

CHALMERS



A Security Evaluation and Internal Penetration Testing Of the CAN-bus

Master of Science Thesis in the program Networks and Distributed Systems

SAREH TALEBI

Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, October 2014

The Author grants to Chalmers University of Technology the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology store the Work electronically and make it accessible on the Internet.

Security Evaluation and Internal Penetration Testing of CAN-bus

SAREH TALEBI

© SAREH TALEBI, October 2014

Supervisor: TOMAS OLOVSSON

Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden October 2014

Abstract

From the early ages of civilization, people have always fought to have safety and comfort in all the aspects of their lives. Contemporary vehicles are not an exception. Nowadays, vehicles contain a number of electronic control units (ECUs) which form networks and provide many different functions. In order to let the driver benefit from the new technology, the automotive manufacturers have created a strong relationship between the vehicle and the fleet management. Remote diagnosis and *firmware updates over the air (FOTA)* are some examples of services brought by the new technology in order to involve a minimum of customer inconvenience. This approach brings a considerable number of advantages: the vehicle needs no longer to be brought in a service station, the update of a firmware is made as soon as it is released, and the time between discovering an error and identifying its causes is reduced. But, in order to assure these functions, in-vehicle networks are connected to external networks, a fact that exposes them to dangerous threats such as cyber attacks.

The in-vehicle network consists of a number of networks where each of them has different impact on the vehicle's mechanism. These networks are formed by a number of ECUs and are used differently. For instance, MOST (media oriented system transport) is used in to transmit voice, audio, and video content. LIN (local interconnected network) is responsible for controlling door locking mechanisms, windows and mirrors. For critical applications, such as engine control and anti-lock braking system (ABS), CAN (controller area network) is used.

This master thesis evaluates the security in in-vehicle networks by focusing on the CAN-bus protocol, since the most critical applications use this protocol for communications. In order to perform this evaluation, the development of a framework for conduction a penetration test is done.

Keywords

In-Vehicle networks, Connected Car, Controller Area Networks (CAN), Penetration Test.

Acknowledgments

I would like to warmly thank my supportive supervisor and examiner Tomas Olovsson and Erland Jansson at Chalmers who kindly helped me with their comments, advices and viewpoints from the very first steps of starting this master thesis.

I am also extremely grateful to Erik Ceder and Patrick Hylén, two Volvo managers. They gave me the opportunity to get access to boxcars and rigs to perform part of the practical work. As a student, it was such a unique experience to be at Volvo. I am so thankful to have such a fortune which was definitely impossible without their favor to me.

My sincere gratitude goes to my wonderful parents and beloved brothers, Nader and Sepehr, Who have always encouraged me to pursue every single one of the dreams I have had no matter how outrageous they were. I will be always indebted for being there for me exactly when I need. You have made an amazing family to me and given me exactly what I need; love, support and complete understanding. I love you.

Contents

Abstract	3
Acknowledgments.....	4
List of Tables	9
Abbreviations	10
1 Introduction	11
1.1 Background	12
1.2 Scope.....	13
1.3 Objectives.....	13
1.4 Limitations.....	14
1.5 Methodology.....	14
1.6 Structure of the report.....	14
2 In-Vehicle Network	15
2.1 Background	15
2.2 In-Vehicle Buses	16
3. Controller Area Network (CAN) Specifications	19
3. 1 History and Application.....	19
3.2 What is CAN?	19
3.3 Overview of CAN's communication	21
3.4 Features	21
3.5 CAN in the ISO/OSI Reference Model	23
3.6 Physical Layer.....	23
3.7 Link Layer	26
4 Security Considerations	39
4.1 Security Properties.....	39
4.2 In-Vehicle Taxonomy.....	40
4.3 ECU classification	44
5. Threats and Attacks	46
5.1 Attacks.....	46
5.1.1 Internal Attacks.....	46
5.1.2 External Attacks	48
6. Penetration test	51

6.1 What is a Penetration Test?	51
6.3 How to perform a penetration Test?	52
6.4 Criteria of Success	54
6.5 Penetration Approaches	54
6.6 Limitations.....	55
7. Practical implementation of the test set-up	56
7.1 Software Simulator	56
7.2 Description of test environment.....	56
7.3 Overview of a CANoe Application.....	59
7.4 Test implementation.....	63
7.4.1 Attacks related to CIA	63
7.4.2 Attacks related to CAN Frame.....	75
7.4.3 Implementation-based flaws	83
8 Countermeasures.....	84
8.1 Intrusion Detection/Prevention Systems.....	84
8.2 Firewalls	84
8.3 Honeypots	84
8.4 Encryption	85
8.5 Authenticity check	86
8.6 Forensics Support.....	86
8.7 Security features related to the network architecture	86
9. Results	88
9.1 Planning and preparation:	88
9.2 Discovery:.....	88
9.3 Attack	89
10. Conclusion.....	92
11. Future Work	93
References	94

List of Figures

Figure 1.Traditional wired infrastructure where the vehicle is brought to a service station and all the tests, updated, configurations and diagnostics are done inside a closed environment [2]	12
Figure 2. Wireless infrastructure model where diagnostics and updates are done over the air [2]	12
Figure 3. Short range communication model where updates and diagnostics are done over the air [3]..	13
Figure 4. In-Vehicle Network	15
Figure 5. CAN effect on decreasing the wire quantity [4]	19
Figure 6. Base Format Data Frame[7]	27
Figure 7.Extended Format Data Frame	29
Figure 8. Remote Frame [8]	30
Figure 9. Error Frame [8]	30
Figure 10. Receiver of the previous message or not an Error Passive station [8]	32
Figure 11. An Error Passive Transmitter Station[8]	32
Figure 12. Data Transmission with Arbitration	33
Figure 13: Medium Access Control [6]	33
Figure 14. Principle of Data Exchange.....	37
Figure 15. LLC Data Frame	38
Figure 16. LLC Remote Frame	38
Figure 17.Computer and Network attacks.....	40
Figure 18. Computer and Network Incident [12]	41
Figure 19.Detailed incident taxonomy adopted to the vehicle environment	42
Figure 20. Read Building Block.....	43
Figure 21. Spoof Building Block.....	43
Figure 22. Drop Building Block	43
Figure 23. Modify Building Block	43
Figure 24. Flood Building Block.....	43
Figure 25. Steal Building Block	43
Figure 26. Replay Building Block	44
Figure 27.4-stage Penetration Testing Methodology [17]	52
Figure 28. Attack Phase of Penetration Testing [17]	53
Figure 29. CAN Network Gateways.....	57
Figure 30. Electronic Control Unit (ECU).....	57
Figure 31. Simulated CAN configuration.....	58
Figure 32. Comfort subnet database	60
Figure 33. PowerTrain subnet database	60
Figure 34. Comfort subnet network nodes.....	60
Figure 35. PowerTrain subnet network nodes.....	60
Figure 36. CAN message structure.....	60
Figure 37.Behaviour of DoorLeft Node	63
Figure 38. Read and Inject Attack - Visual results, radio on	65

Figure 39.Scenario 1, Spoof Attack, ABSdata Message	66
Figure 40. Scenario 2, Spoof Attack, infected ABSdata messages file. The first column shows time of transmission, C9 is the ID transmitting block, and the 6 columns indicate data values.	67
Figure 41 Spoof Attack, First Scenario, Visual results.....	68
Figure 42. Spoof Attack, Second scenario, Result.....	69
Figure 43. Console_2 message.....	70
Figure 44."Turn on headlights" Message.....	71
Figure 45.Replay attack - Visual results	71
Figure 46 DOS Attack, Comfort Network and broken Dashboard	73
Figure 47. DOS Attack, properly working PowerTrain network.....	73
Figure 48. Modify attack, real speed is different with what the indicator shows	75
Figure 49. Sending shortened ABSData messages with all possible DLC values in a real environment.....	77
Figure 50.Sending shortened ABSData messages with all possible DLC values in a simulated environment	77
Figure 51. Part of a filtered traffic of sending a shortened ABSData message in a real environment.....	78
Figure 52.Part of a filtered traffic of sending a shortened ABSData message in a simulated environment	78
Figure 53. Sending a lengthened ABSData with all possible DLC values in a real network	78
Figure 54.Sending a lengthened ABSData with all possible DLC values in a simulated network	79
Figure 55. Part of a filtered traffic of sending a lengthened ABSData message in real network	79
Figure 56. Part of a filtered traffic of sending a lengthened ABSData message in simulated network.....	80
Figure 57. Injecting message with non-existing ID	80
Figure 58. injecting message with non-existing ID	81
Figure 59. Injecting too many ABSData messages to increase the bit-rate (it is part of the injected messages).....	82
Figure 60. Injecting message with high bit-rate in simulated environment.....	83
Figure 61. Secure vehicular communication[3]	85
Figure 62. Secured Message Sending [3]	85
Figure 63. Secured Message Receiving[3]	86

List of Tables

Table 1. Automotive bus comparison	18
Table 2: CAN in ISO/OSI Reference Model [6]	23
Table 3. Attack type and violated Security Property	44
Table 4. SILs according to controllability and probability of a failure [14]	45
Table 5. SEL according to the safety effect of security threads [14]	45
Table 6. SIL values for ECU categories [14]	45
Table 7. CANoe, GNS3 and NETSIM	56

Abbreviations

ABS	1Anti-lock Braking System
ACK	2Acknowledgement
ACL	3Access Control Lists
CAN	1Controller Area Network
CIA	Confidentiality, Integrity, Availability
CRC	2Cyclic Redundancy Code
CSMA	3Carrier Sense Multiple Access
CSMA/CD	4Carrier Sense Multiple Access / Collision Detection
DLC	Data Length Code
DOS	Denial of Service
ECU	Electronic Control Units
FOTA	Firmware updates Over The Air
ID	Identifier
IDE	Identifier Extension
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
KDC	Key Distribution Centre
LIN	Local Interconnected Network
LLC	Logical Link Control
LSDU	Link Service Data Unit
MAC	Medium Access Control
MOST	Media Oriented System Transport
OBD-II	On-Board Diagnostics
PBS1	Phase Buffer Segment 1
PBS2	Phase Buffer Segment 2
PTS	Propagation Time Segment
REC	Receive Error Counter
RTR	Remote Transmission Request
SAE	Society of Automotive Engineers
SEL	Safety Effect Levels
SIL	Safety Integrity Level
SOF	Start of Frame
SRR	Substitute Remote Request
SS	Synchronization segment
TCC	Telematic Call Center
TDM	Time Division Multiplex
TDMA	Time Division Multiple Access
TEC	Transmit Error Counter
TTCAN	Time Triggered CAN
USB	Universal Serial Bus
WMA	Windows Media Audio
WiFi	Wireless Fidelity

1 Introduction

Comfort and safety are objectives that humans have always wanted to have in all aspects of their life. Automobiles are not an exception. After the automobile was invented to make transportation simpler and faster, the desire to have facilities and equipment in cars was expressed. Today the complexity of modern automobiles is rapidly increasing as the possible electronic services provide more and more functionality [1].

Mechanical parts in the modern vehicles are gradually being replaced by electronic and software components. In order to reduce the amount of the required cables a number of Electronic Control Units (ECUs) are connected together and form In-vehicle networks. Depending on the criticality of the transferred messages, different networks exist. Among them, the Control Area Network (CAN) is an event-triggered bus system. The components of engine management system and the Anti-Lock Breaking System (ABS) communicate through this bus. CAN also provides remote diagnostic of electrical parts as well as firmware update over the air (FOTA). Although it has many benefits, it largely lacks security mechanisms. However, safety and security are two different and separated concepts. There is a direct relationship between these two approaches in the vehicular networks area. Since safety is related to human life, it requires a secure system. This is why security is worth a comprehensive and critical evaluation. Previously isolated, vehicles are now connected to the Internet and are exposed to cyber attacks. Such attacks potentially threaten human safety.

A relevant number of surveys have already been done in order to underline security related issues in connected cars. Kleberger et al. [1] brings into light recent research in the security aspects of the In-Vehicle networks. So far, security flaws, possible attacks, protection mechanisms, some security-critical applications as well as experimental tests have been presented. However, to the best of our knowledge, a stability test has never been done before. The robustness of such a test is related to the amount of knowledge the tester of the system has. To narrow the scope, this report focuses on the CAN-bus protocol and has as its final aim to conduct a penetration test in order to find out to what extend ECUs are vulnerable to different possible attacks.

1.1 Background

Contemporary technology has brought a new concept into the vehicular universe: the connected car. There exist two models of environments that involve in-vehicle network approach: a vehicle-to-infrastructure model and a model of wireless communication with the car.

By remote diagnostics, the automotive manufacturers have increased the driver's comfort as the car does not need to be brought to the service station anymore. This process implies diagnosing symptoms or problems from distance. Error codes are extracted from the vehicle in order to determine the defective component. In detail, a diagnostics request is sent to the embedded computers in the vehicle, in order to command them to perform a certain action. After performing the action or reading specified values, the results are sent to the source that generated the request. In this way, defective components or firmware updates are discovered in short time.

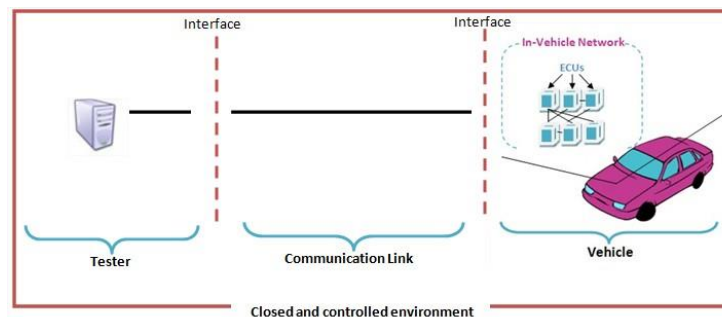
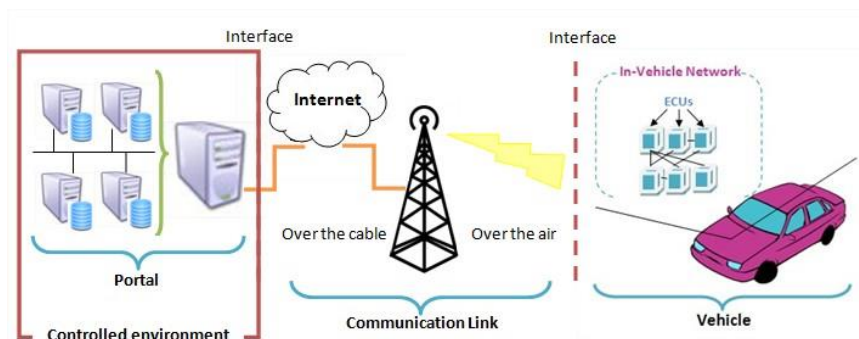


Figure 1. Traditional wired infrastructure where the vehicle is brought to a service station and all the tests, updates, configurations and diagnostics are done inside a closed environment [2]

- **Vehicle-to-Infrastructure Communication**

The vehicle's owner is no longer constrained to let the vehicle to a service station, the contemporary vehicles are equipped with systems that perform required processes over the air. These procedures implied a certain level of inconvenience in the past, because the car needed to be connected with cables inside a service station, in order to be tested. However, today technology has brought a new communication system: Vehicle-to-Infrastructure. This communication system is based on establishing a wireless connection with the vehicle from a central unit. In this manner, diagnosing a problem in a car or performing firmware updates involve minimal inconvenience for the customer and for the service station [2].



1 Figure 2. Wireless infrastructure model where diagnostics and updates are over the air [2]

- **Short-Range Communication with Hand-Held Devices**

There is another model aimed by vehicle manufacturers to provide services that allow hand-held devices to communicate with the vehicle. Contrary to the vehicle-to infrastructure model this communication system, allows communication with only one vehicle at the time.

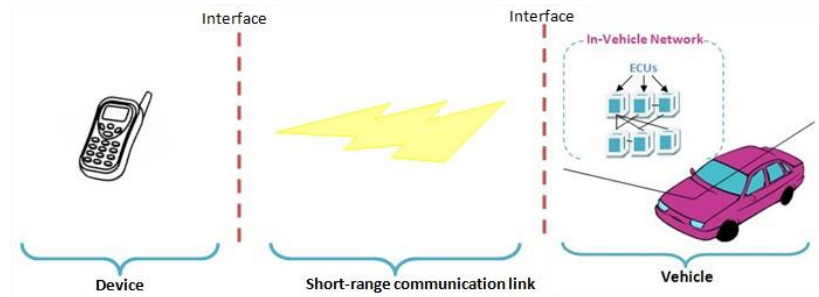


Figure 3. Short range communication model where updates and diagnostics are done over the air [3]

This general presentation of the connected car together with the two principal communications methods used nowadays bring into light the in-vehicle network place into these scenarios. These approaches bring into the light the special attention that has to be accorded to the safety of the connected car, since this has been linked to external networks. An attack over the ECUs of a car can end tragically by causing damaging injuries to the driver or by crashing the car. Thus it is important to understand the safety aspects strongly depend on security aspects [2].

1.2 Scope

The scope of this master thesis is to evaluate In-Vehicle networks by focusing on the CAN-bus protocol. This work is divided in three parts. The first part presents state of the art of the In-Vehicle and CAN-bus protocol security issues and forms the base for the second part which is developing of a framework for penetration testing in this network. Finally, related countermeasures of the vulnerabilities found in the practical part will be discussed.

1.3 Objectives

The aim of this project is to:

- Perform a comprehensive investigation of In-vehicle networks by focusing on the CAN-bus
- Identify all security issues, flaws and vulnerabilities of this network and try to find all the possible attacks
- Conduct a penetration test on a simulated a CAN-BUS mechanism and gather results
- Propose countermeasures
- Evaluate the proposed countermeasures for efficiency and feasibility
- Compare the theoretical and the experimental evaluations

1.4 Limitations

There are different types of penetration tests. Black, gray, and white are three types of tests. Black box testing is a test where the tester has no knowledge of the system, white is referred to a test where an expert or a group of experts have in-depth knowledge of the system, and finally grey testing is a test which is a combination of black and white testing. For this master thesis project, the white test is out of scope. However, it is intended to perform a gray test.

Moreover, CANoe (the software used to implement the tests) is a licensed software, all the implementations have been done in demo version of the software, which added a difficulty in knowing how to create programs by the limitations imposed by the software manufacturer and at the same time accomplishing the targeted scope of the implementation.

1.5 Methodology

The present project is designed to give a clear and complete methodology of how to perform a penetration test in CAN Networks and it is conceived in 3 different parts:

- Presentation of the project context
- Detection of security problems in CAN Network
- Definition of testing framework and implementation of the attack cases

1.6 Structure of the report

This report starts with a short review of all the in-vehicle networks and then focuses on the CAN-bus in chapter 3 by studying its features and structure. Then chapter 4 reviews of all the required security properties and then maps them to an in-vehicle taxonomy. In chapter 5, possible attacks are classified into two categories; internal and external attacks.

After that a penetration test is defined and investigated in chapter 6. Then in chapter 7, test cases introduced and performed mostly on both a simulated environment and a real environment. Finally in chapter 8 possible countermeasures are studied. Last chapters go for conclusion and future works.

2 In-Vehicle Network

2.1 Background

The new technology has brought an explosion of new functionality into the vehicle's world. As becoming more computerized, contemporary vehicles rely on a number of ECUs that enable services and functionality in order to increase the driver's comfort. The interconnection of these ECUs and buses forms the in-vehicle network. These networks are connected through gateways.

In order to control different parts of a vehicle mechanism, there exists three principal types of in-vehicle networks: CAN, LIN and MOST.

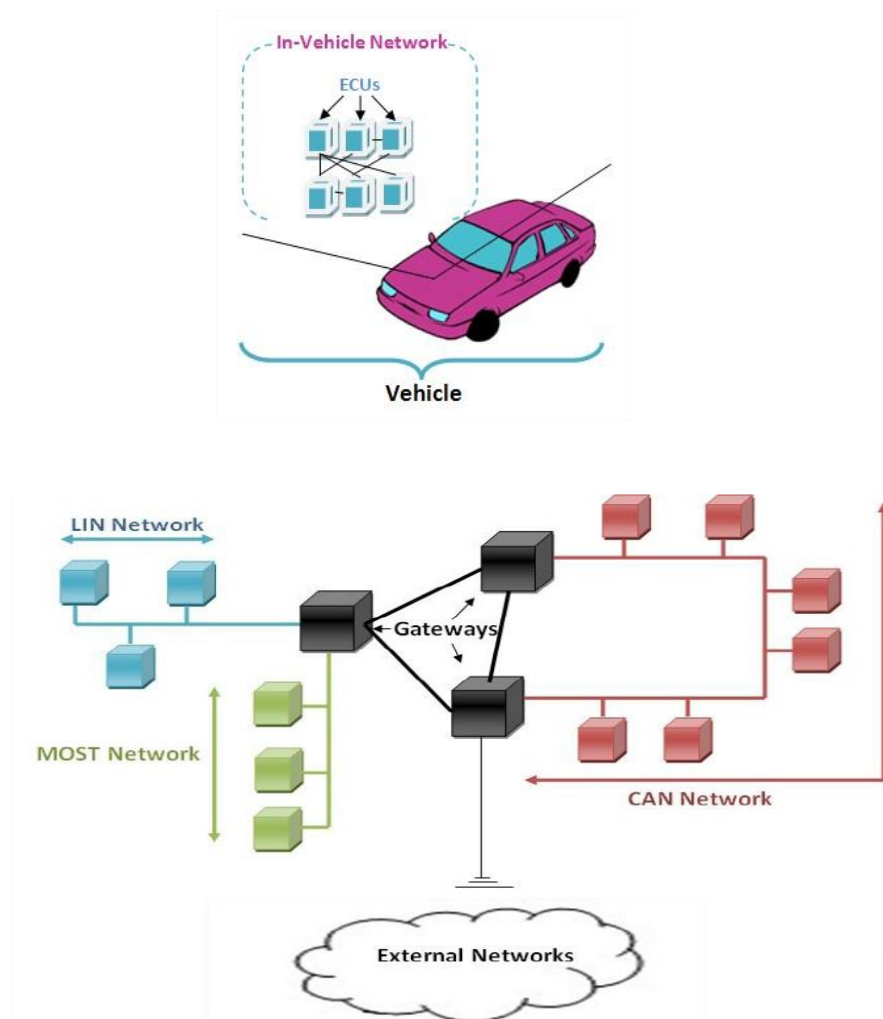


Figure 4. In-Vehicle Network

In order to assure a good and safe functionality of an ECU, the firmware on this unit needs to be kept up-to-date. This operation is aimed to be done over the air in most of the cases and done through the in-vehicle network. Performing remote diagnostics without the need of bringing the vehicle to a service station is another functionality that in-vehicle network assure by communication with fleet management over external networks. Since the in-vehicle network concentrated on meeting safety requirements, the protection against cyber attacks has been neglected. As connected to external networks, in-vehicle networks represents an easy target as these networks focus more on withstanding failures caused by non-malicious or accidental flaws then on intended and malicious ones.

2.2 In-Vehicle Buses

In-Vehicle networks consist of a combination of ECUs, sensors, and actuators. The main goal of developing such networks is allowing sensor information and ECU messages circulate in the entire vehicle system. In order to provide more services and functionality like performing firmware update over the air (FOTA) and remote diagnostics, wireless gateways have become an integral part of the modern cars. Depending on the functionality they provide and on the required speed of ECU messages transfer, different buses exist. The most important ones are mentioned in below.

- **Controller Area Network (CAN)**

CAN is a serial and real-time communication protocol for speeds up to 1 Mbit/s. The principle of this multi-master architecture network is to broadcast messages to all connected nodes (and a generated CAN message will therefore be received by all ECUs). The responsibility to process the message belongs to the node, as CAN messages do not have a recipient address but is classified by its identifier. To ensure a privileged transmission of top priority messages, CAN uses CSMA/CD (Carrier Sense Multiple Access / Collision Detection) access control, the transfer mode is asynchronous. As a protection measure against electromagnetic disturbances, the protocol uses a method to detect transfer errors and to indicate the need of interruption or retransmission of the erroneous messages. The principal target applications of this bus are basically the critical components. Engine control, driving assistants, Antilock Break Systems (ABS) and electronic gear box are some examples that use CAN due to its characteristics (high data rate, redundant network, error handling) [3].

- **FlexRay**

This sub network is a scalable and fault-tolerant communication system designed to exchange high-speed data deterministically. Its high bandwidth of 10 MBit/s helps to deal with the significant burden on the network caused by the increasing the number of electronic systems present in modern vehicles. Similarly to the CAN protocol, FlexRay

consists of a receiver-selective bus system, with up to 64 nodes. As a method of priority driven control for data transmission, FlexRay uses TDMA (Time Division Multiple Access) access control both synchronous and asynchronous transfer modes. Its error protection mechanism is based on channel redundancy, a checksum protocol. This protocol principally targets the next applications shift-by-wire, steer-by-wire, break-by-wire and emergency systems [3].

- **Local Interconnect Network (LIN)**

LIN is a single-wire network providing communication between smart sensors and actuators, and the LIN bus is the simplest communication bus with a data rate of up to 20 Kbit/s. A single master node and a group of up to 16 slaves conduct collision-free communication. For incorrect transferred message detection, parity bits and checksums are included in the messages of this network. LIN is designed with a sleep and wake-up mechanism, since the nodes of the network have a sleep mode in order to decrease power consumption. Automatic door locking, power-window and mirror are units that use a LIN network [3].

- **Media Oriented Systems Transport (MOST)**

MOST is responsible to provide audio, video, stream and control data. It is a high-speed bus designed for messages that contain a sender and receiver address. Access control is provided through TDM (Time Division Multiplex and the data rate of this bus is up to 24 MBit/s [3].

The following table summarizes the most important automotive bus systems.

BUS Type	Architecture	Access Control	Data rate	Error Protection	Examples of usage
LIN	Single-Master	Polling	20 Kbit/sec	Checksum Parity bits	door-locking, power windows, rain, light sensors, climate regulation
CAN	Multi-Master	CSMA/CD	1 Mbit/sec	CRC Parity bits	engine control, ABS, airbag, electronic gear box, driving assistants
FlexRay	Multi-Master	TDMA	10 Mbit/sec	CRC Bus Guardian	Break/Steer/Shift-by-Wire systems, emergency systems
MOST	Multi-Master	TDM	24 Mbit/sec	CRC	multimedia

	System Service	application(Entertainment, navigation, etc.)
--	-------------------	---

Table 1. Automotive bus comparison

As well as in other types of networks, in vehicular networks security is an important which must be seriously taken into account. A secure infrastructure is vital in such networks. As it mentioned before, the CAN-bus carries messages of main units in an In-vehicle network. Consequently a security breach in such a protocol can lead to catastrophic consequences. For example an attacker can gain unauthorized access to the in-vehicle network, control critical components of a vehicle and cause irreparable damage to the vehicle or its passengers.

3. Controller Area Network (CAN) Specifications

3.1 History and Application

In 1986, GmbH introduced the CAN serial bus system at the Society of Automotive Engineers (SAE) congress. Even though the intention of developing such protocol was to use it in the automotive industry, it soon became obvious that this system could be used in a wide range of embedded system appliances such as Medical equipment, aerospace applications, railway applications, lift systems, etc. Instead of a heavy and expensive wired point-to-point communication systems in automotives, CAN came up with the idea of connecting intelligent electronic devices, a more lighter and reasonable way. Since it was quickly adopted by the automotive industry, several higher level protocols have been standardized on CAN. CANopen and DeviceNet are two examples of such standards. CANopen protocol is used in both industrial and non-industrial equipments. It is used to link and control lift devices, such as panels, controllers, doors, and light barriers. Beside industrial usage, it is also employed in nonindustrial applications such as laboratory equipment, sports cameras, telescopes, automatic doors, and even coffee machines [4].

Today, every European made car uses CAN network. In 1993, standard 11898 was published by ISO to define CAN for general industrial usage. Following the rapid growth, in 2000 an ISO defined a protocol for scheduled transmission of CAN messages called Time Triggered CAN (TTCAN). It is expected that the TTCAN extension will push CAN to continue its growth for coming ten to fifteen years into a wide range of other embedded systems applications.

3.2 What is CAN?

Due to different required reliability, previous automobile control networks were configured to use multiple bus lines. This resulted in a huge wire harness. Because of cost and weight aspects, the amount of wire harnesses aimed to be reduced [5].

In order to build an efficient network of ECUs and to have expandable and error resistance nodes, CAN-bus was developed. This led to reasonable costs, short latency in the transmission of the messages between units and high data rates [A]. The figure below shows the reduction of wiring in CAN network.

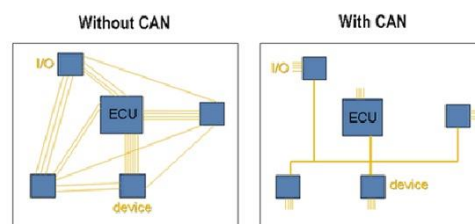


Figure 5. CAN effect on decreasing the wire quantity [4]

CAN is a serial communication protocol that allows ECUs to communicate with each other in a network. Processors are used in an automobile to make it safer, reliable, and easier to maintain. This increases the need of communication. Using a serial bus, CAN system sends and receives messages between the individual nodes (processors) in the network. The nodes are independent which means that if one of them fails, the others nodes in the vehicle will continue to work properly. A node that needs to transmit a message has to wait until the bus is free. Each message has an identifier, and every message is broadcasted in the network so that it is available to every other node in the network. Nodes only select those messages that are relevant for them and ignore the rest.

The CAN protocol was basically designed for the transmission of short messages (no more than eight bytes long) like signals sent to trigger events to lock seat belts during heavy braking, to measure temperature, and to read the pressure. There is no interruption on an ongoing transmission since the messages are assigned different priorities but urgent messages are always transmitted first. Similarly to most of the transmission protocols, CAN has an error checking mechanism in order to assure a highly reliable traffic [7]. Two types of bus levels, which are assumed at any given time, exist in this network: dominant level and recessive level. A transmitting node can send a message to a unit by changing this bus level [5].

In [4] benefits of the CAN are listed as below:

- **Low-Cost, Lightweight Network**

Instead of analog and digital inputs cumbering all devices in the system, having a CAN interface in each ECU decreases the cost as well as the weight of cars.

- **Broadcasted Communication**

In order to add a special intelligence to every device in the system, a CAN controller chip was added in each device. Since each node broadcasts its messages in the network, all the other nodes can receive them. This feature makes CAN more flexible and tolerable to any probable changes in terms of adding or deleting a node in the network.

- **Priority**

Each message is assigned with a priority which helps the network determine the transmission order at a time. This assures the non-interrupted transmission of a high priority message. Moreover, it allows the network to meet deterministic timing constraints.

- **Error Capabilities**

After every transmission the frame content is checked by a Cyclic Redundancy Code (CRC) mechanism. The incorrect messages will be ignored by the nodes. In the case

of too many errors, the nodes can disconnect themselves or even stop transmitting messages.

3.3 Overview of CAN's communication

CAN is a peer-to-peer network. In these networks there is no master node to control accessibility of the other individual nodes to read and write data on the CAN bus. A node that wants to transmit has the responsibility to check the bus in order to determine if the bus is engaged in another transmission. The sender and the receiver addresses are not included in the CAN frames. Instead, the frame is labeled with an arbitration ID which is unique throughout the network. Since frames are broadcasted, all the nodes present in the CAN network will receive them. Based on the arbitration ID of the frame, nodes will decide whether to accept or to ignore the frame.

Moreover, nodes are assigned a unique priority so that if two or more nodes want to transmit a message at the same time, the node with the highest priority (lowest arbitration ID) automatically gets bus access. The node with lower-priority must wait until the bus becomes available [4].

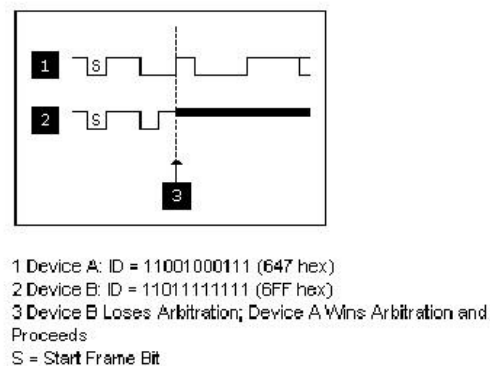


Figure 6. Usage of message priority to avoid conflicts [4]

3.4 Features

CAN protocol has the following features: [5]

- **Multi-Master**

It means that all the units connected to such a bus can try to send a message when the bus is unoccupied. The one with the highest priority will be granted the right to start transmission.

- **Arbitration**

All messages in CAN are assigned a static identifier (ID) during bus access. The priority of the messages is resolved by that identifier. If two or more nodes have to send a message, contention for the bus is arbitrated according to the ID of each message by comparing the

IDs bitwise. The mechanism of arbitration guarantees that time and information are not lost. This mechanism compares the level of the transmitted bit with the one which it monitors on the bus. The winner node of this arbitration (which has the highest priority - dominant level) will continue to send, while the one that lost the competition (recessive level) must stop its transmission and go to a listening operation [8].

- **System Flexibility**

The CAN-bus connected nodes have no specific address. This feature makes adding or removing a node to/from the bus flexible. It means that there is no need to make any changes in software, hardware, or application layer to the other connected nodes to that bus.

- **Data Rate**

In a given system the bit-rate is uniform and fixed. After considering a suitable speed (based on the size of the network) all the nodes must keep this data rate, otherwise an error would arise which hinders communication in the network. This kind of rule only applies to nodes in the same network.

- **Remote Data Request**

There is a mechanism called “Remote Frame” that enables a node to request a Data Frame. Both the Data Frame and the corresponding Remote Frame have the same identifier [8].

- **Error Detection/Notification/Recovery functions**

To have a precise error detection, signaling and self-checking are implemented in every CAN node [8]. An error might be detected by any node in this network by the mechanism called *Error Detection Function*. When a node detects an error in the system it informs other nodes by sending them a notification. This process is called *Error Notification Function*. Finally, if a node encounters an error while sending a message it continues sending the message as well as an error notification. The transmission will be repeated until the messages will be successfully sent. This error recovery function takes at most 29 bit times. This mechanism is called *Error Recovery Function*.

- **Error confinement**

Basically, two types of errors might arise in the CAN-bus. If due to the environment noise, the data on the bus temporarily becomes disturbed and *Temporary Errors* might arise. An internal failure or a disconnection of a node might lead to a *Continual Error*.

To discriminate between these types of error, the CAN protocol uses different functions. By decreasing the priority of the problematic node, the communication of the other nodes will not be disturbed and in case of a continual data error on the bus, the problematic node will be separated and switched off to not disturb the normal communication.

- **Number of Participating Nodes**

There is an indirect relationship between the number of connected nodes and the speed of the BUS. Although there is no explicit limit of the number of connected nodes in the CAN-bus, this number is implicitly limited by the delay time and electrical load in the bus.

- **Data Consistency**

A transmitted message in a CAN network will be accepted by either all other nodes or none. Therefore data consistency is provided by two concepts: multi-master and error handling.

- **Bus Values**

Dominant and recessive levels are two logical values which are accepted by this bus. Dominant bit ('0') always wins when in competition with recessive bit ('1') [8].

3.5 CAN in the ISO/OSI Reference Model

OSI standard provides a comprehensive model for communicating layers in a network. CAN-bus only uses layer 1 and layer 2. While the definition of link layer in all of the standards is similar, there are some differences in physical layer's definition. Table1 demonstrates CAN position in the ISO/OSI reference model.

No. of layer	ISO/OSI ref model	CAN protocol specification
7	<i>Application</i>	Application specific
6	<i>Presentation</i>	Optional: Higher Layer Protocols (HLP)
5	<i>Session</i>	
4	<i>Transport</i>	
3	<i>Network</i>	
2	<i>Data Link</i>	CAN protocol (with free choice of medium)
1	<i>Physical</i>	

Table 2: CAN in ISO/OSI Reference Model [6]

3.6 Physical Layer

The CAN bus uses a more reliable bus topology than star or token ring. Using a hub to connect nodes, a star model has potentially one single point of failure. In token ring, if any of the individual nodes fails the network will be broken. The CAN controller determines the level of a bus by potential difference in two wires that comprise the bus.

As a media CAN bus uses a different driven pair of wires which are called CAN_H and CAN_L. Twisted pairs are used since it reduces electromagnetic interference [7].

- **Bit Representation**

There are various ways to encode bits in digital systems. Here Non-Return to Zero and Manchester encoding are discussed.

- **Non-Return to Zero (NRZ).**

It does not require signal transitions to represent each bit. The signal remains '0' or '1' for the entire time slot. There is no easy way to tell where each bit starts or ends when there are more than two '1's or '0's in a row. A receiver can know the beginning and end of a bit by having a clocking source that is synchronized with the transmitter so that it can decipher the bit stream. This is called *synchronous communication*.

- **Bit Stuffing**

Destuffed bit stream	01011111010	10100000101	01011111000010	10100000111101
Stuffed bit stream	01011111o01 0	10100000l101	01011111o0000l1 0	10100000l1111o01
0 , o = dominant (stuff) bit; 1 , l = recessive (stuff) bit				

Figure 7. Bit Stuffing [14]

NRZ signals might remain unchanged for a long period e.g. during a long constant series of '1's or '0's. This can lead the network's individual oscillators out of synch. CAN inserts a signal transition bit every time five identical bits appear in a row. This signal transition is called a *stuff bit*. The receivers synchronize their clocks with it. After detecting five '0's or five '1's in a row, a receiving unit automatically disregards the next bit.

There are only some fields coded by this method: Start of Frame, Arbitration Field, Control Field, Data Field, and CRC sequence. The remaining bits of the Data Frame like CRC delimiter, ACK field, and End of Frame are fixed and not allowed to be stuffed.

Furthermore, only the Data Frame and Remote Frame are coded by this method. The error Frame and Overload frame have fixed values and are not allowed to be coded by this method.

- **Recessive and Dominant State**

Dominant and recessive are two terms that describe state of the bus. When the CAN_L = CAN_H = 2.5V it is said that lines are at the same potential and it is considered a *recessive* state. The CAN bus is *idle* when it remains in the recessive state. The *dominant* state occurs when there is a difference between them and CAN_L = 1.5V and CAN_H = 3.5V.

It is possible to represent data in binary format '0' and '1'. '0' defines a dominant state while '1' represents a recessive state.

- **Bit Timing and Synchronization**

To make data transmission efficient, all the units in the CAN network are synchronized. It means that every node uses the same clock rate to send and receive data. There is a single reference point which all the nodes set their clocks based on. Without a reference signal it is not simple to synchronize clocks. Oscillator drift, propagation delays, and phase errors are some possible reasons why nodes lose their synchronization. CAN synchronizes clocks using two methods: Hard Synchronization and Resynchronization. All the messages start with a fixed dominant bit which is called Start of Frame (SOF) bit. *Hard Synchronization* occurs during transmission of the SOF. Nodes synchronize their clocks using the transition of this bit. Since clocks will not remain synchronized throughout the entire frame they need to be synchronized continuously.

Transition from a recessive state to a dominant state is called a *Resynchronization*. In case of a string of five '0' or '1', CAN inserts a stuff bit to resynchronize its clock.

- **Bit Timing**

Every CAN system is configured with a *data rate*. If it was possible to eliminate all the desynchronization sources then the *nominal data rate* which is the number of bits per second transmitted by an ideal transmitter could remain constant. But oscillator drift and other problems change the nominal bit rate from the actual bit rate. Consequently, a mean to synchronize message traffic is inevitable. *Bit time* is the amount of time which is required to transmit a bit across the network. The bit time consists of several parts.

Synchronization segment (SS)

Propagation time segment (PTS)

Phase buffer segment 1 (PBS1)

Phase buffer segment 2 (PBS2)

Each segment is further divided into units called Time Quanta.

- **How Synchronization is Achieved**

CAN protocol uses NRZ to communicate. Using bit timing and no synchronizing signals attached at the beginning or end of each bit, the transmitting unit starts sending frames synchronously. The receive unit is synchronized by changing of the bus level as it receives frames.

If the transmitter or the receiver gets out of sync they adjust their operation timing by means of a hardware synchronization or resynchronization [5].

3.7 Link Layer

CAN Protocol Specification

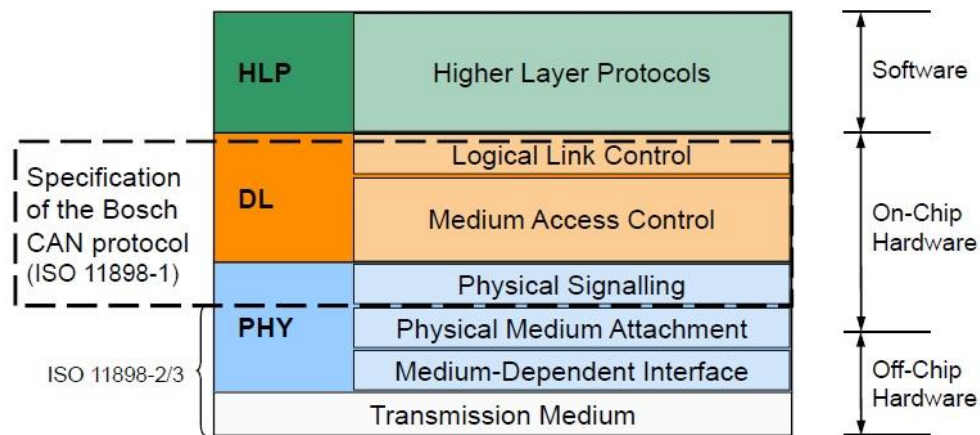


Figure 8: CAN Protocol Specification [6]

3.7.1 Sub-layer: Medium Access Control (MAC)

This sub-layer represents the kernel of the CAN protocol. The most important features of MAC are [6] Message Framing, Arbitration, Acknowledgement, Error Detection and Signaling, and a management entity called Fault Confinement.

- **Message Framing**

Four types of messages are manifested in CAN protocol.

- **Base Format Data Frame**



Figure 9. CAN Message

This is a normal frame which carries up to eight bytes of data from a sender unit to a receiver unit. There are two different versions of Data Frames: Base Format and Extended Format. Extended Format was developed for large system with heavy traffic such as buses or trucks. Each Data Frame has an identifier that assigns a unique number to each message. The base format in such systems could not support the number of required identifier because the number of transmitting message was greater than the possible identifiers. Therefore, a larger identifier field in Extended Format was developed to meet this need. Eight different fields are defined in this frame. The figure below is a visual representation of this frame.

Start of Frame	1 bit
Arbitration Field	11 bits
Remote Transmission Request	1 bit
Identifier Extension	1 bit
r0	1 bit
Data Length Code	4 bits
Data Field	0-8 bytes
CRC Sequence	15 bits
Delimiter	1 bit
Acknowledgement Slot	1 bit
Delimiter	1 bit
End-of-Frame Field	7 bits
Intermission Field	3 bits

Figure 6. Base Format Data Frame [7]

Start of Frame: a dominant bit marks the beginning of a message.

Arbitration Field: includes both the identifier and Remote Transmission Request (RTR) bit.

Identifier: 11 bits indicate message identifier that is transmitted from ID-10 to ID-0. (ID-0 is the least significant bit). It is not allowed to have all 7 most significant bits equal to recessive. 11 bits provide 2^{11} (=2048) unique identifiers. The lower the value of the bit, the higher the priority is.

Remote Transmission Request (RTR) Bit: a dominant bit in Data Frame or a recessive bit in Remote frame marks the RTR bit. It serves a dual goal. It indicates which node has access to the bus and also identifies type of the Frame.

Control Field: 6 bits in form of an Identifier Extension bit, a reserved bit, and Data Length Code represent Control Field.

Identifier Extension (IDE) bit: it is either set as dominant to present a Base Format Frame or recessive to present an Extended Format Frame.

Reserved bit: it is a dominant bit which is never used.

Data Length Code: four bits indicates the length of the message in bytes. Although the length is typically less than eight, it will be considered eight if it is greater than eight.

Data Field: payload with the length from 0 to 8 bytes can be transmitted in this field. It is the only field that does not have a fixed length.

Cyclic Redundancy Check (CRC) Field: a CRC field is used to detect if the message is corrupted during transmission.

CRC Sequence: it is derived from a Cyclic Redundancy Check that suites with frame with bit count less than 127 bits. To carry out CRC calculation, a polynomial is used. Coefficients of this polynomial are a de-stuffed bit stream and consist of SOF, Arbitration Field, Control Field, Data Field.

CRC Delimiter: it is a recessive bit, followed after the CRC sequence.

ACK Field: it contains two bits, ACK Slot and ACK delimiter. The sender will transmit two recessive bits. If no unit acknowledges a message the transmitter will retransmit until the intended unit turns it off.

ACK Slot: In case of a successful reception, receiver node replies with a dominant bit in the ACK Slot bit.

ACK Delimiter: it is a recessive bit right after ACK Slot. The ACK Slot is surrounded by two recessive bits- ACK Delimiter and CRC Delimiter.

End of Frame Field: seven recessive bits mark a complete error-free transmission. A dominant bit in ACK Delimiter or in End of Frame bits mark the beginning of either Error or Overload Frame.

Intermission Field: 3 recessive bits intermission field represents the minimum space between Data or Remote Frame with preceding Frames. Starting with two dominant bits in this field marks an Overload Frame.

- **Extended Format Data Frame**

Besides IDE in the Control Field and the size and order of the Arbitration Field, this Format is similar to the Base Data Frame. Additionally, both can coexist but it has a lower priority over Base Format Data Frame.

Start of Frame	1 bit
Base Message Identifier	11 bits
Substitute Remote Request	1 bit
Identifier Extension	1 bit
Extended Message Identifier	18 bits
Remote Transmission Request	1 bit
r1	1 bit
r0	1 bit
Data Length Code	4 bits
Data Field	0-8 bytes
CRC Sequence	15 bits
Delimiter	1 bit
Acknowledgement Slot	1 bit
Delimiter	1 bit
End-of-Frame Field	7 bits
Intermission Field	3 bits

Figure 7.Extended Format Data Frame

Arbitration Field: to support a 29-bit identifier in this format the Arbitration field is longer. 2^{29} (= 536 million) unique identifiers are possible in this format.

Base message Identifier: it is similar to the same field in Base Format Frame. It presents the most significant bits of the identifier.

Substitute Remote Request (SRR): a recessive bit with no specific use. It is just a placeholder so that the next field remains in same place as it is in the Base Format Frame.

Identifier Extension (IDE) bit: it is a recessive bit in Extended Format Frame and dominant in Base Format.

Extended Message Identifier: it provides 18 bits which together with BaseMessage Identifier satisfy 29 bits.

Remote Transmission Request (RTR): very similar to Base Format.

Control Field: two dominant reserved bits as well as a Data Length Code (DLC) present the base Format. DLC field is identical to the same field in Base Format.

▪ Remote Frame

A node that needs information of another node can trigger it by sending a Remote message. They are mostly sent out on a regular schedule to draw updates from sensors. The format of this frame is similar to the Data Frame. Only RTR bit in Remote Frame is recessive which means that a data frame has priority over an identical Remote Frame.

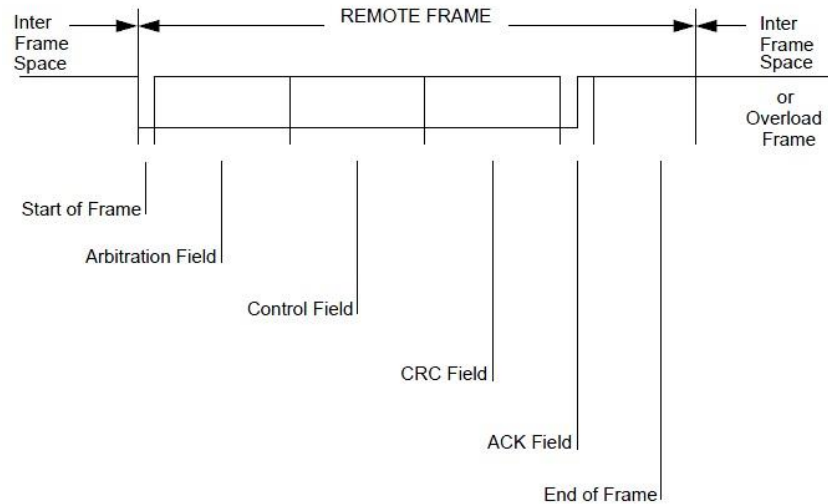


Figure 8. Remote Frame [8]

▪ Error Frame

It consists of two fields- Error Flag and Error Delimiter which is sent out by a receiver node when it detects an Error in a message. Therefore, it can be sent after a Data Frame or a Remote Frame. The sender has to listen to the bus after sending a message in order to detect any probable Error message. In case of detecting an Error message it has to retransmit. Moreover, not all nodes in the CAN network are allowed to transmit an Error message [7].

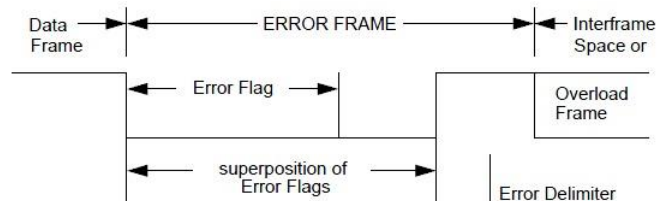


Figure 9. Error Frame [8]

Error Flag: There are two types of Error Flags: Active and Passive. It will be more elaborated in Fault Confinement section.

Active Error Flag which is six consecutive dominant bits.

Passive Error Flag which is six consecutive recessive bits. It might be overwritten by other nodes.

Error Delimiter: it consists of 8 consecutive recessive bits. It also indicates the End of the Frame. After this the bus enters an Idle State that can be the start of nodes competition to get bus.

- **Overload Frame**

It can be considered as an Error Frame but it does not trigger the transmitter to retransmit. It is sent to inform the transmitter to slow down sending further frames based on detecting an internal condition. There are three cases that determine the sending of an overload frame: detecting a 'dominant' bit during first two bit of the intermission, detecting a 'dominant' bit in the last bit of End of Frame, or at the last bit of error delimiter or overload delimiter fields. Overload Frame has the same format as the Error Frame.

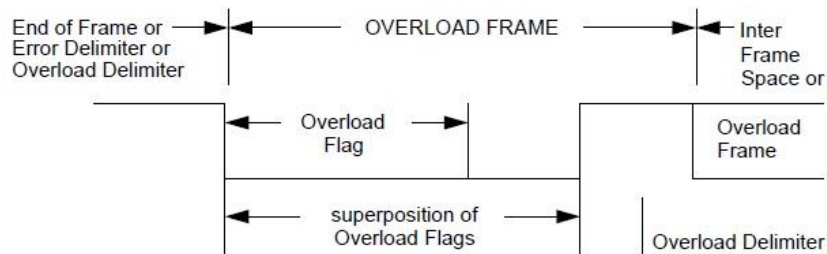


Figure 14. Overload Frame

Overload Flag: it contains a sequence of six dominant bits which correspond to Active Error Flag. After the other nodes detect an Overload Flag on the bus they will transmit their own Overload Flags which effectively stops all the traffic on the bus. Then, nodes listen to hear the Overload Delimiter. After sending an Overload Frame, the maximum time needed to recover is 31 bit times.

Overload Delimiter: it contains a sequence of eight recessive bits.

- **Inter-Frame Space:** Data Frames and Remote Frames are separated from preceding frames by a space field called Inter-Frame Space. Based on being sender or receiver there are different fields. In general three fields are defined: Intermission, Suspend Transmission, Bus Idle.

Intermission: Three recessive bits mark an intermission. An intermission Field represents the minimum amount of spacing between Data or Remote Frames.

Suspend Transmission: after the transmission of a message by an Error Passive station (will be explained in Error Detection Mechanism), eight 'recessive' bits and an Intermission are sent before starting to transmit again. If meanwhile another nodes start a transmission, this Error Passive node stops sending and begin to listen to the bus.

Bus Idle: There is no exact duration of Bus Idle time. When the bus is recognize to be free all the nodes that want to transmit start competition.

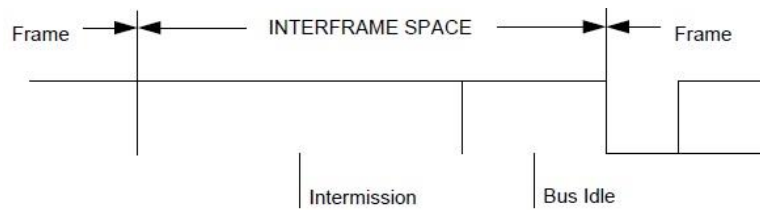


Figure 10. Receiver of the previous message or not an Error Passive station [8]

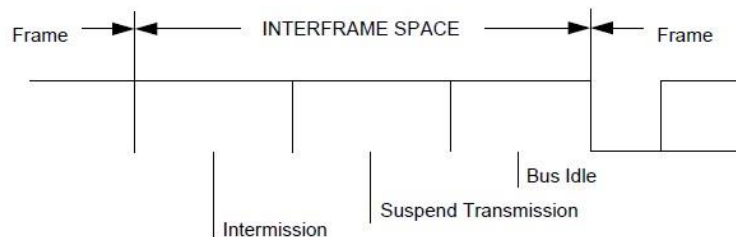


Figure 11. An Error Passive Transmitter Station [8]

- **Arbitration**

MAC sub-layer manages the contest over getting access to the bus between nodes that are ready to send data. In fact, MAC gives the opportunity to the nodes to send a bit at a time and participate in the competition. There are several methods to access the medium. As it is depicted in the figure below, there are two general mechanisms: Deterministic and Stochastic (Random). While predefined access rights assure that there is no conflict in the deterministic access control model nodes can access the bus as soon as it is idle in stochastic access control model.

Deterministic access control uses either centralized access or a decentralize access to the bus. A central entity controls access to the network. This has a potential vulnerable single point of failure. Decentralized access control is more complex as it is needed to assign priorities to the node dynamically.

Stochastic access control can be either collision or non-collision free most often based on Carrier Sense Multiple Access (CSMA) approaches. All the nodes monitor the bus. Once the network is idle, all waiting nodes will attempt to get access to the network. Although only the node with highest priority is able to transmit, a method to find this node is inevitable. CSMA can solve this problem by preventing collisions between messages, which in this case it is a Collision Avoidance method. It also can be set up to take the risk of message conflicts, but it has to intervene to detect and clean up these conflicts, which in this case it is called a Collision Detection method [7].

Bus access can be Destructive or Non-Destructive. In *Non-destructive bus access*, the bus is allocated to one and only one station. *Destructive bus allocation mechanism is not always a successful bus allocation. If there are simultaneous requests for bus access by more than one station, all transmission attempts will be aborted.*

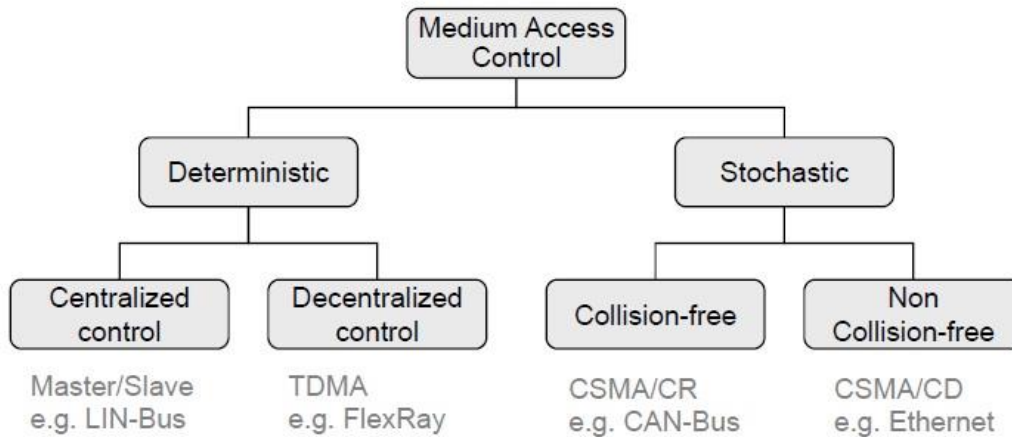


Figure 12: Medium Access Control [6]

CAN is a Non-Destructive bit-wise Arbitration that uses CSMA/Collision Detection method. In fact, it allows messages with higher priority over lower ones. As it is mentioned in previous parts, the Arbitration Field determines priority and identity of the message. Consequently, competing nodes start by transmitting their Arbitration Field and listening to the bus at the same time. If nodes that are transmitting recessive bits see a dominant bit over the bus they will stop sending since a higher priority message is detected. Loser nodes in the arbitration go to a receive mode. By this method, highest priority messages will get the bus and less priority ones will start competition right after the bus is idle again. Besides that, no bandwidth is wasted by this method. All the nodes in the network can hear all the transmission which is absolutely not safe.

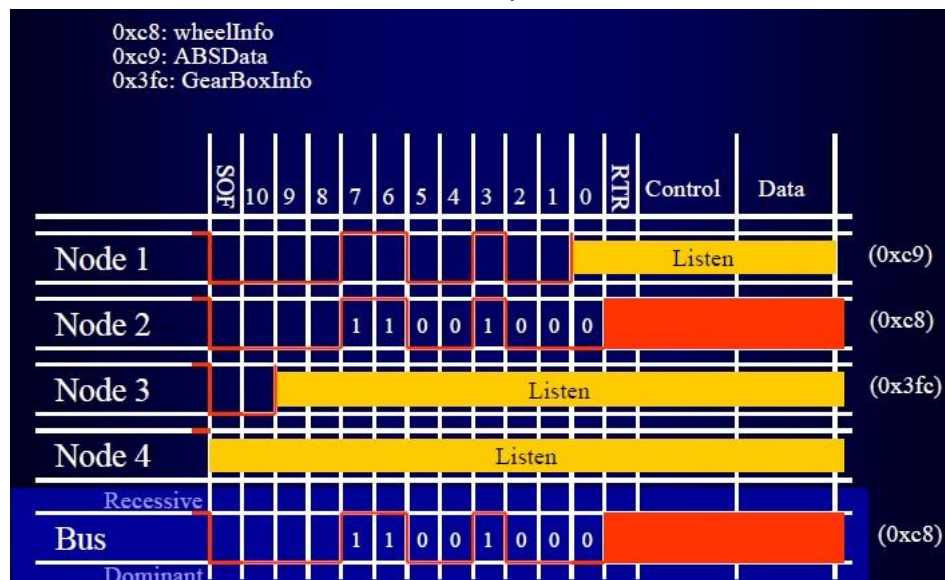


Figure 12. Data Transmission with Arbitration

- **Acknowledgement**

Acknowledgement is a common way that a sender can be sure that its messages are delivered or not. It continues sending a message until it receives an ACK.

- **Error detection, Error handling, and Fault Confinement**

Errors that can lead to catastrophic failures are avoided in CAN, as it is designed to be used in automobile systems. CAN has eliminated the latency concerns by increasing speed in version 2.0. Moreover, a comprehensive integrity check is performed in CAN by means of a variety of error messages. In the CAN protocol, nodes can change their functions based on the severity of the errors they receive. They might continue to send and receive data like a normal node or be completely shut down. This feature is called Error Confinement. In fact, it prevents distributing faults by shutting off a faulty node before such errors bring the network down.

Depending on the type and number of error conditions, five error conditions and three error states that a node can be in are defined in the CAN protocol. The following section describes each one in more detail [11].

Error Detection Mechanisms

CAN uses the following methods to detect errors [7]

- **Bit Check:** in this process each node monitors the bus and checks the frame that was just sent to see if it is correct. A bit error is detected by a node if it sends out a dominant bit when it is called for a recessive bit and vice versa.
- **Frame Check:** Some fields in each type of frames are fixed. For example all the frames must start with a dominant bit. Each type of frame contains specific bit fields that always have fixed values. The process of Frame Check detects Form Errors by monitoring these fixed fields in every frame they receive. A node will detect a frame error if a fixed bit field contains an illegal bit value.
- **Cyclic Redundancy Check (CRC):** Applying a polynomial equation to a block of transmitted data is a very effective and common way to detect errors in a network. The transmitter calculates a CRC value and attaches it in the corresponding field in the frame. Receiving node applies the same polynomial and calculates CRC value again and then compares these two values. If the two values are the same the message has kept its integrity otherwise the message is not trustworthy and the receiver sends a CRC Error message back to the sender.
- **Acknowledgement Check.** Transmitting node expects an acknowledgement in the ACK slot from at least one node in the network and if it is not satisfied then this is considered

an Acknowledgement Error. The node will continue to retransmit the frame until it receives an acknowledgement.

- **Stuff Rule Check.** If more than five identical bits in a row are detected by a node a Stuff Error has occurred and an Error Frame is sent.
- **Error states [11]**

If any of aforementioned errors is detected by any node in the system the transmission can be aborted by sending an Error Flag. This Flag informs other nodes about an error in the system and prevents them accepting the erroneous message. It ensures consistency of the system. The transmitter has to retransmit the message.

In order to count the number of errors two counters are included in the nodes: Transmit Error Counter (TEC) and Receive Error Counter (REC). When a transmission or detection error is detected, based on the error type and the node which caused the error, the respective counter will be increased by a weighted value. On the other hand, the counter will be decreased by one after a successful transmission.

Based on the number of transmitted and received Error Frames, each node is in one of these states: Error Active, Error Passive, or Bus-Off.

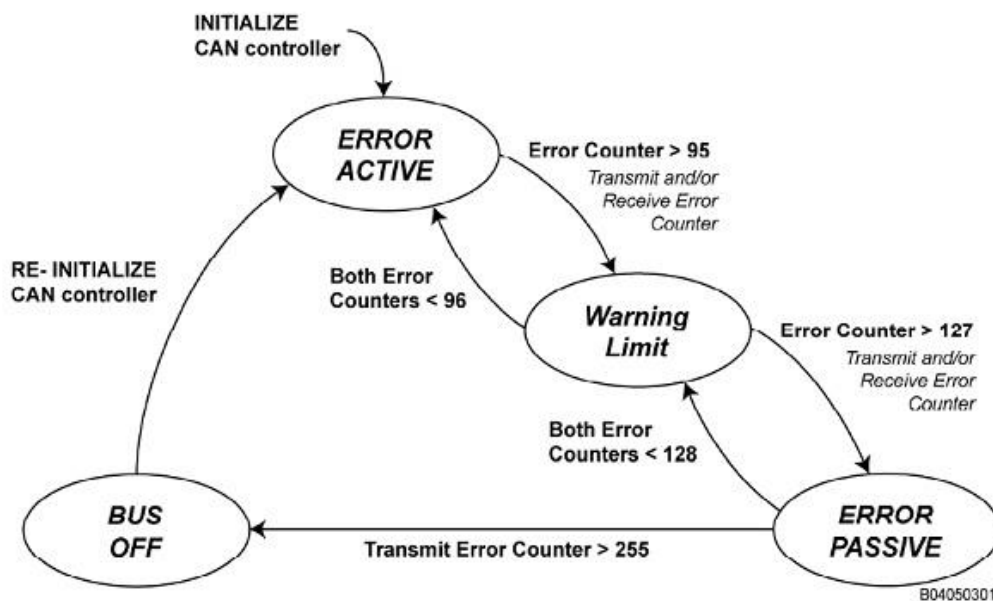


Figure 19 - CAN Error State

- **Error Active**
It is an ordinary operational mode. A node in this mode can actively participate in the ordinary bus communication: send and receive without restrictions. When detecting an error, the node sends an active error flag which is six consecutive dominant bits. Based

on Bit Stuffing rule, nodes are not allowed to send more than five similar bits. This transmission makes other nodes to send an error flag called Error Echo Flag. The sequence of such dominant bits results in a superposition of different error flags sent by other nodes.

A node is Error-Active when both the Transmit Error Counter (TEC) and the Receive Error Counter (REC) are below 128.

- **Error Passive**

If the number of Error messages transmitted or received by a node exceeds 127, it goes to an Error Passive state. In this state, nodes are not permitted to transmit any Active Error flag anymore. They can only send Passive Error Flag which consists of six recessive bits. Similar to Active Error Flag, Passive Error Flags violate Bit Stuffing rule. Again, the receiving nodes will respond with Error Flags of their own. If the Error-Passive node is not the only transmitter or is a receiver, then the Passive Error Flag will have no effect on the bus because it transmits recessive bits as error flag. Nodes in this state must wait for eight bit times before trying to retransmit if they want to send a Data Frame with an error.

- **Bus-Off**

If the TEC is greater than 255 then the node goes into the Bus-Off state. Only Transmit Errors can lead a node to such a state. It means transmitting 32 consecutive messages with errors will lead the node to be turned off. Switching to this mode causes the node to stop sending or receiving messages, acknowledge messages, or transmit Error Frame of any kind [7]. During this state, a CAN controller can be reset either by hardware or by the application. In case of resetting by application it demands a certain amount of idle time. After resetting, TEC and REC will be set to zero. CAN controller state will be Active State.

Fault Confinement

A very probable risk with any serial bus network is that any defective node can shut down the entire network. The CAN protocol deals with it using an automatic detection of a faulty node. It can disconnect a node from the network before the faulty node disorders the communication. Besides automatic detection of a faulty node, it recovers nodes from Bus-Off state and return to Error-Active mode to restart transmission normally. For both issues, the CAN protocol uses Fault Confinement. It distinguishes between sporadic from permanent errors. This function lowers the priority of an error-prone unit messages so it does not hinder communication of other nodes. It might be separated if a continual data error on the bus is occurring [5]. Consequently, an erroneous node cannot monopolize all the bandwidth and it will be shut off before bringing the network down. Through this, it guarantees bandwidth for critical system information [11].

3.7.2 Sub-layer: Logical Link Control (LLC)

This sub-layer describes upper part of the OSI data link layer and deals with issues that do not have anything to do with the medium access methods. It is concerned with Message Filtering, Overload Notification, and Recovery Management.

LLC Services

It provides two types of connectionless mode transmission service [14].

- **Unacknowledged Data Transfer Service**
By this, and with no need of establishing a data link connection, LLC user can exchange Link Service Data Unit (LSDU). Such connection can be point-to-point, multicast, or broadcast.
- **Unacknowledged Data Request Service**
By this, and with no need of establishing a data link connection, LLC user can request a remote node to transfer Link Service Data Unit (LSDU).

LLC Functions

- **Message Acceptance Filtering**
As it is mentioned before, each message has an identifier. This identifier does not indicate anything about the destination of the message, and instead it explains the meaning of the data. Message Acceptance Filtering is a mechanism by which a receiving node decides that the received frame is relevant and then accepts it or just ignores it [14].

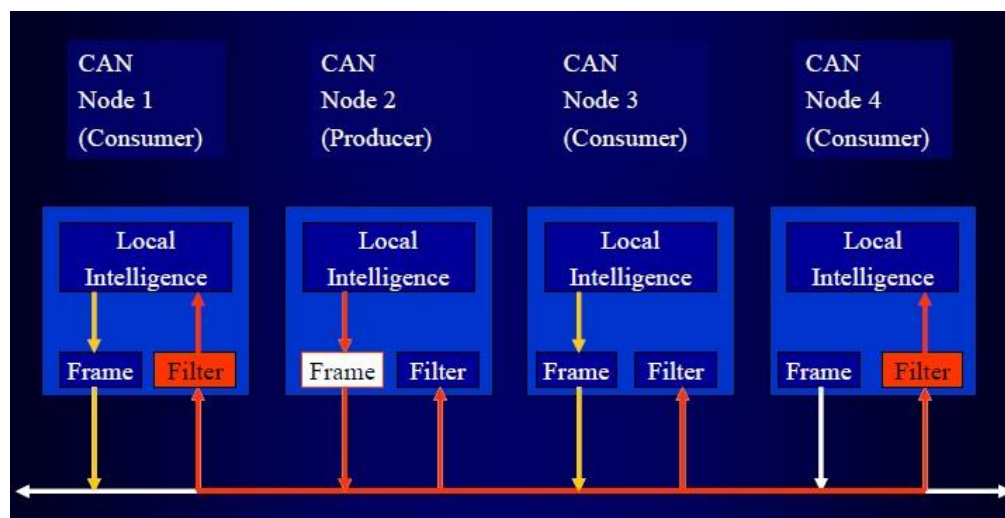


Figure 13. Principle of Data Exchange

- **Overload Notification**
If the internal condition of a receiving node needs delay of the coming LLC data or LLC Remote Frame, the Overload Frame of the MAC sub-layer will be initiated in Logical Layer Control.

- **Recover Management**

If a message loses arbitration or if it is distributed by error during transmission, it is LLC sub-layer who is responsible for retransmitting automatically.

LLC Frame Types

Based on different services off LLC sub-layer there are two types of Frames- Data Frame and Remote Frame.

- **Data Frame:** It carries data from transmitter to receiver. It consists of three fields- Identifier Field, Data Length Code (DLC) Field, and Data Field.

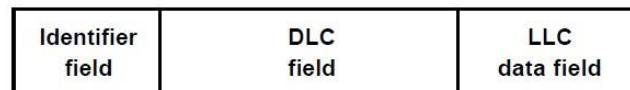


Figure 14. LLC Data Frame

- **Identifier Field:** consists of three sub-fields
 - **Base Identifier:** with 11 bits length is a unique number to identify the frame.
 - **Extension Flag:** it is either '0' or '1'. If it is '0' then the next sub-field will be ignored.
 - **Identifier Extension:** 18 bits additional identifier used together with Based identifier if the number of needed frames exceeds the amount provided by Base identifier.
 - **Data Length Code (DLC) Field:** 4 bits DLC indicates number of bytes.
 - **Data Field:** from 0 t 8 bytes of main data.
- **Remote Frame**
- It is sent to request data from a unit. Both Remote Frame and corresponding Data frame have same identifier. It consists of two fields- Identifier Fields and DLC fields.



Figure 15. LLC Remote Frame

- **Identifier Field:** it is identical to the Data Frame same field.
- **Data Length Code (DLC) Field:** 4 bits DLC indicates number of bytes and may only be transmitted in a system with determined data length code.

4 Security Considerations

The reason for developing in-vehicle networks was originally to be used inside an automobile. In fact, in-vehicle networks used to be isolated, but gradually introducing FOTA and some other changes made these networks to be connected to outside of the automobile. Lack of some vital security properties marked in-vehicle networks to be considered as a non-secure network.

Moreover, when talking about network security, a need to have a common language arises. In this chapter, first security properties will be discussed, and then the security common language as well as a fitted one for in-vehicle networks will be mentioned. Since CAN is a link level and physical level protocol, the security features of these levels will be discussed.

4.1 Security Properties

In this part, some common security properties which are base properties to evaluate network security will be discussed. These properties are used to evaluate security of in-vehicle networks.

- **Data Confidentiality-** To prevent from unauthorized read which leads to disclosure of data, it is needed that the content of the messages is kept confidential so that only intended ECU can read it. In CAN, messages will be broadcast over the bus. All the other ECUs will hear the traffic but they decide to ignore or filter the traffic based on message's identifier. So, keeping the privacy needs a deep consideration in such a network.
- **Data Integrity-** To be sure that a message will be transferred intact with no modification during the path, some functions are needed to ensure message integrity. CAN uses CRC to verify the integrity of messages. Since CRC is not secure enough some potential attacks might launch because of this flaw.
- **Data Availability-** Data on the network is needed to be available at any time. Otherwise a Denial of Service attack is happened. This attack is hard to prevent in CAN because of the function of Fault Confinement. This function detects a faulty node and disconnects it from the network. It is very possible that an intruder user forces an innocent ECU to go to a BUS-Off mode.
- **Data Authentication-** A receiver requires verifying the transmitter of a message. Otherwise a malicious node can spoof infected messages and dump intended ECU to malfunction. Since in-vehicle networks' Frames lacks sender and receiver address, a malicious user can easily enforce ECUs to perform arbitrary actions.

- **Data Freshness-** In lack of some features to ensure the freshness of a message, an attacker can listen to the traffic and capture critical messages. She/he can replay them when it is time.
- **Data Non-Repudiation-** It is needed to be able to differentiate between a faulty ECU from a spoofed message sent by an intruder. This feature is needed since other security properties are not included properly in such network.

4.2 In-Vehicle Taxonomy

Howard et al. [12] have introduced a common language for computer and network security terms to classify information and events into a common taxonomy. They have proposed some general terms as well as a structure for incidents. Since this work addresses in- vehicle networks and specifically CAN network, it is intended to map such taxonomy to vehicular networks and evaluate just related parts.

The Attack scenario by [12] is depicted in the below picture. Five logical steps are involved in such an attack matrix- Tool, Vulnerability, Action, Target, and Unauthorized Result. Although several examples in each category are presented, they believed that it is just a spectrum of possible activities.

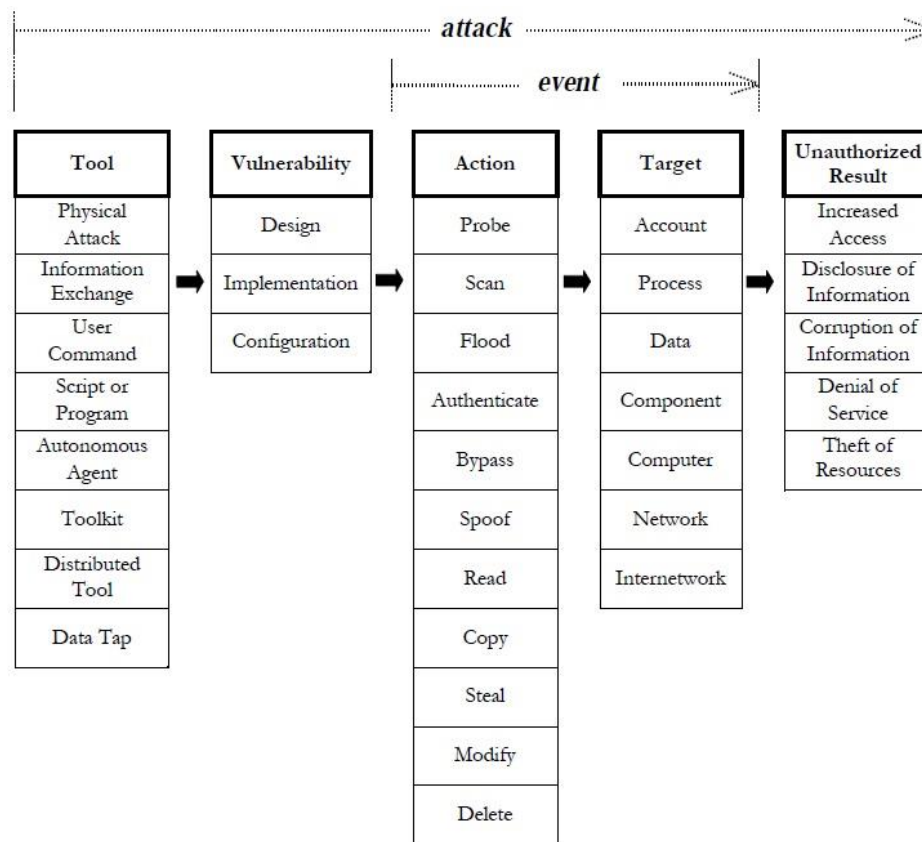


Figure 16.Computer and Network attacks

As it is clear in the picture, the first two steps are used to cause an event on a computer or network. Such an event might reach to an Unauthorized Results. To be more specific, an attacker might use a tool to exploit a vulnerability to perform an action on target in order to achieve unauthorized results. According to the [12] these terms are define as follow:

Tool: it is a means or software that can be used to exploit vulnerabilities or violate some rules.

Vulnerability: it's a flaw or weakness in the system that might be lead to a security breach by a malicious user.

Action: it is an activity to gain result by either a user or a process. It is also conceptualized to be directed toward Target.

Target: it is categorized into two entities- logical entity such as account, process, or data and physical entity such as component, computer, network or internetwork.

Unauthorized Result: to have a successful attack, gaining unauthorized Result is needed.

Two more terms are introduced in this model- Event and Attack. They believe that computer and network's operations consist of varieties of *Events*. An event is a "discrete change of state or status of a system or device" [12] It results from an action which is directed against a target. It is valuable to extract some significant aspects of this definition. Firstly, an action and a target must exist to have an event but it does not mean that the results have to cause a security breach. Secondly, an event is a logical link between an action and a target which represent how it is thought about the event generally with no attention to each individual step. Thirdly, it does not limit the scope of actions to just unauthorized actions. In fact, some routine or authorized actions might lead to an attack. Lastly, all the mentioned events are not possible. On the other hand, an *Attack* happens when a series of steps on a computer or network achieve an unauthorized result. These steps are shown in the above picture. An attack itself is part of an *Incident*. A group of attacks that are distinct from the others by some elements like the Attacker, Objectives, Attacks themselves, Timing, or Sites are considered as Incident. The picture below shows a visual representation of an Incident.

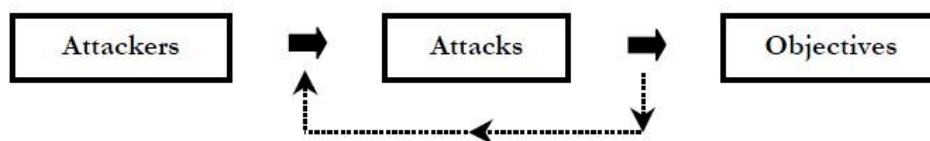


Figure 17. Computer and Network Incident [12]

In this model, an Attacker who is an individual initiates an Attack to gain her/his objectives. These Objectives are the final goal of such an attack. Wolf et al. [3] categorized attackers into three main groups- Car owner, Garage personnel, Third party. They then ranked technical sophisticate garage employee as the most powerful attacker.

Finally two concepts of *Success* and *Failure* of an attack are further discussed. Success is considered as achieving an unauthorized result while Failure happens when the system could resist such attempts and does not allow any unauthorized access. Moreover, the result of an incident might remain *Unknown* when it is neither *Successful* nor *Fail*.

By merging the two previous pictures and adopting it to the vehicle environment, Hoppe et al. [13] proposed the below picture.

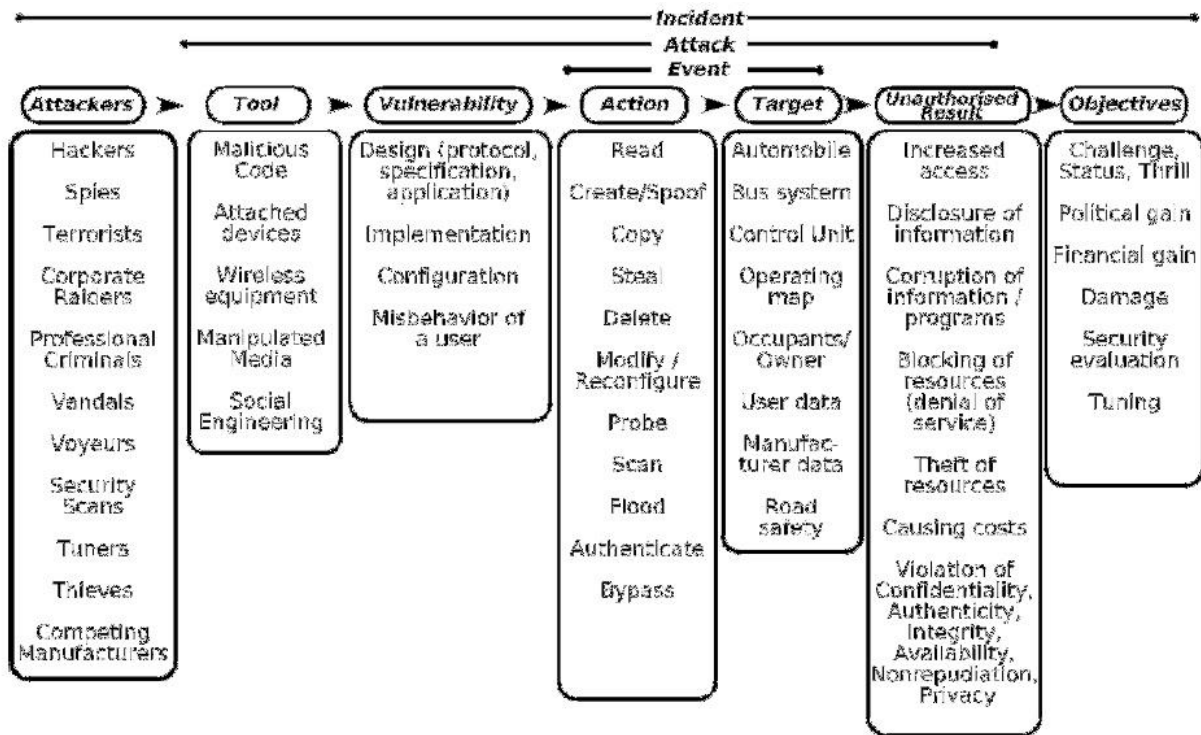


Figure 18. Detailed incident taxonomy adapted to the vehicle environment

Although this taxonomy addresses computer and network incidents, Nilsson et al. [15] have decreased the scope to be fitted to vehicular networks. They concluded that from all the mentioned actions, an attacker in a vehicular network can just Read, Spoof, Drop, Modify, Flood, Steal, and Replay the traffic. Although some of these are applicable in any ECU, actions such as Drop, Modify, and Steal can only happen in gateways. They further define each action and also present a visual building block. On the other hand, some owners of the cars modify the property of their car in an unauthorized way to get some benefit out of it, Hoppe et al. [13] added 'tuner' as an attacker and 'tuning' as a possible attack to the taxonomy to adjust it to the vehicle networks. In the following part a summary of these actions is presented.

Read

In CAN network, data are transferred in plain text which means lack of confidentiality protection. An attacker can gain any transferred information between ECUs since traffic is broadcast.

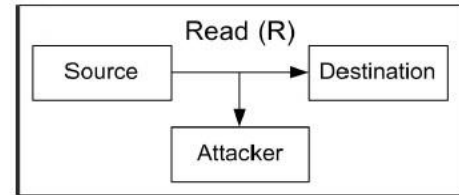


Figure 19. Read Building Block

Spoof

Lack of data authentication, a malicious user can inject infected or unauthorized messages to perform arbitrary actions. It also can blame an innocent ECU for its infections.

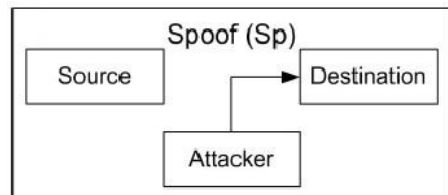


Figure 20. Spoof Building Block

Drop

A malicious node in CAN network is able to drop messages. It can be led to an availability attack. For instance, an attacker can ban forwarding messages in a gateway in CAN to perform a Drop attack.

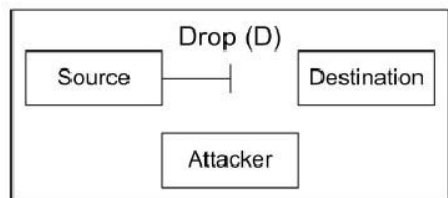


Figure 21. Drop Building Block

Modify

Lack of Integrity mechanism in CAN allows modification. As it is shown in the picture all the previous attacks are involved to conduct a Modify attack.

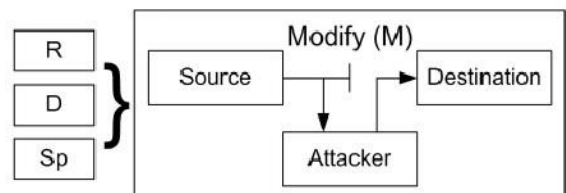


Figure 22. Modify Building Block

Flood

A malicious user can occupy lines by sending out a high rate of spoofed messages. It can lead to a Denial of Service attack. Such an attack is a consequence of availability issues.

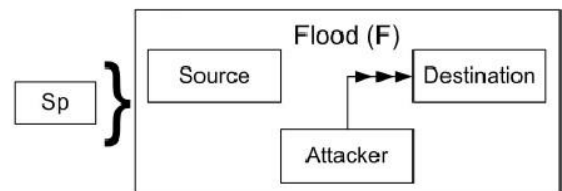


Figure 23. Flood Building Block

Steal

An attacker can use Drop and Read attacks to perform a Steal attack. In fact it abuses lack of confidentiality and availability.

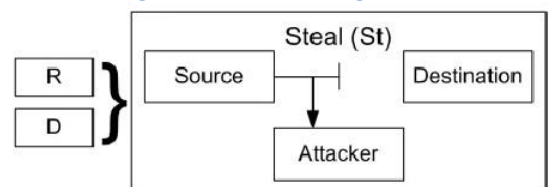


Figure 24. Steal Building Block

Replay

There is no feature to confirm freshness of the messages. An attacker can use it and capture important messages and resend it when it is needed to get arbitrary data in any time.

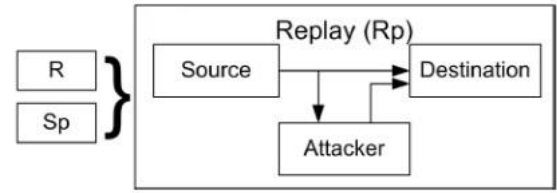


Figure 25. Replay Building Block

Security Property	Read	Modification	Drop	Spoof/ Create	Steal	Flood	Replay
Confidentiality	+	+			+		+
Integrity		+		+		+	+
Authenticity		+		+		+	+
Availability		+	+	+	+	+	+
Non-Repudiation			+			+	

Table 3. Attack type and violated Security Property

According to [15], these attacks can be combined in various ways to launch more complex attacks. They also introduced the concept of a vehicle virus. Such a virus is a means by which an infected piece of code can provide catastrophic results to vehicle and passengers. It can be triggered by numerous events to execute this code. Since viruses are more complex to generate than typical attacks, an intruder needs to have a complete knowledge regarding ECUs connected to the BUS and their responsibilities. Besides, she/he needs to have knowledge about installed firmware on ECUs, remote commands, and also a way to get access to these networks.

4.3 ECU classification

Nilsson et al. [14] classify ECUs based on the safety affect in time of a failure. They defined 5 categories- Powertrain, Vehicle Safety, Comfort, Infotainment, Telematics.

- **Powertrain-** This category consists of the most critical resource controls. Brake system belongs to this category. A failure in ECUs of this category can lead to catastrophic results on driver safety.
- **Vehicle Safety-** ECUs that are used as safety assistance. Anti-lock braking systems, tire pressure monitoring, airbag, and collision avoidance systems are some examples. A failure in such ECUs might endanger driver safety.
- **Comfort-** ECUs that are used as driver assistant. Thermal management, parking assistance belong to this category. A failure in these ECUs does not lead to an immediate safety issues.
- **Infotainment-** ECUs belong to this category support multimedia and transmission and reception to/from external resources. TV, audio streams, traffic and weather information are some examples. Any failure will not provide safety issues.

- **Telematics**- this category provides mobile communication as well as supports networked applications. Any failure in this category might distract driver.

They used safety integrity level (SIL) which is the probability of well functioning of safety related systems under all the required conditions in a stated time. It assigns a number to each level of controllability and probability of a failure in a function.

Controllability	Acceptable Failure Rate	Safety Integrity Level (SIL)
Uncontrollable	Extremely improbable	4
Difficult to Control	Very remote	3
Debilitating	Remote	2
Distracting	Unlikely	1

Table 4. SILs according to controllability and probability of a failure [14]

Moreover, they introduced Safety Effect Levels of security threats (SEL) for “identifying the safety effect of a specific security threat”. The below table represents SEL values of each safety effects.

Safety Effect	Safety Effect Level (SEL)
Disastrous	4
Severe	3
Mediocre	2
Distracting	1

Table 5. SEL according to the safety effect of security threads [14]

And finally they assigned a SIL value to each category of ECUs .The below table shows SIL value and corresponding ECU category [14].

ECU Category	SIL
Powertrain	4
Vehicle Safety	4
Comfort	2
Infotainment	1
Telematics	1

Table 6. SIL values for ECU categories [14]

They also made a classification on the cyber attacks and assigned SEL values to each type of attack. Using this classification and possible safety threats in the next chapters, it is intended to evaluate the affection of same attack on different types of ECUs as well as represent SEL value of that attack.

5. Threats and Attacks

Not only a malicious user connected to a vehicle can circumvent the control systems but also an external attacker can breach security. Exploiting one or more mentioned security requirements, an intruder may get access to the network and try to infect ECUs or gateways. For instance, a physical (direct) access or an indirect access by infected CD or DVD can harm MOST. Since vehicular networks are interconnected, infection of one of them may affect functionality of the others. The attacker can also inject malicious code into them or conduct one of the attacks that have been discussed in previous chapters. Gateways may provide hazardous communication that can endanger whole vehicle. Besides getting direct access, an attacker may use wireless communication to get unauthorized access to the vehicle.

In this chapter previously done attacks on CAN-bus are going to be reviewed. It is intended to have an overview of what has been done till now in order to extract remaining area.

5.1 Attacks

Basically, in some of the investigated papers, it is assumed that an intruder could get a physical access to the network, while in the others the attacker could inject malicious code and reduces the need to a direct access for example by exploiting unprotected diagnostics interfaces, infecting media systems by manipulating update discs or exploiting potential wireless communication system vulnerabilities. Consequently, the in-vehicle networks need to be secured enough and satisfied all the aforementioned security properties.

5.1.1 Internal Attacks

Replay Attack

Hoppe et al. [13] have implemented two sub-networks of CAN- Powertrain and Comfort. They have defined two scenarios to get unauthorized access to the window lifting system by performing a replay attack. In their first scenario, they capture state message from the window system and injected it (Replay attack) at later time. Although the window system sends its state periodically, the replayed malicious message enforces it to open the window. In the second scenario, malicious code that performs the same attack is triggered when the car's speed exceeds 200 Km/h. Speed information needs to pass from the gateway between the powertrain subnetwork to be broadcasted within the comfort sub-network. Consequently, it can be sniffed by any ECU in the comfort CAN bus.

Both examples exploit the same design vulnerability; CAN does not provide authentication neither for sender nor for receiver which opens up for spoofing and sniffing attacks. Lack of freshness in the first scenario and broadcasting messages in CAN in the second lead to these security breaches.

Moreover, Hoppe et al. [13] have also described another target used in order to perform a reply attack: the warning lights. Used for triggering and intrusion into a parked car, the warning lights turn on, for example, in the case of an unauthorized opening of a door. Looking from a more technical angle, in the CAN network, this event triggers a message in the Comfort system ECU to set or unset the warning lights. Every component connected to Comfort CAN subnetwork, can interfere

by sending an “off” message, once an “on” message is seen on the network. This led the authors to discover that indicator bulbs (objects used in order to simulate warning lights) “stay completely dark most of the time” and “sometimes only a short, weak glowing appeared” even if the “on” message was not removed from the network.

The same paper described an analysis of the Airbag Control System. The authors removed the system and emulated the behavior of a fully functional one. Another idea was to identify all CAN messages expected from an ABS system and fooling CAN by replying to messages in the network sub-network. Even if this does not reduce the driver’s control of the car, the safety implications are enormous in an emergency situation.

Read Attack

Nilsson et al. [15] performed a couple of attacks on CAN’s ECUs such as door, engine, remote lock etc. by inserting malicious code in an infected ECU. To perform a Read attack on this bus, it was enough to listen to the traffic and capture the intended messages. Such message can then be sent to a remote location via the wireless interface. In [15] they capture ‘Locking Remote Control Request’, ‘Vehicle Motion’, ‘Window State’, etc.

Lack of confidentiality in CAN provides messages to be sent in plain text. Consequently, an intruder may get access to the network and easily listen to the traffic and capture whatever message he/ she is looking for.

Spoof Attack

An attacker who knows a message can create and send it when he wants to abuse it. In [15] it is shown that an ‘UnlockingRemoteControlRequest’ message can be created and sent to unlock the door.

Lack of data confidentiality, an attacker can read a message and at a later time retransmit it. Due to no authentication, this fake message may trigger unwanted actions to occur at inappropriate time.

Modify Attack

In [15] by changing the data value of a ‘LockingRemoteControlRequest’ to become an ‘UnlockingRemoteControlRequest’ an attacker can simply open the door of the vehicle. A message can be modified when it is leaving the ECU creating it or at the gateway when it is sending to other CAN networks. A captures message can also be modified and replayed at a later time.

Lack of data integrity check in the CAN protocol can lead to such attacks.

Denial of Service (DOS) Attack

CAN uses CSMA/CD access control mechanism which is priority base. To conduct a DOS attack, it is possible to produce messages with topmost priority to prevent lower priority messages. Moreover, using fault confinement mechanism in CAN, a malicious user can disconnect an ECU by sending several error flags.

Vehicle Virus

Nilsson et al. [15] combined different types of attacks to make a virus. The virus will be triggered after sending a 'LockingRemoteControlRequest'. It generates a 'UnlockingRemoteControlRequest' and then start the engine.

5.1.2 External Attacks

In contrast to the internal attacks, there are a broad range of external attacks. In this kind of attacks it is assumed that attacker does not have a direct physical access to the vehicle. Koscher et al. in [16] have divided this type of attack into three possible classes- indirect physical access, short-range wireless access, long-range wireless access.

- **Indirect physical access**

Two main interfaces have been introduced by [16]

- **On-Board Diagnostics (OBD-II) port-** which service personnel uses it to diagnostics and re-flashing (updating) ECUs. It provides a direct access to all CAN networks in a vehicle. Either a scan tool or a PC-centric approach is typically used to access this port. In the second approach a hardware device which is referred to as PassThru plugged into OBD-II and also connected to a laptop via USB or WiFi. In both cases a laptop or a computer manages connection to the vehicle. The attacker can compromise this system to get access to the vehicles.

Vulnerabilities:

There are two vulnerabilities in PassThru device:

- 1) Lack of authentication, an attacker on same WiFi network as PassThru can connect to it. If it is connected to a vehicle, he/she can get access to the vehicle as well.
- 2) It is feasible to infect a PassThru itself and then compromise a wide range of vehicles.

- **Media Players-** recent vehicles provide means to play a CD or DVD with a variety of audio formats and also a digital multimedia port to play files on USB or iPod/iPhone. An attacker might conduct his/her attacks through an infected CD or audio file or compromise a user's phone beforehand to infect internal networks when it is connected to it.

Vulnerabilities:

- 1) This media player can recognize a special formatted CD contains specific files that prompts user with a cryptic message. In case of an incorrect answer it can re-flash related ECU.
- 2) Such media players can parse complex files which gives an ideal opportunity to the attacker. A WMA audio file that sends CAN packets when playing in a vehicle player was made.

- 3) **Buffer overflow:** one of the file read functions accepts arbitrary length input. But it does not make an important threat since the overflowed buffer is not in a stack.

- **Short-range wireless access (from 5m to 300m)**

It is considered that an attacker is able to place a wireless transmitter close to the target car to send malicious codes over Bluetooth, Remote Keyless Entry, Tire Pressure, RFID Car Key, Emerging Short-Range Channels (802.11 WiFi).

They focused on Bluetooth capability which is built into telematics unit.

Vulnerabilities:

1) Indirect: using a compromised smartphone as an intermediary device, the attacker can have paired Bluetooth device to the vehicle. They used a Trojan Horse Application on the HTC Dream (G1) phone running Android 2.1 to infect car's telematics units and then the critical ECUs.

2) Direct: attacker needs to know the vehicle's Bluetooth MAC address and then pair her device with the car. They used Bluesniff and a specific software to retrieve mentioned information from previously paired device. After that, using brute force attack, they found shared secret.

- **Long-range wireless access (greater than 1 Km)**

There are two categories in this type of external attacks- Broadcast channel and addressable channels.

- **Broadcast Channel-** can be heard by a receiver on demand. It also can be used as a control channel for triggering attacks.
- **Addressable Channels-** a continuous connectivity via cellular networks to support safety, early alert of mechanical issues, anti-theft and conveniences features. These channels are accessible over arbitrary distances.

Vulnerabilities:

AqLink protocol is used in the telematic unit to transfer/receive data and voice to/from Telematic Call Center (TCC). Reverse engineering aqLink protocol, they find following vulnerabilities:

- **Vulnerabilities in Gateway:** While aqLink supports packet size up to 1024 bytes, custom codes that glue aqLink to the Command program assumes maximum packet size 100 bytes. This can lead to a stack-based buffer overflow.
- **Vulnerabilities in Authentication:** When a car receives a call, it promptly responds by a three byte authentication challenge which is a random value. Then TCC send back hashes the value and adds a 64-bit pre-shared key to the vehicle. In case of any fault or error, the system will resend it until it receives an acknowledgement. During this time,

the system is locked. The problem with this system is that the random challenge is not really random. In fact it is static and identical. Moreover, calling to a vehicle while the telematic unit is off, generate same challenge which enables an attacker to perform a replay attack. And lastly, one out of every 256 calls which has an incorrect formatted message will be considered as valid. It can provide exploitation situation.

6. Penetration test

The purpose of this section is to give a brief and basic overview of penetration testing. The aim is to bring to light the aspects which have to be analyzed when performing such a test. In order to assure protection to a network, strong knowledge about current and past vulnerabilities is required. Patching all equipment as soon as patches are made available could bring a short term solution for protecting a system but the environment can still remain vulnerable to attacks even after patching it. In this case, penetration testing can be a very useful tool to determine potential impacts on a system.

6.1 What is a Penetration Test?

A penetration test is an authorized and scheduled process aiming to verify that applications, networks or systems are not vulnerable to a security risk that could allow unauthorized access to resources. It consists of using an automated or manual toolset to test a resource, host or network [21].

A penetration test usually involves the use of attacking methods that used by hostile intruders or hackers. In order to resolve the vulnerabilities identified by the test, the results are documented and presented to the owner of the system [20].

A penetration test does not represent a full security audit as it is basically an attempt to breach the security of a network or system. It worth to keep in mind that such a test represents just a snapshot of the system at a moment in time.

Such a test can be useful for determining:

- To what degree the system can tolerate real world-style attack patterns
- What level of ability an attacker needs to have to compromise the system successfully
- Additional countermeasures that could secure the system
- The ability of network administration to detect attacks and respond appropriately

It is important to know that such a test can never be eliminated but it can mitigate the risks.

6.2 Internal VS External Penetration Test

Tests should be in form of upmost probable and damaging attacks patterns including worst-case scenarios such as a malicious administration. A penetration test scenario typically simulates either inside or outside attacks. Both types of attacks can be tested as well which testing outside attacks has higher priority. Depending on what should be tested and by whom this process has to be done, two types of penetration tests exist: internal and external.

- **An Internal penetration test** aims to discover vulnerabilities that physical access or social engineering exposures. By simulating internal attacks, an internal penetration test is used to

determine what vulnerabilities exist for systems that are accessible using authorized network connections (inside the organization network domain). Testers act as a malicious insider who has been granted some level of accessibility. They always try to gain a greater level of access to the network. Moreover, Testers are provided with network information that someone with their level of access would normally have—generally as a standard employee or even a system or network administrator.

- **An External penetration test** is intended to determine the vulnerabilities that exists in the connections that on organization has with the Internet (firewall, gateways). Moreover, it simulates attacks by the third party who does not have specific knowledge of the target. To perform such a test, testers basically do not have any information about the system and might get required knowledge by collecting information from different resources.

6.3 How to perform a penetration Test?

6.3.1 Process and Methodology

As it is depicted in the below picture, conducting a penetration test involves systematic steps as: [20]

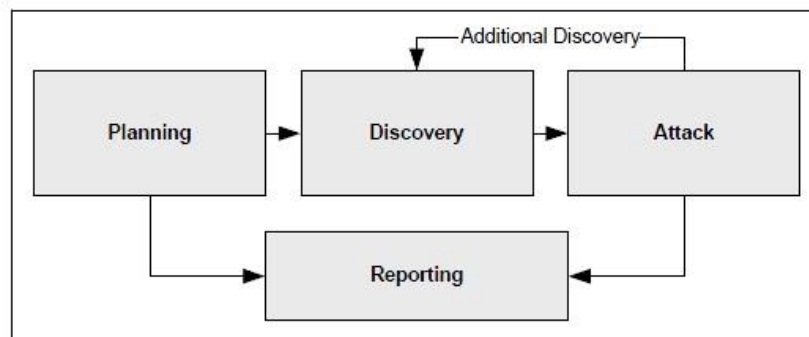


Figure 26.4-stage Penetration Testing Methodology [17]

- **Planning and preparation** – determining the scope and the objective of the test, the targeted systems and networks, the staff involved and the form in which the results of the test is presented. There is not any actual test in this phase.
- **Discovery** - It involves two parts:
 - **Information Gathering and Analysis** – it starts the actual testing with getting as much information as possible about the system targeted (there exist an important number of online resources and tools available for gathering information)
 - **Vulnerability Detection** – after analyzing the information gathered at the previous step, in this phase existing vulnerabilities in the targeted systems have to be determined. It is involved with comparing the information gathered of the target

scanned system with vulnerability data base. Having a full list of available vulnerabilities and patches for the system under such a test would be absolutely helpful [18].

- **Attack** – it is the main part of the penetration test and involves different individual steps. The previously identified vulnerabilities in this phase are the goal to exploit. In case of a successful attack, the vulnerability is verified and needed to be mitigated. Moreover, tester can install more tools on the target system to get access to additional recourse on the target network. Such attacks have to be conducted on multiple systems to determine the level of accessibility of a malicious user. The picture below represents a full process of this part.

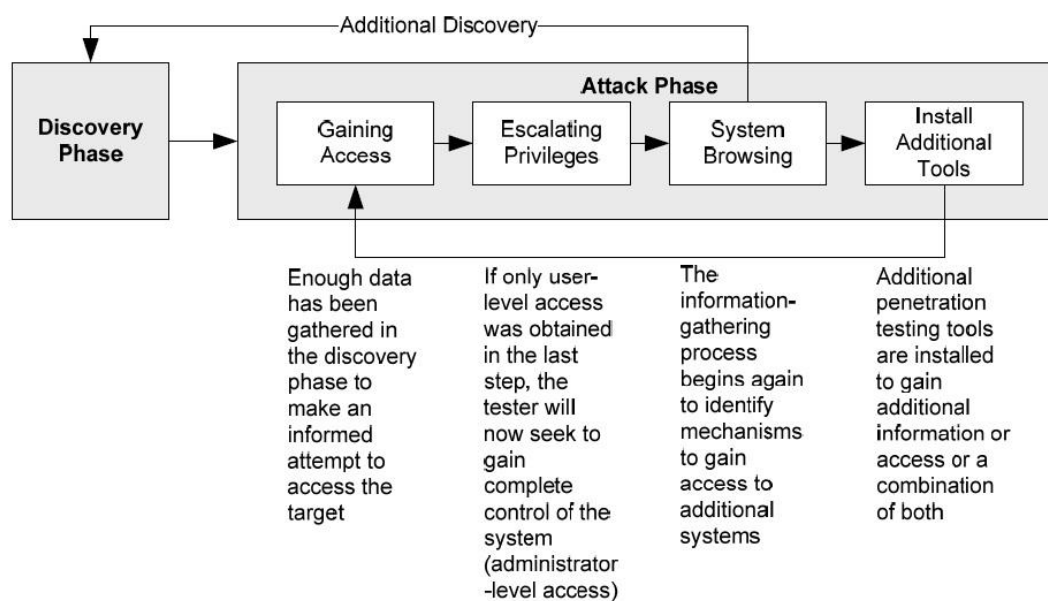


Figure 27. Attack Phase of Penetration Testing [17]

- **Analysis and Reporting** – it is important to provide the results of the penetration attempt in order to give a detailed analysis of how an attacker could exploit the vulnerabilities founded in the system (keep in mind that separating vital vulnerabilities from less vital ones would help an organization to take a decision).
- **Cleaning Up** – all changes or additions done in order to conduct the test have to be cleaned up at the end of the process.

As it is shown in the both pictures there is a link from Attack part to the Discovery part. It presents the fact that a Penetration testing is an iterative process that leverages minimal access to gain greater access.

6.3.2 Rule of Behavior

The 'rules of behavior' provide authorization to proceed the penetration test and define [21]:

- The scope of the test consist of the limitations related to the target
- The type of the test, using approaches and techniques
- The risks involved by performing a penetration test

6.4 Criteria of Success

As a penetration test requires special preparation and environment, it is important to respect the timeframe and the conditions agreed in the 'rules of behavior'. Any test procedure no respecting this timeframe should be avoided. Once the success criteria (prescript in the 'rules of behavior') have been accomplished, penetration attempts should be rapidly and safety terminated.

There are different goals for a penetration test [21]:

- Reading restricted files
- Altering restricted files
- Reading transaction data
- Access to internal resources
- Controlling network management and systems
- Demonstrating ability to control the network
- Access to any user account
- Access to supervisor privileges

It is important to keep in mind that success criteria have to be well defined as failure to properly define conditions can result in unmet expectations and could lead to a false sense of security.

6.5 Penetration Approaches

Basically, there exist three types of penetration approaches:

- **Zero knowledge** - this type of test provide the most realistic penetration test as the testers have no real information about the target environment. In most of the cases this attack has to begin with information gathering.
- **Partial knowledge** - this type of attack is done by already having an information base that an organization can provide to the test team in order to save time and expenses. To perform this kind of attack the organization might provide network topology documents and policy or any valuable information and could make the test team focus on just a part of their system.

- **Full knowledge** - this type of approach is designed to simulate an attack that can be done by someone who has intimate knowledge about the organization system like an actual employee.

6.6 Limitations

There exist several restrictions on the validity of a penetration testing. Time restriction and expense are among the most important ones. A penetration test only evaluates security of the presented network or system at the moment. Since the system or network might change by adding a new application or systems and also identified security issues are limited, the result of such a test will change by time. In fact, this test is only performed on the current vulnerabilities that are known by the tools and packages [19]. The amount of the collected data in a given period of time is the most important factor to evaluate the validity of the test.

Limitation of the attack and the approaches used to attack the system which are included in the “rule of behavior” serve severity of the test.

7. Practical implementation of the test set-up

7.1 Software Simulator

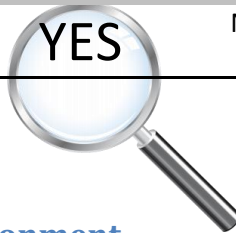
As the network simulator software, CANoe 7.6.68(SP3) from Vector Informatik is used to implement CAN networks. Vector Informatik provides solutions for the networking of distributed systems, automotive networking, truck and vehicle networking, aviation networking. Their principal product, CANoe, is a software tool for design, simulation, testing, analysis and diagnostics of the entire ECUs networks but also of individual ECUs. This product is widely used in the automobile industry (only business clients can get access to a complete version) for developing and testing of embedded automotive systems or performing diagnostics [9].

Several car manufactures use CANoe to model entire board networks of their car series. Since it enables us to simulate a complete automotive network with several ECUs, we have decided to use this platform in order to implement and simulate the penetration tests and demonstrate that hypotheses that have been made on CAN security problems are real and can lead to safety problems.

The table below shows the differences between CANoe, GNS3, and NETSIM. In the project description it was mentioned that either GNS3 or NETSIM will be used as network simulators. According to the below table, CANoe seems as the best alternative.

Table 7. CANoe, GNS3 and NETSIM

	CANoe	GNS3	NETSIM
Principal functionality	Simulation	Simulation	Simulation
Complexity	Medium	Low	Cannot judge(Never used)
Ease of use	Medium	Low	Cannot judge(Never used)
Price	Not known	Free	\$1500
Experience in using the software	0	0	0
Special designed for In-Vehicle Networks use	YES	No	No



7.2 Description of test environment

CANoe is a software which is widely used in the industry environment. This master thesis leads its investigations through this software. Conceived as a base of how a penetration test could be implemented, this report is also designed to give the reader an overview of how to develop a CANoe application. Though, this section will give the most important steps to follow in order to have an overview of how CANoe works.

Conceived as a base of how a penetration test could be implemented, this report is also designed to give the reader an overview of how to develop a CANoe Application. Though, this

section will give the most important steps to follow in order to have an overview of how CANoe works.

Before explaining the development process of a CAN Application in detail, it is important to be aware of how CAN networks are designed and how they are dealt with in this software. As presented in the previous sections, CAN Networks consist of an interconnection of ECUs. In a network the ECUs are connected to one another through their Network Nodes and exchange information over the bus.

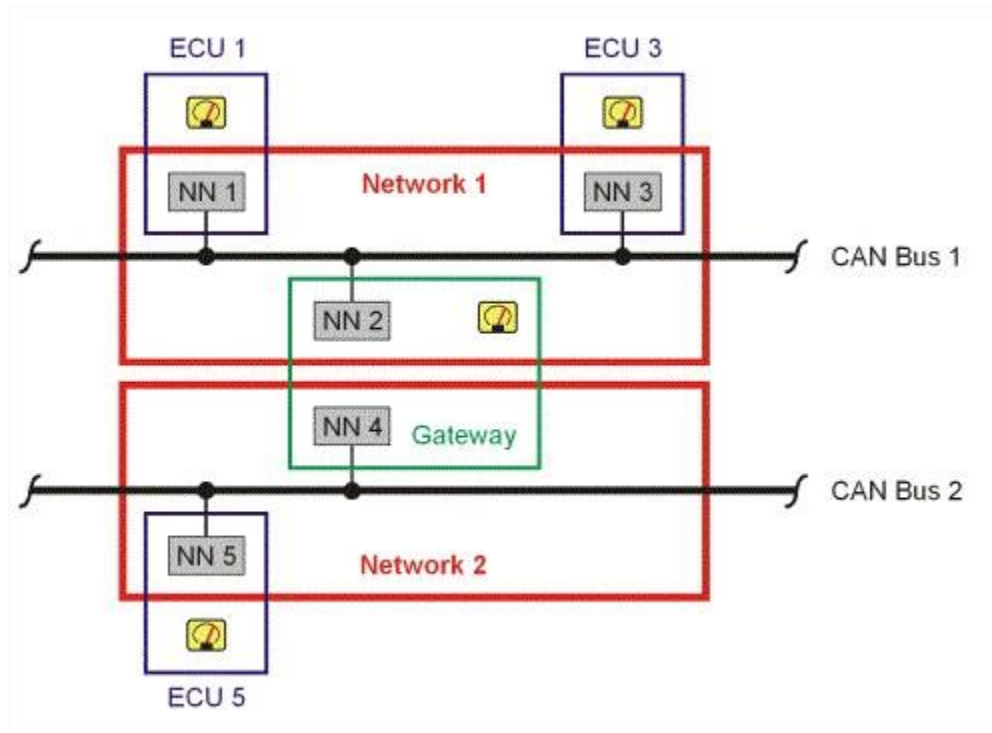


Figure 28. CAN Network Gateways

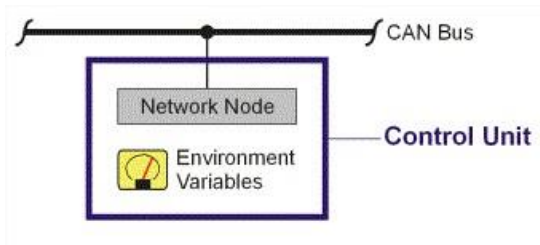


Figure 29. Electronic Control Unit (ECU)

Each ECU represents a distributed processing unit in the network. The network node has the role of performing the information exchange between the ECUs. Each ECU has its own network node and network nodes represent the control unit's interface to the CAN bus.

Environment Variables consist of input and output variables of network nodes, such as switch position, sensor signals and actuator signals.

As the test cases that have been proposed are oriented into the Power Train and Comfort CAN buses, one of the subset of a simplified car's automotive components provided by CANoe was preferred as a network topology for simulating the attacks. This environment needed some changes in order to be adapted to the project needs and be more suitable to the project expectations.

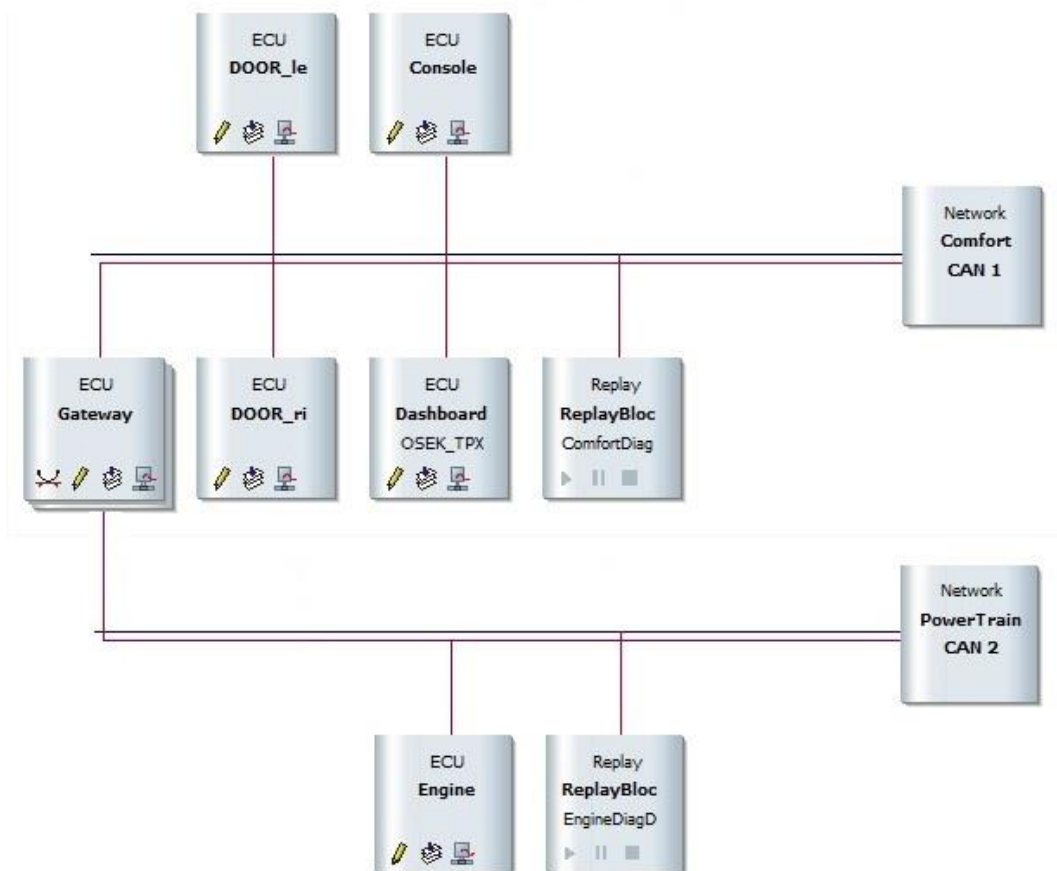


Figure 30. Simulated CAN configuration

The configuration shown in the Figure above uses as a base the demo-environment provided by CANoe but differs from the original one by removing some blocks (NM_Tester and NM_Tester_C, blocks used in network management) that are not interesting in the present test implementation. Also, the target of this project is to test security issues of some specific ECUs: Engine, Door Left, Door Right, Dashboard, and Console. The architecture was therefore simplified in order to narrow the scope on the targeted ECUs but also to simplify software and CPU resources.

The setup consists of two separated CAN bus networks- The Powertrain (CAN2) and the Comfort (CAN1). These two subnets are connected via a Gateway which assures that messages are passed from one subnet to another. For example, car speed, engine speed or the temperature of

the engine are the messages generated by the Powertrain network which have to be sent through the Gateway, in order to arrive in the Comfort network and to be displayed to the driver.

7.3 Overview of a CANoe Application

In the following sections, the development process of the CANoe application will be shortly explained. Technical details can be found in the CANoe Tutorial provided by Vector Informatik and they will not be discussed in this paper. To have a better understanding of the chosen architecture, each step will be explained and exemplified through the setup used in this project.

To make a configuration using CANoe, it is needed to follow these steps:

- 1) Database creation
- 2) Database nodes creation
- 3) Database messages and signals creation
- 4) Database association and nodes addition to the network
- 5) Panels creation
- 6) Node behavior creation

Step 1: Database Creation

Each CANoe Application needs a database. All information that is processed in a networked CAN bus system, as well as the interrelationships between different units of information, are managed in a database. CANDb++ is a data management program which can be used to create and modify these databases.

Practical example: database used in this project

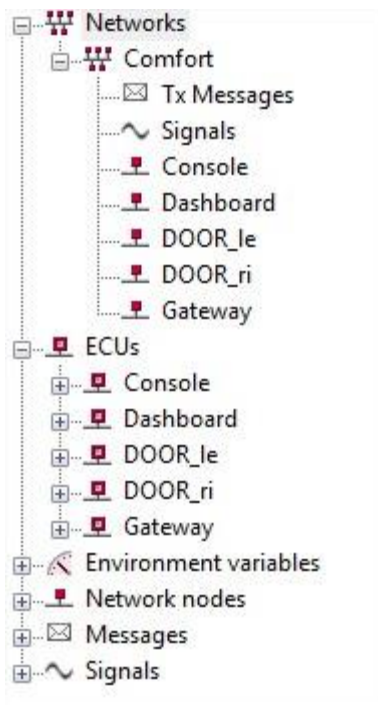


Figure 31. Comfort subnet database

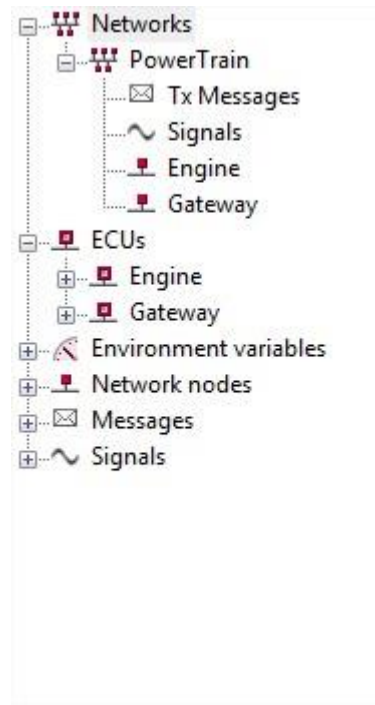


Figure 32. PowerTrain subnet database

Step 2: Database nodes creation

As explained before, each ECU contains a network node, that represents the ECU interface with the CAN bus. Figures below illustrate the network nodes presented in the project configuration. It is easy to see that the ECUs, presented in figure above have the same lines as network nodes. In fact, every time a network node is defined, CANdb automatically defines an ECU with the same name, and a link is installed between the new network node and the ECU. As system parameters, a network node has a symbolic name and an address. The address must be unique between the networks.



Figure 34. Comfort subnet network nodes

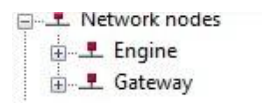


Figure 33. PowerTrain subnet network nodes

Step 3: Database messages and signals creation

In order to exchange information over CAN bus, specific messages are created. As explained in the CAN

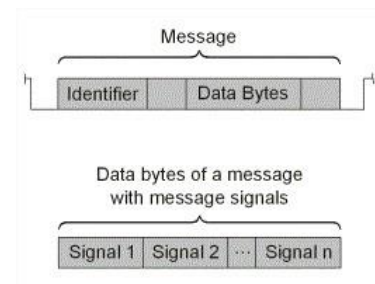


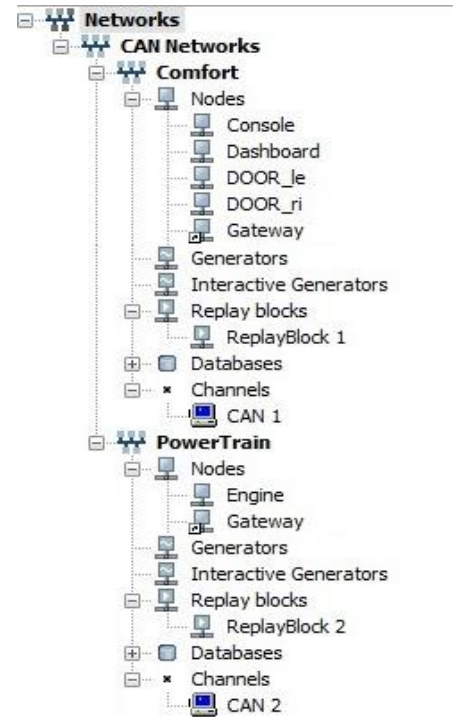
Figure 35. CAN message structure

specifications chapter, CAN messages are defined by several fields (Start of frame, ID, RTR, IDE, DLC, CRC, etc). Most of these fields are also defined in the CANdb. What is important to notice is that each message has a unique identifier (CAN ID). Each message is assigned to a network node, hence only the assigned node can send the message. In the Data field of the message, several signals can be included.

Step 4: Database association and node addition to the network

After the database has been completed and ECUs Network Nodes Messages have been created, the next step is to associate that database to a CANoe Application and to add the nodes to the networks. This figure gives an overview of how the final result could be after these steps. The image will be presented, in more details, later in this report.

Figure 5 contains a hierarchical overview of the available object types and objects. CANoe represents each architecture as a representation of the databases that has been defined previously for the work environment.



Step 5: Panels creation

Panels provide a simple and clear way to display and control interfaces, to visualize signal values, manipulate the simulation model and to evaluate measurements. The environment provided by CANoe and used in this project consists of several panels. Below Figure shows the principal ones, used in order to visualize and test the proposed attacks.

Figure shows the Control, Console, Window Position, and DashBoard panels. For example, the Dashboard Window presents the visual representation of the Dashboard ECU contained in the work environment. As this ECU is contained in the CAN1(Comfort) it cannot have access to specific values of the CAN2(PowerTrain). Since a gateway connects two or more CAN network together, data can be transferred from one network to another.



On the other hand, the Control Window represents the visual representation of the Powertrain subnet. The user can interact with the position of the key, the gear box and also with the Break. All these interactions lead to change in the Dashboard.

The Window Position panel allows the user to change the position of the window by pressing the Control Units buttons (up and down, for the right or for the left window).

Lastly, the Console Panel, enables the user to turn on and off the radio. He/she can change the channel, signal left or right turns, turn on and off the head-lights and the hazards.

Step 6: Node behavior creation

In order to give behavior to a node, the CAPL language is used. A CAPL program is usually developed in the CAPL Browser. The Browser window is subdivided into three distinctive panes. The left pane contains a tree view of all important elements for which a CAPL program can be written. The area on the upper right is where global variables will be placed for the CAPL program, and the area below it is where the actual source code for each event procedure is written. Nodes can therefore be programmed to react to messages, to set their own timers, to send messages, etc.

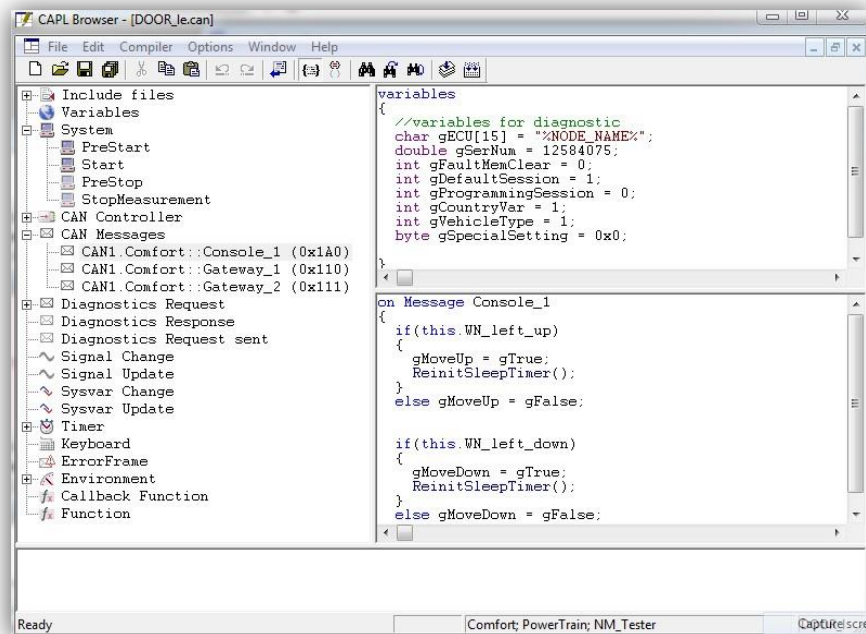


Figure 36.Behaviour of DoorLeft Node

7.4 Test implementation

To evaluate CAN bus security in this master thesis, we have divided probable vulnerabilities into two classifications- Internal and External vulnerabilities. Due to lack of equipments, only internal vulnerabilities are practically investigated. Furthermore, internal vulnerabilities themselves have been classified into three groups of flaws- Security Properties (CIA) based flaws and CAN Frame based flaws, and Implementation based flaws. In the following Sections these flaws are going to be considered as the target of the attacks to see how vulnerable is the CAN.

7.4.1 Attacks related to CIA

Attacks in this part are designed based on the lack of security properties. Unfortunately, these attacks are implemented only on simulated environment. Although it was intended to perform them on real a device, I never got the chance. Available automobiles that I got access to were differently assembled than what is considered in theory. Therefore, I only had to perform the attacks on a simulated environment. Consequently, lack of real environment result I could not compare virtual and real area's result. Here we have only mentioned basic attacks which lack only one of the security properties. It is of course possible to combine these properties to make a more complicated attack. However, this master thesis intended to investigate CAN basic shortages.

1. Read and Injection attack

The first penetration test was designed to exploit what an ECU can do to its own functionality if malicious code is injected in its behavior. This attack exploits the confidentiality property as the value of the speed is broadcasted to all nodes in the Comfort bus. Console ECU which normally deals with Radio or head-lights functionality, is then able to receive the speed value, normally used only by the Dashboard ECU. In this way, the injected code can be run.

Attack Case: When the car reaches 30 km/h, the radio displays “Penetration successful” and any attempt to try to change the radio channel is impossible once this message appears.

- The speed triggering the attack was chosen randomly in this test (the attack can be done at any speed)
- The message displayed by the radio screen was also chosen randomly (any message could be displayed, the only limit is the screen size)

Technical implementation

The speed value is transmitted from the PowerTrain Bus, through the Gateway into the Comfort Bus in order to be displayed on the Dashboard. As CAN messages are broadcasted to all the ECUs connected to a bus, the speed value can be read by all ECUs connected to the Comfort bus.

In a normal configuration, only the Dashboard has to be programmed to respond to this value by displaying it to the user. The others ECUs, normally drop this message, but an attacker can inject code in these ECUs, telling them not to drop the message but instead to do what he wants.

The following code has been injected to the Console (this ECU does not use the speed value in any way so the code was “hidden” here). As the speed value is broadcasted on the bus by the Gateway (in this configuration Gateway_2) the Console ECU responds to this message by reading the speed value and configuring the display value of the radio to “Penetration successful”.

```
on message Gateway_2  
{  
if (this.CarSpeed>30){  
putValue(EnvSetRadioChannelDSP, "Penetration successful");  
}  
}
```

Car behavior after attack

In the moment the car reaches 30km/h, independently of the radio state (on or off), the message is displayed on the radio screen. Only if the speed decreases under 30 km/h the radio car behaves normally.

The 30km/h speed was chosen because of the discomfort it can produce: it is a low speed that triggers the attack, so it will happen very fast after starting of the car (nothing guarantees that the driver will reach high speeds, but it is likely he will exceed “30 km/h” sooner or later). Besides, if a high speed would have been chosen, it is easy to go under that speed, to make the radio behave normally.

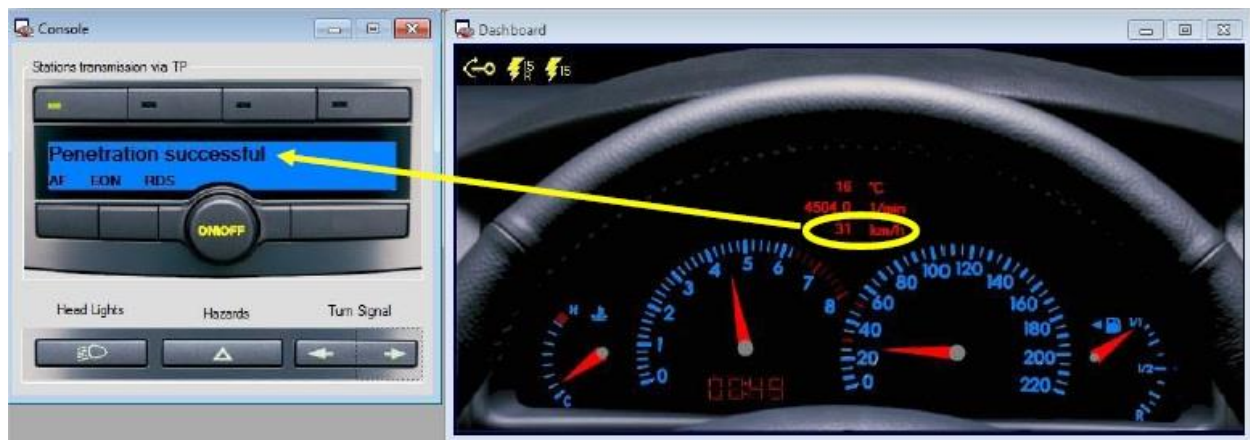


Figure 37. Read and Inject Attack - Visual results, radio on

2. Spoof attack

The second test case was designed to spoof messages and inject them into the CAN bus. To perform such an attack two scenarios have been considered.

Scenarios

- 1- Attack case 1: when the brake is pushed a timer is triggered to send spoofed messages, indicating a low-speed in the Dashboard (e.g. 20 Km/h). (Using a timer)
- 2- Attack case 2: using a Replay Block, a number of ABSdata messages contain random and meaningless values are injected from second 160 to second 164. (Using a Replay Block)

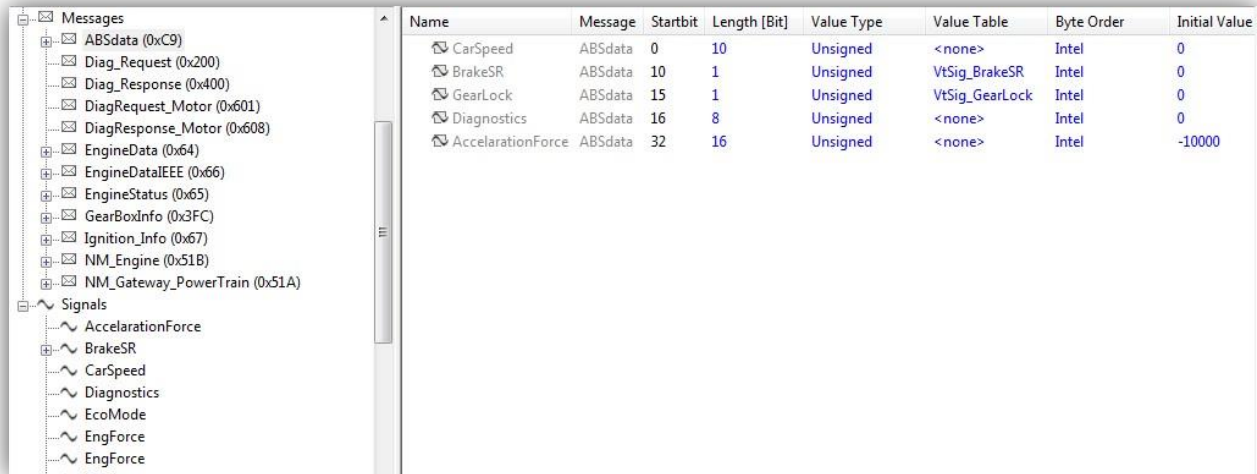
Scenarios' Comparison

While the first scenario is using a time to create infected messages, the second one is using a Replay Block to inject infected message. Moreover, the first one needs to add a new signal to the data base

Technical implementation

1- In order to build this attack, a new signal was added to the demo-environment.

In the database associated to the Comfort bus, the “BrakeSR” signal was created. Then, this signal was assigned to the “ABSdata” message (Figure), which is sent by the Engine (belongs to PowerTrain bus). In this way, each ECU in the Comfort bus can have access to the “Brake” value (“Break_Active” or “Break_Passive”).



Name	Message	Startbit	Length [Bit]	Value Type	Value Table	Byte Order	Initial Value
CarSpeed	ABSdata	0	10	Unsigned	<none>	Intel	0
BrakeSR	ABSdata	10	1	Unsigned	VtSig_BrakeSR	Intel	0
GearLock	ABSdata	15	1	Unsigned	VtSig_GearLock	Intel	0
Diagnostics	ABSdata	16	8	Unsigned	<none>	Intel	0
AccelerationForce	ABSdata	32	16	Unsigned	<none>	Intel	-10000

Figure 38.Scenario 1, Spoof Attack, ABSdata Message

Then the behavior of the network node corresponding to Door Left ECU was added. Door left was chosen, in order to dissimulate the attack (Door Left ECU has nothing to do with the brake value so the attack will be well “hidden” here).

The following CAPL lines were added to the node:

```
variables{
    (...)                                //definition of additional variables
    msTimer timerSpoofedSpeed;          //variable for the timer attack
}
on message Gateway_2
{
    if(this.BrakeSR != 0){
        setTimer(timerSpoofedSpeed,20); //launch the window timer when brake is pushed
    }
}

on timer timerSpoofedSpeed
{
    message Gateway_2 spoofedCarSpeed; //define a spoofed message
    spoofedCarSpeed.CarSpeed = 20;      // set the spoofed speed value
    output(spoofedCarSpeed);           //send data onto the bus
}
```

```

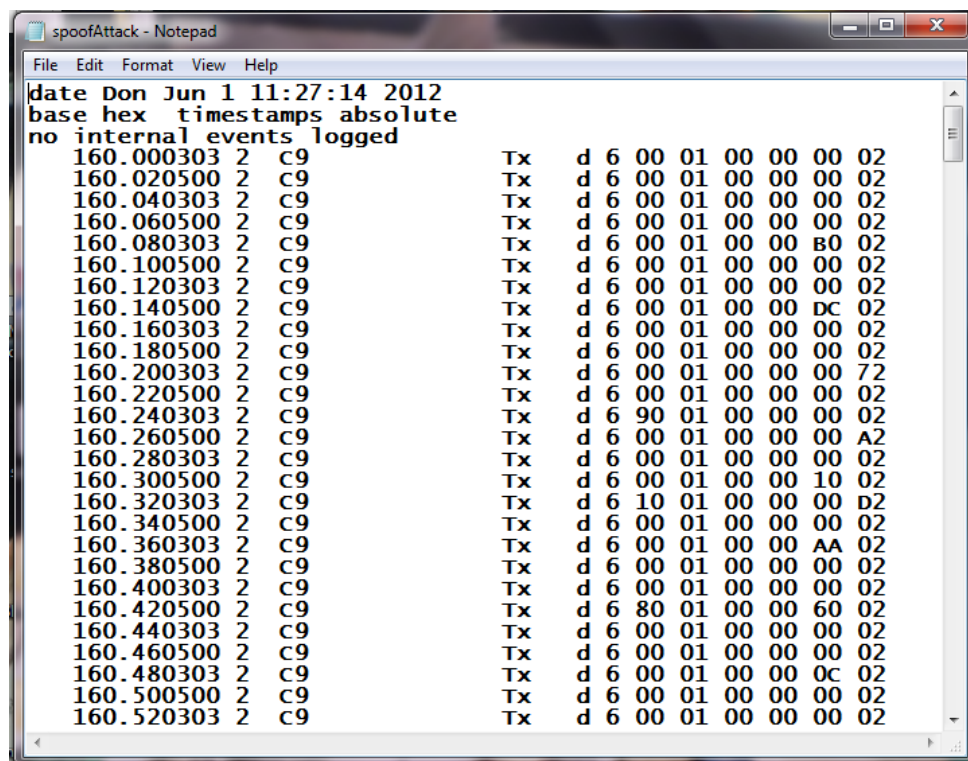
cancelTimer(timerSpoofedSpeed);           // restart the timer to repeat
setTimer(timerSpoofedSpeed,5);             // set the timer every 5 ms
}

```

- 2- As it is shown in the above picture, ABSdata contains a couple of signals to carry data. A .ASC file contains a number of ABSdata messages, has added to the Replay Block connected to the Powertrain CAN network. The format of the messages is shown below. It injects too many ABS messages with random values. Flooding too many messages in a period of time can affects car speed indicator in a way that it not only shows an incorrect value but also jumps from one value to another.

Here is the picture of the injected file contains infected ABSdata messages. Using a log of the network while it is working normally, the format of an ABSdata is extracted. Based on the extracted message, some messages with infected values are designed and used in this attack.

Figure 39. Scenario 2, Spoof Attack, infected ABSdata messages file. The first column shows time of transmission, C9 is the ID transmitting block, and the 6 columns indicate data values.



Car's behavior after attack

- 1) The below figure shows the dashboard behavior around $t = 41$ seconds after the start of the car. At some moment in time, before t the brake was active. This triggered the abnormal behavior of the speed indicator. The purpose of the "Trace PowerTrain" window is to record the PowerTrain bus activities during measurement. In the figure, it can easily be seen that that speed indicator is not synchronized with the real value of the speed. In the Trace PowerTrain, the real speed is shown, in this case: 64.5 mhp. This value is equivalent to 104 km/h but in the DashBoard the speed indicator is showing 16Km/h. This attack causes the speed indicator to jump between the real value and the spoofed value every second since real values are also transmitted on the bus.

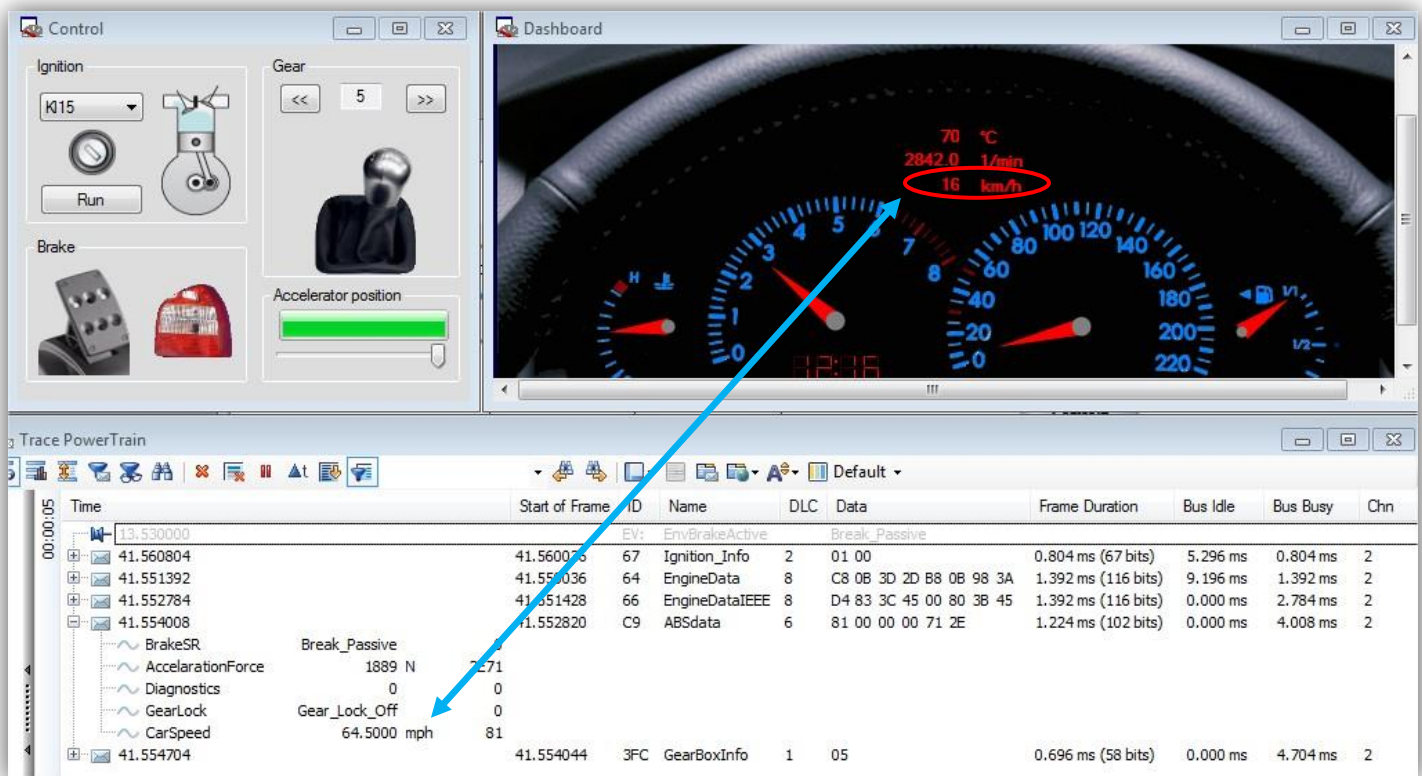


Figure 40 Spoof Attack, First Scenario, Visual results

- 2) The below picture shows that in some seconds after spreading of the injected messages in the network, CarSpeed experiences a huge distortion. This distortion is depicted in the below pictures in carSpeed diagram clearly.

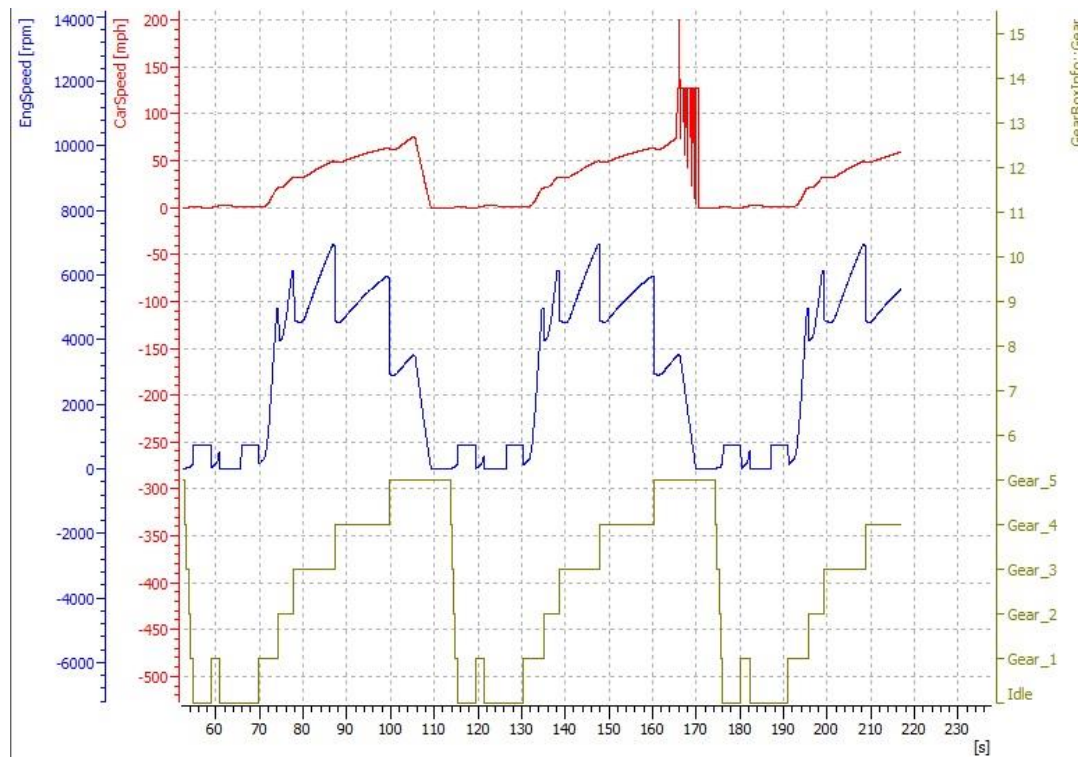


Figure 41. Spoof Attack, Second scenario, Result

3. Replay attack

This third attack was designed to use the Replay Block. This block provides the means for reproducing measurement sequences which have already been recorded. This block uses a Log file and introduces the messages contained in this file into the data flow.

Attack case: The attack is launched at the car start by turning on the replay block. This block is configured to read a log file. The lines of this file contain data to enable the headlights.

Technical implementation

This attack needs to have a preparation phase. Before launching the attack a log file is created. Depending on the implementation, messages can differ and therefore, if an attacker does not have any knowledge about the way the messages have been configured, he has to “sniff” the traffic. This is the reason, messages that the console sends when the head-lights are turned on, are “sniffed” and recorded into the log file (as this attack is focused only on the head-lights functionality, only particular messages like “turn-on the head-lights” are analyzed and logged in the file). Then these messages can be replayed into the CAN1 bus using the Replay block functionality and configuring it to use the log file.

In this attack “turn-on head-lights” messages have been aimed but there are other possible attacks that can be done. For the present test, the following file was used:

30.000303	1	1A1		Tx	d 2	01 00
30.020500	1	1A1		Tx	d 2	01 00
30.040303	1	1A1		Tx	d 2	01 00
30.060500	1	1A1		Tx	d 2	01 00
30.080303	1	1A1		Tx	d 2	01 00
30.100500	1	1A1		Tx	d 2	01 00
30.122004	1	1A1		Tx	d 2	01 00
30.141056	1	1A1		Tx	d 2	01 00
30.161056	1	1A1		Tx	d 2	01 00
30.180303	1	1A1		Tx	d 2	01 00
30.200500	1	1A1		Tx	d 2	01 00
30.220303	1	1A1		Tx	d 2	01 00
30.240500	1	1A1		Tx	d 2	01 00
30.260303	1	1A1		Tx	d 2	01 00
30.280500	1	1A1		Tx	d 2	01 00
30.300303	1	1A1		Tx	d 2	01 00
30.320500	1	1A1		Tx	d 2	01 00
30.340500	1	1A1		Tx	d 2	01 00
30.360303	1	1A1		Tx	d 2	01 00
30.380500	1	1A1		Tx	d 2	01 00
30.400303	1	1A1		Tx	d 2	01 00

Time, value
increased
every 20ms

Sender of the
message (here
Console_2)

Message

The “01 00” message means “activate the headlights”. First of all the replayed message is sent by Console_2 (ID = 0x1A1 in hexadecimal representation). This message contains the following signals: “Phase”, “Light”, “Active” as presented in figure 21. “Phase” and “Active” are not interesting here as this test focus on the “Light” signal. The length of this message is 2 bytes and the byte order is Intel (Little Indian). Therefore, when it comes to send a signal to “turn on” the head lights the message is “10000000 00000000” in bit wise representation and as “01 00” byte representation.

Message 'Console_2 (0x1A1)'						
Definition Signals Transmitters Receivers Layout Attributes Comment						
Signal	Message	Multiplexing/Group	Startbit	Length [Bit]	Byte Order	Value Type
Light	Console_2	-	0	1	Intel	Unsigned
Active	Console_2	-	1	2	Intel	Unsigned
Phase	Console_2	-	8	1	Intel	Unsigned

Figure 42. Console_2 message

signal																
bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
light "on"	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
byte				01								00				

Figure 43. "Turn on headlights" Message

Car behavior after attack

The replay block is set to begin sending messages around 30 seconds after the car has started. Therefore, till this time, the car behaves normally but after 30 seconds, the replay block begins to send messages to turn on the head-lights. The frequency of this messages is set to be around 20 ms. That explains why even if the real console (Control panel) sends messages to the ECU in order to shut the lights off, the lights still remain on.

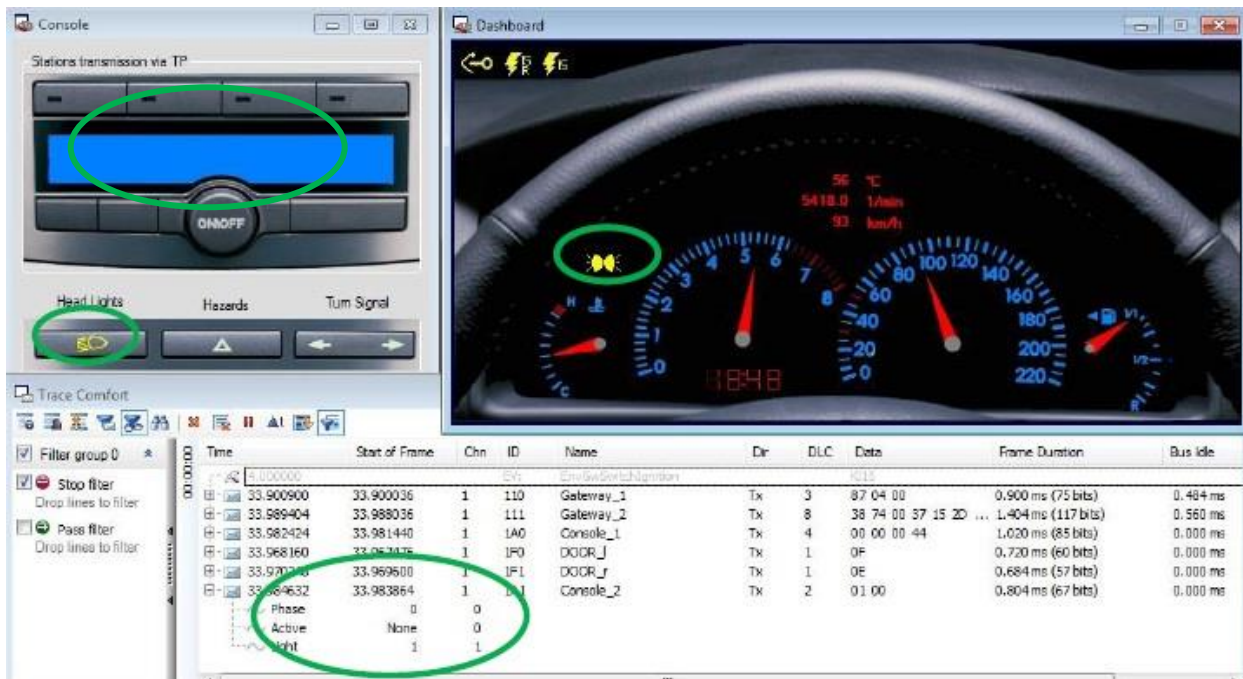


Figure 44. Replay attack - Visual results

4. Drop Attack

This attack is intended to perform a Denial of Service. Since CAN broadcasts all the messages in the network, to run a DOS attack, it is needed to prevent spreading data all around this network. As it is mentioned before, in-vehicle networks are connected to each other by gateways. Consequently, this gateways seems to be the best place to prevent broadcasting data.

Attack Scenario: When the car's speed exceeds 40 Km/H, the speed indicator stops working.

Technical implementation

To run such an attack, a new signal is defined and added to the comfort database. Gateways act as a bridge between the two in-vehicle networks. To pass data from one network to another, signals from initiating networks are needed to be mapped to the signals defined in the destination network. The idea to do this attack is that carSpeed signal from ABSdata message will be mapped to this new signal. Because this signal is an unknown signal in comfort network, it does not pass the value of car speed. Consequently car speed value will not be passed to the dashboard.

According to the below code which is triggered every time that ABSdata message changes, if the value of car speed changes and the engine is running, the new signal will be mapped to the carSpeed signal. Through this the car speed value is banned from propagation to the comfort network.

On signal ABSdata::CarSpeed

```
{ // DOS attack

double lastval;

    SetBusContext(gBusContext_Comfort);

double lastval;

if(lastval != this.phys && gEngineIsRunning)

    $Gateway_2::CarSpeed = $ABSdata::fakedsignal;

lastval = this.phys;

}
```

Car behavior after attack

This attack is located in the gateway that connects Powertrain network to Comfort network. It tries to ban carSpeed signal of an ABSdata message. This signal is provided by Engine ECU in Powertrain network. It will be broadcasted to the other networks so that the Dashboard shows the car speed to the driver. After performing this attack, the car speed's indicator seems totally broken.

As it is depicted in the below picture, while the engine is working, car speed's indicator seems broken and it does not work properly. The table besides it shows that no CarSpeed signal is propagated to the comfort network.

The picture below is a trace window of Powertrain network. It demonstrates all the messages traveling in this network. It shows that message with ID 111 contains car speed value which means that the car is moving while the responsible network and module to show nothing.

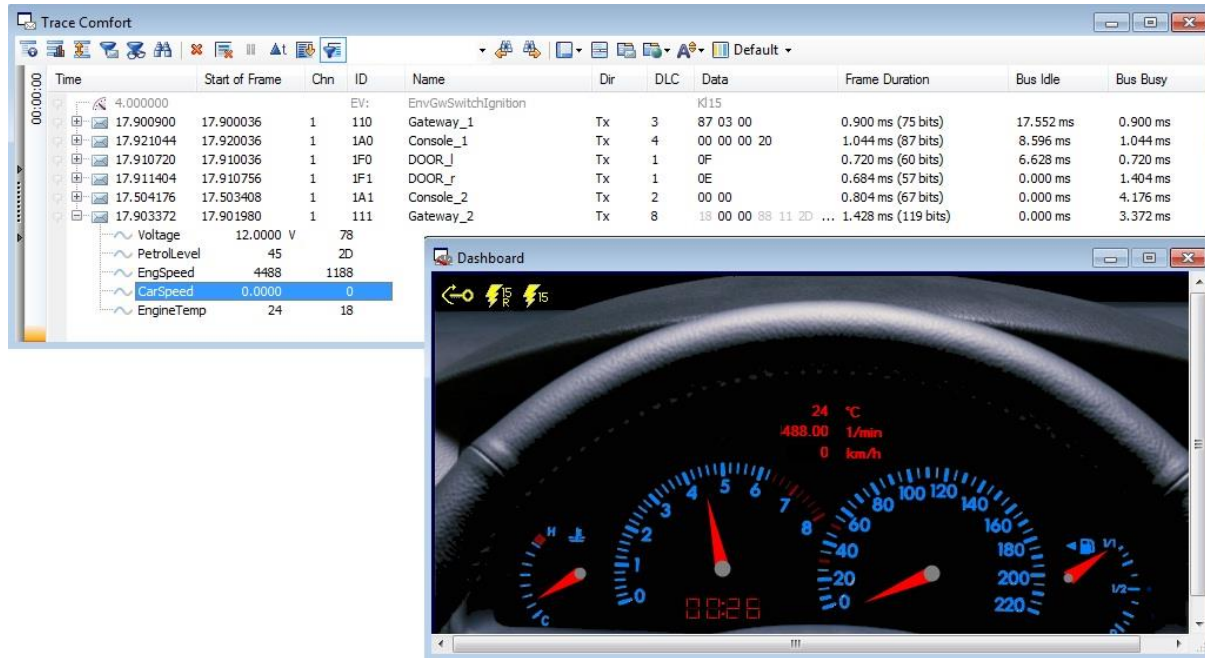


Figure 45 DOS Attack, Comfort Network and broken Dashboard

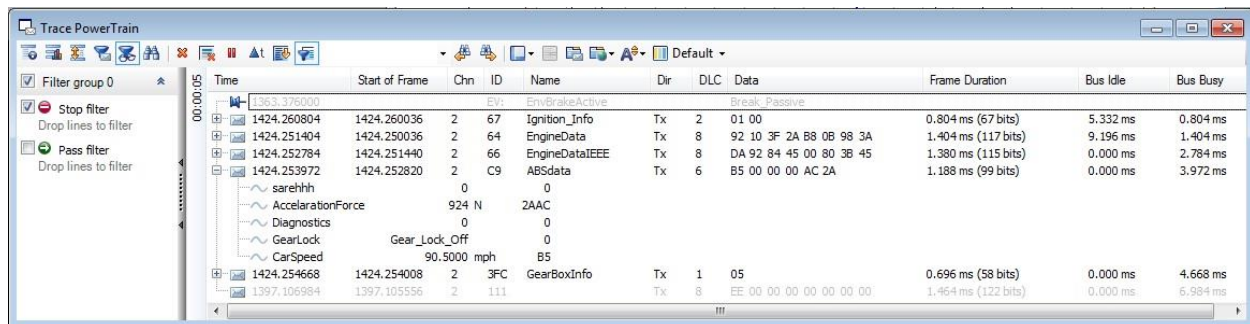


Figure 46. DOS Attack, properly working PowerTrain network

5. Modify Attack

This attack is intended to change the data in a way that the driver misunderstands the car's situation. Affecting integrity of the messages, it tries to deceive the drivers. Since data is transferring in clear text, a malicious user can make change in it.

Attack Scenario: when the car's speed exceeds 40 Km/h, the speed indicators shows a fake value which equals to a half of the real speed.

Technical implementation

To run such an attack, ABSData message will be checked in Dashboard module every time that the car's speed changes. Once the car's speed exceeds 40 km/h, signal change event will assign a value with half of the real car's speed value.

The below code is added to ABSdata signal change and trigger every time that the cars speed change.

On signal ABSdata::CarSpeed

```
{  
  
double lastval;  
  
if(this.phys * 1.6 >40)  
{  
  
    if(lastval != this.phys )  
  
        $Gateway_2::CarSpeed = $ABSdata::CarSpeed* 0.5;  
  
}  
  
lastval = this.phys;  
  
}
```

Car behavior after attack

As the car starts and speeds up to 40 km/h the car's speed indicator jumps to 20 km/h while according to the Trace power Train table real speed is something different. By this, the driver will never understand real speed and may go faster than what is allowed. Picture below

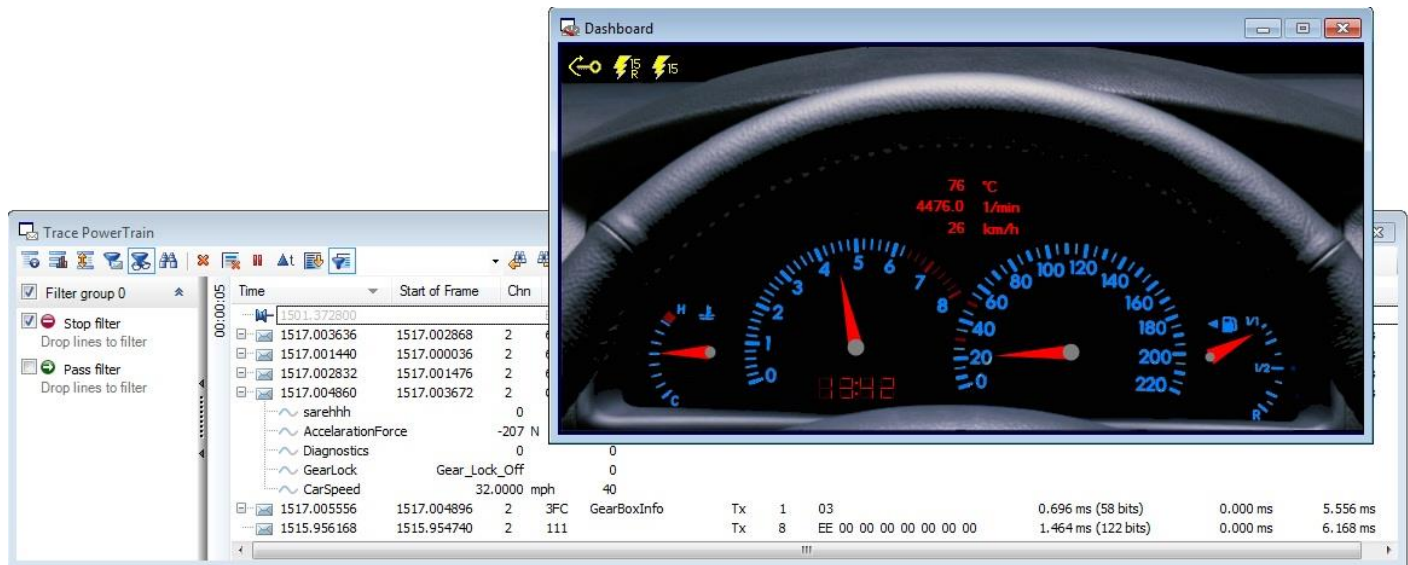


Figure 47. Modify attack, real speed is different with what the indicator shows

7.4.2 Attacks related to CAN Frame

In this section, probable flaws with CAN message's fields are going to be evaluated. According to the CANoe software, it is not possible to make change in all the fields (as test case) since the value of some of them are fixed. Therefore, only ID, Data Length Code (DLC), and Message Type Fields are available to make some changes. It is also possible to assign signals to the message's Data field by CANoe which has already been investigated in the previous part. Fortunately, attacks in this part are performed both on simulated and real environment. Consequently, a comparison on results is intended to see how different they can be.

In the following picture the boxcar which is used to run the attacks on it is depicted. All the electronic parts are connected to each other to simulate a real car. As it is clear there are only electronic parts since there is nothing to do with mechanical parts in this study.



Figure 48. Boxcar

- **DLC Field Evaluation**

As it is mentioned before, DLC field is responsible to highlight the length of the Data field. Changing this field in a way that makes inconsistency in the propagated messages may make crash or increase the network traffic dramatically. This part is going to evaluate probable affections.

1. **Injecting message with deliberately shortened Data field than normal with all possible DLC.**

Attack Scenario – Sending an ABSData (break module) message with shorter data length than what it really should be and with different possible DLC values.

- **Performing on real environment**

Considering a ABSData message with normally 8 bytes data value and DLC equal to 8, a set of ABSData messages is sent with 5 bytes Date and DLC value equal to 2, 5, and 8 (less than, equal, and higher than data length). Moreover, to have a clear result Break module is disconnected completely. A replay module is used instead.

```

date Tue Jun 12 12:54:18 am 2012
base dec timestamps absolute
internal events logged
// version 7.2.0
Begin Triggerblock Tue Jun 12 12:54:18 am 2012
0.000000 Start of measurement
0.004571 CAN 1 Status:chip status error active
0.004789 CAN 2 Status:chip status error active
1.801363 1 328 Rx d 8 238 123 0 179 144
2.441361 1 328 Rx d 5 238 123 0 179 144
2.601397 1 328 Rx d 2 238 123 0 51 176
2.621261 1 328 Rx d 5 238 123 0 35 180
2.641361 1 328 Rx d 8 238 123 0 19 184
3.221241 1 328 Rx d 2 238 123 0 67 172
3.401257 1 328 Rx d 5 238 123 0 179 144
End TriggerBlock
|

```

Figure 49. Sending shortened ABSData messages with all possible DLC value in a real environment

- **Performing on simulated environment**

Using a Replay block, same attack as previous environment is performed on this network. The differences are the ID of message which is different in each environment and the normal data length. In the simulated environment a normal ABSData message has 6 bytes data. Consequently, various ABSData messages with 5 bytes Data and DLC value equal to 2, 5, and 8 are sent(less than, equal, and higher than data length).

```

date Don Jun 1 11:27:14 2012
base hex timestamps absolute
no internal events logged

40.100303 2 c9 Tx d 6 00 00 00 02 00
40.120500 2 c9 Tx d 5 00 00 00 02 00
40.140303 2 c9 Tx d 6 00 00 00 00 00
40.160500 2 c9 Tx d 8 00 00 00 02 00
40.180303 2 c9 Tx d 2 00 00 00 02 00

40.200303 2 c9 Tx d 5 00 00 00 02 00
40.220500 2 c9 Tx d 6 00 00 00 02 00
40.240303 2 c9 Tx d 5 00 00 00 00 00
40.260500 2 c9 Tx d 6 00 00 00 02 00
40.280303 2 c9 Tx d 8 00 00 00 02 00

```

Figure 50. Sending shortened ABSData messages with all possible DLC value in a simulated environment

Result

According to the results of both environments, among all the aforementioned messages the one that its Data length is equal to DLC, is showed up on the traffic, while the two other messages were never shown up.

2.220708	1	328	Tx	d 8	238	123	0	83	168	0	0	0
2.240784	1	328	Tx	d 8	238	123	0	67	172	0	0	0
2.260917	1	328	Tx	d 8	238	123	0	51	176	0	0	0
2.280795	1	328	Tx	d 8	238	123	0	35	180	0	0	0
2.300657	1	328	Tx	d 5	238	123	0	19	184	0	0	0
2.320636	1	328	Tx	d 8	238	123	0	3	188	0	0	0
2.340724	1	328	Tx	d 8	238	123	0	243	128	0	0	0
2.360936	1	328	Tx	d 8	238	123	0	227	132	0	0	0
2.380694	1	328	Tx	d 8	238	123	0	211	136	0	0	0

Figure 51. Part of a filtered traffic of sending a shortened ABSData message in a real environment

42.620804	2	Ignition_Info	Tx	d 2 01 00	Length = 768000	BitCount = 67
42.640804	2	Ignition_Info	Tx	d 2 01 00	Length = 768000	BitCount = 67
42.651416	2	EngineData	Tx	d 8 BC 0C 3E 2D B8 0B 98 3A	Length = 1380000	BitCount = 118
42.652796	2	EngineDataIEEE	Tx	d 8 9E C7 4B 45 00 80 3B 45	Length = 1344000	BitCount = 115
42.653984	2	ABSdata	Tx	d 6 8B 00 00 00 CB 2D	Length = 1152000	BitCount = 99
42.655136	2	ABSdata	Tx	d 5 00 00 00 02 00	Length = 1116000	BitCount = 96
42.655832	2	GearBoxInfo	Tx	d 1 05	Length = 660000	BitCount = 58
42.656552		EnvSetRadioChannelDSP := "0"				
42.656552		EnvCarSpeedEuro := 0				
42.660804	2	Ignition_Info	Tx	d 2 01 00	Length = 768000	BitCount = 67

Figure 52. Part of a filtered traffic of sending a shortened ABSData message in a simulated environment

2. Injecting message with deliberately lengthened Data field than normal with all possible DLC

Attack Scenario – Sending an ABSData (break module) message with longer data length than what it really should be and with different possible DLC values.

- **Performing on real environment**

Considering an ABSData message with normally 8 byte data value and DLC equal to 8, in both environments this message is sent with 10 bytes Date and DLC value equal to 9, 10, 12 (less than, equal, and bigger than data length). Same as previous attack and to have a clear result Break module is disconnected completely. A replay module is used instead.

```

date Tue Jun 12 14:31:21 pm 2012
base dec timestamps absolute
internal events logged
// version 7.2.0
Begin TriggerBlock Tue Jun 12 14:31:21 am 2012
0.000000 Start of measurement
0.004571 CAN 1 Status:chip status error active
0.004789 CAN 2 Status:chip status error active

1.791483 1 328 Rx d 9 238 123 0 179 144 0 0 0 0 0
2.243399 1 328 Rx d 10 238 123 0 179 144 0 0 0 0 0
2.506907 1 328 Rx d 9 238 123 0 51 176 0 0 0 6 4
2.961983 1 328 Rx d 8 238 123 0 179 144 0 0 0 11 0
3.767393 1 328 Rx d 12 238 123 0 131 156 0 0 0 0 0

End TriggerBlock

```

Figure 53. Sending a lengthened ABSData with all possible DLC value in a real network

- **Performing on simulated environment**

Using a Replay block, same attack as previous environment is performed on this network. The differences are the ID of message which is different in each environment and the normal data length. In the simulated environment a normal ABSData message has 6 bytes data. Consequently, various ABSData messages with 10 bytes Data and DLC value equal to 9, 10, and 12 are sent (less than, equal, and higher than data length).

```

date Don Jun 1 11:27:14 2012
base hex timestamps absolute
no internal events logged
40.000303 2 c9 Tx d 9 00 00 00 00 00 22 22 22 22 02
40.020500 2 c9 Tx d 6 00 00 00 00 00 22 22 22 22 02
40.040303 2 c9 Tx d 6 00 00 00 00 00 22 22 22 22 02
40.060500 2 c9 Tx d 10 00 00 00 00 00 22 22 22 22 02
40.080303 2 c9 Tx d 9 00 00 00 00 00 22 22 22 22 02

40.100303 2 c9 Tx d 12 00 00 00 00 00 22 22 22 22 02
40.120500 2 c9 Tx d 6 00 00 00 00 00 22 22 22 22 02
40.140303 2 c9 Tx d 10 00 00 00 00 00 22 22 22 22 02
40.160500 2 c9 Tx d 6 00 00 00 00 00 22 22 22 22 02
40.180303 2 c9 Tx d 12 00 00 00 00 00 22 22 22 22 02

```

Figure 54. Sending a lengthened ABSData with all possible DLC value in a simulated network

Result:

According to the result, the ABSData message that is shown on the log files is none of these messages. An ABSData message with DLC equal to 9 and Data value equal to 8 is the only one which transferred.

```

2.500918 1 328 Tx d 8 238 123 0 115 160 0 0 0
2.520778 1 328 Tx d 8 238 123 0 99 164 0 0 0
2.540770 1 328 Tx d 8 238 123 0 83 168 0 0 0
2.560913 1 328 Tx d 8 238 123 0 67 172 0 0 0
2.580933 1 328 Tx d 9 238 123 0 51 176 0 0 0
2.600791 1 328 Tx d 8 238 123 0 35 180 0 0 0
2.640916 1 328 Tx d 8 238 123 0 3 188 0 0 0
2.660907 1 328 Tx d 8 238 123 0 243 128 0 0 0
2.680864 1 328 Tx d 8 238 123 0 227 132 0 0 0
2.700587 1 328 Tx d 8 238 123 0 211 136 0 0 0

```

Figure 55. Part of a filtered traffic of sending a lengthened ABSData message in real network

41.920804	2	Ignition_Info	Tx	d 2 01 00	Length = 768000 BitCount = 67
41.940804	2	Ignition_Info	Tx	d 2 01 00	Length = 768000 BitCount = 67
41.947012	2	NM_Gateway_PowerTrain	Tx	d 4 1B 12 01 FF	Length = 984000 BitCount = 85
41.951380	2	EngineData	Tx	d 8 49 0C 3D 2D B8 0B 98 3A	Length = 1344000 BitCount = 115
41.952772	2	EngineDataIEEE	Tx	d 8 0E 8C 44 45 00 80 3B 45	Length = 1356000 BitCount = 116
41.953972	2	ABSdata	Tx	d 6 86 00 00 00 1B 2E	Length = 1164000 BitCount = 100
41.955400	2	ABSdata	Tx	d 9 00 00 00 00 00 22 22 22	Length = 1392000 BitCount = 119
41.956096	2	GearBoxInfo	Tx	d 1 05	Length = 660000 BitCount = 58
41.956804		EnvSetRadioChannelDSP := "0"			

Figure 56. Part of a filtered traffic of sending a lengthened ABSdata message in simulated network

• ID Field Evaluation

A malicious user may install an ECU to send infected messages on the CAN network. To do such an attack it is important to know if it is possible to inject message with non-existing ID.

Attack Scenario: Inject message with non-existing ID.

Technical implementation:

A message with non-existing ID is defined. Again a Replay block is used to inject these new messages.

```

date Don Jun 1 11:27:14 2012
base hex timestamps absolute
no internal events logged
55.300303 2 222 Tx d 4 00 00 00 02
55.320500 2 222 Tx d 4 00 00 00 02
55.340303 2 222 Tx d 4 00 00 00 02
55.360500 2 222 Tx d 4 00 00 00 02
55.380303 2 222 Tx d 4 00 00 00 02

55.400303 2 222 Tx d 4 00 00 00 02
55.420500 2 222 Tx d 4 00 00 00 02
55.440303 2 222 Tx d 4 00 00 00 02
55.460500 2 222 Tx d 4 00 00 00 02
55.480303 2 222 Tx d 4 00 00 00 02

55.500303 2 222 Tx d 4 00 00 00 02
55.520500 2 222 Tx d 4 00 00 00 02

```

Figure 57. Injecting message with non-existing ID

Car behavior after attack

This attack performed once in a simulated environment and once in a real environment. Both networks let injecting such a message. Consequently, an attacker can install an infected ECU in a CAN network and try to send malicious messages. Tracing log files reveals existing such an ECU though.

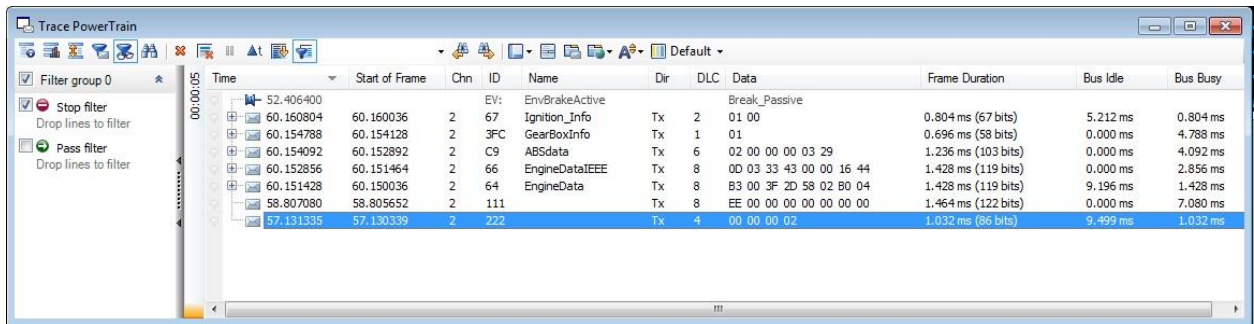


Figure 58. injecting message with non-existing ID

- **Flipping the bit-rate**

CAN messages of a given ECU are broadcasted by a universal unique ID and a constant frequency. This attack is intended to affect the functionality of an ECU by injecting too much message more than what is expected. It is thought that it may disorder the related ECU by making too much messages which leads the receiver to make too much error messages or ACK and therefore forces the sending ECU to stop sending message. By this a DOS attack is expected.

Attack Scenario: injecting too much ABSData message to the network when the speed exceeds 40 Km/h.

Technical implementation

A replay block is considered to inject a huge number of ABSData message. The picture below depicts part of such a file in t=50.

```

date Don Jun 1 11:27:14 2012
base hex timestamps absolute
no internal events logged
50.000303 2 c9 Tx d 6 00 00 00 00 00 00
50.000393 2 c9 Tx d 6 00 00 00 00 00 00
50.000403 2 c9 Tx d 6 00 00 00 00 00 00
50.000500 2 c9 Tx d 6 00 00 00 00 00 02
50.000503 2 c9 Tx d 6 00 00 00 00 00 00
50.000593 2 c9 Tx d 6 00 00 00 00 00 00
50.000603 2 c9 Tx d 6 00 00 00 00 00 00
50.000700 2 c9 Tx d 6 00 00 00 00 00 02
50.000703 2 c9 Tx d 6 00 00 00 00 00 00
50.000793 2 c9 Tx d 6 00 00 00 00 00 00
50.000803 2 c9 Tx d 6 00 00 00 00 00 00
50.000900 2 c9 Tx d 6 00 00 00 00 00 02
50.000973 2 c9 Tx d 6 00 00 00 00 00 00
50.001303 2 c9 Tx d 6 00 00 00 00 00 00
50.001393 2 c9 Tx d 6 00 00 00 00 00 00
50.001403 2 c9 Tx d 6 00 00 00 00 00 00
50.001500 2 c9 Tx d 6 00 00 00 00 00 02
50.001503 2 c9 Tx d 6 00 00 00 00 00 00
50.001593 2 c9 Tx d 6 00 00 00 00 00 00
50.001603 2 c9 Tx d 6 00 00 00 00 00 00
50.001700 2 c9 Tx d 6 00 00 00 00 00 02
50.001703 2 c9 Tx d 6 00 00 00 00 00 00
50.001793 2 c9 Tx d 6 00 00 00 00 00 00
50.001803 2 c9 Tx d 6 00 00 00 00 00 00
50.001900 2 c9 Tx d 6 00 00 00 00 00 02
50.001973 2 c9 Tx d 6 00 00 00 00 00 00
50.010303 2 c9 Tx d 6 00 00 00 00 00 00
50.010393 2 c9 Tx d 6 00 00 00 00 00 00

```

Figure 59. Injecting too much ABSData message to increase the bit-rate (it is part of the injected messages)

Car behavior after attack

This attack performed once in a simulated environment and once in a real environment. Both networks never let injecting such a huge number of data. They instead started to stop it by sending an alarm. In simulated environment the software sent below error message which shows Demo version's restrictions. To be able to run such an attack, the number of messages is lowered. Consequently the attack was performed which made disorder in functionality of car's speed indicator. There is also the sign of such malfunction in the car speed. Surprisingly there is no extra message in the log file after the attack. This can help a malicious user to inject too much malicious messages and therefore makes mitigation harder.

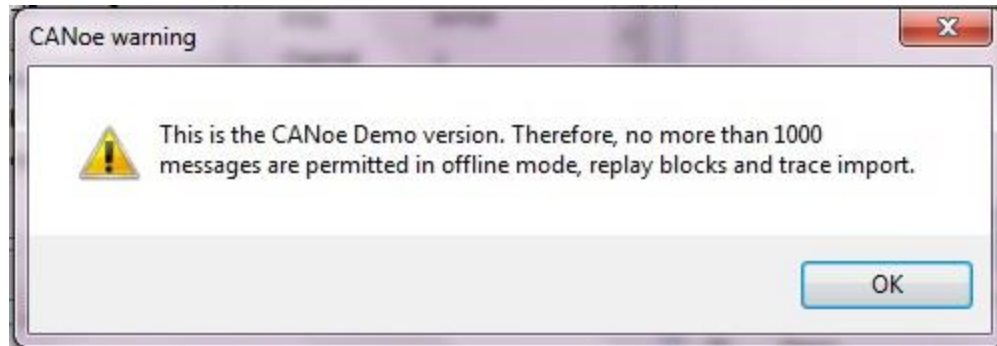


Figure 60. Injecting message with high bit-rate in simulated environment

On the other hand running such an attack in real environment made the voice alarm beeping which means the bit-rate is higher than what is expected.

7.4.3 Implementation-based flaws

Implementation phase sometimes involves with making some changes in theory. In fact, there are some new needs to make a match between reality and theory. Although till here CAN network is considered as a peer-to-peer network, in reality to reduce the power consumption a master node is used. This master node is called Central Electronic Module (CEM) and it determines ECUs' state. Different states such as Sleep, Working and etc are defined. A master node in CAN network acts exactly same as the human brain to coordinate all the ECUs to decides proper state of an ECU at the moment and also traveling between states. Consequently, CAN in practice acts as a Master Slave node.

Although Master – Slave networks provide the likelihood of being a single point of failure, it is deniable that it dramatically saves a considerable amount of power. In the trade-off between power consumption and security issues in practice vendors took power reduction which can open new doors to a malicious user.

8 Countermeasures

Up to the here many probable and possible flaws, vulnerabilities, and attacks in a CAN network have been investigated. Protection of automobile system against such potential hazardous problems is inevitable. In this last part of the thesis, countermeasures, the available ways of decreasing or preventing attacks are evaluated.

8.1 Intrusion Detection/Prevention Systems

Similar to the typical desktop computer networks, Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) are options to protect and prevent attacks in embedded automotive networks. They can be implemented as host based systems in ECUs to monitor the behavior of running code at a host or network based systems to analyse the ongoing data traffic on the network to detect visible signs of an attack [13]. Using an IDS could be helpful to generate warning in case when it is not possible to thwart it. Due to autonomous reaction of a vehicle, using an IPS needs extra care otherwise it can lead a safety issue.

Two types of IDSs, Specification-based detection and Anomaly-based detection are introduced. One possibility is to Place a detector in each to control ongoing traffic based on protocol stack information and the object directory of the CAN-protocol at each ECU. The alternative would be to implement an anomaly-based detection an IDS placed in each controller to monitor the traffic and listen to it. It monitors bit-rate and compare it with what is allowed [1].

8.2 Firewalls

There are different networks in a modern car which are connected by gateways. To have a secure communication between networks, messages must be checked before broadcasting on to the networks. For example, ECUs connected to LIN and MOST networks should be prevented from sending frames into high safety-relevant bus systems as CAN or FlexRay. Firewalls check the incoming traffic as well. The use of firewalls seems as the best choice to control messages. Firewall rules are based on the capabilities of ECUs. Vehicular units which are enabled to implement digital signatures or MACs, may have the rules of its firewall based on the authorizations given in the certificates of each controller. This makes only authorized units able to transmit valid frames into vehicular bus systems. On the other hand, if the ECUs do not have the capabilities to use digital signatures or MACs, the rules of the firewall can be established only on the authorizations of each subnet [3].

8.3 Honeypots

This technique is a security protection which monitors the intruder and analysis and collect data of the attacker behavior in a network. This data can be analysis later. It also provides the means of learning new attacks. The vital fact about honey pot is that to what extend it is closed to the reality. It should be implanted in a way that the intruder gets the feeling that it is walking in a real network and not in a simulated one. Moreover, specifically talking about vehicular networks,

for safety and security means various hardwares are needed [1]. Verendel et al. [1] suggest such an approach in a vehicle by attaching a simulated in-vehicle network to the wireless gateway in the car.

8.4 Encryption

To have a secure communication it is vital that the transferred information can be seen only by the right receiver without any modification in its way. Having combination of symmetric and asymmetric encryption not only provide required security but also has high performance. Symmetric encryption is fast enough to meet internal broadcasting speed while asymmetric provide key distribution mechanism. Wolf et al. [3] proposed a secure vehicular communication as depicted in below picture. There is a centralized super gateway processor which connects all the internal car networks as well as Bluetooth network together and has a memory to store secret keys and a list of trusted ECUs. A Trusted Computing Module application can provide such a memory. It is supposed that every successful verified ECU holds the symmetric bus group key K_i and its public and private keypair PK_j and SK_j and also the gateway's public key PKG . The gateway has the certificates and each bus internal group key K_i .

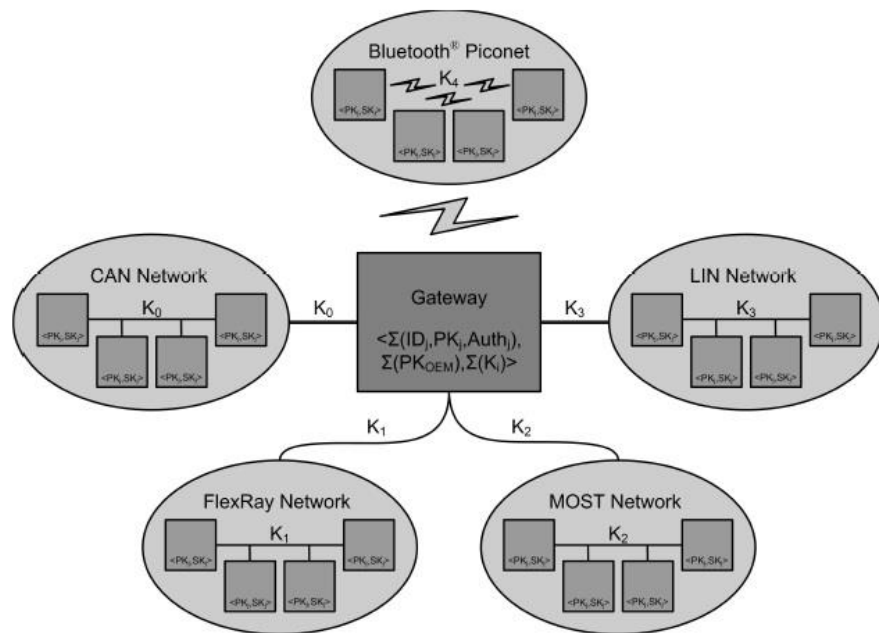


Figure 61. Secure vehicular communication [3]

Internal bus messages are encrypted by K_i so that only units which have a valid K_i can decrypt local broadcasted messages. As shown in below table, to provide message integrity and sender authentication every controller may include a digital signature SM . Message integrity is possible using an asymmetric message authentication code (MAC)

C1 = Encrypt(M;K_i)	Encrypt message M with symmetric key K _i
SM = Sign(C1; SK_j)	Sign C1 with secret key SK _j (optional)
C = C1 SM	Send C composed of C1 and SM (optional)

Figure 62. Secured Message Sending [3]

Table below depicts reception of an encrypted message C by an ECU or the gateway. Internal ECUs decrypt only the symmetric part C1 of C while gateways have to verify enclosed signature SM. The gateway encrypts the message again and forwards it to the intended subnet. If the sending unit verifies successfully and if it possesses appropriate authorizations, the gateway forwards the message encrypted again into the targeted subnet[3].

M = Decrypt(C1;Ki)	Decrypt C1 to message M with symmetric key Ki
Verify(SM; PKj)	Verify SM with public key PKj (gateway only)
Target € Authj	Forward M into target subnet if Authj allow (gateway only)

Figure 63. Secured Message Receiving [3]

8.5 Authenticity check

Since CAN-bus does not provide authenticity check, an intruder may use such a flaw. Recognition of sending ECU by its network characteristic would be really helpful to ban authenticity related attacks. Authentication of all parties in a CAN-bus can ensure that only valid controllers are communicating in a car and any suspicious or unauthorized message may immediately be discarded or processed separately. Various information provided by OSI model layers such as timing behavior, signal or attenuation quality could be used to classify allowed ECUs [13]. Moreover, every unit may be assigned a certificate containing ID, a public key, an authorization of the respective controller to authenticate itself. The gateway is needed to have a list of trusted public keys [3].

8.6 Forensics Support

The more IT devices get popular, the more IT security related attacks grow up. Although diagnostics only detect safety violations (unintended failures), it hardly detects security violations (intended attacks). Besides that, it would be time-consuming to find an infected or malicious module in a car since an attacker may have hidden it in a smart way. Consequently, there would be a need for logging of all possible safety and security related events such as flash operations, connecting to outside and etc. to protect this sensitive data, there would be some kind of black box which is configured to be privacy preserving for the driver. There, however, would be some downsides with this approach. Although required memory for such a technique will not be an issue, physical protection may make such a black box relatively expensive. Moreover, malicious code which once infiltrates such a system, it may affect all information in different ways- overwriting, dropping [10].

8.7 Security features related to the network architecture

All the aforementioned countermeasures can be used to detect or prevent malfunctions by means of a technique. Kleberger et al [1] have evaluated different types of architectural security features. Here is the summary of ongoing research so far. It is suggested to improve security features by enforcing authentication of units to the gateway by means of a certificate. In case of performing a successful authentication, the given unit will receive a shared symmetric key with other units.

The first architecture proposes using of OSI model features to secure CAN_BUS. Although the OSI model expects all five security features, confidentiality, integrity, authentication, non-repudiation, and access controls only the first 3 features are met in this model and the two remaining are not useful enough in this context. Another proposed model covers integrity by using of a MAC and data authentication in the CAN communication. It provides a 128-bit shared key between each communicating pairs. The MAC is calculated using four consecutive CAN-messages and the resulting MAC is divided into four 16-bit blocks and transmitted in the Cyclic Redundancy Code (CRC)-field of the next four CAN-messages. At least eight messages are required for a complete verification.

Another architecture proposes a communication between trusted parties which share same symmetric key for encrypting and decrypting. The trusted groups are defined by Access Control Lists (ACLs) which are signed by the automotive company. Each ECU has an ACL to define the trusted groups that the ECU belongs to. A Key Distribution Centre (KDC) is responsible for creating and distributing the symmetric keys for each group. It is located within the car. The symmetric keys will propagated within the trusted groups to communicate to an ECU. This ECU sends its ACL to the KDC. If KDC verified the signature on the ACL, the symmetric keys will be sent to those trusted groups defined by the ACL [1].

9. Results

As it was discussed in chapter 6 a penetration testing consists of 5 systematic steps- Planning and Preparation, Discovery, Attack, Reporting.

In this thesis we conducted a penetration test on CAN network. In each chapter we have investigated one of the steps. Chapter one covers Planning and Preparation step. Chapters two, three, four and five cover Discovery step and chapter seven covers Attack step. This chapter is dedicated to present the result of this test by discussing each step. By this we are going to form the last step, Reporting.

9.1 Planning and preparation:

In this step the scope of the test and its final goal as well as the target system was studied.

As it was previously discussed CAN network is divided into 5 smaller categories based on the communication speed and safety affect in time of a failure. [4] In order to narrow down the scope of this master study we have only focused on Powertrain and Comfort networks. Powertrain consists of the most critical ECUs. Any security breach in this network can be leaded to a catastrophic result on driver and passenger safety. Comfort on the other hand, has those ECUs with safety assistance related type. Any failure in these ECUs might not immediately affect the driver safety. Moreover, we have not considered all the ECUs in these two networks. We have only tried to simulate the most important ones. In reality depending on the car's model the available ECUs in such sub-networks can be slightly different. According to this fact we assumed that it shall not affect the result of the penetration testing.

In the beginning of this work we were not sure that we can get to perform our tests on any real car or even a boxcar. Therefore we started to simulate this network. The software that we used to simulate CAN network was CANoe. It is the most common and comprehensive software in car's industry which can be used for simulating of in-vehicle networks, analysis of the data traffic in each network and developing of entire ECU networks as well as individual ECUs. It can also be used for diagnosing. However, it was unfortunately not possible to get the hold of the complete version of CANoe so we had to use its demo version. Later when we performed all of the testing using the simulated network we could have the chance to re-perform part of it on a boxcar in Volvo Car Corporation (VCC). We could not find any boxcar in Volvo which has the exact same platform as our sub-networks. Anyway, it shall not be a problem since the communication bus is the same.

A grey level of penetration testing was intended in this master study. It means that tester shall have a reasonable level of knowledge about the system, its security's flaws and the capability of the simulating software.

9.2 Discovery:

Tester shall have a good understanding about the system. Having a higher level of knowledge regarding the target system will probably help to discover system's flaws and vulnerabilities. Therefore in this step the tester shall study the system deeply and then analysis the lessons to find out the vulnerabilities of the system. Consequently the following sub-sections were formed in this step.

Information gathering and analysis:

In order to do a gray level penetration testing we needed to discover the target system as much as it was possible. In chapter 2, in-vehicle networks were shortly discussed and compared. After that, in chapter 3, the most focus has been on CAN network since it was the target of this study. CAN has the most critical ECUs such as brake system and engine control which makes it an interesting point to conduct a stability test. It can lead to a series of catastrophic consequences.

The more knowledge that the tester has about the target system the more security breach could be eventuated. Therefore in chapter 3 we tried to get to know the bus communication, communication level compare to OSI model, the CAN's messages types, the structure of each message and a in-depth CAN protocol features.

Vulnerability Detection:

In essence, in-vehicle network supposed to be isolated but introducing FOTA and other functionality connected this network to outside of the car and therefore made it a potential point for security attacks. In this step existing and known vulnerabilities of such a bus was studied. Therefore, in Chapter 4, security properties and an in-vehicle common language were discussed. In chapter 5 internal and external attacks were studied. Depending on the attacker's access to the CAN network the possible attacks are classified into internal and external attacks categories. Any possible read, spoof, modify, DoS or even a combination of two or some of them can be considered as an internal attack. External attacks assume the attacker does not have a direct physical access to the vehicle. [16]

9.3 Attack

As it was discussed before, in this master study attacks were designed based on internal vulnerabilities. They were classified into three categories- security property (CIA) based flaws, CAN's frame based flaws, implementation based flaws. In the following we are going to discuss each group of attacks.

Attacks related to CIA

Each attack is designed based on lack of one of the security properties.

- 1- Read attack- designed based on lack of CAN's confidentiality. As CAN transfers plain text data on the communication bus a malicious user can read out information by listening to the communication bus or changing filters on any other ECU. Assuming the attacker has got access to the network we injected two lines CAPL code to Console ECU. Basically, Speed is transmitting from PowerTrain to be displayed on the Dashboard. We tried to read out speed information by Console ECU. Console belongs to the Comfort sub-network and receives PowerTrain through the gateway. Eventually due to lack of CAN's confidentiality the attack was successful.

- 2- Spoof Attack- designed base on lack of CAN's authentication. Since there is no such a security feature in CAN network it is possible to inject infected unauthorized messages. We considered two different scenarios to perform this attack. The first one triggers whenever the brake pedal is pushed. Then it sends speed messages with values far lower than what it is to fool the driver. The second one uses a replay block to flood speed messages with random values. The intention is infecting speed indicator. CAN bus does not check the authenticity of the sender therefore the malicious user can easily abuse it.
- 3- Replay Attack- Designed base on lack of checking freshness of CAN messages. Therefore it is possible to listen to the network traffic, capture arbitrary messages and reproduce them whenever it is intended. We listened to the network traffic, recoded headlight enable/disable messages and tried to enable them at car's startup using a replay block. CAN-bus never checks the freshness of the messages.
- 4- Drop Attack- designed based on lack of availability. We tried to perform DoS attack by preventing spreading of CAN messages all around the network. The best place to initiate such an attack is probably gateways. Gateways map data from the initiating network to the other networks. We tried to make a new signal in the receiver network and pass the value from the initiating network to the new signal. By this we prevented spreading the real value to other networks.
- 5- Modify Attack- designed based on lack of integrity of CAN messages. The intention is to deceive the driver by changing data. When the speed reached to a certain value the speed indicator shows half of the real speed.

Attacks related to CAN Frame:

The value of CAN's messages fields are either fixed or changing from message to message. Among them ID, DLC, Message type fields are not fixed. The probable flaws with these fields are the base of these types of attacks. On the other hand these types of attacks are performed on both simulated and real environments. The result in each environment will be discussed.

DLC field

We tried to inject shortened CAN messages than normal with different possible DLC values meaning less than, equal and higher than data length. Result showed only the one with data length equal to DLC showed up on both network's traffic.

CAN messages with lengthened data field than normal with different possible DLC values meaning less than, equal and higher than data length were injected to both networks. None of these messages were appeared on the network.

ID field

It examined the system to see if it is possible to send messages with non existing IDs. Therefore such a message was defined and injected to the both environments. The message was appeared on both networks. This lets the attacker to install a new infected ECU in the system for potential malicious activities.

Flipping the bit rate

Each ECU in CAN network sends messages with unique ID in a constant frequency. We tried to inject too many messages to see if it is possible to disorder the sending or receiving ECU. A replay blocked was used to flood brake messages on both environments. We noticed that both environments did not let injecting this number of same messages. Both stopped it by send an alarm message. Even though it was not possible to transmit all the messages we could disorder the network functionality. Also there was no extra message in the log file after this attack. It helps attacker since there is no track of malicious activity left on the bus.

Attacks related to Implementation:

Even though CAN-bus is considered as a peer-to-peer network in theory, in reality usually there is a master node. It is used to reduce the power consumption by changing ECUs states. It makes such a network more or less the same as a master-slave network. Master-slaves networks provide the single point of failure.

10. Conclusion

While bringing high performance and efficiency to the automotive world, the computerization of vehicles has opened up for a new range of security risks. The security evaluation we have performed makes it clear that automotive security must be considered carefully. The CAN bus protocol, which is widely used in intra-vehicle communication, has a series of weaknesses that makes it vulnerable to different kinds of attacks, such as: read, spoof, replay, modify and DoS attacks.

This master thesis is aimed to evaluate security aspects of the in-vehicular networks. Therefore a theory part which studies this area and known vulnerabilities is performed. Accordingly, a practical part to ensure known flaws in reality is done.

In the theoretical part of this master thesis, in-vehicular networks are described and shortly studied. The focus of this master thesis is on CAN network therefore CAN structure, features, specifications, and communication pattern are studied in more detail.

On the other hand, security considerations for a network were also studied. Since CAN is located in the lower layers of the OSI model, link layer and physical layer, only required security properties of these layers were discussed. Therefore, CIA, data authentication, data freshness and data non-repudiation were studied. Using a common taxonomy, information and events are classified to form a common language for computer and network security terms. This taxonomy specified for vehicle. So that an attack scenario involves with four steps- tools, vulnerabilities, event and unauthorized results. Obviously, only attacks related to the vehicular networks were discussed. This part provided a basis knowledge for the tests which are done in the practical part.

The practical parts tried to firstly simulate such a network and then attack it. To attack the network, a penetration testing was done. Attacks are classified into three classifications. A test case is defined for each attack. Technical implementation and car behavior for each attack is discussed as well. Even though a penetration testing does not fully monitor the security level of the network we could breach the security of the CAN network.

Lastly, possible countermeasures and protection methods are studied in the last chapter.

11. Future Work

The tests developed in this study and the collaboration started with VOLVO represents a starting point for future works.

This study can be further developed by incorporating the following ideas:

- Continue the collaboration with VOLVO and implement more tests on a CANoe Complete Version.
- Investigate the limitations of implementing penetrations tests on CANoe Demo Version and compare them with the complete version results.
- Implement the tests on a real car and extend them on other ECUs not targeted in the present report due to the CANoe Demo Version.
- Evaluate the results provided by a real environment, identify problems and propose appropriate countermeasures.
- Evaluate other in-vehicle networks by conducting similar penetration testing.

References

- [1] Pierre Kleberger, Tomas Olovsson, and Erland Jonsson, "Security aspects of the In-Vehicle Network in the Connected Car", IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, June 5-9, 2011.
- [2] Nilsson, Dennis K. (2009) How to secure connected car. Göteborg: Chalmer University press
- [3] M. Wolf, A. Weimerskirch, and C. Paar, "Security in Automotive Bus Systems," in Workshop on Embedded IT-Security in Cars, Bochum, Germany, November 2004.
- [4] National Instruments, "Controller Area Network (CAN) Overview", <http://www.ni.com/white-paper/2732/en> (April 2012).
- [5] Renesas Electronics Corporation, "Introduction to CAN", <http://www.renesas.com> , April 2010.
- [6] Tobias Hoppe, Stefan Kiltz, Jana Dittmann: "Automotive IT-Security as a challenge: Basic Attacks from the Black box perspective on the Example of Privacy Threats", 2009.
- [7] Daniel Mannisto, Mark Dawson: "An Overview of Controller Area Network (CAN) Technology", <http://www.parallax.com/dl/docs/prod/comm/cantechovew.pdf> (April 2012)
- [8] S. Savage, "Experimental Security Analysis of a Modern Automobile", In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 447-462, 16-19 May 2010.
- [9] Vector Informatik, <http://www.vector.com/> (May 2012)
- [10] T. Hoppe, S. Kiltz, and J. Dittmann. Security threats to automotive CAN networks practical examples and selected short-term countermeasures. In Proc. of the Int'l Conf. on Computer Safety, Reliability and Security (SAFECOMP), pages 234–248, 2008.
- [11] Keith Pazul: "Controller Area Network (CAN) Basics", <http://ww1.microchip.com/downloads/en/AppNotes/00713a.pdf/> , Microchip Technology Inc. Preliminary DS00713A-page 1 AN713 (1999).
- [12] J. D. Howard and T. A. Longstaff, "A Common Language for Computer Security Incidents," no. Sandia Report: SAND98-8667, 1998.
- [13] Tobias Hoppe, Jana Dittmann: "Sniffing/Replay Attacks on CAN Buses: A Simulated Attack on the Electric Window Lift Classified using an Adapted CERT Taxonomy", <http://omen.cs.uni-magdeburg.de/automotiv/cms/upload/ACM.pdf> (April 2012)

- [14] Dennis K. Nilsson, Phu Phung, and Ulf E. Larson: "Vehicle ECU Classification Based on safety-Security Characteristics", In Proceedings of the 13th International Conference on Road Transport and Information Control (RTIC), 2008.
- [15] Dennis K. Nilsson, Ulf E. Larson: "Simulated attacks on CAN buses: vehicle virus", AsiaCNS '08 Proceedings of the Fifth IASTED International Conference on Communication Systems and Networks, Pages 66-72
- [16] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno: "Comprehensive Experimental Analyses of Automotive Attack Surfaces", USENIX Security Symposium, August 2011.
- [17] Karen Scarfone, Murugiah Souppaya, Amanda Cody, Angela Orebaugh: " Technical Guide to Information Security Testing and Assessment", National Institute of Standards and Technology, September 2008.
- [18] Dave Burrows: "Penetration 101 - Introduction to becoming a Penetration Tester", GIAC Security Essentials Certification (GSEC) Version 1.3, SANS Institute 2002.
- [19] SANS Institute InfoSec Reading Room: "Penetration Testing - Is it right for you?", SANS Institute 2002.
- [20] SANS Institute InfoSec Reading Room: "Conducting a Penetration Test on an Organization", SANS Institute 2002.
- [21] SANS Institute InfoSec Reading Room: "Guidelines for Developing Penetration Rules of Behavior", SANS Institute 2001.