Script Your Car
2013-12-28
Felix Domke
30C3
7.0 °C
76505 km
579.1 trip
Why Would Anyone Want to...?

- I love hacking. I love programming. I love Python.
- I don’t love cars.
- But I’m spending way too much time in a car to not try to combine these.
- Also this:
Let’s Hack the Car...

• “If You Can’t Open It, You Don’t Own It.”™
• Yeah, but I’m not a car mechanic. Plus, the car better still works after I’m done (or I’ll be done).
• What’s more interesting than an engine?
• ADDING PYTHON TO YOUR CAR.
A python in your car.

http://stuffaroundyou.blogspot.de/2012/05/picture-of-python-in-car-engine.html
Python in your car.
Entry Point: Bluetooth Kit?

- Why would hacking the bluetooth kit be interesting at all?
  - It shows up in the car’s dashboard menu.
  - It supports Internet. (And Forbes said that Internet-enabled cars are the future!)
  - It can play audio.
  - If it breaks, I can still drive.
Where Are We?

- Most cars have “schematics” available.
  - Well, mostly wiring diagrams, but still.
- Volkswagen/Audi/Seat/Skoda has “erWin”
  - Allows downloading official documents for a small fee. No need to pirate them!
A (Semi-)Modern Car.
A (Semi-)Modern Car

Powertrain CAN (A-CAN) 1 Mbit/s

Radar Sensors
ECU
ABS
Airbag

Convenience CAN (K-CAN) 125 kbit/s

Doors
Park Distance Control
Steering Wheel
HVAC

Infotainment CAN (I-CAN) 100 kbit/s

Radio/Navigation
Phone Kit
Audio Amp
Rear Cam Ctrl.

CAN Gateway

OBD-II

Saturday, December 28, 13
novero GmbH

• For the Volkswagen-Group (VW/Audi/Seat/Skoda...), current Bluetooth Kits are built by Novero GmbH.

• Novero GmbH was Nokia’s Automotive Group, but split out in 2008.
• ARM11
• Linux and WinCE supported
  • (Let's hope it's Linux)
• Why would they need so much power?
i.MX 31

- ARM11
- Linux and WinCE supported
- (Let's hope it's Linux)
- Why would they need so much power?
HT-4 Features (User)

- It bridges a remote SIM card to the mobile phone via rSAP.
- It allows a 3G DUN connection via Bluetooth.
- It does speech recognition and synth.
- It plays A2DP.
HT-4 Features (Hacker)

- RSAP: It has software control over the SIM card
- DUN: It has software control over the PPP session
- Voice Control: It has a lot of CPU power
- A2DP/HFP: It can play audio, and receive audio
“Hacking In My Car”

- “Let’s just get a laptop and a handheld scope.”; yeah, did you ever try that?
- It’s cold!
- Tradeoff: CO poisoning or a dead battery
- ... can I brick it?
- Urg. Let’s rather fix that first...
“Hacking In My Car”

• “Let’s just get a laptop and a handheld scope.” yeah, did you ever try that?

• It’s cold!

• Tradeoff: CO poisoning or a dead battery

• ... can I brick it?

• Urg. Let’s rather fix that first...
Novero HT-5

- Got a “spare” bluetooth kit, an HT-5
- HT-5 is used in MY’14 cars, can be retrofitted into any car with an HT-4
- Additional end-user features: WiFi sharing
- Additional hacker features: WiFi access point, builds up own PPP connection, Router+NAT (hence more likely to use real OS)
- Turns out - different hardware platform.
Marvell 88CP95N

- 88CP955 gives a few more results
- Seems to be a PXA955 + Communication
- Cortex-A8 at ~1GHz
- 3G modem
- Found in some Android tablets
Marvell 88CP95N

- 88CP955 gives a few more results
- Seems to be a PXA955 + Communication
- Cortex-A8 at ~1GHz
- 3G modem
- Found in some Android tablets
Let’s Boot it on My Desk.

- Plan:
  - Connect 12V.
  - Hope it boots.
  - Find UART.
  - Hope it’s Linux.
  - Install Python.

Saturday, December 28, 13
Car Interface

- 54 pin connector
- Pinout can be derived from car schematics
- GND, 12V, CAN_H/CAN_L available
Powering Up

- Demo: what happens when I apply 12V?
- Spoiler: nothing
- Well, the S12X is powered, the main CPU is not.
- Looks like we have to send... the right CAN message first?

Freescale MC9S12XEP768CAG

Marvell 88CP95N-BKA2

SRAM

Marvell 88PM8607 Power Management
CAN Bus Crash Course

Device 1

Device 2

Device 3

120Ω Termination

120Ω Termination

CAN_H

CAN_L
CAN Bus Crash Course

CAN_H
CAN_L

0 - Dominant
1 - Recessive

5V (driven)
2.5V (non-driven)
0V (driven)

dominant bit wins!
CAN Bus Crash Course

- Base frame format:
  - 11-bit Identifier (also used for priority)
  - 0 to 8 bytes of data
  - And lots of other stuff the Transfer Layer cares about (but we don’t).
Make This Work kthx.

- Revised Plan:
  - Connect CAN while device in car
  - Capture CAN traffic
  - On desk, replay CAN traffic
  - Hope it boots.
  - UART, Linux, Python etc.
CAN PC Interface

- “Any” CAN interface adapter will work.
- I used a LAWICEL CANUSB, which is... okay.
- (Has a rather low packet rate limit, but OK for our purpose.)
- Any microcontroller with native CAN support will work, but please don’t try to bitbang.
CAN PC Interface

• “Any” CAN interface adapter will work.
• I used a LAWICEL CANUSB, which is... okay.
• (Has a rather low packet rate limit, but OK for our purposes)
• Any microcontroller with native CAN support will work, but please don’t try to bitbang.
Emulating Vehicle Power

- Also called “S-Kontakt” or “KL15” (Klemme 15, DIN 72552)
- Let’s try!
- Device draws reasonable power now!
- Isolated CAN message:
  - ID=661, “03 00 00 00 00 00 00 00 00”
Now, WTF?

```
[ 0.000000] Console: colour dummy device 80x30
[ 0.000000] ram_console: buffer (null), invalid size 0, datasize 4294967284
[ 0.000213] Calibrating delay loop... 593.48 BogoMIPS (lpj=289792)
[ 0.019989] pid_max: default: 32768 minimum: 301
[ 0.020080] Mount-cache hash table entries: 512
[ 0.020416] CPU: Testing write buffer coherency: ok
[ 0.021026] devtmpfs: initialized
[ 0.026031] regulator: core version 0.5
[ 0.026184] NET: Registered protocol family 16
[ 0.026214] Tauros2: Disabling L2 prefetch.
[ 0.026245] Tauros2: Disable L2 write buffer coalescing
[ 0.026245] Tauros2: L2 cache support initialised in ARMv7 mode.
[ 0.028686] regulator_init: select saarb v12 ldo map
[ 0.029876] create_proc_file PMIC_ID proc file created!
[ 0.029907] create_proc_file Audio Power proc file created!
[ 0.030883] hw perfevents: no hardware support available
[ 0.035919] bio: create slab <bio-0> at 0
[ 0.036804] SCSI subsystem initialized
[ 0.037261] s12x_probe called
[ 0.037414] Device spi1.0 probed
[ 0.037597] S12X Character device sucessfully created
[ 0.037811] usbcore: registered new interface driver usbfs
[ 0.037902] usbcore: registered new interface driver hub
```
GPL

• Oh, why didn’t I see the GPL in the owner’s manual?

• More importantly: Who has to send me the source code?

• It’s complicated. But IANAL.
Local Access == root?

• Boots into Linux - but we don’t know the root password. "$1$JN.iQytI$b1EbtEaRL2xSgZVri6dU/" if you have some spare time.

• Flash modification would help, but meh, BGA flash. Such an effort.

• U-Boot used in non-secure configuration (allows entering console by hammering ^C)

• Traditional “init=/bin/sh” trick should work?
Local Access == root!

- It uses initrd. So it’s “rdinit=/bin/sh”.
- Doesn’t work. We don’t know the FS layout. (And there’s no /dev/console)
- U-Boot allows dumping memory, so we manually extract the initrd.
- Initscript, yay. Directly booting real rootfs, then `init=/bin/sh` works. `mount -o remount,rw /; passwd` and there we go.
Hardware Tricks

- HT-5: Micro-SD slot can be added and “just works”. Even easier!
- SIM slot (requires reconfiguration via Diag)
- USB OTG on ext. accessible Micro-USB
  - Firmware Upgrade, Diagnostics
  - HT-4 host-only, HT-5 is real OTG
  - serial by default, can be changed to usb-ethernet
Emulating The Display

- To hack on the desk, we need to emulate the car.
- Or at least:
  - Steering wheel buttons
  - Display
  - Everything to keep it alive
S12X

• I was hoping to find CAN messages arriving in Linux, and then being parsed by a binary with symbols and excessive debug spew. No such luck.

• Finding “other” end on device for CAN messages proved... interesting.

• The S12X abstracts all of that to a very high level.
BAP

• Very simplified:

• BAP allows a control unit to provide “values” (like “screen content”), and a display unit to use these “values”.

• BAP caches and synchronizes changes, and manages lifecycle (heartbeat, errors) in a well-defined way.
BAP

- “Bedien- und Anzeigeprotokoll” - German Engineering is in da house!
- Hard to find anything more technical than THIS, but it has OSI layers, so it must be good:

<table>
<thead>
<tr>
<th>OSI Layer</th>
<th>BAP Name</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Layer</td>
<td>BAP Application Layer</td>
<td>“Value X is Y”</td>
</tr>
<tr>
<td>Presentation Layer</td>
<td>BAP Application Layer</td>
<td></td>
</tr>
<tr>
<td>Session Layer</td>
<td>BAP Protocol Layer</td>
<td>“Update Value X”</td>
</tr>
<tr>
<td>Transport Layer</td>
<td>BAP Communication Layer</td>
<td>Single BAP messages</td>
</tr>
<tr>
<td>Network Layer</td>
<td>CAN or LIN</td>
<td>CAN messages</td>
</tr>
<tr>
<td>Data Link Layer</td>
<td>CAN or LIN</td>
<td></td>
</tr>
<tr>
<td>Physical Layer</td>
<td>CAN or LIN</td>
<td></td>
</tr>
</tbody>
</table>
Reversing BAP

<table>
<thead>
<tr>
<th>ID</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>62c</td>
<td>80374c0103002f00</td>
</tr>
<tr>
<td>62c</td>
<td>c0030108003803cf</td>
</tr>
<tr>
<td>62c</td>
<td>c1ff000000000a02</td>
</tr>
<tr>
<td>62c</td>
<td>c20008001cc00b00</td>
</tr>
<tr>
<td>62c</td>
<td>c300000000030001</td>
</tr>
<tr>
<td>62c</td>
<td>c40000fffffffffffffff</td>
</tr>
<tr>
<td>62c</td>
<td>c5ffff00000000ff</td>
</tr>
<tr>
<td>62c</td>
<td>c6ffff0100000200</td>
</tr>
<tr>
<td>62c</td>
<td>c70000</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>62c</td>
<td>0c0203002f000301</td>
</tr>
</tbody>
</table>

Opcode=4
LsgId=48
FctId=1

03002f00030108003803cfff00000000
0a020008001cc00b
0000000000030001
0000fffffffffffffff
ff00000000fffffff
01000002000000

Opcode=0
LsgId=48
FctId=2

03002f000301
<table>
<thead>
<tr>
<th>Direction</th>
<th>Timestamp</th>
<th>CanId</th>
<th>Opcode</th>
<th>LsgId</th>
<th>FctId</th>
<th>Data</th>
<th>Text</th>
</tr>
</thead>
</table>
| FROM SG   | 15:58:29  | 6Ah   | 41     | 2     | BAP_Config | 0300290000300 | ...
| FROM SG   | 15:58:21  | 66h   | 43     | 2     | BAP_Config | 03002b000301  | +...
| TO SG     | 15:57:09  | 67h   | 43     | 1     | GetAll    | 0            |      |
| FROM SG   | 15:58:25  | 66h   | 43     | 15    | Fst-Operation State | 00     |      |
| FROM SG   | 15:57:09  | 66h   | 43     | 1     | GetAll    | 0            |      |
| TO SG     | 15:57:09  | 67h   | 43     | 16    |            | 0003        |      |
| FROM SG   | 15:57:09  | 66h   | 43     | 17    |            | 0003        |      |
| TO SG     | 15:57:10  | 67h   | 43     | 17    |            | 0100        |      |
| FROM SG   | 15:58:27  | 66h   | 43     | 17    |            | 0100        |      |
| TO SG     | 15:57:27  | 67h   | 43     | 18    |            | 0207080000000000 | ..... |
| FROM SG   | 15:58:19  | 66h   | 43     | 19    |            | 0100000701ff   | ..... |
| TO SG     | 15:57:26  | 67h   | 43     | 19    |            | 0100000701ff   | ..... |
| FROM SG   | 15:58:29  | 66h   | 43     | 23    |            | 01          |      |
| FROM SG   | 15:58:20  | 63h   | 41     | 3     | FunctionList | 3803ffce000 | 8.... |
| FROM SG   | 15:58:28  | 66h   | 43     | 18    |            | 0207080000000000 | ..... |
| FROM SG   | 15:58:30  | 67h   | 43     | 23    |            | 01          |      |
| FROM SG   | 15:58:19  | 66h   | 43     | 22    | CurrentScreen | 01006464 |      |
| FROM SG   | 15:58:26  | 66h   | 43     | 20    | ScreenData | 01000007000400800c54656c6564 | ..... |
| TO SG     | 15:58:26  | 67h   | 43     | 21    |            | 01          |      |
| FROM SG   | 15:58:29  | 66h   | 43     | 19    |            | 010001070101 |      |
| FROM SG   | 15:58:21  | 63h   | 41     | 4     | Heartbeat  | 0a          |      |
| FROM SG   | 15:58:22  | 63h   | 41     | 14    | Fsg-Setup  | 02          |      |
| FROM SG   | 15:58:23  | 63b   | 41     | 15    | Fst-Operation State | 00     |      |
| FROM SG   | 15:58:24  | 63h   | 41     | 16    |            | 1cc0ff3f07000000 | ...? |
| FROM SG   | 15:58:25  | 63h   | 41     | 17    |            | 00000000     | ..... |
| FROM SG   | 15:58:23  | 66h   | 43     | 3     | FunctionList | 3801ff0000000000 | 8.... |
| FROM SG   | 15:58:26  | 66h   | 41     | 22    |            | 024f00      | ..... |
| FROM SG   | 15:58:23  | 66h   | 41     | 4     | Heartbeat  | 0a          |      |
| FROM SG   | 15:58:27  | 63b   | 41     | 25    |            | 00          |      |
| FROM SG   | 15:58:28  | 63h   | 41     | 27    |            | 00000000     | ..... |
| FROM SG   | 15:58:30  | 66h   | 41     | 21    |            | 00          |      |

**Note:**
- The highlighted text appears to indicate a message about a telephone being started:
  `...Telefon wird...gestartet.........`
KI Simulator

• Demo KI Simulator
D-Bus

• D-Bus used very extensively
• S12X messages are posted on D-Bus
• We see screen updates, key presses, bluetooth events...
• Only very high level view of CAN, though.
• NAVPOS, Speed signals are visible.
Higher-Level menus

- Manages screen priorities, key presses for selection updates
- Localizes and converts screen data to what the car expects
- Sends them to S12X -> CAN
import dbus

session_bus = dbus.SessionBus()

s12x = session_bus.get_object("com.nokia.s12xrouter", "/com/nokia/s12xrouter")

s12x.ScreenData(dbus.ByteArray("008001000070004000000004043300120000800b48656c6c6f20576f726c64330007010080003300070200800033000703008000".decode('hex')))}
Regular Data Flow

bluetoothprovider

callprovider

settingsprovider

genericdisplay

kpbddisplay

S12X

BAP

CAN

Unlocalized Screen Data

Screen Data

Keys

"Screen Id"

Hijacked Data Flow

bluetoothprovider

??
callprovider

"Screen Id"

settingsprovider

genericdisplay

Unlocalized Screen Data

kpbdisplay

Screen Data

S12X

hello.py

BAP

CAN
Ok, but...

• Conflicts with original Menu
• a few more lines required...
  • create logic screen and set as active
• All key presses will be forwarded to us
• no more screen updated from original logic
class MainMenu(Menu):
    HEADER = ["My Custom Menu"]
    MENU_ENTRIES = ["Choice 1", "Choice 2", "Exit"]

    def __init__(self):
        Menu.__init__(self, self.HEADER, self.MENU_ENTRIES)

    def selected(self, result):
        print "MAIN MENU SELECTED, res=", result
        if result is None or result == 2: # back or exit
            self.closed = True
            self.result = None
        else:
            print "menu menu invoke", result

bap_display = BapDisplay(genericdisplay, s12x, MainMenu())
bap_display.ActiveHandleChangedNotification(dbus.UInt32(2))
mainloop = gobject.MainLoop()
mainloop.run()

File helloworld.py saved.
root@Gemini:~ $ python helloworld.py
Why This?  (and not a car PC)

• Affordable, automotive-proven hardware
• Lifecycle management (no battery draining!), clean startup and shutdown
• Well-behaving lower layers implemented - misbehavior on the Linux side should never do any damage due to abstractions
Can This Brick My Car?

- You won’t hear me saying “no”.
- Cars are designed for safety, not security.
- The S12X has fairly extensive sanity checking - sending too many or too few messages will fail “gracefully”.
- However, if you intentionally tunnel the right messages through the gateway, ...
“[…]
Ich hatte nach einer Anleitung im Internet einen Adapter gebaut, der während der Fahrt Daten aus den div. Steuergeräten ausliest und die dann in einem Datenlogger auf dem PDA ablegt. […]
Irgendwann auf dem Weg von Frankfurt nach Cuxhaven ist dann […] erst die Instrumententafel ausgefallen. […] nachdem die die Instrumente wieder Werte gezeigt haben […] ging der Motor aus, die Knöpfe an den Türen hoch und die vorderen Airbags gingen alle auf. (Fahrer-/ Beifahrerairbag, Sitzairbag und Gurtstraffer)
[…]
Der Wagen war Schrott (Beide B-Säulen von Gurtstraffern verzogen […] und irgendwie kam der Gutachter von der Versicherung dahinter, das der gespeicherte Unfall mit Seitenaufprall nicht über Beschleunigungssensoren im Airbag-Steuergerät kam. Dann natürlich noch die ganze Elektronik mit den Datenkabeln zum Diagnosestecker hinter der Instrumententafel.

War also ein teures Experiment, weil die Versicherung keinen Cent bezahlt hat und der Wagen noch einen Zeitwert von 15.000 Euro hatte.”

http://www.mikrocontroller.net/topic/33809#247235
For The Bad Guys...

• Access to the microphone
• Access to the Internet
• Access to GPS

• Ideal hardware for a tracking bug!
• Short physical access to unit is enough. (But so is for deploying a dedicated bug.)
For The Good Guys...

- All sorts of geofencing applications:
  - home automation
  - electronic parking tickets (which may be a bad idea by itself)
- Weather
- Traffic (Google/Bing traffic to TMC FM modulator, anyone?)
Thanks!

• tmbinc@elitedvb.net
• github.com/tmbinc/car