Linux and the Automotive Security Lab

Nathan Willis
nate@lwn.net

The what?

- A survey of real-world automotive security exploits
- Analysis for common threads
- Places where Linux can help significantly

The why...

I'm a mild-mannered reporter by day, but the 2011 CAESS report grabbed my attention.

http://www.autosec.org/pubs/cars-usenixsec2011.pdf

The why...

I'm a mild-mannered reporter by day, but the 2011 CAESS report grabbed my attention.

http://www.autosec.org/pubs/cars-usenixsec2011.pdf

What differentiates this paper? Using an actual car.

The why...

Further research on my part reveals that actual cars are a rarity.

The why...

Further research on my part reveals that actual cars are a rarity.

So I collect research that uses them.

While we're on the subject...

∃ A plethora of academic research

...mostly about CAN bus.

While we're on the subject...

∃ A plethora of academic research
...mostly about CAN bus.

And CAN bus is Theseus's ship.

And now, let the fun begin!

RFID in the early years

Bono et al., August 2005

Security analysis of a cryptographically-enabled RFID device

https://www.usenix.org/legacy/event/sec05/tech/bono/bono.pdf

Defeated TI Digital Signal Transponder (DST) used in ignition/entry and mobile payment systems.

RFID in the early years

Bono et al., August 2005

Security analysis of a cryptographically-enabled RFID device

https://www.usenix.org/legacy/event/sec05/tech/bono/bono.pdf

* Short key length, security-by-obscurity challenge-response

Are we there yet?

Barisani and Bianco, 2007

Unusual car navigation tricks: injecting RDS-TMC traffic information signals

http://dev.inversepath.com/rds/cansecwest_2007.pdf

Inject arbitrary messages to navigation system: traffic jams, road closures, accident delays, terrorist attacks, bullfights (?).

Alerts interrupt navigation; some redirections not displayed to driver.

Used off-the-shelf parts.

Are we there yet?

Barisani and Bianco, 2007

Unusual car navigation tricks: injecting RDS-TMC traffic information signals

http://dev.inversepath.com/rds/cansecwest_2007.pdf

* Message sources unauthenticated. Plaintext messages (except for location codes...). Broadcaster IDs easy to find.
* Weak crypto, software accepts unencrypted messages just fine.
* Road incidents hidden from user (even suspicious ones)

Key decisions

Indesteege et al., April 2008

A practical attack on KeeLoq

http://www.cosic.esat.kuleuven.be/publications/article-1045.pdf

Eisenbarth et al., August 2008

On the power of power analysis in the real world: A complete break of the KeeLoq code hopping scheme.

http://www.iacr.org/archive/crypto2008/51570204/51570204.pdf

* "Learning" mode. Guessable keys (derived from serial numbers).

Car talk

Krucker and Bitzi, 2009

Communication with a Toyota Prius

http://students.asl.ethz.ch/upl_pdf/151-report.pdf

Injecting CAN bus commands; controlled steering and throttle. No modification to car hardware required. But only because they were unwilling to try.

Car talk

Krucker and Bitzi, 2009

Communication with a Toyota Prius

http://students.asl.ethz.ch/upl_pdf/151-report.pdf

* Classic CAN bus problems: no authentication of sender, no encryption of message contents, no message integrity check, etc.

The Road to Ruin

Koscher et al., May 2010

Experimental Security Analysis of a Modern Automobile

http://www.autosec.org/pubs/cars-oakland2010.pdf

Center for Automotive Embedded Systems Security (CAESS)

Disable/engage brakes, A/C, door lock, instrument panel, windshield wipers, trunk lid, horn, disable cylinders, all lights, shift-lock, power steering, starter, idle RPM . . . basically, you name it.

The Road to Ruin

Koscher et al., May 2010

Experimental Security Analysis of a Modern Automobile

http://www.autosec.org/pubs/cars-oakland2010.pdf

* CAN bus weaknesses; CAN bus flooding
* Bus segments not isolated (OBD-II port used)
* Remote write access to ECU firmware
* Ability to erase ECU firmware after attack (self-modifying code)

Half in the bag

Hoppe et al., July 2010

Security threats to automotive CAN networks—Practical examples and selected short-term countermeasures

http://link.springer.com/chapter/10.1007/978-3-540-87698-4_21

Controlled window motors, warning lights, airbag system, sniffed traffic on CAN bus gateway. Executed denial-of-service attack on bus.

Half in the bag

Hoppe et al., July 2010

Security threats to automotive CAN networks—Practical examples and selected short-term countermeasures

http://link.springer.com/chapter/10.1007/978-3-540-87698-4_21

* Previously-seen CAN bus weaknesses
* Non-repudiation of CAN bus messages mentioned

Tired. So tired.

Roufa et al., 2010

Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study

http://ftp.cse.sc.edu/reports/drafts/2010-002-tpms.pdf

Read TPMS data from 40m. Spoofed TPMS messages remotely, including warnings.

Tired. So tired.

Roufa et al., 2010

Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study

http://ftp.cse.sc.edu/reports/drafts/2010-002-tpms.pdf

* No message encryption, static module identifiers. No authentication of message sender. Excess communication range.
* Mentions privacy issues (vehicle tracking).

Hear hear!

Francillon et al., February 2011

Relay attacks on passive keyless entry and start systems in modern cars

http://eprint.iacr.org/2010/332.pdf

Intercepting-and-forwarding RF messages between key and car is sufficient to unlock, lock, or start vehicle. Protocol-independent!

* No time/distance bounds. No access control at lock (e.g., waiting).

Distance learning

Checkoway et al., 2011

Comprehensive Experimental Analyses of Automotive Attack Surfaces

http://www.autosec.org/pubs/cars-usenixsec2011.pdf

CAESS follow-up; addresses external attack surface.

Exploits CD player (update mechanism and buffer overflow in DAC), to modify firmware and send CAN messages. Reprogrammed service center PassThru devices to replicate worm.

Distance learning

Checkoway et al., 2011

Comprehensive Experimental Analyses of Automotive Attack Surfaces

http://www.autosec.org/pubs/cars-usenixsec2011.pdf

(continued...)

Paired Bluetooth devices to car, circumventing UI, and exploited 3G telematics unit; gained access to CAN bus in both cases.

Loaded trojans triggered later by TPMS, RDS, 3G, and Bluetooth.

Distance learning

Checkoway et al., 2011

Comprehensive Experimental Analyses of Automotive Attack Surfaces

http://www.autosec.org/pubs/cars-usenixsec2011.pdf

* Many attacks were complex, but hinged on CAN weaknesses.
* Unsafe functions like strcpy, debugging symbols, error strings, etc.
* Lack of simple countermeasures (stack cookies, ASLR)
* No behavioral analysis (e.g., unlimited Bluetooth pairing attempts)

Distance learning

Checkoway et al., 2011

Comprehensive Experimental Analyses of Automotive Attack Surfaces

http://www.autosec.org/pubs/cars-usenixsec2011.pdf

* Unnecessary services running (ftp, vi, telnet?)
* Glue code at interface boundaries is weak (shell scripts?)
* ECUs updatable in the field
* Separately-sourced components without integration testing

Point-n-click

VandenBrink, July 2012

Dude your car is pwned

https://isc.sans.edu/presentations/sansfire2012-Rob_Vandenbrink-obd-preso.pdf

Uses OBD-II port for full CAN bus access. Not much in the way of new exploits, but he does release Python/Qt code.

You read it here first

Miller and Valasek, August 2013

Car Hacking

http://illmatics.com/car_hacking.pdf

Attempts to be a comprehensive primer on CAN bus and related exploits. Demonstrates instrument panel control, navigation message injection, steering DOS and control, engaging/disabling brakes, acceleration, light control, killing cylinders, horn, seatbelt motor, door locks, fuel gauge, ECU firmware upload.

Trends

A decade ago, there were usable cryptographic attacks.

Those days are mostly behind us, but that is no reason to slack off.

Trends

Common themes in modern attacks include:

- sender authentication and message encryption
- assumption that a physically disjoint bus offers protection
- guessable or discoverable identifiers
- testing and debugging features not removed
- failures hidden from the user
- no countermeasures against active attackers

Where Linux can help

A common (and open) stack can help guard against the multi-source-components problem.

i.e., bad and untested interface boundaries

Where Linux can help

A common (and open) stack can help guard against the multi-source-components problem.

However: as ECUs get consolidated, this becomes a sandboxing issue (rather than disappearing).

Virtualization / LXC not just for user apps.

Where Linux can help

Access control: which ECUs (or processes) can send what messages to whom.

i.e., start fighting over LSMs.

I naively find SMACK easier to follow on this front. But you probably shouldn't let me tell you anything.

Where Linux can help

Access control: which ECUs (or processes) can send what messages to whom.

i.e., start fighting over LSMs.

I naively find SMACK easier to follow on this front. But you probably shouldn't let me tell you anything.

The big issues I see: extra services & trusting the net

Where Linux can help

Basic countermeasures from desktop/server systems:

- ASLR, etc.
- Logging
- Passive countermeasures like throttling
- Smarter gateway controllers

Where YOU can help

#1 – Obviously, writing the software

#2 – Tell me what I'm missing