Linux-Stack Based V2X Framework: All You Need to Hack Connected Vehicles

Duncan Woodbury, Nicholas Haltmeyer
{p3n3troot0r@protonmail.com, ginsback@protonmail.com}

July 29, 2017
State of the World: (Semi)Autonomous Driving Technologies

- Vehicular automation widespread in global industry
- Automated driving technologies becoming accessible to general public

- Comms protocols used today in vehicular networks heavily flawed
- New automated technologies still using CANBUS and derivatives
Stages of Autonomy

- Today: Stage 2 Autonomy - Combined Function Automation

- V2X: Stage 3 Autonomy - Combined Function Automation
  ⇒ Leverage vehicular ad hoc mesh network for exchange of safety and actor/world state information
Critical Aspects of V2X

- Reliable high-throughput ad hoc mesh networking and exchange in a real-time cyberphysical environment
- Standardization of Stage 3 automation in federal and consumer transportation systems
- Enhanced safety and traffic optimization technologies leveraging V2X
Technologies Using V2X

- Collision avoidance (Forward Collision Warning) systems
- Advanced Driver Assistance Systems (ADAS)

- Cooperative adaptive cruise control
- Automated ticketing and tolling
Impact of V2X: Why Care?

- Most importantly: self-driving cars are shiny
- Your children will (would) ride in these
- Enables safety functions not possible with onboard systems
- V2X technologies applicable across range of cyberphysical systems
- NHTSA V2V NPRM: V2V to be standardized in light vehicles
- It’s happening: You want in?
Tangible Benefits

According to the USDOT,

- **Safety**
  - Prevent 25,000 to 592,000 crashes annually
  - Save 49 to 1,083 lives
  - Avoid 11,000 to 270,000 injuries
  - Prevent 31,000 to 728,000 property damaging crashes

- **Travel time**
  - 27% reduction for freight
  - 23% reduction for emergency vehicles
  - 42% reduction on freeway (with cooperative adaptive cruise control & speed harmonization)
Impact on Automotive Security

- Huge attack surface for car hacking
- Responsible for governing and optimizing traffic flow
- Attacks propagate within the mesh network
  - Hack one car to hack the whole road
  - 1609.2 PKI incomplete - proposed solutions not scalable

- Will drive adoption and development of autonomous vehicles
- Homogeneous use of WAVE and V2X enables unprecedented complexity in transportation systems
V2X Protocol Stack

Management Plane

Security Services

Layer-specific management entities

Data Plane

UDP / TCP

IPv6

LLC

WAVE MAC (including channel coordination)

PHY

WSMP

Figure 1—WAVE reference model
IEEE 802.11p

Wireless Access in Vehicular Environments

- Amendment to IEEE 802.11-2012 to support WAVE/DSRC
- PHY layer of V2X stack
- No association, no authentication, Wildcard BSSID = {ff:ff:ff:ff:ff:ff}
- 5.8-5.9GHz OFDM
IEEE 1609

WAVE Short Message Protocol (WSMP)

- 1609.2 Security Services
  - PKI, cert revocation, misbehavior reporting
- 1609.3 Networking Services
  - Advertisements, message fields
- 1609.4 Multi-Channel Operation
  - Channel sync, MLMEX
- 1609.12 Identifier Allocations
  - Provider service IDs
IEEE 1609.3: WSM

Packets encoded as WAVE Short Messages (WSMs)
## IEEE 1609.3: WSM Example

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Length (octets)</th>
<th>Value (hex)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subtype, Option Indicator, WSMP Version</td>
<td>1</td>
<td>08</td>
<td>Subtype = 0 (4 bits) Option Ind = 1 (1 bit) Version = 3 (3 bits)</td>
</tr>
<tr>
<td><strong>COUNT</strong></td>
<td>1</td>
<td>03</td>
<td>Info Elem Count = 3</td>
</tr>
<tr>
<td>Info Element 1</td>
<td>Channel Number</td>
<td>3</td>
<td>0F 01 AC</td>
</tr>
<tr>
<td>Info Element 2</td>
<td>Data Rate</td>
<td>3</td>
<td>10 01 0C</td>
</tr>
<tr>
<td>Info Element 3</td>
<td>Transmit Power Used</td>
<td>3</td>
<td>04 01 9E</td>
</tr>
<tr>
<td><strong>TPID</strong></td>
<td>1</td>
<td>00</td>
<td>Address Info (PSID) only, no Info Elem Ext field present.</td>
</tr>
<tr>
<td><strong>WMP-T-Header</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provider Service Identifier</td>
<td>3</td>
<td>C0 03 05</td>
<td>PSID: 0pC0-03-05</td>
</tr>
<tr>
<td><strong>WSM Length</strong></td>
<td>1</td>
<td>0D</td>
<td>Length = 13 Length of WSM Data</td>
</tr>
<tr>
<td><strong>WMP-Data</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSM Data</td>
<td>13</td>
<td>48 55 6C 6C 6F 20 57 6F 72 6C 64 21 00</td>
<td>ASCII content: ‘Hello World!’ 0x48 = H 0x55 = E 0x6C = l etc.</td>
</tr>
</tbody>
</table>
Participants broadcast WAVE Service Advertisements (WSAs)
Control and service channels (CCH and SCH)
- DSRC message set and data elements
- ASN1 UPER encoding (latest rev does not compile)
- Basic Safety Message (BSM), Emergency Vehicle Alert (EVA), etc.
Subtleties in Protocol Specifications

- Ambiguous parse rules for certain frames (Information Element Extension)
- Services are gated only by PKI permissions
  - Proprietary applications using ad hoc permissions
- Messages leak S/PII
  - Misbehavior reporting just randomizes the send address in an attempt at privacy
  - WSA fingerprinting prevents privacy
- Channel switching for single-antenna systems desync
  - >0.1s delay with 200 cars/km²
IEEE 802.11-2012 details 802.11p
  - Not supported by majority of COTS WiFi hardware
IEEE 1609.\{3,4\} stable, under development
IEEE 1609.2 (PKI, misbehavior reporting) incomplete, under development
SAE J2735 stable, under development
  - SAE J2735 ASN1 (2016) not stable
  - Example of PKI Brilliance: ‘Another aspect of the privacy and non-tracking of vehicles becomes apparent here as the MAC address needs to be randomly changed to prevent vehicle tracking.’
Possibly Unintentional Obfuscation of the Standards

- Removal of message CRC from J2735
- Continued revisions would likely make in-the-field devices obsolete
  - WSMPv3 (current) has no backwards compatibility
- Standards vague in best practices, favoring proprietary implementations

Consider:

7.3.3.4 Effect of receipt

No behavior is specified.
Major Changes to the Standards

- Standards still in development after decades
  - Spectrum allocated by the FCC in 1999
  - WAVE first codified in 2005
  - J2735 in 2006
- Rewrites of security services to change certificate structure
- Rewrites of management plane to add services (P2PCD)
- Incomplete safety message dictionary
- No standards for application-layer services
Physical Manifestations of V2X: Deployment

Three USDOT pilot studies: NYC, Tampa (THEA), Wyoming
V2X in the Linux Networking Subsystem
802.11p: Driver and Kernel Tree Modifications

- **WiFi driver modifications:**
  - Add support for ITS-G5 channel spectrum, 5/10MHz-width channels
  - Add support and error checking for OCB mode
  - Force usage of user-specified regulatory domain

- **/net/wireless modifications:**
  - Add support wildcard broadcast transmission
  - Add support for 5/10MHz-width channels, channel state definitions for OCB mode
  - Force usage of user-specified regulatory domain

- **mac80211 modifications:**
  - Add (fix) support and error checking for OCB mode
    - Initialization and de-initialization
    - Channel filter configuration, disable beaconing and association

- **cfg80211 modifications:**
  - Channel filter configuration for OCB mode
  - Add support for 5/10MHz-width channels

- **nl80211 modifications:**
  - Channel filter configuration for OCB mode
  - Add support for 5/10MHz-width channels
IEEE 1609: 1609 in the Linux Kernel

Kernel module to pack, parse, and broadcast messages

- Relevant data structures
  - WSM, WSA, WRA, SII, CII, IEX
- Full control of fields
  - subtype, TPID, PSID, chan, tx power, data rate, location, etc.
  - Operating modes for setting degree of compliance to standard (strict, lax, loose)
- Channel sync, dispatch
- Netlink socket interface to userspace (af_wsmp)
Error Checking and Corrections Implemented

- Parser short circuiting
- Domain checks on each field
- Operating modes for standard compliance
  - Will reject messages where domain is non-compliant
- Relevant error handling (EBADMSG, EINVAL, EFAULT, etc.)
SAE J2735: Userspace J2735 Message Dictionary

- socketv2v utility suite: v2vsend, v2vdump, v2vsniffer
- Fully implements BSM, EVA, RSA, CSR J2735 message formats
- Communicates with 1609 kernel module via Netlink socket
- Enables VANET participation with generic Linux box and 5GHz WiFi
Platform Requirements: Linux!

- V2X stack integrated in mainline Linux kernel
  - No proprietary DSRC hardware/software required
  - V2X stack current - deployed V2X ‘solutions’ obsolete
- Currently supports ath9k/ath9k_htc, rtlwifi
- Fully implements 802.11p, IEEE 1609.\{3,4\} in Linux networking subsystem
- IEEE 1609.2 to be integrated upon completion
Capabilities Leveraging V2X Stack: Hacking Connected Vehicles

- Rapidly prototype new V2V applications
- Penetrate commercial implementations
- Analyze real V2V network data
  - Pilot studies, protocol analysis
Developing Connected Vehicle Technologies

- Widespread access enables engagement of security (1337) community in standards development
  - History lesson: CANBUS sucks (for automotive)
- Interact with existing V2X infrastructure
  - Pressure manufacturers and OEMs to implement functional V2V
- Deploy ahead of market - experimental platforms
  - UAS, maritime, orbital, heavy vehicles
- Opportunity for empirical research: See what you can break
  - Straightforward to wardrive
  - Hook DIY radio (Pi Zero with 5GHz USB adapter) into CANBUS (for science ONLY)
(You can) Use J2735 DSRC over 802.11p with Linux

- Participate in connected VANETS
  - v2vsend: Craft and inject messages into ITS spectrum
  - v2vsniffer: Sniff particular messages from specific actors
  - v2vdump: Sniff all communications on ITS channels
DSRC Sniffing/Wardriving

- DSRC is a broadcast protocol
- Dimensions, directionality, orientation, acceleration, display etc. provide means to easily fingerprint and track participants
- From the SAE DSRC Implementation Guide:
  - ‘The VII Probe Data Service collects anonymous probe data from all Mobile Users, and distributes it to any authorized Network User or Roadside Infrastructure User that requests it.’
  - ‘Applications are initialized by matching the locally registered AID with an advertised AID (application announcement) received on the radio link’
- Highly distributed infrastructure - attacks propagate across the network easily
Understanding the Adversary

- **Passive**
  - Determine trajectory of cars within some radius
    - Few stations required to monitor a typical highway
  - Determine services provided by peers
  - Characterize network traffic for regions of the road
  - Uniquely fingerprint hardware being used
    - RF signature
    - Probe responses

- **Active**
  - Deny service
  - Manipulate misbehavior reports
  - Exploit bad hardware/software to access CANBUS
    - Different regional configurations can lead to undefined behavior
  - Disrupt vehicle traffic
  - Parade as a moving toll station
    - Ad hoc PKI for application-layer services
Hacking ITS

- Level 1: Denial of Service
  - Single-antenna DSRC systems susceptible to collision attack
- Level 2: DSRC spectrum sweep, enumerate proprietary (custom) services available per participant
- Level 3: Impersonate an emergency vehicle
- Level 4: Become mobile tollbooth
- Level 1337: Remotely execute platooning service
  - Assume direct control
Protocol Exploitation

Use design flaws in the VANET to create rapidly propagating effects

- Privacy mitigations put in as an afterthought
- PKI/trust management doesn’t scale
- XML-driven J2735 safety message dictionary
- Any RSU deployed is a single point of failure for the region
Global Access to V2X

- Provides vehicle to streamline testing and development of V2X
- Mainline Linux kernel integration \( \equiv \) V2X stack easily applied to UAS, maritime, heavy truck, communications systems, etc.
- V2X now tangible, scalable, accessible: Now it is up to us to fix it!
What: V2X in Your Hands

- Shape the future development of ITS
  - Fix/mitigate systemic issues in VANET security
- Push toward free as in freedom solutions
- Reduce global attack surface through engineering of good standards

Engage and participate in public VANET (hack the planet)
Acknowledgments
References

- Check me out on github: https://github.com/p3n3troot0r/Mainline-80211P


