ISSN 2347 - 3983
Volume 8. No. 10, October 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter658102020.pdf
https://doi.org/10.30534/ijeter/2020/658102020

CAN Bus Protocol Permeating and Exploiting using ICSIM and CANSniffer

Chaithanya B N[1], Dr Dayanand Lal[2], Geetha K[3], Nida Kousar[4],MrsManjula R[5]
[1]Assistant Professor, Department of CSE, GITAM School of Technology, Bengaluru, India, cnagaraj@gitam.edu
[2]Assistant Professor, Department of CSE, GITAM School of Technology, Bengaluru, India, dnarayan@gitam.edu
[3]Assistant Professor, Department of CSE, GITAM School of Technology, Bengaluru, India, gkallesh@gitam.edu
[4]Assistant Professor, Department of CSE, GITAM School of Technology, Bengaluru, India, knida@gitam.edu
[5]Assistant Professor, Department of CSE, GITAM School of Technology, Bengaluru, India, mrudresh@gitam.edu

ABSTRACT
With the growing prevalence of technologies for internet and drivers and the rise of semi - independent and auto-driving vehicles on roads worldwide, safety for intelligent cars is a timely problem that should be addressed in the modern world. CAN (Control Area Network) the predominant intra-vehicle network for electronic control units (ECU's), which regulate one or more in-vehicle devices, enables time sensitivity. The issue of safety of intelligent cars is a priority for public and personal security, as many researchers have reported on attacks against autonomous cars and on a very small amount of incident involving autonomous cars and the preservation of knowledge of such 'datacenters on wheels' is of more importance than ever. Throughout this point furthermore, the implementation of cyber security research for car hacking was huge learning curve. This paper explains the car-hacking module with the complete installation and implementation of all open-source software's. This research highlights the usage of a mixture of open-source software and a common CAN to USB (Universal Serial Bus) cable or wireless adapter, to check a vehicle for flaws causing repetitive attacks. Results will effectively demonstrate the ICSim screen, the panel controller and the sniffer page, and how a single CAN bus communicates through sensors, actuators and ECUs.

Key words: CAN, ECU, Autonomous car, Hacking, Security.

.

1. INTRODUCTION

Progress in automation has focused on a major portion of connectivity development in recent years, bringing multiple kinds of digital controls to current automobiles. Modern cars have many network forms like CAN, FlexRay, LIN (Local Interconnect Network) and MOST (Media Oriented Systems Transport). [1] Since CAN was launched in 1986, a car was already an autonomous device, so it had no external link. All devices are managed by an ECU. It facilitates connectivity between ECUs and the various internal networks in operation, so certain ECUs are linked to external interfaces like other cars, local networks and Internet as well. In addition, new modern cars with 20 to 100 ECUs [2] are capable of organizing, managing and tracking internal vehicle loads. Each subsystem may share details with the nearby components through engine systems to the brake system and essentially to the wireless system. As a consequence of this contact, an internal vehicle network was eventually created. The CAN is the best known auto communication device standard for the internal vehicle network in a standard fuel-based automobile. It enables security-critical ECUs to adequately disperse information across many gateways through CAN packets between them and other linked buses (i.e. FlexRay, MOST, LIN).

Additionally, embedded connections in modern vehicles may now enable the outside world to connect both by wireless (for example WiFi, Bluetooth, etc.) and wired (USB) interfaces. With potential vehicle-to-infrastructure (V2I) and Vehicle-to-Vehicle (V2V) communications [3], this phenomenon will continue to increase within the automotive sector. However, due to the growing amount of communication exchanges inside the CAN bus network and busses communicating with the outside world, it creates a variety of security risks battling their way into the framework, such CAN bus flaws might not only damage the driver's safety, but would also affect other vehicles.

As shown in the fig 1, the connected car indicates that the cars are still associated to an external network. The linked automotive system consists of foregoing components [4]:

(i) The car in which ECUs have been mounted and an in-vehicle network has been designed.

(ii) The portal for delivering different vehicle services.

(iii) The connectivity channel for connecting vehicle to the portal.

In such a scenario, vehicles are equipped with several ECUs which are linked to an in-vehicle network via a CAN BUS system. Another part linked to the CAN bus is the On Board Diagnostics (OBD) connection. This plug offers accessibility to the diagnostic functions required to monitor or troubleshoot vehicles. This link is established by a wireless networking system like a telematics ECU. The portal is subdivided into web-based services and applications for smartphones. Vehicles are vulnerable to extreme threats if linked car ecosystems are built without addressing in-vehicle network vulnerabilities; a protection framework for this issue is then suggested.

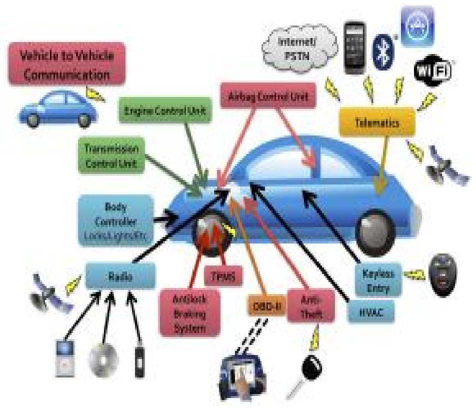


Figure 1: Compatibility in modern car

Control Area Network

The CAN bus in a vehicle establishes communication with the automotive sensors and the various ECUs. There could be more than 70 ECU or more available for new vehicles, motor fuel, airbags, brake anti-lock system, lights, multimedia and much more. The international electronics and engineering company Bosch created CAN in 1991 as a CAN 2.0 message-based norm, in reality. [5]

It can be stated that all vehicle control systems will not operate over the CAN bus. There are a range of other structures in current cars, along with multiples bus architectures, namely LIN, Flex Ray and MOST. Such network communications may include Ethernet, GSM / LTE, GPS navigation, Satellite radio and vendor-specific services such as OnStar. [6].The benefits of CAN are Low-Cost, Lightweight Network, Broadcast Communication, Priority and Error Capabilities.

Although CAN provides many techniques for error detection, data integrity and data accuracy, it doesn't provide any form of safe communication all over the network. Computerization of cars has provided many benefits, such as driver convenience, vehicle reliability and efficiency. Amid these advantages, increasing reliance on these apps has expanded possible attack thresholds and also revealed several vulnerabilities. [7] Until now, most vendors have followed the 'hidden protection' strategy, which ensures the execution is held secret so the no one can completely comprehend and potentially exploit it. Moreover, it is no longer valid in today's automobile environment, because certain flaws can be well exposed to attackers. For eg. Non-authentication, Network optimization exclusion, Data encryption deficit, Susceptible to denial of service attack.

CAN data frame structure

A message dependent protocol is Controller Area Network, that is also called a CAN bus. This has been designed to permit the communication in the entire vehicle-system via a single / double-wire bus with a variety of electrical components, e.g., micro controllers, electronic controls (ECWs), sensors, devices and actuators. CAN operates by transmitting the packets in general, meaning that any nodes connected to the CAN bus will accept all transmissions of the packets. In addition, a CAN packet frame is identified as a framework and holds a CAN series (bytes) of data on the network. From each CAN system transferred, the arbitration identifier (ID) field shows priority packets. The lesser the value of the ID bit denotes the packet's highest priority. Such protocol will prevent collisions throughout the traffic of the CAN bus.

CAN data frame consist of 4 kinds: data frame, error frame, frame overload, and remote frame [8]. Each frame's generic structure contains these fields: identifier for arbitration, data, acknowledgement, and a few still.ID field of the CAN packet, bit size is 11 bits and 29 bits if extended. The data field of a CAN includes between 0 and 64 bits. The semantic specifics of CAN ID and data are unique and are held very secure by car manufacturers. For regular and extended versions of the arbitration code, the simple CAN data structure will be shown in figure 2. The following table 1 defines each area in the CAN data framework:

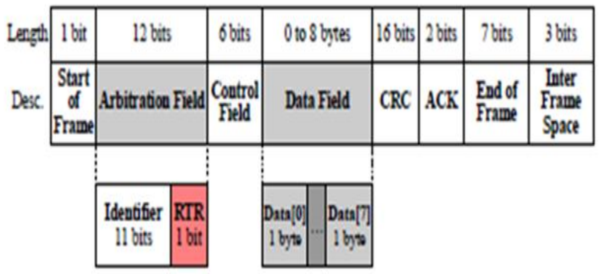


Figure 2: CAN data frame work

Table 1:Can Data Frame

Start of frame (SOF).	It determines the start of a dominant-bit CAN message and informs each nodes of a starting of CAN message transmission
Arbitration	This comprises of 11 bits as an identifier and RTR (Remote Transmission Request) as one bit.Throughout the arbitration procedure the identifier is often used as a primary concern and the RTR is defined as per the CAN frame.
Control	It's also defined as test field; it provides the recipient with details to validate that all expected packets are effectively handled.
Data.	This data area includes specific knowledge about behavior conducted by CAN nodes. It may range between 0 and 8 bytes.
Cyclic Redundancy Code(CRC)	It's regarded as cyclic redundancy code or protection zone, a 15-bit fault identification system that tests the integrity of packets
Acknowledge (ACK)	Often recognized as region of proof.When an error is found during the transmission phase, the receiver must automatically alert the sender for the data packets to be transmitted again. This sector guarantees accurate reception of CAN packets by the recipient nodes.
End of frame (EOF)	This field displays the end of the CAN frame by flag of a recessive portion.

A CAN data frame has been the only system used for the information transmission of CAN packets.Whenever the RTR (remote request) bit flag is dominant, it becomes a CAN data framework.Once the RTR (remote request) bit flag is prominent, it becomes a CAN data framework.

2. LITERATURE SURVEY

Big efforts to find possible faults in the automotive communication network, in particular for CAN bus protocol was initiated by many researchers. They found that a number of vehicle security-critical components can be changed by direct access, nonphysical access and connect directly to a vehicle for both shorter and longer distances. It is discussed below

Sen Nie et. al, [9] first noticed browser exploit, therefore they thought a contactless solution is required. The Wi-Fi SSID, Tesla Operation, is installed in every Tesla car, and the password is plain text stored in the QtCarNetManager. They considered, however, that it could not be linked automatically in normal mode. At the moment, Tesla Visitor arrived to their sights, this is a Wi-Fi hotspot given by Tesla Body Shop and Superchargers. The information on this SSID is kept in a range of customers' cars in order to auto link in the future. If we spoof this Wi-Fi hotspot and divert QtCarBrowser traffic with our own domain, secretly hacking Tesla cars could be possible. Including Wi-Fi tricks, as we assume in cellular mode the phishing and app mislabeling may even contribute to remotely activating flaws in our software if we create sufficiently custom domains. Since it's focused on a browser-borne assault, we might assume that remotely distributing exploit despite physical access is confined to imagination.

S. Checkoway et. al has proposed a system where [10], the real assault is against the monitoring system being inserted into the port like OBD, CD, USB. The so-called transmission system, connected to the ODB port, operated from a laptop remotely via Wi-Fi, has been jeopardized: the Communication API flaws have allowed shell code injection into the system from some other computer on same Wi-Fi network. Afterwards, the system passes through the network every time a new car is linked to the network with malicious packets. Worse still, the infected system may in effect also target other devices with the same WiFi network that are similar. Two flaws have been found in the CD player being studied. Initially, inserting a CD that contains a file with a certain name made the player think that it is a firmware upgrade and thus installs new and malicious software. In USB port many scenarios that can be invented. The former case is possible, where a compromised USB key file is accessed by the car media player. A further choice is to attach a damaged device (for example a mobile or an MP3 player), which are then used to target the ECU with which the device is linked.

N. Courtois et.al has discussed that in most cars today activate their doors or alarms remotely activated. Although some authentication is provided to these instructions that are sent through the air, it can be broken or bypassed. For instance, recorded assaults could be identified at [11] towards KeeLoq, a block cipher used by many manufacturers. In addition, Passive keyless entry and start systems (PKES) enable the users, though holding their keys in the pockets, to open and start cars. A group conducted relay assaults on ten various versions of the PKES systems. This allowed them to activate an antenna nearby the key holders (in 8 m) and the other in front of the target car, and start its engine then when the keys are 50 m from vehicle. The engine was then unlocked.

C. Miller et. al has proposed a system to [12] display some manufactured messages as CAN packets that were sent from a distance by possible entry sources like the TPMS. As a response, some of the essential components of the car were disabled, including the braking mechanism. In addition, when the driver was driving the steering wheel rotated 180 °.Across these attack thresholds in the car system, security experts have eventually found that almost all new vehicle networks have been engineered with protection but with no safety in mind. Therefore it is important that a comprehensive approach to the protection of the CAN bus is proposed in a vehicle network to ensure long-term safety.

3. METHODOLOGY

There's nothing now, as you drive a vehicle, that isn't controlled by a computer. But at the root of it all is the CAN, a car's central nervous system that is liable for intra vehicular interaction For CAN-Bus service we can use a Craig Smith ICSim kit. ICSim contains speed meter screen, door lock signs, turn signals and control panel. The control panel helps the consumer to communicate with the vehicle simulated network, add speed, brakes, monitor door locks, and turn signals. Steps for exploitation are shown in figure 3

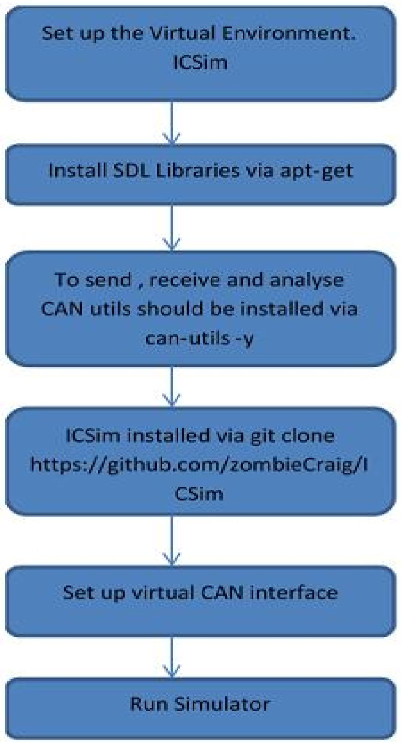
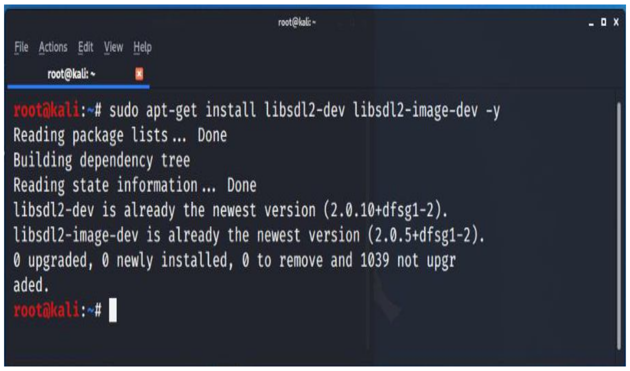

Set up the Virtual Environment. ICSim
Install SDL Libraries via apt-get
To send , receive and analyse CAN utils should be installed via can-utils -y
ICSim installed via git clone https://github.com/zombieCraig/ICSim
Set up virtual CAN interface
Run Simulator

Figure 3: Exploitation steps

- **Set up Virtual Environment**

Car Hacking is done by using Instrumentation cluster simulator. ICSim makes it very easy to configure and cost-effective to learn how to use a CAN-Bus. Simulator Instrument Cluster needs SDL libraries. SDL is a multiplatform computer graphics and audio software library. This is needed because ISCim draws and imitates a virtual dashboard. It will be mounted using apt-get as in figure 4.

Sudo apt-get install libsdl2-dev libsdl2-image-dev –y



```
root@kali:~# sudo apt-get install libsdl2-dev libsdl2-image-dev -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
libsdl2-dev is already the newest version (2.0.10+dfsg1-2).
libsdl2-image-dev is already the newest version (2.0.5+dfsg1-2).
0 upgraded, 0 newly installed, 0 to remove and 1039 not upgr
aded.
root@kali:~#
```

Figure 4: Installing ICSim

Installing CAN Utils

CAN utilities are required for sending, receiving and evaluating packets of CAN. CAN-Utils is indeed a Linux related series of tools that allows Linux to connect with the

CAN device on the car. The canutils are made up of 5 big tools which we use quite often are shown in figure 5.

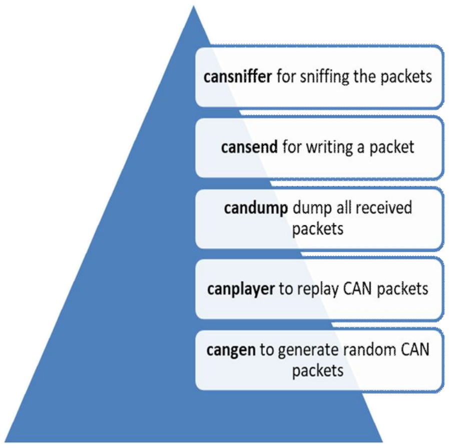
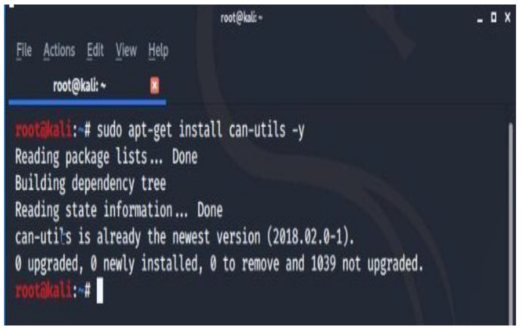

cansniffer for sniffing the packets
cansend for writing a packet
candump dump all received packets
canplayer to replay CAN packets
cangen to generate random CAN packets

Figure 5: CAN Utils.

Command used for installing canutil is shown in figure 6.
sudo apt-get install can-utils –y



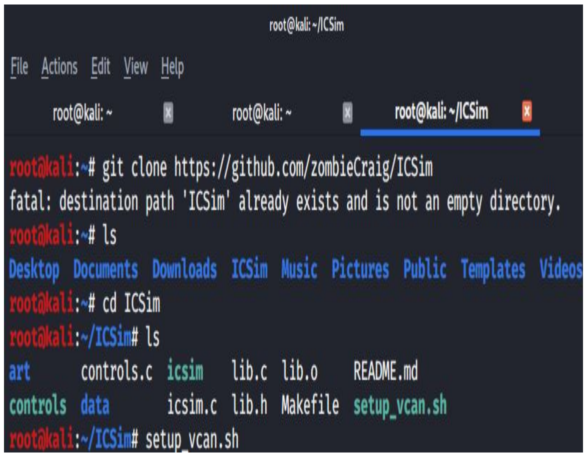
```
root@kali:~# sudo apt-get install can-utils -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
can-utils is already the newest version (2018.02.0-1).
0 upgraded, 0 newly installed, 0 to remove and 1039 not upgraded.
root@kali:~#
```

Figure 6: Installation of CAN Utils

Next step is to download and update the ICSim package as in figure 7 after loading LibSDL and the Utilities dependencies. It is done using the Git Command

git clone https://github.com/zombieCraig/ICSim.git

Git can copy the ICSim project files to a repository in your home directory called ICSim. Shift to the folder of ICSim, and describe the contents using cd ICSim/ and ls command respectively. Many files are seen within the ICSim folder, including two executable files named Controls and icsim, as seen below.

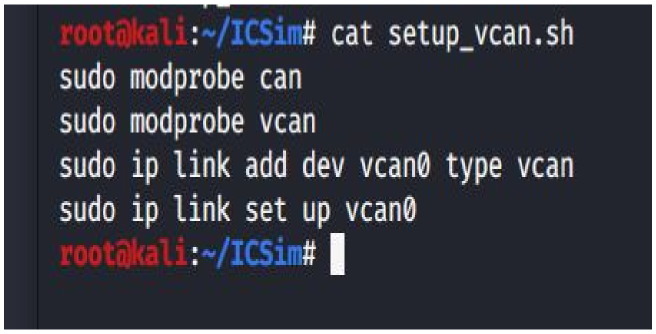


```
root@kali:~# git clone https://github.com/zombieCraig/ICSim
fatal: destination path 'ICSim' already exists and is not an empty directory.
root@kali:~# ls
Desktop  Documents  Downloads  ICSim  Music  Pictures  Public  Templates  Videos
root@kali:~# cd ICSim
root@kali:~/ICSim# ls
art       controls.c  icsim    lib.c  lib.o     README.md
controls  data        icsim.c  lib.h  Makefile  setup_vcan.sh
root@kali:~/ICSim# setup_vcan.sh
```

Figure 7: Updating ICSim packages.

- **Simulated CAN Network Development**

There will be a shell script named setup_vcan.sh when you traverse within the ICSim directory.



```
root@kali:~/ICSim# cat setup_vcan.sh
sudo modprobe can
sudo modprobe vcan
sudo ip link add dev vcan0 type vcan
sudo ip link set up vcan0
root@kali:~/ICSim#
```

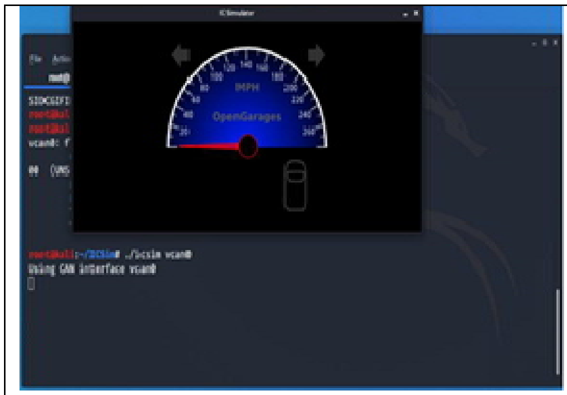
The modprobe function loads kernel modules, such as the CAN and vCAN network components from the CAN library and the first couple lines of the script load both modules so that they can interact with our car hacking simulator by means of CAN protocols via a virtual computer (vCAN) network. The two last lines establish the vCAN-type network system vcan0 and allow the connection. Run shell script by following command.

sh setup_vcan.sh

ifconfig is used to verify that the network vcan0 connection is active.

- **Running the ICSim Software**

First console
Open a simulated instrument cluster programme, icsim, we built ~/ICSim /icsim vcan0 on the vcan0 virtual network interface

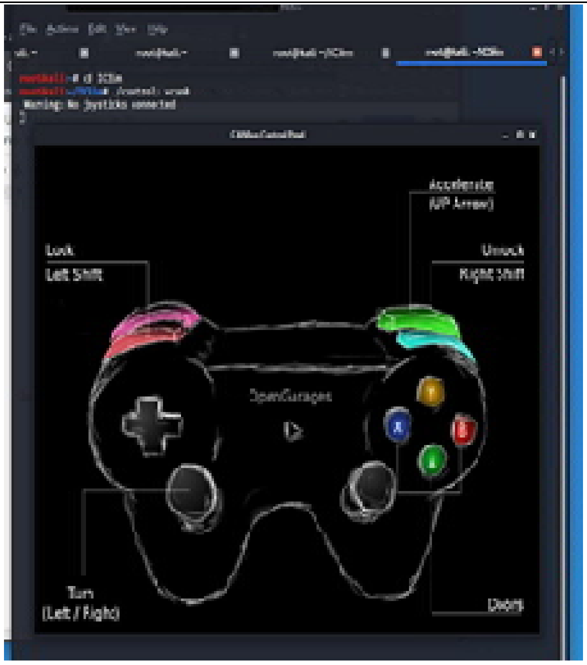


```
root@kali:~/ICSim# ./icsim vcan0
Using CAN interface vcan0
```

The controller for the dashboard

Second console

open the controls app
: ~/ICSim/controls vcan0

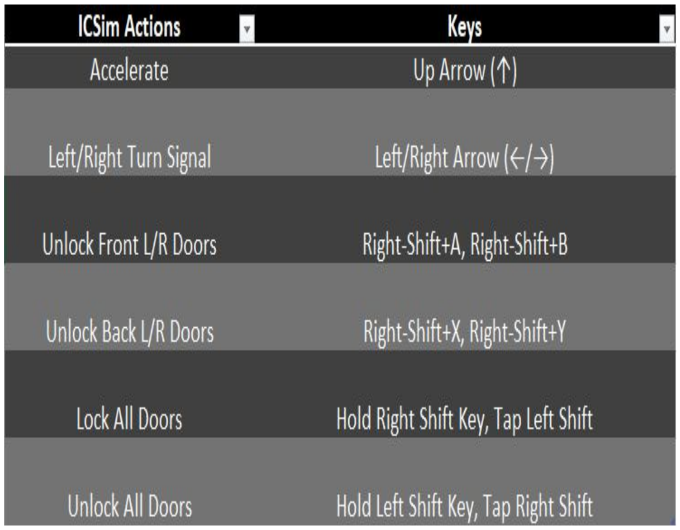


```
root@kali:~# cd ICSim
root@kali:~/ICSim# ./controls vcan0
Warning: No joysticks connected
```

ICSim's fundamental specification consists of at least two elements, an ICSim Instrument Cluster Simulator software file which simulates the car dashboard instrument panel, and controls which can be used to provide virtual vehicle user control (including the rpm, driving, door locks and turning signals). The first time for start-up auto hackers is to access packets in this latest automated CAN network through a third console window that functions as a network sniffer. Open three consoles

4. RESULTS

The cursor controls displayed in Table 2 actually operate when the program is chosen for the CAN Bus Control Panel. If the interface does not appear to react, tap on the CAN Bus Control Panel display of your mouse and then use the keyboard given underneath to display the IC Simulator Dashboard modifications: Change the ICSim Dashboard with the following key combinations.

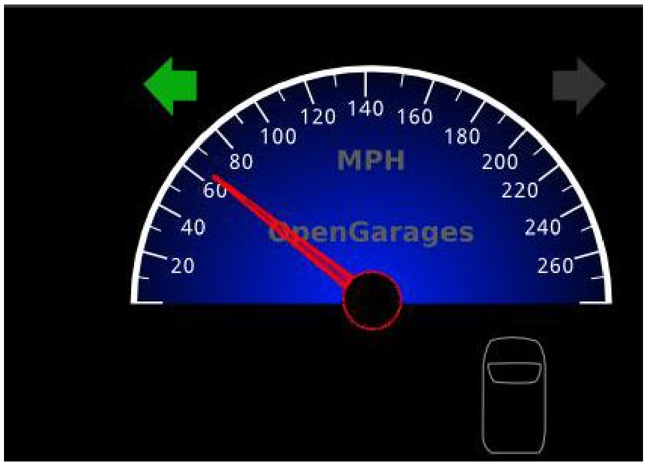


ICSim Actions	Keys
Accelerate	Up Arrow (↑)
Left/Right Turn Signal	Left/Right Arrow (←/→)
Unlock Front L/R Doors	Right-Shift+A, Right-Shift+B
Unlock Back L/R Doors	Right-Shift+X, Right-Shift+Y
Lock All Doors	Hold Right Shift Key, Tap Left Shift
Unlock All Doors	Hold Left Shift Key, Tap Right Shift

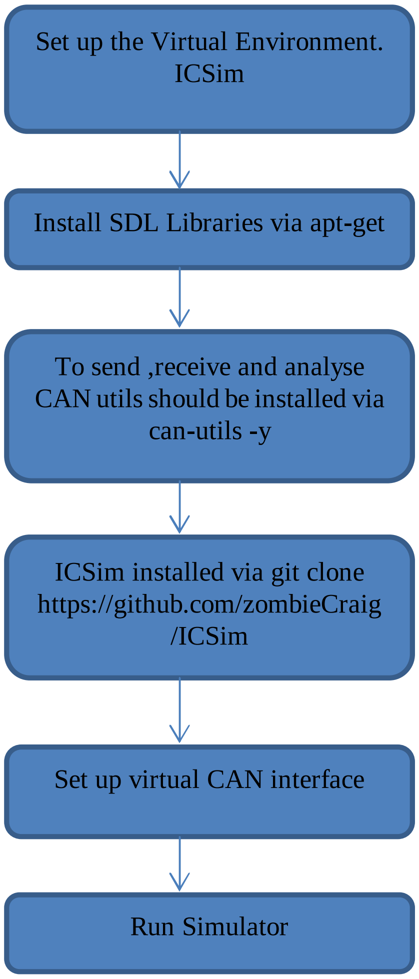
Table 2: Keyboard controls for CAN Bus control App

To show the simulated car take-offs, signal to right and left turns, open and release doors and so forth, select a button in the CANBus Control Panel and click several mouse buttons mentioned in Table 1.

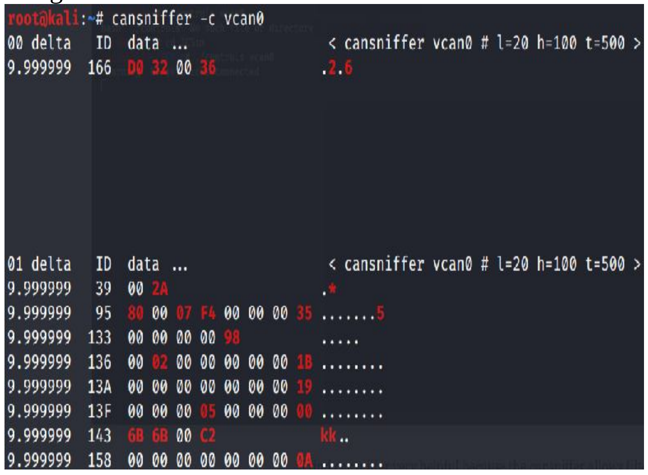
You will see this once the up arrow and left arrow buttons are clicked.



Sniffing of the CAN frames produced by ICSim:


Set up the Virtual Environment. ICSim
Install SDL Libraries via apt-get
To send ,receive and analyse CAN utils should be installed via can-utils -y
ICSim installed via git clone https://github.com/zombieCraig/ICSim
Set up virtual CAN interface
Run Simulator

Let us eventually see how the communication packets the control panel software sends by can sniffing to the dashboard simulator, the CAN function sniffer used in the previous programme CAN. We can use CANSniffer, a can-utils service, to sniff packets. Open the console for the third and start the vcanCANSniffer. The alternate c is used to signify the change of bytes in the frame. The data traffic is a straightforward tcpdump or a list of wireshark packets as seen in figure 8.



```
root@kali:~# cansniffer -c vcan0
00 delta   ID  data ...                  < cansniffer vcan0 # l=20 h=100 t=500 >
9.999999  166  D0 32 00 36               .2.6



01 delta   ID  data ...                  < cansniffer vcan0 # l=20 h=100 t=500 >
9.999999   39  00 2A                     .*
9.999999   95  80 00 07 F4 00 00 00 35 .......5
9.999999  133  00 00 00 00 98            .....
9.999999  136  00 02 00 00 00 00 00 1B ........
9.999999  13A  00 00 00 00 00 00 00 19 ........
9.999999  13F  00 00 00 05 00 00 00 00 ........
9.999999  143  6B 6B 00 C2               kk..
9.999999  158  00 00 00 00 00 00 00 0A ........
```

Figure 8: Can sniffer provides a network view of CAN simulated packets

You can display how a single CAN bus communicates through sensors, actuators and ECUs via this open triplicate display, the ICSim computer, panel controller and the sniffer tab. Take 90 MPH to get the engine, set the governed high speed, lock the doors and open the doors when you are up and click left and right. The dashboard is intended to show if these combinations are safe to use in a real car.

5. CONCLUSION

The paper provides precise instructions on how to established open source car hacking apps free of charge for educational or research purposes. We have also described how to link a low-cost wireless connection to current automobiles using the OBD-II on-board diagnostic port.We are effectively able to demonstrate the ICSim screen, the panel controller and the sniffer page, you can see how a single CAN bus communicates through sensors, actuators and ECUs. Take 90 MPH to drive, set up the regulated top speed, lock and unlock the doors, and press the left and right signals while you move up. Students, as well as scholars, can use these tools in the assessment and adaptation of the source code and technologies and basic specifications of various car hacking scenarios using the open source ICsim package OpenGarages.org and ScanTool 's free ScanTool.net..

REFERENCES

[1] Controller Area Network and Its Reduced Wiring Technology
[2] C. Miller, C. Valasek, in Black Hat USA, 2014. A survey of remote automotive attack surfaces (2014), p. 94
[3] S. Al-Sultan, M.M. Al-Doori, A.H. Al-Bayatti, H. Zedan, A comprehensive survey on vehicular ad hoc network. J. Netw. Comput. Appl. 37, 380–392 (2014)
[4] Enhanced Android App-Repackaging Attack on In-Vehicle Network.
[5] Cook, J.A. & Freudenberg, J.S. (2008). Controller Area Network (CAN). Retrieved September 5, 2018 from https://www.eecs.umich.edu/courses/eecs461/doc/CAN_notes.pdf
[6] Car Hacking: Accessing and Exploiting the CAN Bus Protocol
[7] R. Currie, ‘Developments in Car Hacking’, SANS Institute, 2016.
[8] H. Lee, S.H. Jeong, H.K. Kim, in 2017 15th Annual Conference on Privacy, Security and Trust (PST). OTIDS: A novel intrusion detection system for in vehicle network by using remote frame (Calgary, 2017), pp. 57–5709.
[9] FREE-FALL: HACKING TESLA FROM WIRELESS TO CAN BUS
[10] S. Checkoway, D. McCoy, B. Kantor, D. Anderson ,H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T.

Kohno et al., “Comprehensive experimental analyses of automotive attack surfaces,” in Proc. 20th USENIX Security, San Francisco, CA, 2011.
[11] N. Courtois, G. Bard, and D. Wagner, “Algebraic and slide attacks on keeloq,” in Fast Software Encryption.Lausanne, Switzerland: Springer, 2008, pp. 97–115.
[12] C. Miller, C. Valasek, in Black Hat USA, 2015. Remote exploitation of an unaltered passenger vehicle (2015), p. 91
[13] Mr. Parikshith Nayaka S K, Mrs. Shobha Rani, Dr. Dayanand Lal, Dr. M Anand. (2020). Convert Channel and Information Hiding in TCP/IP. International Journal of Control and Automation, 13(02), 582 - 591. Retrieved from http://sersc.org/journals/index.php/IJCA/article/view/11199
[14] Jacob, I. Jeena. (2020). Ensuring Network Security using Secured Privileged Accounts. International Journal of Emerging Trends in Engineering Research. 8. 1959-1963. 10.30534/ijeter/2020/80852020.
[15] Parikshith2020, A modern themed system for patients security of data exposure in semi-convinced servers in the cloud. International Journal of Emerging Trends inEngineering Research, 2020,8, 4123-4127. https://doi.org/10.30534/ijeter/2020/15882020
[16] Mr. Manoj K, Ms. Akhila R, Mr. Ganesh M, Mr. Parikshith Nayaka S K. (2020). Social Media Sentimental Analysis using Machine Learning. International Journal of Advanced Science and Technology, 29(05), 2654 - 2662. Retrievedfrom http://sersc.org/journals/index.php/IJAST/article/view/11364