Linux-Stack Based V2X Framework: SocketV2V
All You Need to Hack Connected Vehicles

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

DEFCON 25: 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 - Limited Self-Driving Automation
Barriers to Stage 3+ Autonomy

- Ownership of ethical responsibilities - reacting to safety-critical events
- Technological infrastructure
  - Installing roadside units, data centers, etc.
- Adaptive and intuitive machine-learning technology
V2X Concept

- Vehicles and Infrastructure use WAVE over 5.8-5.9GHz adhoc mesh network to exchange state information
- Link WAVE/DSRC radios to vehicle BUS to enable automated hazard awareness and avoidance
- Technological bridge to fully autonomous vehicles
Critical Aspects of V2V

- High throughput vehicular ad hoc mesh network (VANET)
- Provide safety features beyond capability of onboard sensors
- Geared for homogeneous adoption in consumer automotive systems
- Easy integration with existing transportation infrastructure
- First application of stage 3 automation in consumer marketplace
Transportation network impacts all aspects of society
Impact of V2X Technologies: on Consumers

- **Safety benefits:**
  - Prevent 25,000 to 592,000 crashes annually
  - Avoid 11,000 to 270,000 injuries
  - Prevent 31,000 to 728,000 property damaging crashes

- **Traffic flow optimization:**
  - 27% reduction for freight
  - 23% reduction for emergency vehicles
  - 42% reduction on freeway (with cooperative adaptive cruise control & speed harmonization)
Impact of V2X Technologies: Global Industry

- Scalable across industrial platforms
  - Optimize swarm functions
  - Improve exchange of sensor data
- Enhance/improve worker safety
  - Vehicle-to-pedestrian
  - Construction, agriculture, maintenance
- Improve logistical operations management
  - Think: air traffic control for trucks
Impact of V2X Technologies: Critical Infrastructure

- Provide interface for infrastructure to leverage VANET as carrier
- Increase awareness of traffic patterns in specific regions
  - Analysis of network traffic facilitates improvements in civil engineering processes
- Fast widespread distribution of emergency alerts
- Reduce cost of public transit systems
Impact of V2X Technologies: Automotive Security

- Wide open wireless attack vector into transportation network
- Injections easily propagate across entire VANET
- Wireless reverse engineering using 1609 and J2735
- Easy to massively distribute information (malware)
Technologies Using V2X

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

- Cooperative adaptive cruise control
- Automated ticketing and tolling
Vision of SocketV2V

- Security through obscurity leads to inevitable pwning
- Security community must be involved in development of public safety systems

- Catalyze development of secure functional connected systems
- Provide interface to VANET with standard COTS hardware
Background on SocketV2V

- Linux V2V development begins November 2015
- Large body of existing work found to be incomplete
  - No open-source implementation exists
  - Attempts at integration in linux-wireless since 2004
- Abandon efforts to patch previous attempts mid-2016
- Two years of kernel debugging later, V2V is real
Motivation for V2X Development

- Current standards for onboard vehicle communications not designed to handle VANET
- Increase in automation $\Rightarrow$ increase in attack surface
- Auto industry calling for proprietary solutions
  - Leads to a monopolization of technology
  - Standards still incomplete and bound for change, proprietary solutions become obsolete
  - Multiple alternative standards being developed independently across borders
- Imminent deployment requires immediate attention
Lessons Learned from Previous Epic Failure

1. You keep calling yourself a kernel dev, I do not think it means what you think it means

2. Sharing is caring: closed-source development leads to failure

3. Standards committees need serious help addressing unprecedented levels of complexity in new and future systems
V2X Stack Overview

DSRC Protocol Stack with Standards

- Basic Safety Message (SAE J2735)
- Minimum Performance Requirements (SAE J2945)
- Non-Safety Applications

DSRC Security
(IEEE 1609.2-2013)

- DSRC WSMP with safety subnet
  (IEEE 1609.3-2010)
- TCP/UDP
- IPv6

DSRC WAVE
Architecture Guide
(IEEE 1609.0-2013)

- DSRC Multi-Channel (IEEE 1609.4-2010)

- DSRC PHYSIC + MAC (IEEE 802.11p-2010)

DSRC Protocol Stack
V2X Stack Overview: 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, no encryption
- Multicast addressing with wildcard BSSID = {ff:ff:ff:ff:ff:ff}
- 5.8-5.9GHz OFDM with 5/10MHz subcarriers
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: WAVE Short Message

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Length (octets)</th>
<th>Value (hex)</th>
<th>Description</th>
</tr>
</thead>
</table>
| Subtype, Option Indicator, WSMP Version | 1               | 0B          | Subtype = 0 (4 bits)  
Option Ind = 1 (1 bit)  
Version = 3 (3 bits)                                                      |
| Count                               | 1               | 03          | Info Elem Count = 3                                                           |
| Info Element 1 Channel Number       | 3               | 0F 01 AC    | WAVE Element ID = 15  
WAVE Elem Length = 1  
Channel: 172                                                              |
| Info Element 2 Data Rate            | 3               | 10 01 0C    | WAVE Element ID = 16  
WAVE Elem Length = 1  
Data rate: 6 Mb/s                                                         |
| Info Element 3 Transmit Power Used  | 3               | 04 01 9E    | WAVE Element ID = 4  
WAVE Elem Length = 1  
30 dBm                                                                    |
| TPID                                | 1               | 00          | Address Info (PSID) only, no Info Elem Ext field present.                   |
| Provider Service Identifier          | 3               | C0 03 05    | PSID: 0pC0-03-05                                                           |
| WSM Length                          | 1               | 0D          | Length = 13  
Length of WSM Data                                                          |
| WSM Data                            | 13              | 48 65 6C 6C 6F 20 57 6F 72 6C 64 21 00 | ASCII content: ‘Hello World!’  
0x48 = H  
0x65 = e  
0x6C = l  
extc. |
V2X Stack Overview: SAE J2735

- Message dictionary specifying message formats, data elements
  - Basic safety message, collision avoidance, emergency vehicle alert, etc.
  - ASN1 UPER specification
  - Also supports XML encoding

- Data element of Wave Short Message
- Application-layer component of V2X stack
State of V2X Standards: Evolution

- WAVE drafted in 2005, J2735 in 2006
- WAVE revisions not backwards compatible
- IEEE 1609.2 incomplete
- J2735 revisions not backwards compatible
  - Encoding errors in J2735 ASN1 specification
- 3 active pilot studies by USDOT
- Experimental V2X deployment in EU
- Developmental status still in flux
Major Changes to the Standards

- Refactoring of security services to change certificate structure
- Refactoring of management plane to add services (P2PCD)
- Refactoring of application layer message encoding format
  - Multiple times: BER ⇒ DER ⇒ UPER
- Refactoring of application layer ASN1 configuration
- Revision of trust management system - still incomplete
(Possibly Unintentional) Obfuscation of the Standards

- No specification of handling for service management and RF optimization
- Minimal justification given for design choices
- Introduction of additional ambiguity in message parsing (WRA IEX block)
- Redlining of CRC data element in J2735 messages
- Refactoring of J2735 ASN1 to a non standard format
- Proposed channel sharing scheme with telecom
Subtleties in WAVE/J2735

- Ordering of certain fields not guaranteed in 1609
- Type incongruities in 1609
- Wave Information Element contains nested structures
- Channel synchronization mechanism based on proximal VANET traffic
- Channel switching necessary with one-antenna systems
- Implementation-specific vulnerabilities can effect entire network
V2X Attack Surfaces

- VANET accessible from a single endpoint
  - Attacks propagate easily across network
- Entry point to all systems connected to V2X infrastructure
  - Manipulation of traffic control systems: lights, bridges
  - Public transportation
  - Tolling and financial systems
- DSRC interface acts as entry point to onboard systems
  - Wireless access to vehicle control BUS
  - Transport malware across trafficked borders
- Privilege escalation in PKI
  - Hijacking emergency vehicle authority
  - Certificate revocation via RSU
Understanding the Adversary: Passive

- Determine trajectory of cars within some radius
  - Few stations required to monitor a typical highway
- Enumerate services provided by peers
- Characterize network traffic patterns within regions
- Uniquely fingerprint network participants independent of PKI
  - RF signature
  - Probe responses
  - Behavior patterns
- Perform arbitrage on economic markets
Understanding the Adversary: Active

- Denial of service, MITM
  - Impersonate infrastructure point
  - Manipulate misbehavior reports
- Disrupt vehicle traffic
  - Target specific platforms and individuals
  - Economic warfare: manipulation of supply/distribution networks
  - Behavior modeling and manipulation
- Privilege escalation in VANET
  - Parade as an emergency vehicle or moving toll station
    - Ad hoc PKI for application-layer services
  - Assume vehicle control via platooning service
V2X Threat Model: Applied

- Network traffic can be corrupted over-the-air
- Ad hoc PKI can allow certificate hijacking
- Diagnostic services in DSRC implementation expose vehicle control network
- Valid DSRC messages can pass malicious instructions to infotainment BUS
- Fingerprinting possible regardless of PKI pseudonym scheme
- Trust management services vulnerable to manipulation
  - Misbehavior reporting
  - Certificate revocation
  - Denial of P2P certificate distribution
Our Solution: Just Use Linux

- Platform-independent* V2X stack integrated in mainline Linux kernel
  - No proprietary DSRC hardware/software required - uses COTS hw
  - Extensible in generic Linux environment
  - *Currently supports ath9k

- Implements 802.11p, IEEE 1609.{3,4} in Linux networking subsystem
  - mac80211, cfg80211, nl80211
  - 1609 module to route WSMP frames

- Underlying 802.11 architecture compatible with Linux networking subsystem
  - Mainline kernel integration leads to immediate global deployment
  - This enables rapid driver integration
SocketV2V: Implementing 802.11p in Linux Kernel

- Add driver support for ITS 5.825-5.925GHz band
- Define ITS 5.8-5.9GHz channels
SocketV2V: Implementing 802.11p in Linux Kernel

- Modify kernel and userspace local regulatory domain
- Force use of user-specified regulatory domain
Enable filtering for 802.11p frames

Require use of wildcard BSSID

```c
case NL80211_IFTYPE_OCB:
    /* We accept: MGMT, TSF, action
     * CTL: ~(PSPOLL,CFEND,CFENDACK)
     * DATA: data, null, QoS data, QoS null
     */
    if (!bssid)
        return false;
    if (!multicast &
        !ether_addr_equal(sdata->vif.addr, hdr->addr3))
        { /* Group addr have wildcard BSSID
            return false; */
        }
    if (!multicast &
        !ether_addr_equal(sdata->dev->dev_addr, hdr->addr1))
        { /* Do not add \n
            return false; */
        }
    if (!ieee80211_is_ocb(hdr->frame_control))
        { /* Wrong frame type for OCB */
            return false;
        }
    if (!rx->sta) {
        int rate_idx;
        if (status->encoding != RX_ENC_LEGACY)
            rate_idx = 0;
        else
            rate_idx = status->rate_idx;
        ieee80211_ocb_rx_no Sta(sdata, bssid, hdr->addr2,
                         BIT(rate_idx));
    }
    return true;
```
SocketV2V: Userspace Utility Modifications for 802.11p

- Add **iw** command for joining 5GHz ITS channels using OCB

```c
static const struct chanmode chanmode[] = {
    { .name = "5MHz",
      .width = NL80211_CHAN_WIDTH_5,
      .freq1_diff = 0,
      .chantype = NL80211_CHAN_5MHz },
    { .name = "10MHz",
      .width = NL80211_CHAN_WIDTH_10,
      .freq1_diff = 0,
      .chantype = NL80211_CHAN_10MHz },
};
```

- Add **iw** definitions for 5/10MHz-width channels in OCB

- Use **iw** to join ITS spectrum with COTS WiFi hardware!

```bash
phy1
Interface wlx001f53d73df
ifindex 4
wdev 0x100000001
addr e0:91:75:3d:73:df
type outside context of a BSS
channel 170 (5890 MHz), width: 10 MHz, center1: 5890 MHz
txpower 17.00 dBm
```
SocketV2V: Implementing WAVE in Linux

Functions 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 switching, dispatch
- Netlink socket interface to userspace
- Userspace link to the 802.11p kernel routines
- Manage channel switching using iw
WAVE Usage

- Message fields set in struct wsmp_wsm
- WSMs encoded to bitstream through wsmp_wsm_encode
- WSM bitstream decoded through wsmp_wsm_decode
- Functions for generating random WSMP frames to specified compliance
- Opens socket to 802.11p wireless interface
- Inject WSMs with prefix EtherType 0x86DC
WAVE Structs: WSM

```
struct wsmp_wsm {
    /* N-Header */
    uint8_t subtype; /* 0 to 15 */
    uint8_t version; /* WSMP_VERSION */
    uint8_t tpid; /* 0 to 5 */
    bool use_n_iex;
    struct wsmp_iex *n_iex;

    /* T-Header */
    union {
        uint32_t psid; /* 0x0 to 0xFFFFFFFF */
        struct {
            uint8_t src[2];
            uint8_t dst[2];
        } ports;
    }
    bool use_t_iex;
    struct wsmp_iex *t_iex;
    uint16_t len; /* 0 to 16383 */
    uint8_t *data;
};
```

- WAVE Short Message: message encapsulation and forwarding parameters (N-hop, flooding)
WAVE Structs: IEX

- Information Element Extension: optional fields for RF, routing, and services

```c
struct wsmp_iex {
    uint16_t count; /* 1 to 255 */

    /* WSM Elements */
    uint8_t chan; /* 0 to 200 */
    uint8_t data_rate; /* 2 to 127 */
    uint8_t tx_pow; /* -128 to 127 */

    /* SII Elements */
    struct wsmp_ie_psc psc;
    uint8_t ip[16];
    uint8_t port[2];
    uint8_t mac[6];
    int8_t rcpi_thres; /* -110 to 6 */
    uint8_t count_thres;
    uint8_t count_thres_int; /* 1 to 255 */

    /* CII Elements */
    struct wsmp_ie_edca edca;
    uint8_t chan_access; /* 0 to 2 */

    /* WSA Elements */
    uint8_t repeat_rate;
    struct wsmp_ie_2d loc_2d;
    struct wsmp_ie_3d loc_3d;
    struct wsmp_ie_advert_id advert_id;

    /* WRA Elements */
    uint8_t sec_dns[16];
    uint8_t gateway_mac[6];
    bool use[WSMP_EID_CHANNEL_LOAD+1];

    /* Raw elements not defined in 1609.3-2016 */
    uint16_t raw_count;
    struct wsmp_ie_raw *raw;
};
```
(More) WAVE Structs

```
struct wsmp_wsa {
    uint8_t version;          /* 0 to 15 */
    uint8_t id;               /* 0 to 15 */
    uint8_t content_count;    /* 0 to 15 */

    bool use_iex;
    struct wsmp_iex *iex;

    uint16_t sii_count;       /* 0 to 31 */
    struct wsmp_sii **sis;

    uint16_t cii_count;       /* 0 to 31 */
    struct wsmp_cii **cis;

    bool use_wra;
    struct wsmp_wra *wra;
};

struct wsmp_sii {
    uint32_t psid;
    uint8_t chan_index;      /* 0 to 31 */

    bool use_iex;
    struct wsmp_iex *iex;
};

struct wsmp_cii {
    uint8_t op_class;        /* 802.11 */
    uint8_t chan;            /* 802.11 */
    int8_t tx_pow;           /* -128 to 127 */
    uint8_t adapt;           /* 0 to 1 */
    uint8_t data_rate;       /* 0x02 to 0x7f */

    bool use_iex;
    struct wsmp_iex *iex;
};

struct wsmp_wra {
    uint8_t lifetime[2];     /* IETF RFC 4861 */
    uint8_t ip_prefix[16];   /* IETF RFC 4861 */
    uint8_t prefix_len;      /* IETF RFC 4861 */
    uint8_t gateway[16];
    uint8_t dns[16];

    bool use_iex;
    struct wsmp_iex *iex;
};
```
SocketV2V: Implementing J2735

```c
/* BasicSafetyMessage */
struct BSM {
    enum DSRCmsgID msgID;
    struct {
        uint8_t msgCnt;
        uint32_t tid;
        uint16_t secMark;
        uint32_t lat;
        uint32_t longt;
        uint16_t elev;
        uint32_t accuracy;
        uint32_t txstate;
        uint16_t speed;
        uint16_t heading;
        uint8_t angle;
        uint64_t accelSet[4];
        uint16_t brakes;
        uint32_t size;
    } blob1;
};
```

```c
struct BSM *bsm1;
bsm1 = calloc(sizeof(struct BSM), 1);
bsm1->msgID = basicSafetyMessage;
bsm1->blob1.msgCnt = 0xAB;
bsm1->blob1.tid = 0x12345678;
bsm1->blob1.secMark = 0x6358;
bsm1->blob1.lat = 0x67BA2100;
bsm1->blob1.longt = 0xA4B263D;
bsm1->blob1.elev = 0xAAAA;
bsm1->blob1.accuracy = 0x1221ABAB;
bsm1->blob1.txstate = 0x3;
bsm1->blob1.speed = 0xA2B;
bsm1->blob1.heading = 0x1212;
bsm1->blob1.angle = 0x42;
bsm1->blob1.accelSet[0] = 0x0A08;
bsm1->blob1.accelSet[1] = 0x0C0D;
bsm1->blob1.accelSet[2] = 0x0FA;
bsm1->blob1.accelSet[3] = 0x1AFBC;
bsm1->blob1.brakes = 0xA134;
bsm1->blob1.size = 0x01323456;
```
SocketV2V: Implementing J2735

- Generate WSM
- Pack J2735 msg in WSM data element
- Serialize WSM
- Tx using pcap_inject!
SocketV2V: Implementing J2735

Wireshark WSMP plugin incomplete per current 1609 encoding
SocketV2V: Implementing J2735

```c
struct RoadSideAlert *rsal;
rsal = calloc(sizeof(struct RoadSideAlert),1);
rsal->msgID = roadSideAlert; // A 1 byte instance
rsal->msgCnt = 0x0A;
rsal->typeEvent = 0x0DEFC;
rsal->position.lat = 0x67BA2100;
rsal->position.longt = 0x1A4B2630;

struct EmergencyVehicleAlert *eval;
eval = calloc(sizeof(struct EmergencyVehicleAlert),1);
eval->msgID = emergencyVehicleAlert; // A 1 byte instance
eval->timeStamp = 0x515; // optional
eval->tid = 0x12345678;
eval->rsaMsg = rsa1;
eval->basicType = 0x04;

struct wsmp_wsm *wsm2;
wsm2 = calloc(sizeof(struct wsmp_wsm),1);
wsm2->subtype = 0x0;
wsm2->version = 0x3;
wsm2->tpid = 0x00;
wsm2->use_n_iex = 0;
wsm2->psid = 0xC00305;
wsm2->use_t_iex = 0;
wsm2->len = sizeof(*rsal);
wsm2->data = (uint8_t *) rsal;

struct wsmp_wsm *wsm3;
wsm3 = calloc(sizeof(struct wsmp_wsm),1);
wsm3->subtype = 0x0;
wsm3->version = 0x3;
wsm3->tpid = 0x00;
wsm3->use_n_iex = 0;
wsm3->psid = 0xC00305;
wsm3->use_t_iex = 0;
wsm3->len = sizeof(*eval);
wsm3->data = (uint8_t *) eval;
```
Future Pwning of ITS: You Wanna be a Master?

- 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
Additional Forms of Pwning

Pandora’s box:

- Mass dissemination of malware
- Passive surveillance with minimal effort
- Extract RF parameters for imaging
- Reverse engineer system architectures given enough data
- Epidemic propagation model
- Built in protocol switching
  - Exfiltration over comm bridges!
Developing Connected Vehicle Technologies

- Widespread access enables engagement of security (1337) community in standards development
- 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)
Acknowledgments
References

- Check me out on GitHub: https://github.com/p3n3troot0r/socketV2V


- Papernot, Nicolas, et al. ”Practical black-box attacks against deep learning systems using adversarial examples.”