(PEN)TESTING VEHICLES WITH CANTOOLZ

YACHT - YET ANOTHER CAR HACKING TOOL

By ALEXEY SINTSOV (@asintsov)
#whoami

**WORK:** Principal Security Engineer at

**Community:** co-founder of DC group

WARNING: I am not a HARDWARE/CAR guy... my past is about JIT-SPRAY, shellcodes, ROP, BoF, UAF and WEB things like SQLi... but now all these things came into automotive world ;)
#CarPWN community from RUSSIA (of course with love)

Bunch of CAR hackers/enthusiasts, just a Telegram community who are interested in automotive internals:

- CAN/LIN/Ethernet/Wi-Fi/BT research
- Reverse Engineering of ECU/HU
- Bug hunting
- Creating own tools and hardware modules (DIY)

They are active CANToolz users and my best testers and bug-reporters, thx to them:

- Michael Elizarov
- Sergey Horujenko
- Anyton Sysoev
- Dmitry Nedospasov (@nedos)
- Sergey Kononenko (CANToolz dev)
- Ilya Dinmuehametov

.. and more more more...
#Pentesting?

DISCLAIMER:
This is not a FUD talk, I am not going to ‘sell’ any devices or services. Automotive Security Engineers are doing a good job right now and they are trying to address all issues. So it is not SO bad as you could read in mass-media. There are some challenges and problems – yes. But people are working on making this world a more secure place and tomorrow is always better than yesterday.

Oh... not this picture AGAIN! Enough....
#Attack surface

Direct attacks

- Local I/O
  - CAN interfaces
  - Ethernet
  - WiFi
  - OBD-II

- Wireless components and ECUs
  - Long Radio:
    - GSM/UMTS
    - Radio/RDS
    - GPS
  - Short Radio:
    - WiFi/Bluetooth
    - TPMS
    - Keyless lock/start
    - Radars/Sensors/Cameras

- HeadUnit
  - Software components
    - WEB Browser
    - MP3/etc
    - RDS
    - Applications
    - Connected Car services
#Attack surface

Connected Car

- CSRF
- MITM
- Internet Backend services hacking
- ...

- Wireless components and ECUs
  - Long Radio:
    - GSM/UMTS
    - Radio/RDS
    - GPS
  - Short Radio:
    - WiFi/Bluetooth
    - TPMS
    - Keyless lock/start
    - Radars/Sensors/Cameras
- HeadUnit
  - Software components
    - WEB Browser
    - MP3/etc
    - RDS
    - Applications
    - Connected Car services

- Local I/O
  - CAN interfaces
  - Ethernet
  - WiFi
  - OBD-II
#Attack surface local interfaces

- Local I/O
  - CAN interfaces
  - Ethernet
  - WiFi
  - OBD-II

- Wireless components and ECUs
  - Long Radio:
    - GSM/UMTS
    - Radio/RDS
    - GPS

- Short Radio:
  - WiFi/Bluetooth
  - TPMS
  - Keyless lock/start
  - Radars/Sensors/Cameras

- HeadUnit
  - Software components
    - WEB Browser
    - MP3/etc
    - RDS
    - Applications
    - Connected Car services
#CAN Bus

CAN gateway/switch*

* Different topology possible
# CAN Bus

https://en.wikipedia.org/wiki/CAN_bus
#Attack vector ("remote" example)

1. From the Internet/GSM/Wi-Fi into HU (RCE)
2. From HU to “intermediate device” like GatewayECU or another computer (for example privileged access to CAN bus)
3. PROFIT
#Backdoors

- Backdoor connected to OBD2 (external, not stealth mode)
- Backdoor connected to CAN BUS (internal, stealth)

.. or you could always compromise Internet back-end services (for connected car) and spy remotely.
#Local vector

- CAN/K-LINE and UDS (over OBD2 or unauthorized CAN access)
  - “Fake” ECU
  - Firmware “update”
  - Keys reset/rewrite
#Unauthorized CAN access

http://signalochka.ru/ ©

http://elauto-spb.ru/ ©
#OBD-II and UDS

ISO 14229-1
#Not only for BAD things, like theft and hacking...

UGLY THINGS:
- ‘Paid’ features unlock (illegal)
- Resets: VIN, mileage ...

GOOD THINGS (on you own risk):
- Custom anti-theft systems
- Custom/ DIY connected-car systems
- MOD’s, custom firmware for ECU...
#Tools for CAN

A lot of really good tools:

- [http://illmatics.com/content.zip](http://illmatics.com/content.zip) - with examples from Charlie and Chris talk about CAN
- [https://github.com/ericevenchick/CANard](https://github.com/ericevenchick/CANard) - Abstract CAN lib with UDS/ISO TP support
- [https://github.com/zombieCraig/UDSim](https://github.com/zombieCraig/UDSim) - Fuzzing, traffic simulator and more

Moarrrr: [https://github.com/jaredmichaelsmith/awesome-vehicle-security](https://github.com/jaredmichaelsmith/awesome-vehicle-security)

BUT, my needs are different:

- **HARDWARE independent** software for CAN bus reverse engineering and black-box analysis
- **Flexible and powerful framework** with multi interface support, for MITM, fuzzing and scanning
- **Module based framework**, where all modules could be used the way I want (like GNURadio design)
- **Features**: like data-type analysis, stats-analyses, UDS detection, CAN network emulator
- **API interface**

Nmap + MetaSploit + BurpSuite + GNURadio + “something like that”, but for CAN network....
Open Source Python Framework (Apache 2)
Works on Windows/Linux/Mac OS
All code in Python3

CANToolz core engine:
  • Multi interface/bus support
    • MITIM supported
  • As python lib (dev API)
  • WEB API
  • Console/ WEB GUI (extendable)

Ready-to-use modules:
  • CAN firewall
  • Simple Fuzzer, and proxy/MITM-fuzzer
  • UDS Scanner/sniffer
  • Stats analysis features... a lot of
  • Dump and replay
  • Extendable CAR/CAN emulator (ECU level)
  • TCP2CAN I/O module, tunnels and more!
  • USBTin I/O module
  • CANBus Triple I/O module
  • CAN Socket I/O module (linux only)

Ready to use for CAN traffic reverse engineering and black-box analysis!

https://github.com/eik00d/CANToolz
#CANToolz design: pipe’s concept

In config file:

- mod_firewall
- hw_USBtin
- hw_USBtin~2

CANToolz framework (MITM example)

Step 1
- mod_firewall
  - Block CAN frames A,B,C
- hw_USBtin~2
  - read
- hw_USBtin
  - read

Step 2
- hw_USBtin~2
  - write

Step 3
- mod_firewall
  - Block CAN frames X,Y,Z
- hw_USBtin~2
  - write

Step 4
- hw_USBtin
  - write

Step 5
- hw_USBtin~2
  - write

Step 6
- hw_USBtin
  - write

Loop

- You can use module in different instances (~2)
  - Different memory, state
- You can use same instance multiple times
  - Same memory/state

https://github.com/eik00d/CANToolz
# CANToolz design: pipe’s concept

In config file:

```plaintext
# Load modules
load_modules = {
    'hw_USBtin' : ,
    'hw_USBtin~2' : ,
    'mod_firewall' : {},
}

# Scenario with steps

actions = [
    {'hw_USBtin' : {'pipe': 1, 'action': 'read'}},
    {'hw_USBtin~2': {'pipe': 2, 'action': 'read'}},
    {'mod_firewall' : {'pipe': 1, 'black_list': [1337, 31337]}},
    {'mod_firewall' : {'pipe': 2, 'black_list': [0x1122, 0x2211]}},
    {'hw_USBtin~2': {'pipe': 1, 'action': 'write'}},
    {'hw_USBtin' : {'pipe': 2, 'action': 'write'}}
]
```

https://github.com/eik00d/CANToolz
#DEV API example: one-byte proxy-fuzzer

def counter(self):
    if self._i >= 255:
        self._i = 0
        self._active = False
    self._i += 1
    return self._i - 1

def do_init(self, params):
    self._active = False

def do_start(self, params):
    self._i = 0

# Change one byte to random

def do_fuzz(self, can_msg, byte_to_fuzz):
    if 0 < byte_to_fuzz < 9:
        can_msg.CANFrame.frame_data[byte_to_fuzz - 1] = self.counter()
    return can_msg

# Effect (could be fuzz operation, sniff, filter or whatever)
# can_msg - CANToolz message from the pipe (IN)
def do_effect(self, can_msg, args):
    # can_msg.CANData - boolean, if CANFrame in the Message
    if can_msg.CANData and can_msg.CANFrame.frame_type == CANMessage.DataFrame:
        if can_msg.CANFrame.frame_id in args.get('fuzz', []) and 'byte' in args:
            can_msg = self.do_fuzz(can_msg, args['byte'])
            can_msg.bus = self._bus
        elif 'nfuzz' in args and can_msg.CANFrame.frame_id not in args.get('nfuzz', []) and 'byte' in args:
            can_msg = self.do_fuzz(can_msg, args['byte'])
            can_msg.bus = self._bus
    # can_msg - CANToolz message TO the pipe (out)
    return can_msg
#Documentation

WIKI on GitHub:
- Use-cases and usage examples
- Modules documentation (not for all..eh, outdated sometimes)
- API documentation

Blog: https://asintsov.blogspot.com
- Developer’s blog
#CANToolz CAN over TCP

Remote access to CAN

Remote access to WEB API

TCP/IP Networks

CAN over TCP TUNNEL

Module: hw_CAN2TCP
Inspired by @dn5__ VIRcar emulator (https://github.com/dn5/vircar), I have created ECU modules as CANtoolz modules and connect them as they are connected in real car.
So I have ECU devices emulators, worked on different CAN buses and connected via CAN gateway.

- Door sensors
- Lights sensors
- RPM sensors
- CANGateway routing rules
- VIN fingerprint
- UDS support (auth, key)
- Engine start – control auth.
#CANToolz car emulator
What can we do over CAN?

- Data analysis (if exists, on some new cars OBD2 interface has no ‘live traffic’)
- OBD2 (boring)
- UDS
  - MITM – Session Hijacking
  - Proxy - Just sniffing
  - SCAN - Black-Box search (like nmap)
# UDS Scan — black box

CANToolz modules: `gen_ping/mod_stat` (examples/uds_scan.py)

Generate UDS pings with chooses services and subcommands

Analyze all traffic and detect UDS sessions

UDS based on ISO TP, sessions, authentication via SecurityAccess

ISO TP: first byte(s) of CAN data used as fragmentation flag and index/counter: ISO 15765-2
#UDS Scan – DEMO on emulator

Analyzing captured traffic after scan for devices that has UDS SecurityAccess:

![UDS Scan Demo on Emulator](image)

Output:

- ID: 0x701 Service: 0x27 Sub: 0x1 (Send request)
  - Request: 
  - Response: 3d171440
In real life it is not so simple as described in books:

- padding,
- UDS offset is not 8...

And more devices were found...
#UDS tester tools sniffing results

Sent by Anton Sysoev

Output:

UDS Detected:

ID: 0x70a Service: 0x10 Sub: 0x3 (Extended Diag Session)
Request: 
Response: 003201f4

ID: 0x70a Service: 0x22 Sub: 0x01 (Read Data By Identifier)
Request: 00
Response: 045556574550485641313456573338383030303030
ASCII: .EV_FHMAY1W3600000.

ID: 0x70a Service: 0x22 Sub: 0x6 (Read Data By Identifier)
Request: 01
Response: 0103

ID: 0x70b Service: 0x10 Sub: 0x3 (Extended Diag Session)
Request: 
Response: 003201f4

ID: 0x70b Service: 0x22 Sub: 0x01 (Read Data By Identifier)
Request: 00
Response: 045556574550485641313456573338383030303030
ASCII: .EV_RDKB0U30.

ID: 0x70b Service: 0x22 Sub: (ALTERNATIVE INTERPRETATION if no SUB) (Read Data By Identifier)
Request: 0001
Error: Request out of range

ID: 0x70c Service: 0x10 Sub: 0x3 (Extended Diag Session)
Request: 
Response: 002201f4
#UDS SecurityAccess auth sniff

Sent by Anton Sysoev

Auth.PIN = 0x20d3 – 0xfd1 = 0x1102

Seed from ECU...

Auth. code from tester to ECU...
#UDS SecurityAccess and immo bypass on CANToolz emulator

DISCLAIMER: this is not a REAL hack, not real data, and even not a real situation. This is just a simulation of what could be in real, So this is just my imagination and bugs in my VIRTUAL car, only for education purposes!

SCENARIO:
1. Attacker drilled a door and connect to the CABIN bus. Now they can unlock doors. (probably they need to activate bus, but in my emulator let’s think we have the bus active all the time)
2. Attacker has no immo keys, which are unique. But he could use UDS to reset these keys to his own (not a real situation right now, I hope). Because Immobilizer keys are unique for each car but SecurityAccess code is not, it could be shared and could be known (by RE). Access to OBD2 is enough for UDS access...
3. Now an attacker can sniff VIN from the same CAN bus and try to start engine with “new keys”.

DEMO
#CAN Reverse Engineering/BlackBox analysis

- With what command I can unlock car?
- What are door status signals?
- How to get RPM?
- And all things like that: data analysis

Simple techniques, like DUMP and BINARY-SEARCH via REPLAY... but then I understood that I could do it better in CANToolz...
#BUS search
#CANToolz: DIFF method

Frames that probably related to ACTION

- New ID
- New DATA values

Also we could set-up a filter for amount of frames in the second set with “new values”

CAN frames set 1: Recorded dump of CAN traffic. CAR in one stated, nothing happened. I call it ‘noise’.

CAN frames set 2: another traffic dump, when tester did an action (door unlock)

*All next examples and demos are search for LOCK/UNLOCK
#CANToolz: DIFF method - demo

Diff between two sets, with filter:
- only 2 values detected in new set
- Lock/Door statuses: CAN command for lock

Diff between two sets, with filter:
- only 1 value detected in new set
CANToolz: DIFF method - demo

Diff between two sets, with filter:
only 2 values detected in new set

Diff between two sets, with filter:
only 1 value detected in new set

Hmmm... diff works, but you still need more analysis
#CANToolz: STATS. ABNORMALITIES METHOD

Stage 1. Learning.

Each CAN frame for chosen ID has a profile:

- Bits that has been changed (max 64 bits)
  
  \[ \text{BIT\_MASK} = \text{BIT\_MASK OR PREVIOUS\_FRAME\_DATA\_BITS XOR NEW\_FRAME\_DATA\_BITS} \]

- Minimum time between CAN frames with same ID

Stage 2. Compare.

Each CAN frame for chosen ID has a profile:

- In bit mask of changed bits new bits detected.
- Time between frames less than minimum.

Stage 3. Correlation.

Find dependences on those changes between different ID (correlation). Remove all other changes from the result.

\[ \text{EVENT/ACTION SESSION selection!} \]
### #CANToolz: STATS. ABNORMALITIES METHOD DEMO

Extracted event as session

#### SELECTED SESSION (ready to dump into file now and for ACTIVE check):

<table>
<thead>
<tr>
<th>TIME</th>
<th>ID</th>
<th>LENGTH</th>
<th>MESSAGE</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2089</td>
<td>0x2FC</td>
<td>7</td>
<td>21000000000000</td>
<td>first event, changed from 00810000000000000000</td>
</tr>
<tr>
<td>2.2091</td>
<td>0x2a0</td>
<td>8</td>
<td>22222201fffffffff</td>
<td>first change from 8888881ffffffffff, probably because of</td>
</tr>
<tr>
<td>2.2125</td>
<td>0xe0</td>
<td>8</td>
<td>22222000000000f0c0ff</td>
<td>first change from 1111000000000fc0f0ff, probably because of</td>
</tr>
<tr>
<td>2.2575</td>
<td>0x2FC</td>
<td>7</td>
<td>22000000000000</td>
<td>additional changes, probably because of: ['0x2fc', '0x2e']</td>
</tr>
<tr>
<td>2.3077</td>
<td>0xea</td>
<td>8</td>
<td>82ffe7fcffffffff</td>
<td>first change from 01ffe7fcffffffff, probably because of</td>
</tr>
<tr>
<td>2.3079</td>
<td>0xe2</td>
<td>8</td>
<td>f2ffe7fcffffffff</td>
<td>first change from f1ffe7fcffffffff, probably because of</td>
</tr>
<tr>
<td>2.3081</td>
<td>0xee</td>
<td>8</td>
<td>f2ffe7fcffffffff</td>
<td>first change from f1ffe7fcffffffff, probably because of</td>
</tr>
<tr>
<td>2.3522</td>
<td>0x2a0</td>
<td>8</td>
<td>22222201fffffffff</td>
<td>'impulse' rate increased abnormally: EVENT</td>
</tr>
<tr>
<td>2.3678</td>
<td>0x2FC</td>
<td>7</td>
<td>22000000000000</td>
<td>'impulse' rate increased abnormally: NEW STAGE</td>
</tr>
<tr>
<td>2.3722</td>
<td>0xe0</td>
<td>8</td>
<td>22220000000000f0c0ff</td>
<td>'impulse' rate increased abnormally: EVENT</td>
</tr>
<tr>
<td>2.408</td>
<td>0x2FC</td>
<td>7</td>
<td>22000000000000</td>
<td>'impulse' rate increased abnormally: EVENT</td>
</tr>
<tr>
<td>2.4677</td>
<td>0x2FC</td>
<td>7</td>
<td>22000000000000</td>
<td>'impulse' rate increased abnormally: EVENT</td>
</tr>
<tr>
<td>2.5228</td>
<td>0x2a0</td>
<td>8</td>
<td>22222201fffffffff</td>
<td>'impulse' rate increased abnormally: EVENT</td>
</tr>
<tr>
<td>2.5326</td>
<td>0xe2</td>
<td>8</td>
<td>22220000000000f0c0ff</td>
<td>'impulse' rate increased abnormally: EVENT</td>
</tr>
<tr>
<td>2.5679</td>
<td>0x2a0</td>
<td>8</td>
<td>88888881fffffffff</td>
<td>released value back</td>
</tr>
<tr>
<td>2.6313</td>
<td>0xea</td>
<td>8</td>
<td>f2ffe7fcffffffff</td>
<td>first event, changed from f1ffe7fcffffffff</td>
</tr>
<tr>
<td>2.7211</td>
<td>0x2FC</td>
<td>7</td>
<td>82000000000000</td>
<td>'impulse' rate increased abnormally: EVENT</td>
</tr>
<tr>
<td>2.8823</td>
<td>0x2FC</td>
<td>7</td>
<td>82000000000000</td>
<td>additional changes, probably because of: ['0xea6']</td>
</tr>
</tbody>
</table>
#CANToolz: STATS. ABNORMALITIES METHOD

Stage 4: automatic detection.

Automatic replay one by one CAN frames from extracted session and then check if OTHER ID have changed BIT_MASK, like on stage 2.

If this happened then it is mean that our last replayed frame caused those changes and it was our target!

DEMO on emulator

Probably STATUS messages

Lock button pressed and released...?
#CANToolz: features in-dev

- Automatic fields extractor based on changed bits rate
- Automatic data-type identification: signal, counter, integers, ASCII

Those features already in CANToolz partially and could work for some limited situations. But still not stable and could produce “false positives”.
#CANToolz: fields detection
**#CANToolz: meta-data**

**Meta-data** — project’s ‘notes’. This gives tester ability to set labels and some data-extraction rules for already known data.
# CANToolz: loops detection with counters and ASCII

Looks like VIN...
#Other side

- Validation tests
- Fuzzing
  Also usable by researchers, but only when you know WHAT are you fuzzing! Do not do it in real CAR’s bus without understanding what you are doing!
- Research and developing (prototyping and tests)
#CANGateway scan

Check/validate routing/filtering rules, in unit-test/silenium style...

Test frames:
- Frames that should be routed
- Frames that should be modified and routed
- Frames that should not be routed
- Random set of frames (that should not be routed)

Here we do validation

DEMO ON EMULATOR
#Fuzzing

**WARNING, again: only if you know what are you doing.**

- Fuzz known bytes/bits (if “2” – lock, “1” – unlock, then 3 - ?)

- SecurityAccess bruteforce (almost stupid idea, most ECU has anti-bruteforce, but DoS/Reset+Bruteforce maybe could work)

- DoS/Reset combinations...

**Problem:** No debugger attached, so how can we get ‘feedback’ from the ‘crashed’ ECU?

- Attach a debugger (captain!)
- *Check abnormalities in the CAN traffic after Fuzz sample has been sent (slower):*
  - Signal lost
  - New pattern/mask of changed bits
  - Time delay between signals changed
- Check if normal typical CAN ‘request’ causes the same type of ‘response’ (if applicable)

(all these solutions are not implemented in CANToolz yet, but...actually we can do something... demo)

Found interesting paper with close ideas: [https://www.escar.info/images/Datastore/2015_escar_EU_Papers/3_escar_2015_Stephanie_Bayer.pdf](https://www.escar.info/images/Datastore/2015_escar_EU_Papers/3_escar_2015_Stephanie_Bayer.pdf)

Stephanie Bayer, Alexander Ptok ©
#AntiTheft prototype (DIY)

WARNING: only if you know what are you doing. If your ‘device’ crashes then you will lose your engine connection!

1. Prototype in CANToolz
2. Testing on real CAR in CANToolz (MITM in the bus)
3. Creating own device based on any available hardware platform, like Arduino.
4. Deploying into the bus (cut-in, MITM)
#AntiTheft prototype (DIY)
WARNING: only if you know what are you doing.

1. Prototype in CANToolz
2. Testing on real CAR in CANToolz (no MITM)
3. Crating own device based on any available hardware platform, like Arduino.
4. Deploying into the bus (parallel connection)

* Anyway good antitheft system is not ONLY CAN based... just as PoC
#IDS


© Jun Li
#Questions...

Contributors:

Sergey Kononenko
@kononencheg

Boris Ryutin
@dukebarman

Svetlana Sintcova

If you think this project could be helpful for you:
Contributors are WELCOME!
Testers are WELCOME!
Developers/users/researchers are welcome!