Pentesting Cars

Oliver Nettinger

SANS Pen Test Hackfest Europe 2019
22/07/2019
About

Research & Development at NVISO in Germany
Embedded and Automotive Security Research

Passion for transportation since the nineties
Boards, Bikes and Cars

Automotive Security since 2015
Penetration testing and incident response for OEM
Content

Car security basics

Getting started

Challenges

What can I do with the results?

Car Salesman: *slaps roof of car* this bad boy can fit so many $\#$ vulns in it!
How it all began
Car Security Basics

The CAN Bus

- One of the main communication busses in a car

<table>
<thead>
<tr>
<th>CAN ID (11 bit)</th>
<th>Length (4 bit)</th>
<th>Payload (0..64 bit)</th>
<th>CRC (15 bit)</th>
</tr>
</thead>
</table>

A multitude of tools exist for attacking the CAN Bus

- Multiple tools developed by Craig Smith / Open Garages, e.g. ICSim, UDSim
- Kick-Starter Project for DIY OBD-II Device
- Canalyzat0r, Canalyzer, YACHT, ...

ID Len Payload

(000,000255) can 60B [8] 00 4E 7B EE 88 20 01 82
(000,000257) can 60B [8] 01 4B 0B F8 80 20 01 82
(000,000259) can 60B [8] 00 7D AF A0 8B 85 E5
(000,000262) can 60B [8] 01 7D 0F A1 70 80 8A 6C
(000,849146) can 201 [8] 4C 20 8B 00 80 E4 88 80
(000,008724) can 70F [4] 04 1E 81 00
(000,824629) can 60A [4] 02 38 E9 10
(000,800499) can 60B [8] 01 4E 6B FE 88 20 01 77
Car Security Basics

The CAN Bus

- One of the main communication busses in a car

A multitude of tools exist for attacking the CAN Bus

- Multiple tools developed by Craig Smith/Open Garages, e.g. ICSim, UDSim
- Kick-Starter Project
- Canalyzat0r, Canalyzer

<table>
<thead>
<tr>
<th>CAN ID</th>
<th>Length</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>(11 bit)</td>
<td>(4 bit)</td>
<td>(0.)</td>
</tr>
</tbody>
</table>

ID    Len Payload
Car Security

An overview

Cloud

Cars

Embedded

Car Security
In Car

Multiple Subsystems and Buses
• CAN, LIN, MOST, FlexRAY, Ethernet (Flashing, Video, Audio, OTA)
• WiFi, Bluetooth, NFC, FM, DAB, ...

Protocol Examples
• Diagnostics: UDS
• In-Car Communication: Some/IP (RPC)
• Backend: HTTP/S, MQTT

Technology Examples
• Node.js, Android Auto, NFS, IPsec,
Getting started...
Why

Phase 1: Attach to OBD-II
Phase 2: ?
Phase 3: Profit
Available Methodology and Tooling

Current Situation

OWASP IoT Testing Guide & Tester Guidance
OWASP Embedded Application Security Project
Available Methodology and Tooling

Software and Hardware

Multiple CAN-specific tools available (see resources)

Training ECU developed by Quarkslab
(Open Source)
https://blog.quarkslab.com/development-of-a-training-ecu.html

PASTA Car Pentesting Framework developed by Toyota
(Open-Source, 28.300 $)
Announced on Black Hat, available since July
https://github.com/pasta-auto
NVISO Automotive Testing Framework and Tooling

Methodology: 10 Modules with detailed test-steps to perform a full penetration test

- CPF-M1 Hardware analysis, side-channel attacks, glitching
- CPF-M2 Firmware analysis
- CPF-M3 Hardening of ECUs, stacks (e.g. AUTOSAR) and underlying operating systems
- CPF-M4 Diagnostic scanning and fuzzing via CAN/Ethernet, OBD-II or remote diagnostics
- CPF-M5 Bus communication, e.g. CAN, FlexRay, Ethernet, MOST and analyse of (proprietary) protocols
- CPF-M6 Radio connections, e.g. Bluetooth, WiFi, FM/DAB, NFC, key fob radio
- CPF-M7 Telematics and Connected Car (GSM, Backend/Cloud, CE-Apps, Onboard-Apps, OTA)
- CPF-M8 Cryptanalysis
- CPF-M9 HAF/VAF - autonomous driving (sensor networks, Car2X communication, machine learning algorithms)
- CPF-M10 Red-Teaming, e.g. for complete vehicle assessments
How to get started?

Various Approaches

ECU → Sub-system → Car → + Ecosystem
How to get started?

Test Equipment And Setup

What you need [minimal setup]
- 12V Power Supply
- Multimeter
- CAN Adapter
- Spare Wires
- Target ECU

Helpful
- Soldering Iron
- Oscilloscope
- Logic Analyzer
- Debugger
- Radio Stuff
- USB Serial Adapter
- ...
First Steps

What could be interesting...

Unprotected Debug Interfaces and Services
Hardware and Software

Diagnostics
Via CAN or OBD

Sensitive Information in Firmware
Passwords, keys

Outdated Software
Services and Libraries

Everything Remote

→ Leverage your existing knowledge!
Getting started

Debug and other useful Ports

Retrieve datasheets
Probe existing connectors
Find test points on PCB

Tools like Bus Pirate, JTAGulator can help identifying the right pins
Getting started

Diagnostics

Directly connect to the CAN interface or go via OBD Port
Use tools to scan for available services, e.g. Caring Caribou
Manufacturer Tools are a very useful resource

Standardized information available on Services and Error Code

Diagnostic Messages:
<ArbitrationID>#<Len><Service><Subfct><Data>

Example: Reset
60b#02 11 03
Getting started

Sensitive Information in Firmware

Retrieve firmware by dumping flash chips on the board

Look for firmware, e.g. in:
- open-source tools,
- delivered with manufacturer software,
- or in forums

Firmware files might need to be reversed due to a custom format

Can contain bootloaders, ELF Binaries, filesystems...

Binwalk and hex editors are your friend

Worth reading:
https://publicism.info/engineering/penetration/7.html
NVISO Automotive Testing Framework and Tooling

Gives complete guidance for performing a pentest with thorough coverage

Also applicable for IoT/Embedded Testing
Large parts of framework can be used as a guideline for general IoT and embedded testing

Structure
1. Vulnerabilities / Weaknesses
2. Test steps with examples
3. Recommendations for Mitigation

Example: Hardware Module

- **CPF-M1**: Hardware analysis, side-channel attacks, glitching
  - **CPF-M1.1**: Debug Ports
    - **CPF-M1.1.1**: JTAG
  - **CPF-M1.2**: UART
  - **CPF-M1.3**: I²C
  - **CPF-M1.2.1**: Test Points
    - **CPF-M1.2.1.1**: USB Port Connector
  - **CPF-M1.2.2**: Serial Port Connector
  - **CPF-M1.3.1**: Memory
    - **CPF-M1.3.1.1**: Unencrypted Flash Storage
  - **CPF-M1.4**: SoC Communication
    - **CPF-M1.4.1**: UART/Serial Communication between components
  - **CPF-M1.5**: Communication Module
    - **CPF-M1.5.1**: OnBoard eSIM Card
  - **CPF-M1.6**: Insecure Boot Modes
    - **CPF-M1.6.1**: Recovery Boot Mode
  - **CPF-M1.6.2**: Firmware Flash Mode
  - **CPF-M1.7**: Secure Boot
    - **CPF-M1.7.1**: Key extraction
  - **CPF-M1.8**: Local Firmware Update
  - **CPF-M1.9**: Side Channel Attacks
    - **CPF-M1.9.1**: Power Analysis
  - **CPF-M1.9.2**: Time analysis
  - **CPF-M1.10**: Glitching
    - **CPF-M1.10.1**: Clock glitching
  - **CPF-M1.10.2**: Voltage glitching
  - **CPF-M1.10.3**: Temperature based glitching
  - **CPF-M1.10.4**: Laser/optical glitching
Challenges

Get a **working** test setup

Equipment can be expensive

Information can be hard to find

Vendor tooling is not officially available

Rate vulnerabilities

You want to avoid that your car ends up like this...
Bug Bounty & Responsible Disclosure
Responsible Disclosure

Manufacturers

**Daimler**
https://www.daimler.com/whitehat/

**BMW**

**AUDI/Porsche/VW, Opel, Mazda, ...**
Unknown?

**Examples**
Tencent Keen Security Labs, ADAC

---

**Disclosure**

Please refrain from publishing technical details of any vulnerability you find to give us an opportunity to fix it. We try to work out a disclosure timeline with you.

Please be aware that other than standard IT systems we cannot force somebody to install an update as the vehicles belong to our customers and are not under our control. Therefore it can take a long time after a patch is released before a significant part of vehicles on the road are patched.

**Note:** If you found a flaw in our vehicles, please note that fixing a bug in a vehicle is a substantially different process than fixing a bug in classic IT systems. Vehicle software needs to meet high safety and regulatory requirements, therefore fixing a bug takes significantly more time.

Source: Daimler
Monetize Your Results

Bug Bounty Programs

**bugcrowd**

**FCA**
https://bugcrowd.com/fca, $150 - $7.500

**Tesla**
https://bugcrowd.com/tesla, $100 – $15.000 / CanSecWest/Pwn20wn $1M and Model 3

*Tesla says they won’t void warranty when you „hack their software“.*

**hackerone**

**GM**
https://hackerone.com/gm
Public Examples

Responsible Disclosure in Automotive

Tencent/Keen Security Labs
Responsible Disclosure of several vulnerabilities for Tesla, BMW and more...

Sam Curry
XSS Vuln via Car Name executed in dealership backend
https://samcurry.net/cracking-my-windshield-and-earning-10000-on-the-tesla-bug-bounty-program/
More Useful Resources

**Car Hacking Handbook by Craig Smith**
http://opengarages.org/handbook/

**Tooling by Craig Smith**
https://github.com/zombieCraig

**Jailbreaking Subaru**
https://github.com/sgayou/subaru-starlink-research

**NVISO Automotive Pentesting Methodology**
https://github.com/NVISO-BE

**Auto Forums**
Various

**OWASP IoT & Embedded Security Testing Guides**
https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project
https://www.owasp.org/index.php/OWASP_Embedded_Application_Security
Thank you

onettinger@nviso.de
@one1