Remote Vehicle Interaction

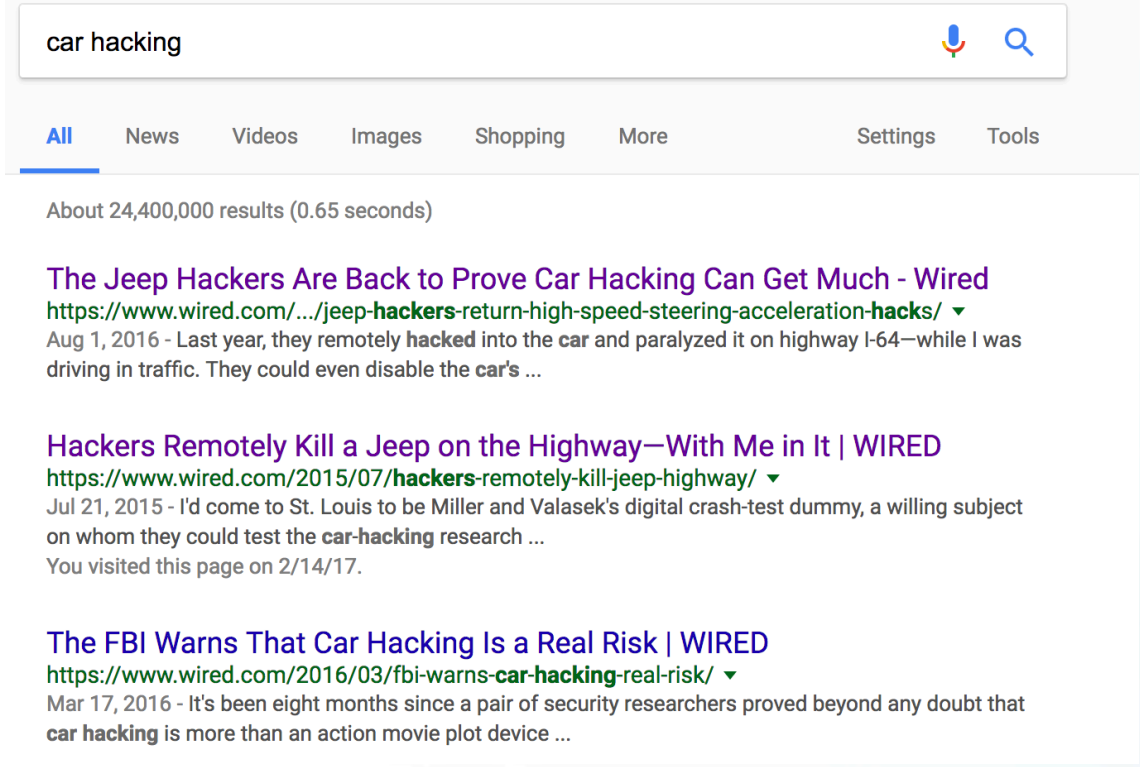
February 23, 2017 | Securing the Connected Car

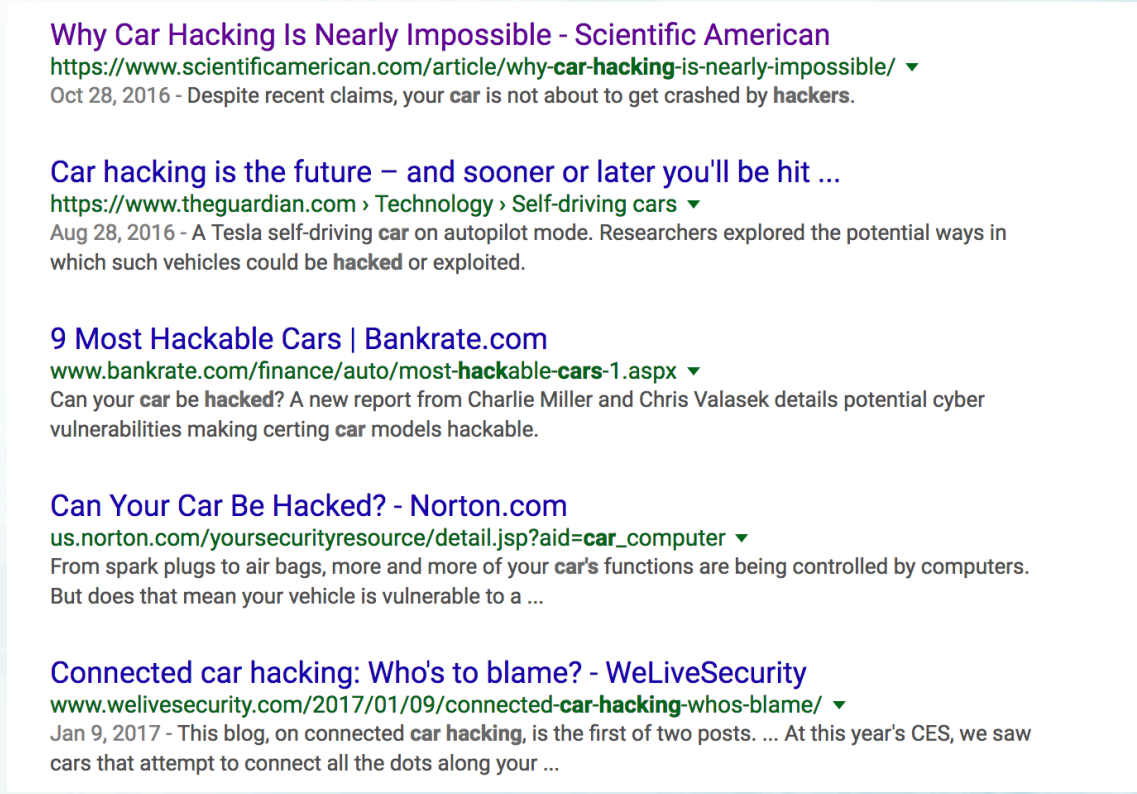
Tatiana Jamison

Open Source Software Architect, Jaguar Land Rover, GENIVI Alliance

Connected cars may be vulnerable

Everyone’s talking about car hacking…





GENIVI®

Chrysler recalls 1.4M cars after Jeep exploit

- Miller & Valasek discovered exploit via cellular network
- Remote control of critical systems, including brakes
- One scan found over 2,500 vehicles

Photo credit: Andy Greenberg, Wired

GENIVI®

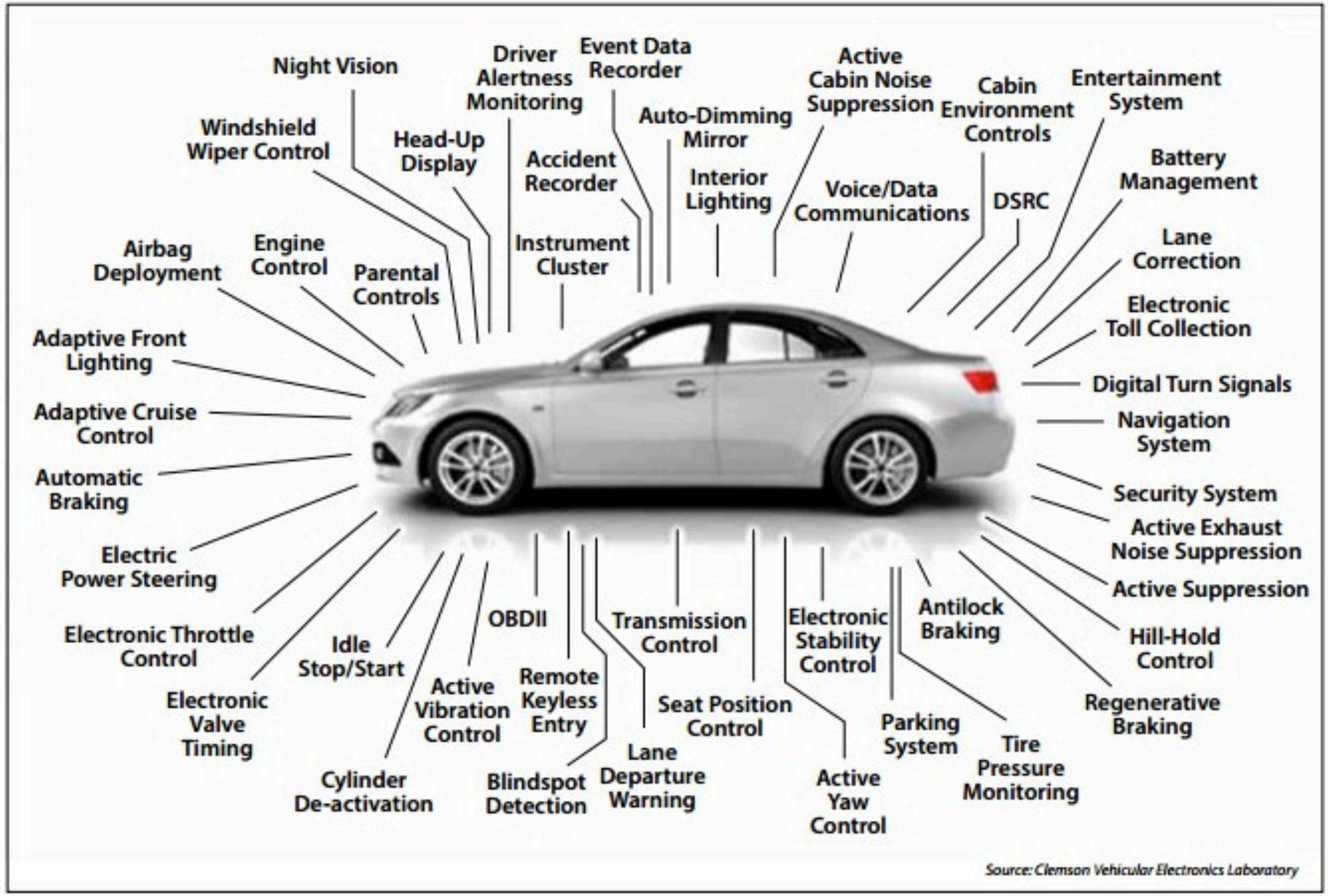
Tesla hit by malicious wifi exploits in 2016

- Sep: Keen Security Lab announces remote exploit via in-car browser
- Nov: Promon AS announces remote exploit via Android app

Photo credit: Darrell Etherington, TechCrunch

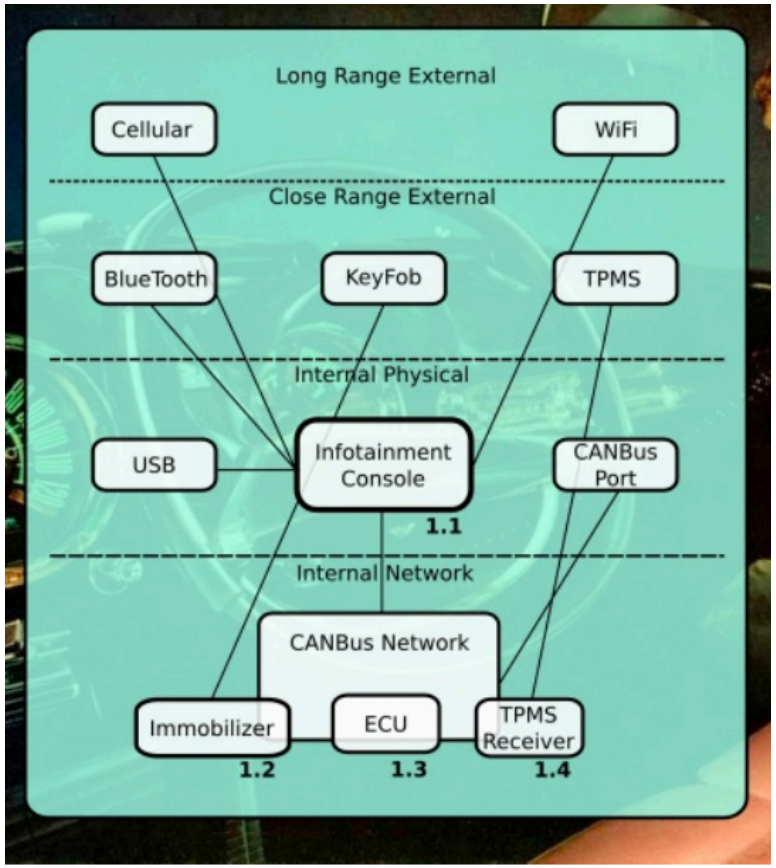
GENIVI®

Automotive software architecture is complex


Night Vision
Driver Alertness Monitoring
Event Data Recorder
Active Cabin Noise Suppression
Cabin Environment Controls
Entertainment System
Windshield Wiper Control
Head-Up Display
Auto-Dimming Mirror
Accident Recorder
Interior Lighting
Voice/Data Communications
DSRC
Battery Management
Airbag Deployment
Engine Control
Parental Controls
Instrument Cluster
Lane Correction
Electronic Toll Collection
Adaptive Front Lighting
Digital Turn Signals
Adaptive Cruise Control
Navigation System
Automatic Braking
Security System
Active Exhaust Noise Suppression
Electric Power Steering
Active Suppression
Electronic Throttle Control
OBDII
Transmission Control
Electronic Stability Control
Antilock Braking
Hill-Hold Control
Idle Stop/Start
Remote Keyless Entry
Seat Position Control
Regenerative Braking
Electronic Valve Timing
Active Vibration Control
Parking System
Tire Pressure Monitoring
Cylinder De-activation
Blindspot Detection
Lane Departure Warning
Active Yaw Control
Source: Clemson Vehicular Electronics Laboratory

GENIVI®

Add external sources…



… and it just gets more complex

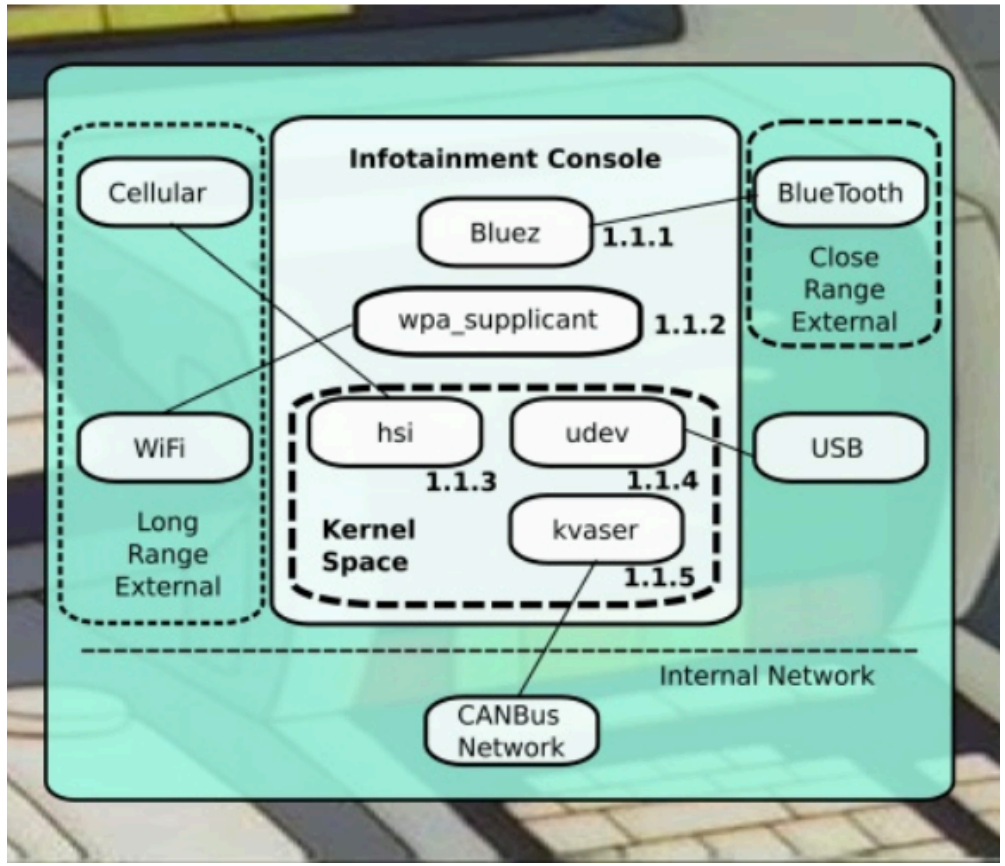
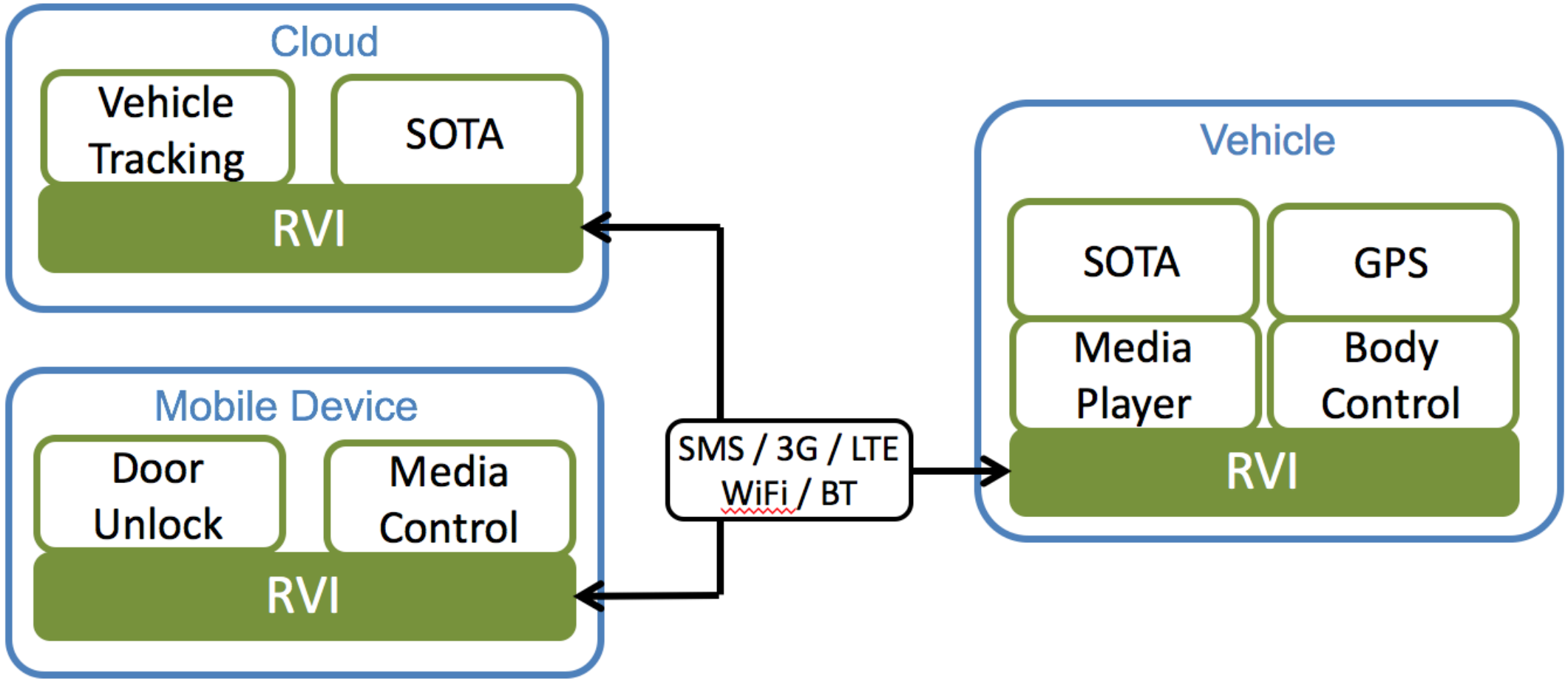


Photo credit:
Craig Smith, “The Car Hacker’s Handbook”, 2014

GENIVI is standardizing how cars connect securely to remote devices

GENIVI®

RVI is middleware for service-oriented arch



GENIVI®

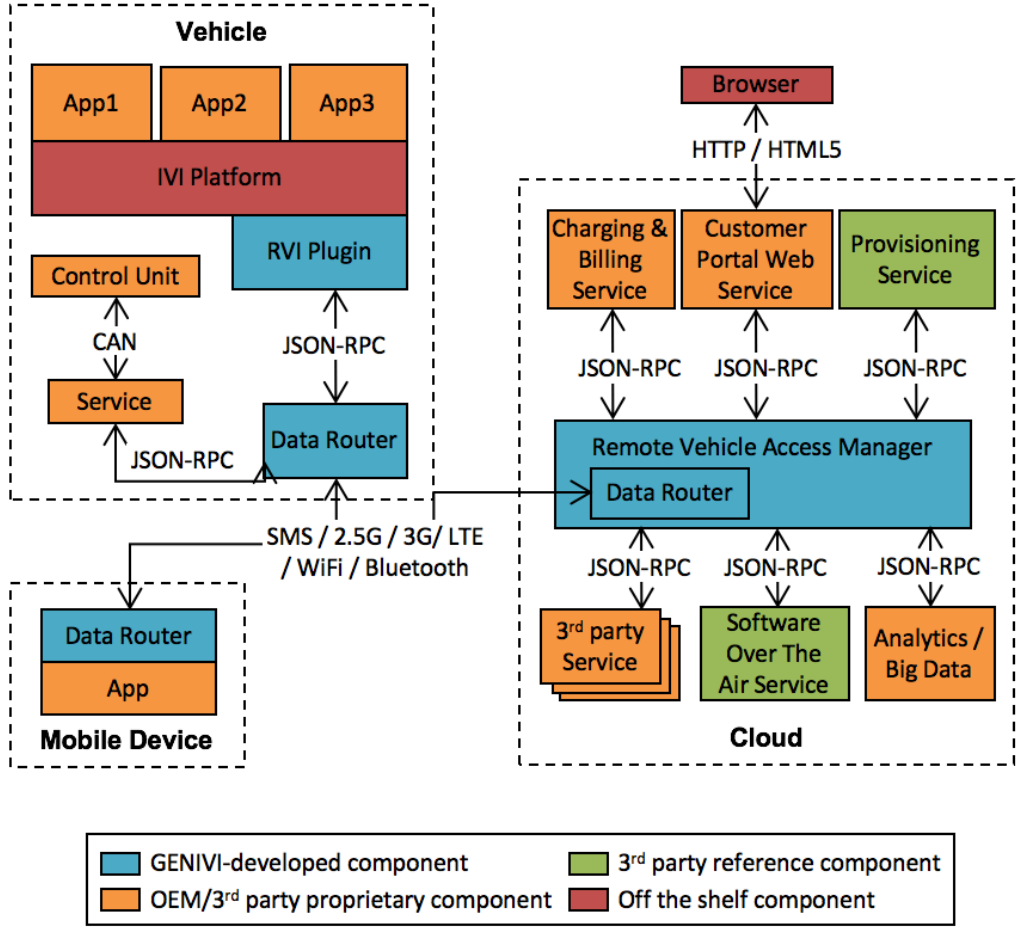
RVI Architecture Overview

API based
The API is the driving technology. Implementation is secondary.

Data Router commonality
Data Router connects all services on all devices.

Mix of open and closed source
Components can be off the shelf, OSS, proprietary, or a combination of the above.

Network complexity shielding
A clean transaction API alleviates services and applications from connectivity concerns.



GENIVI®

RVI has been implemented in several ways

- Proof of Concept implementations exist for:
  - Erlang: cross-platform executable and message bus
  - Objective-C: iOS SDK
  - Java: Android SDK
  - C: cross-platform library
- All implementations are available on GitHub:
  - https://www.github.com/GENIVI?q=rvi

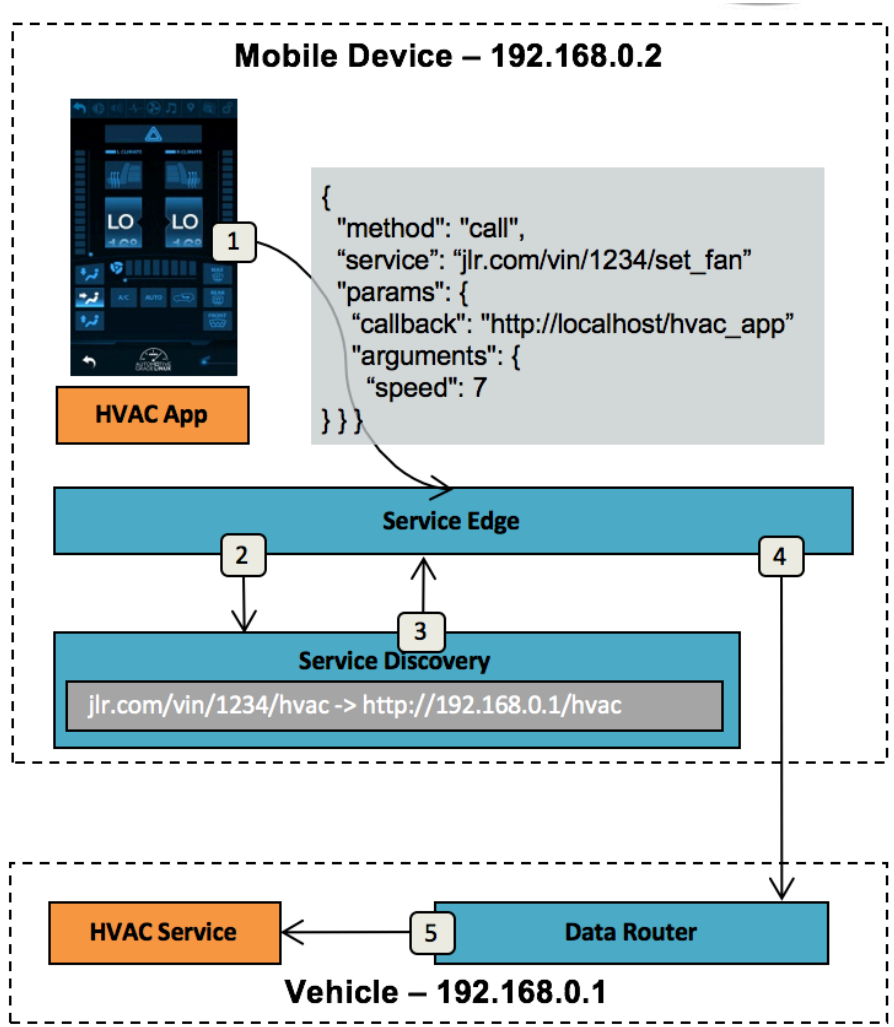
Why RVI?

- Completely open source
- MPL 2.0 licensing supports commercial integration
- Reference implementations exist for Software, Firmware Over The Air (SOTA/FOTA) and Big Data
- Demos for HVAC, and Mobile Unlock

GENIVI®

How does an app developer work with RVI?

1. **Mobile Application Sends Command**
   HVAC App sends an message, targeting a given service URI, to Service Edge.
2. **Locate Target Node**
   Service Edge asks local service Service Discovery to resolve service name to a network address.
3. **Return Network Address**
   Specifies where the target service can be reached.
4. **Send Request to Vehicle**
   The vehicle data router processes the command.
5. **Forward Request to HVAC Service**
   The HVAC Service in the vehicle executes the command.



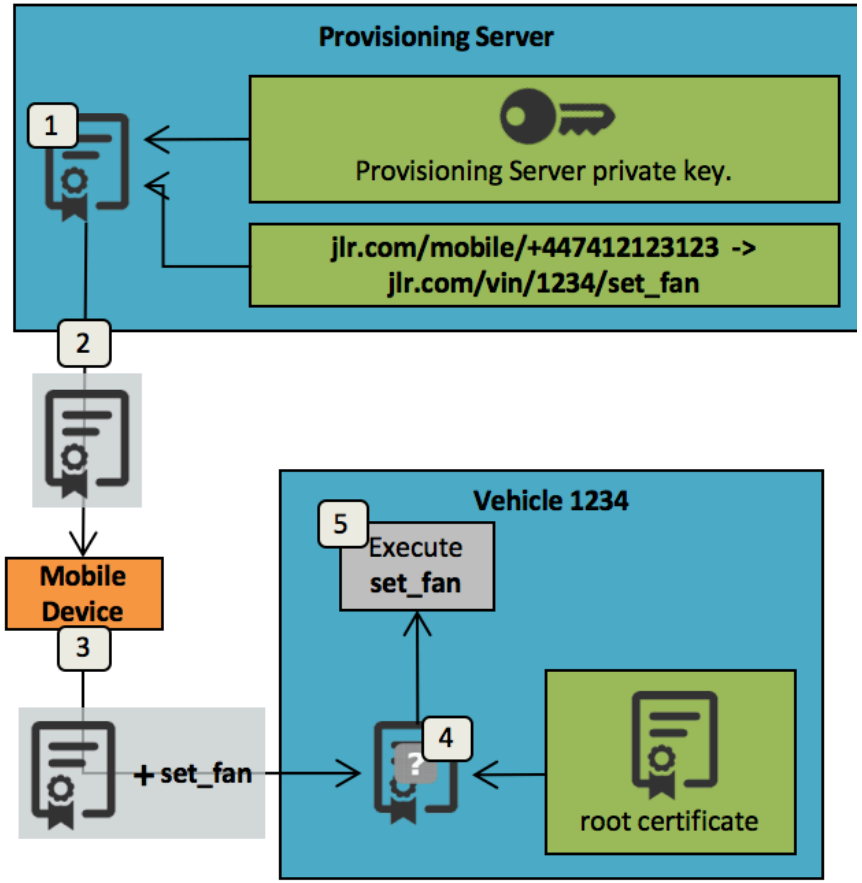
GENIVI®

What security features are present in RVI?

- Require TLS/DTLS v1.2 or higher to secure connections
- Asymmetric cryptography with Public Key Infrastructure
- Access controls are self-carried in JSON Web Tokens signed by Root of Trust to safeguard against tampering
- Access control checked before sending and upon receipt

Authorization - overview

1. **Create, sign credential (JSON Web Token)**
   A JWT granting access to the mobile device is created and signed with provisioning server's private key.
2. **Distribute credential to mobile device**
   The targeted device receives its certificate
3. **Mobile sends request, credential to Vehicle**
   The credential states that mobile device has the right to execut the given request
4. **Validate credentials**
   The JWT and request is validated by the vehicle using the public key of the trusted provisioning server
5. **Execute request**
   The validated command is forwarded to the target service for execution



GENIVI®

What's next for RVI?

- Work to extend & mature proof of concepts continues
- GENIVI project to field test RVI in smart city pilot
- Big Data demos and IoT integration
- Visit GitHub repos to give it a try – no car required!

GENIVI®

Links

- C Proof of Concept: https://github.com/GENIVI/rvi_lib
- Erlang POC: https://github.com/GENIVI/rvi_core
- Mobile: https://github.com/PDXostc/rvi_core_android

  https://github.com/PDXostc/rvi_core_ios

GENIVI Projects: http://projects.genivi.org/

GENIVI®

Further Reading on Automotive Security

- Craig Smith, “The Car Hacker’s Handbook” http://opengarages.org/handbook/
- Dr. Charlie Miller, Chris Valasek, “Remote Exploitation of an Unaltered Passenger Vehicle” http://illmatics.com/Remote%20Car%20Hacking.pdf
- Lee Pike, et al, “Securing the Automobile: A Comprehensive Approach” http://www.galois.com/~leepike/pike-car-security.pdf
- Got a recommendation? Email me: tjamison@jaguarlandrover.com

GENIVI®

Thank you!

Tatiana Jamison: tjamison@jaguarlandrover.com

Visit GENIVI at http://www.genivi.org or http://projects.genivi.org

Contact us: help@genivi.org

GENIVI®