to pay attention to the area of vehicle security [2]
- Hackers are beginning to pay very close attention to the area of vehicle security [3]

CAN bus

- CAN became the standard in 2003 and is now the required protocol
- The standard for outside interfacing to cars is ISO11898-2, which is high speed CAN
- High speed can is limited to a linear bus structure

CAN bus

- Being a two wire differential protocol, if you can read you can write
- A malicious device on the high speed CAN bus can halt communications, or fake transmissions from critical sensors

Firmware updates

- Vulnerabilities are being announced at an ever increasing rate
- Updating the device firmware is even more important in a critical application such as this
- A firmware update is useless if the process can be subverted to load malicious code

Firmware updates

- The Progressive device was hacked by someone emulating a cell tower. What seems like a great deal of effort to hijack firmware is not as difficult as we may believe

Power

- There is no way to detect that a car is in the Accessory state from the OBD port
- Accidentally waking from sleep and staying on is unacceptable, nobody wants to come back to a flat battery

Power

- Reading the voltage present on the OBD port is dicey, you aren’t guaranteed anything about the car’s battery chemistry or setup, you would just be making assumptions

Reliability

- People treat their cars, largely, as an appliance
- If it doesn’t work, they aren’t interested in troubleshooting, they just need to get it to a working state. They will remove your device if they think it causes a problem

Boot time

- People want starting the car to be a seamless experience, everything starts up together
- Boot time is difficult to minimize on embedded systems due to slow flash and lack of suspend

How does Linux solve these challenges?

- It doesn’t. Linux is a kernel. The system you design must solve these challenges.
- Linux is, however, a powerful tool to help you solve them.

Do one thing and do it well

- The tendency to want to have one system do everything is dangerous in this context
- One of Linux's greatest tools is that it integrates well into a larger system
- This flexibility lets you pick the best solutions, as opposed to the ones your vendor supports

How we approached the problem

vinli™

Improving Security - Define the threat

- “Everything” is not a valid answer
- Neither is “the bad guys”
- Physical attacks against the device are very difficult to mitigate in a consumer setting, but also are far more rare than remote attacks over a network interface

Improving Security

- Even then, just as a lock on the door of your house will not stop a determined attacker, neither will our security measures guarantee that the device cannot be broken
- Our best course is then to limit the amount of damage that can be done if someone does gain control of a device with a network interface

Separation of responsibilities

- An STM32F1 is responsible for communication with the car. It streams data back to the rest of the system via UART and receives single byte commands only.
- The programing pins are not exposed, its firmware is sealed at manufacture.

Separation of responsibilities

- Bluetooth communications with a phone are handled with a CC2451 which sits in line with the Linux SoM and the STM32. It retransmits messages going in both directions, and send the data from the STM32 to the phone

Separation of responsibilities

- The Linux SoM is an AR9331 based module with 64M of RAM and 16M of Flash
- Kernel version 3.10
- Provides a WiFi hotspot via a USB LTE module, and handles packaging data to send to the back end web service

The Firmware

- The STM32 runs a custom OBDII stack capable of any standard supported by the OBDII standard
- The CC2541 has a minimal firmware to pass data back and forth, and populate a GATT database for collection by a phone
- The firmware on both of these are fixed at manufacture

Choice of distribution

- OpenWRT has mainline support not only for the AR9331 but for our particular SoM
- Mainline support is huge
- Evil vendor trees are just that, evil

Package updates on the SoM

- In 16M of flash there wasn’t enough room for us to update the entire OS image over the air while having a fallback image, so the kernel is fixed at manufacture
- The packages, however, can be updated
- Packages are signed by us, and updates are hosted on our server

Securing back end communications

- Each device is issued a private key at manufacture, the public key is sent to the back end along with the device ID
- Each message is signed when it is sent, inside of an HTTPS connection with certificate verification

Power management

- Handled by the STM32
- Successful communication with the ECU will start up the CC2541 and the SoM
- Loss of communication will send a graceful shutdown message to the rest of the system before the STM32 shuts off power and goes into a sleep cycle itself

Boot time

- Suspend to disk is a huge help here, but unfortunately there is not enough space
- Suspend to ram is not supported on many embedded processors
- Boot time is more than just kernel and userspace, the LTE module takes time to come up as well

Boot time

- OBDII interface and BLE are on almost immediately, SoM boot time is about 30 seconds
- OpenWRT uses plain init scripts, optimizations have been to bring the LTE interface up as fast as possible

Conclusions

- Security is paramount. All the nifty features in the world won’t make up for damaging someone’s car or worse.
- Linux won’t solve your problems, but it will be a powerful tool in solving them.
- Connected cars are another opportunity to show the flexibility of Linux based solutions.

Questions

More info

- Slides at voidptr.org
- @carpman on twitter, Daniel Jackson on google+ (for as long as Google lets it live)