



I'm!

Andrei Borcea

20 anni . Grandissimo appassionato di tecnologia fin da piccolo. Amo programmare e risolvere problemi. Solitamente lavoro con Python, C, C++, Java, C#, Javascript (Node e tecnologie varie lato web), SQL, Assembly. Amo ambiente linux lato server ma uso windows quotidianamente (non mi odiate)

Sono un appassionato di sicurezza e passo il tempo a trovare bug,problemi nei codici degli altri.Ho basi minime di Assembly in versioni x86,x64,ARM,ELF.Il reverse engineering è il mio amico da sempre.

Attivo nella comunità del jailbreak Apple, PlayStation jailbreak e tool di utilità generale.

Tool: IDA,Radare2,CFF Explorer,API Monitor,HexRays,Visual Studio e la suite completa di Jetbrains

- Github:@hustlercoder
- Twitter:@hustlercoder



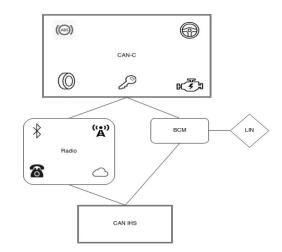
DEFINIZIONI

CAN BUS: Il Controller Area Network, noto anche come CAN-bus, è uno standard seriale per bus di campo (principalmente in ambiente automotive), di tipo multicast, introdotto negli anni ottanta dalla Robert Bosch GmbH, per collegare diverse unità di controllo elettronico (ECU).

ECU: una **unità di controllo elettronico** (*electronic control unit* o *ECU* in inglese) è un sistema di controllo software che è integrato direttamente nel sistema che controlla. Solitamente la chiamiamo centralina.

CAN-C: sottoinsieme del CAN BUS,questa è la parte a basso livello del CAN BUS che si occupa del controllo di sistemi come ABS, iniettori, ESP e altre tecnologie.

CAN-IHS: seconda categoria di CAN BUS che si occupa della parte del controllo dei dispositivi di confort: AC,WIFI,Bluetooth,Comandi al volante.ecc



Message frame

| Field name | | Priority | | | | | Message type ID | | | | | | | | | | | | Service not message Source node ID | | | | | | | | | | |
|----------------|----|----------|----|----|----|----|-----------------|----|----|----|----|----|----|----|----|----|----|----|------------------------------------|---|---|---|---|---|---|----|----|---|---|
| CAN ID bits | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Allowed values | | | | | | 8 | | | | | | | | 57 | | | | | | | | 0 | | | 1 | 12 | 27 | | |
| CAN ID bytes | | | 3 | | | | | | : | 2 | | | | | | | | i | | | | | | | | 0 | | | |

Anonymous message frame

| Field name | | Priority | | | | Discriminator | | | | | | | | Lower bits of message type ID | | | | | | | ID | | Service not message Source node ID | | | | | | |
|----------------|----|----------|----|----|----|---------------|----|----|----|----|----|----|----|-------------------------------|----|----|----|----|----|---|----|---|---------------------------------------|---|---|---|---|---|---|
| CAN ID bits | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Allowed values | | | | | | | | | | | | | | | | | | | | | | 0 | | | | 0 | | | |
| CAN ID bytes | | | 3 | | | | | | | 2 | | | | | | | | 1 | | | | | | | (|) | | | |

Service frame

| Field name | Field name Priority | | | | | | | Sei | vice | tyne | חוי | | | | Re | que | st no | t res | spon | se | | | S | ervio | e no | t me | ssa | ge | |
|----------------|---------------------|----|-------|----|----|----|----|-----|------|-------|-----|----|----|-----|----|-----|-------|-------|------|------|---|---|---|-------|------|------|-------|----|---|
| | | | 1,01, | -, | | | | 00. | *100 | .,,,, | | | | | | Des | stina | tion | node | e ID | | | | S | ourc | e no | ide I | D | |
| CAN ID bits | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Allowed values | | | | | | | | | | | | | | | | | 1 | 12 | 7 | | | 1 | | | 1 | 12 | 7 | | |
| CAN ID bytes | | | 3 | | | 2 | | | | | | | | 1 0 | | | | | |) | | | | | | | | | |

Informazioni sul sistema

- > Basato su sistema operativo QNX 32 bit ARM
 - Microcontrollore Texas Instruments OMAP-DM3730 per comunicazione tra CAN IHS e CAN C
 - > Sviluppato da Harman Kardon(in alcune versioni)
- > File System e servizi
 - IPL(Initial Program Loader) usato per caricare il firmware della ECU
 - IFS-ETFS: IFS contiene l'immagine del sistema operativo con le relative configurazioni, ETFS un file system temporaneo usato nelle operazioni di rw dalla ECU sul bus
 - MMC(Multimedia Card) usato per memorizzare i file temporanei oppure le impostazioni del microcontrollore(vulnerabile ad attacchi di privilege escalation)





Punti vulnerabili

Entry Point

- → RKE
- > TPMS
- Bluetooth
- > FM/AM/XM
- Cellular
- Internet/Apps

ECU

- > RFHM
- > RFHM
- Radio
- Radio
- Radio
- Radio

Bus

- > CAN C
- > CAN C
- > CAN C CAN IHS

ROOT ACCESS INFOTAINMENT

Jailbreaking del sistema infotainment

Necessario:

- emulatore QNX per virtualizzare il sistema operativo e cercare bug
- modificare il codice per rendere possibile l'utilizzo della shell
- D-Bus Python lib
- Reset della lingua e impostarla in inglese







No checksum

Il sistema operativo esegue una verifica sul checksum per confermare che il firmware sia autentico.

Per bypassare questo blocco dobbiamo virtualizzare il firmware con : <u>memifs2 -q -d</u> <u>/fs/usb0/usr/share/swdl.bin /</u> dentro l'ambiente QNX Emulator.

```
Size
                                Name
    Offset
                                *.boot
                                Startup-header flags1=0x9 flags2=0 paddr bias=0
         108
                     22008
                                startup.*
      22110
                                Image-header mountpoint=/
      2216c
                                Image-directory
                                Root-dirent
                       ----
                                proc/boot/procnto-instr
      23000
                     8a000
                                proc/boot/.script
                                                                           602c bin/i2c-omap35xx
               dev/console -> /dev/ser
                                                                  17d000
                                                                           da08 bin/devb-mmcsd-omap3730teb
               tmp -> /dev/shmem
            10 usr/var -> /fs/etfs/usr/var
                                                                           dd3 bin/dev-ipc.sh
            16 HBpersistence -> /fs/etfs/usr/var/trace
                                                                  18c000
               var/run -> /dev/shmem
                                                                           2198 bin/mmc.sh
             a var/lock -> /dev/shmem
                                                                  190000
                                                                          1208f bin/devc-seromap
             a var/log/ppp -> /dev/shmem
            15 opt/sys/bin/pppd -> /fs/mmc0/app/bin/pppd
                                                                  1a3000
                                                                           323d bin/rm
            15 opt/sys/bin/chat -> /fs/mmc0/app/bin/chat
            18 bin/netstat -> /fs/mmc0/app/bin/netstat
                                                                  1a7000
                                                                           ffa2 bin/devc-pty
            16 etc/resolv.conf -> /dev/shmem/resolv.conf
                                                                  1b7000
                                                                            4eb bin/startSplashApp
            16 etc/ppp/resolv.conf -> /dev/shmem/resolv.conf
            18 etc/tuner -> /fs/mmc0/app/share/tuner
                                                                            692 bin/startBackLightApp
             8 var/override -> /fs/etfs
                                                                           1019 bin/mmc chk
             c usr/local -> /fs/mmc0/app
               usr/share/eq -> /fs/mmc0/eq
                                                                           42fe usr/bin/adjustImageState
  b1000
          12af etc/system/config/fram.conf
                                                                  1c0000
                                                                          12c81 usr/bin/memifs2
           38c etc/system/config/nand partition.txt
               etc/system/config/gpio.conf
                                                                  1d3000
                                                                           284 usr/bin/loadsecondaryifs.sh
  b5000
          247b bin/cat
                                                                  1e0000
                                                                          77000 lib/libc.so.3
          1fed bin/io
          2545 bin/nice
                                                                             9 lib/libc.so -> libc.so.3
          216a bin/echo
          38eOf hin/ksh
                                                                  260000
                                                                           b0e4 lib/dl1/devu-omap3530-mg.so
          41bb bin/slogger
                                                                  26c000
                                                                           9d17 lib/dl1/devu-ehci-omap3.so
          531b bin/pipe
                                                                           4705 lib/dll/spi-omap3530.so
          5e02
               bin/dev-gpi
                                                                          14700 lib/dll/fs-qnx6.so
          1f675 bin/io-usb
                                                                           36e6 lib/dll/cam-disk.sc
 160000
          29eb bin/resource seed
          3888 bin/spi-master
                                                                          2b7ba lib/dll/io-blk.so
          48a0 bin/dev-memory
                                                                          5594f lib/dll/charset.so
          9eab bin/dev-mman
                                lib/dll/libcam.so.2
     330000
                                 lib/dll/libcam.so -> libcam.so.2
     350000
                                lib/dll/fram-i2c.so
Checksums: image=0x702592f4 startup=0xc11b20c0
```

Invalidare il checksum

Invalidare il checksum è un operazione piuttosto semplice in questa versione del firmware poiché basta cambiare via hex editor un pezzo del codice.

Bypassare la validazione del firmware si esegue una modifica nella **all offset 128(0x80) in 'S' (0x53)**



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | В | Ç | D | E | F | 0123456789ABCDEF |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------------------|
| 0000h: | 0C | В0 | 47 | 9C | 27 | F6 | 2D | 93 | 3B | EC | 74 | 5A | 8D | A5 | E3 | 98 | .°Gœ'ö-";ìtZ.¥ã~ |
| 0010h: | В6 | 9F | BD | F7 | C8 | 57 | 6B | 89 | C2 | C7 | 6B | E7 | BC | 6B | 21 | EA | ¶Ÿ₩÷ÈWk%ÂÇkç¼k!ê |
| 0020h: | 96 | DC | OC | D8 | DD | F4 | B9 | 9B | 64 | CE | 8F | 8B | 2A | DO | 8A | 47 | -Ü.ØÝô¹>dî.∢*ĐŠG |
| 0030h: | 2F | 44 | F7 | D2 | 3F | 45 | 06 | 4D | 48 | 96 | 9E | 6E | 7D | 7F | 23 | 17 | /D÷Ò?E.MH-žn}.#. |
| 0040h: | 49 | C9 | FE | D1 | F1 | 22 | A0 | 34 | 20 | C6 | DO | 5E | DF | DD | E8 | 14 | IÉÞÑā" 4 ÆÐ^BÝè. |
| 0050h: | A6 | A6 | 2B | 08 | B1 | 47 | 41 | 03 | 79 | 18 | F0 | 8C | 3D | E2 | 6D | BC | +.±GA.y.ðŒ=âm⅓ |
| 0060h: | 49 | 5D | A0 | CE | EB | CE | F7 | C7 | 8A | 1E | A5 | 68 | FB | 8E | 3A | 98 | I] ÎëÎ÷ÇŠ.¥hûŽ:~ |
| 0070h: | 78 | FE | 40 | EA | 10 | 4D | 38 | 30 | 07 | 5A | BC | D4 | E8 | B9 | 1D | 34 | xþ@ê.M80.Z¼Ôè¹.4 |
| 0080h: | 53 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | S |
| 0090h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00A0h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00B0h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00C0h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00D0h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00E0h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00F0h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 0100h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 0110h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 01.001 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |



WIFI Hack

- Crittografia: hard coded credentials e password generate con la data e ora odierna
- > Il chip del wifi vulnerabile a DFU downgrade
- > Nessun controllo dell'input

| # netstat | -n | gr | cep | LISTEN | | |
|-----------|----|----|-----|----------------|-----|--------|
| tcp | 0 | | 0 | *.6010 | *.* | LISTEN |
| tcp | 0 | | 0 | *.2011 | *.* | LISTEN |
| tcp | 0 | | 0 | *.6020 | *.* | LISTEN |
| tcp | 0 | | 0 | *.2021 | *.* | LISTEN |
| tcp | 0 | | 0 | 127.0.0.1.3128 | *.* | LISTEN |
| tcp | 0 | | 0 | *.51500 | *.* | LISTEN |
| tcp | 0 | | 0 | *.65200 | *.* | LISTEN |
| tcp | 0 | | 0 | *.4400 | *.* | LISTEN |
| tcp | 0 | | 0 | *.6667 | *.* | LISTEN |
| | | | | | | |

Bus Daemon Process

La porta 6667 è un IP D-Bus che permette alle applicazioni di comunicare tra loro.

In informatica sono noti come IPC e RPC che sono due tecniche standard di comunicazione e invocazione comandi tra processi.

E importante menzionare il system bus daemon e session bus che sono i più importanti.Tra questi c'è un processo che è fondamentale NavTrailService.

Tool per scripting: D-Bus library Python

```
function methods.rmTrack(params, context)
  return {
    result = os.execute("rm \"" .. trail_path_saved .. params.filename .. "\"")
  }
}
```

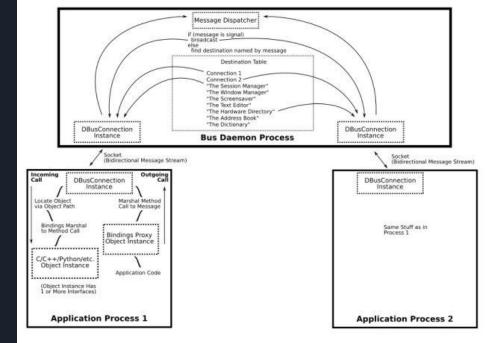


Foto: http://dbus.freedesktop.org/doc/diagram.png

```
$ telnet 192.168.5.1 6667

Trying 192.168.5.1...

Connected to 192.168.5.1.

Escape character is '^]'.

a

ERROR "Unknown command"
```

Dfeet

Basta un semplice comando su Dfeet per avere tutti i servizi attivi sul CAN BUS e tutti i dettagli su come accedere alle loro informazioni.

Le altre applicazioni vengono eseguite sempre sulla porta 6667 e attraverso metodo TCP, però per semplicità mi sono litato al caso base.ll resto è test e studio

https://github.com/GNOME/d-feet



| Execute Invol | e(s method, s parameters) - (s result) |
|--|--|
| Parameters | in python syntax) |
| *getServices* | |
| Output u'{"com.harn | nan.service.platform.launcher":{"name":"com.harman.service.platform.launcher","methods": |
| ("launch": "lau ("stop": "stop ("name": "cor ("getPropert | .inch"}},"com.harman.service.Control":{"name":"com.harman.service.Control","methods": "," getModules":"getModules","start":"start","getServices":"getServices","setDebug":"setDeb n.harman.service.PersonalConfig","methods": ies":"getProperties","getAllProperties":"getAllProperties","setProperties":"setProperties");, |
| {"setSHF_Mu | n.harman.service.psseControlService","methods": te":"setSHF_Mute","getSHF_Scenario":"getSHF_Scenario","setSHF_Scenario":"setSHF_Scenari n.harman.service.readPPS","methods": |
| {"getPropert {"name":"cor | ies"; "getProperties", "getPPSValue"; "getPPSValue"}}, "com.harman.service.LayerManager": n.harman.service.LayerManager", "methods": |
| {"name":"cor {"getPropert | MI2":"viewDTV_HMI2","viewBlackScreen";"viewBlackScreen","viewVES2Input":"viewVES2Input n.harman.service.RemoteStart","methods": ies":"getProperties"}},"com.harman.service.Climate": |
| | n.harman.service.Climate", "methods": |
| Pretty Print | Source |
| | Close Execute |

Prossimo passo?

HVAC (Heating, Ventilation and Air Conditioning)

Cosa possiamo controllare?

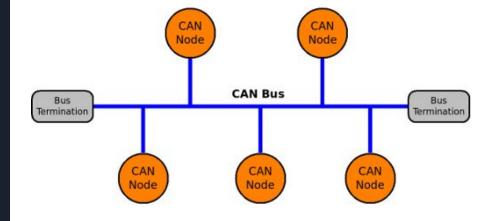
- Aria condizionata
- Radio Volume
- Radio FM
- Bottoni Radio e Riscaldamento

Ritorno al passato

Il CAN BUS permette la trasmissione di 4 tipologie di frame.Ogni tipo di frame è unico e con una struttura ben precisa.

- **Data frame**: frame contenente i dati che il nodo trasmette.
- **Remote frame**: frame che richiede la trasmissione di un determinato identificatore.
- **Error frame**: frame trasmesso da un qualsiasi nodo che ha rilevato un errore.
- **Overload frame**: frame che introduce un ritardo fra data frame e/o remote frame.

Sul CAN BUS si possono trasmettere fino a 2^{29} = 536 870 912.



| Field name | Length (bits) | Purpose |
|--|------------------|--|
| Start-of-frame | 1 | Denotes the start of frame transmission |
| Identifier (green) | 11 | A (unique) identifier which also represents the message priority |
| Remote transmission request (RTR) (blue) | 1 | Must be dominant (0) for data frames and recessive (1) for remote request frames (see Remote Frame, below) |
| Identifier extension bit (IDE) | 1 | Must be dominant (0) for base frame format with 11-bit identifiers |
| Reserved bit (r0) | 1 | Reserved bit. Must be dominant (0), but accepted as either dominant or recessive. |
| Data length code (DLC) (yellow) | 4 | Number of bytes of data (0–8 bytes) ^[a] |
| Data field (red) | 0-64 (0-8 bytes) | Data to be transmitted (length in bytes dictated by DLC field) |
| CRC | 15 | Cyclic redundancy check |
| CRC delimiter | 1 | Must be recessive (1) |
| ACK slot | 1 | Transmitter sends recessive (1) and any receiver can assert a dominant (0) |
| ACK delimiter | 1 | Must be recessive (1) |
| End-of-frame (EOF) | 7 | Must be recessive (1) |

Inviare messaggi sul CAN BUS

Per riuscire a inviare messaggi sul CAN dobbiamo sapere su quale canale comunica i dati.....ne ha di più però quello più frequente è il 7.Il dispatcher fa da ponte tra kernel-mode e user-mode e quindi la funzione che chiamerà sarà diversa...in questo caso corrisponde alla location 0x7EE1C dove jr chiama un address space 4AEA8.

```
0007EDD0
                                                      3[r28], r15
                                                     4[r28], r8
                                                     0x12, r8
                                                     r8, -0x2CC4[gp]
                                                      ROM CAN DATA PTRS, r14 -- ROM CAN DATA PTRS is a list
0007EDE8 2E 37 9D 00
                                                     0x9C[r14], r6
                                                     6, r28, r7
                                                     r15. r8
                                                     r15, r8
                                                     r15, r8
                                                     r15, r8
                                                     0x16C[r14], r6
                                                     memcpy, lp
                                                     2[r28], r6
                                                     r15, -0x2CC8[gp]
                                                     r15, -0x2CC5[gp]
                                                     check stuff then transmit CAN, lp
                                                     0x18, r0, r18
```

```
ROM: 88839D32
ROM: 00039D32 loc 39D32:
                                                          -- CODE XREF: checksum calc+7Ai
ROM: 00039D32
                                        r28, r7
ROM: 00039D34
                                        r26, r7
ROM: 00039D36
ROM: 00039D38
                                         0x80, r2, r6
ROM: 00039D3C
                                        r8, r7
ROM: 00039D3E
                                         1oc 39054
                                         0x10, r0, r7
ROM: 00039D44
                                        r0, r6
                                                          -- if(checksum != 0) { u curr = 1; }
ROM: 00039D46
                                        1oc 39D4A
ROM: 00039D48
                                        1, r7
ROM: 00039D4A
ROM: 00039D4A loc 39D4A:
                                                              ODE XREF: checksum calc+52<sup>†</sup>i
ROM: 00039D4A
                                         1, r2
ROM: 88839D4C
                                        1, r2, r6
                                         r6, r7
                                         loc 39060
```

Final Hack

Cosa possiamo controllare?

- Motore
- Freni
- Sterzo
- Chiusura
- > Indicatori direzione

Hacking 3G/4G



```
# ifconfig
100: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33192
        inet 127.0.0.1 netmask 0xff000000
pflog0: flags=100<PROMISC> mtu 33192
uap0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        address: 30:14:4a:ee:a6:f8
        media: <unknown type> autoselect
        inet 192.168.5.1 netmask 0xffffff00 broadcast 192.168.5.255
ppp0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1472
        inet 21.28.103.144 -> 68.28.89.85 netmask 0xff0000000
```

192.168.5.1 = l'indirizzo gateway per i dispositivi che si connettono al hotspot wifi 68.28.89.85= Indirizzo che serve nel caso la macchina si voglia connettere con qualcuno.La porta 6667 non è aperta quindi è poco utile.

21.28.103.144 = l'indirizzo effetivo con cui i pacchetti escono dalla macchina tramite il NAT.funziona però soltanto con i provider di rete abilitati. Questi sono fissi perché vengono assegnati dal provider della rete(in Italia difficile trovare macchine con 3G/4G), nonostante l'accensione e lo spegnimento della macchina.

Abbiamo finito

- 1. Controllo delle luci (tutte)
- 2. RPMS
- 3. Kill engine
- 4. No brakes
- 5. Steering

Domande?