A Real-time In-Vehicle Network Te
Learning-based IDS Training and V

Hesamaldin Jadidbonab[1], Andrew Tomlinson[1], Hoang Nga Nguyen[1], Tran Doan[2] and Siraj Ahmed Shaikh[1]

[1]Systems Security Group, Centre for Future Transport and Cities
[2]CEM School, EEC, Coventry University, Coventry, UK

Abstract

Modern vehicles are built of onboard networked computers, kn range of functionality and features. Due to their connectiv various vectors including wired (e.g., CAN, Lin) and wire To facilitate building and validating security measures a on-board networks and their components, along with a conn required use cases and driving scenarios in an instrumental we propose a multi-component testbed representing a fexibl training and validating machine learning-based Intrusion

Keywords

Automotive security, testbed, machine learning

1. Introduction

Modern vehicles comprise multiple networked comput (ECUs), to realise a range of functionality and featu connectivity, sensing and body modules. Such ECUs networks, including typically a data bus known as the modern vehicles are an example of a Cyber-Physical Sy

Due to a combination of connectivity (over various and features), modern vehicles are also subject to cy nections of sensors, actuators and devices, modern c wired (e.g., USB) or wireless (e.g., Bluetooth, WiF world. Therefore, the various computing systems emb considered as a closed network, and opportunities f automotive networks have become a reality. It has bee seriously impact the safety of a vehicle if packets ar wired network [1]. A number of articles have investigated t modern vehicles focusing on specific [2, 3, 4, 5, 6, 7, 8, 9, 10, 11] goals and v

AI-CyberSec 2021: Workshop on Artifcial Intelligence and Cyber

Email: abc@coventry.ac.uk (H. Jadidbonab); abc@coventry.ac.uk (A. Tomlinson); abc@coventry.ac.uk (H. N. Nguyen); abc@coventry.ac.uk (T. Doan); abc@coventry.ac.uk (S. A. Shaikh)

Counteracting automotive cybersecurity threats re
countermeasures, detection methods and resolution
lenges specifc to the automotive feld. First, it is c
systems. Conducting tests on live vehicles is costly
to property and life. It also presents challenges reg
researchers may not have access to rolling roads or
is understandable why research into automotive cybe
using of ine experiments on previously captured data
challenge concerns ensuring the scope and validity o
range of conditions and operating scenarios would
or controlling factors such as the weather, environ
challenges may be compounded when testing attack de
learning, where the behaviours of the method
vigorous testing needs to account for system interac
studying behaviours of the vehicle components in iso

To facilitate building and validation of security
automotive CPS, it is vital to replicate the on-board
with a connected external environment representing
in an instrumental and controllable environment. Su

- assessing the resilience of components, subsystem
  as part of a mix of commodity and bespoke multi-vec
- tracing and auditing low-level behaviours for ev
- amassing datasets for industry use and wider expe
  demic and developer communities.

While several in-vehicle datasets have been [propose
recently, they only focus on the in-vehicle network c
realistic driving scenarios. This means any dataset
for training and validating machine learning-based
are used to rapidly detect cyber attacks on in-vehicl

In this paper, we propose a multi-component testbed
in-vehicle architecture for real environment trial
solutions. Our contribution is the logical design an
allows cybersecurity researchers and engineers an i
network components. This testbed will provide vario

- A CAN Bus representing a full-scale functional in
- Support for generating realistic driving scenari
- A plug-and-play facility for testing and evaluati
  telematics, sensors, infotainment, in-cabin and
- Support for integrating security solutions for t

The paper is structured as follows. Section 2 presents
automotive cybersecurity testing. It focuses on th

Table 1
Types of Automotive Cybersecurity Testbeds

Name of Testbed	Test Platform	Reference
Open Car Testbed and Network Experiments (OCTANE)	Hybrid	[17]
Resistant Automotive Miniature Network	Hybrid	[23]
Design, Implementation, Validation and Implications of a Real-world V2I Prototype Testbed	Hybrid	[22]
Developing a QRNG ECU for automotive security: Experience of testing in the real-world	Hybrid	[24]
Testbed for Security Analysis of Modern Vehicle Systems	Hybrid	[21]
Mobile Testing Platform	Hardware	[5]
Hardware-in-loop based Automotive Embedded Systems Cybersecurity Evaluation Testbed	Simulator	[19]
Portable Automotive Security Testbed with Adaptability	Simulator	[20]
A Cyber Assurance Testbed for Heavy Vehicle Electronic C	Simulator	[18]

packages. It also discusses existing approaches t
challenges specifc to detection on CAN networks. The
present the relevant sofware and hardware used to im
vehicle simulators and automotive in-vehicle communic [illegible] 3. The implementation [illegible]
detail of our testbed [illegible] Section 4. [illegible] Section 5 [illegible] integra
testbed with an exemplary IDS, which shows how the te
The IDS is used to detect typical attacks to in-vehicl
concluded in Section 6.

2. Related work

2.1. Automotive Cybersecurity Testbeds

Testbeds can generally be categorised into three di
based, and hybrid. Simulation-based testbeds rely so
the behaviour of ECUs and in-vehicle networks. Since
components, simulation-based testbeds are genera
environment for the testers. Hardware-based testb
emulated hardware components. As opposed to sofwa
testbeds enable testers to study interactions betwe
outputs. Hybrid testbeds include both sofware and har
of simulation-based and hardware-based testbeds. [illegible] 1 provides an overview of [illegible]
testbeds, indicating whether they are simulation-ba
testing platform [5] is the only testbed that uses real physical c

for investigating cybersecurity threats. OCTANE, R
Real world V2I Prototype Testbed and Developing a QR
hybrid testing environments. All other testbeds rely
1 lists the most recent and relevant testbeds (introd
categorised initially based on their platforms struc
Testbeds should also be evaluated based on their var
portability, fdelity, cost and types of attacks, atta
protocols supported. This characterisation is esse
regard to the testing scenarios.
To facilitate the testbeds for more thorough and opti
based on existing and emerging standards for cybers
introduce [18]. This framework enables the automotive cyb
automated security tests and a systematic process
cybersecurity based on ISO/SAE DIS 21343. This frame
beyond its common structure to utilise additional to

2.2. Automotive Cybersecurity Attacks and Dete

Attacks to the CAN network entail the attacker tampering
might disrupt the packet broadcast rates, for examp
flooding the network with fabricated packets, thus in
Although immediately disruptive, such crude attack
Potentially more difcult to detect are attacks that s
or meaningfully alter information presentation. For
with misleading data payloads. Such attacks might
systematically change the car performance or efcie
dissatisfaction. The same attacks might entail the fabricati
from attack systems masquerading as legitimate cont
A well-publicised Jeep Cherokee required the attacke
packets that contained plausible, but altered spee
believing the car was travelling more slowly – this,
park-assist, to be activated. A problem in detecting
values remain within legitimate bounds., so might not
Two common approaches to detecting attacks on any co
are signature approaches [3, 28] and anomaly approaches [
trafc patterns against rule bases. Although these ap
ability to determine, encode and distribute evolving
automotive CAN systems because (i) the data derivatio
and (ii) cars have diverse locations, usage patterns
and update signature databases; and, iii) the atta
approaches for detecting CAN attacks are. Amongst many
approaches seek to use statistical or machine lear
indicative of attacks. They tend to be more prone to f
detect novel attacks, are less reliant on maintaini

across car models. But they assume that ample data is

As discussed in Section 1, generating test data and testing detec
llenges when attempted on real vehicles. This has le
validate security methods using data from previous
nipulated offline to fabricate the likely results of a
(e.g., [12, 13, 14, 15]). Such methods are safer than attackin
give the researcher control over the attack manifest
researcher to only testing scenarios that have prev
attack data may be difficult to fabricate; and packet
connected systems or ECUs are difficult to accurately
complication is presented by the packet arbitration a
protocol (discussed in Section 3.1). The ID governs the priority, curtai
CAN packets, and for the validity of testing would ne

3. Background

In this section, we recall the CAN-BUS protocol, th
testbed. We also briefly review the two simulators em
the CAN-BUS network activity. We also outline an exam
as a demonstration of integration into our testbed.

3.1. CAN-BUS - an in-vehicle communication prot

Internal vehicle networks which provide the ECUs c
standard. The CAN 2.0 and SAE J1939 protocols are the t
for passenger and heavy-duty vehicles, respective
automotive applications are its range of adoption, it
with FlexRay, TTE, e.t.c.), its robustness, and its b

The physical structure of CAN comprises a shared phy
are connected. These nodes can transmit/receive da
any node transmits a logical zero then the bus only re
nodes transmit a logical one. As CAN is an event-base
a synchronised start of communication for all of its
frames which can be transmitted periodically or occ
CAN frame includes seven distinguished fields (Figure 1). Most importantly, e
arbitration holds an 11-bit Identifier and is used to
smaller Identifiers having higher priority [35].

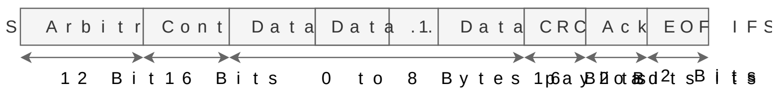

Figure 1: CAN data frame format

The CAN-BUS protocol specifcation is able to format e
of these frames includes two sections of fag and delim
interrupts the normal frames, thus indicating the er
are used for fow control by the receiver [38].

Despite the popularity of the CAN, several related
reported. The[35] re is no message authentication mechani
vulnerabilities are associated with this. This me
message with its sender and thus fails to distingui
message. Additionally, the absence of any encryption
message's integrity and confdentiality [37]. Implementation of an encryptio
challenging due to the low computational power of the
such as lack of backward compatibility, cost and ins
others [38]. These reasons make it convenient for any devi
CAN bus to listen and intercept messages.

3.2. CANoe - an in-vehicle network simulator

CANoe [39], from Vector, is a sofware application primari
and ECU suppliers for developing, simulating, and t
ECUs. An advantage of CANoe is the faithful simulat
timing accuracy and the support of multiple vehicl
using Communication Application Programming Langua
developed by Vector Informatik [40]. Additionally, CANoe
hybrid network confgurations where one part of an on
while the other is physically set up.

In the testbed, the physical CAN bus network of the
simulated one in CANoe via a CAN bus interface device
CANoe enables communication with 3rd party applica
outputs for the simulation. In particular, CANoe uses
enables bi-directional data exchange with other ap
provides car physical data which are sent to CANoe si
CARLA and CANoe, the timers in Python program is depl
virtual ECUs simulated in CANoe to function and genera
as well as the physical one. Conversely, any CAN messag
at creating efects on the simulated vehicles are tra
forwarded to CARLA also by FDX. They enable CARLA to si
the simulated car. Therefore, CARLA functionalitie
data and secondly to realise the efects of received CAN
car.

CANoe enables the researchers with tools for real-ti
options such as message timing, easy observation of
analysis, time-synchronous analysis of multiple bu
sages and various types of stimulation of network tra
hence more realistic scenarios can be implemented.

3.3. CARLA - a car simulator

CARLA [41] is an open-source simulator for autonomous ve
include autonomous driving baselines, Robot Opera
repeatable and customisable trafc scenarios, fexi
the simulation, including trafc generation, pedes
generation, trafc scenarios simulation and a wide r
images, depth, semantics, Lidar). Furthermore, si
using car-like inputs such as steering wheels and foo
exchanged with 3rd party applications via Python pr
with CARLA. CARLA runs on a separate PC.

3.4. Example CAN bus IDS classifier

Later in this paper we show how IDS methods can be int
demonstration and validation. Our goal here is not
particular IDS method, but we use for demonstration a
[42], where the reader can fnd a full explanation of its

The approach assumes that the role of each CAN packe
which is likely given the manufacturer's secrecy re
data variations specifc to the make and model of cars
of CAN data felds which might be combined to provide a
likely sensor data felds (including felds that might b
the statistical profles and pairwise characteristic
packets identifed. Next, a clustering algorithm is a
sensor data felds might be split into functional clus

Having determined a candidate cluster, a bufer is u
from each feld in the cluster. Thus, at each packet br
cluster are examined together as a snapshot. This pr
a contextual gauge for evaluating each of its compon
the cluster can be used for anomaly detection using
in some of the data felds might appear anomalous acco
whole cluster.

A one-class classifer is used since this might dete
anomaly data is not yet available, or cannot be obtain
one-class classifers have been considered for in
demonstration, we use a One-class Support Vector Mach
try to fnd the largest margin between the projected tr
data-dense regions. Target outputs are thus class
position relative to the margin. For this demonstrat
be optimal with comparable CAN networks tested on oth
0.015 and gamma 0.5). For classifer training data, mu
ofine. During online testing, the current snapshot
classifer.

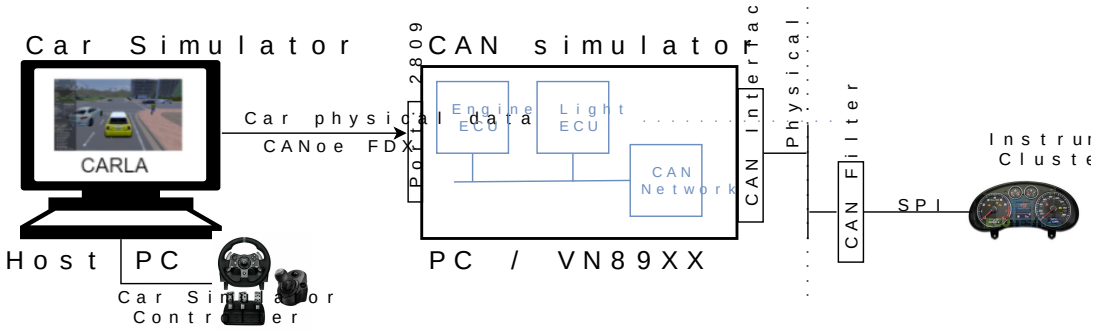

Figure 2 Test-bed configuration

4. Testbed

This section presents our testbed which facilitates l on in-vehicle networks. First, the logical design c provide the detail of the testbed implementation.

4.1. Logical Design

The section outlined the basic requirements that any au regarding vehicle-security technology developmen demonstrator consists of a car simulator, an on-boa and a data storage device. Figure 3 depicts the logical setup of the tes used to generate car physical data for the on-board n information such as engine rev speed, vehicle veloc status. The on-board network simulator is used to s network.

The simulated network contains several virtual ECUs as Engine ECU, Transmission ECU and Light ECUs. The vi car physical data from the car simulator to generate physical network is attached to the simulated networ

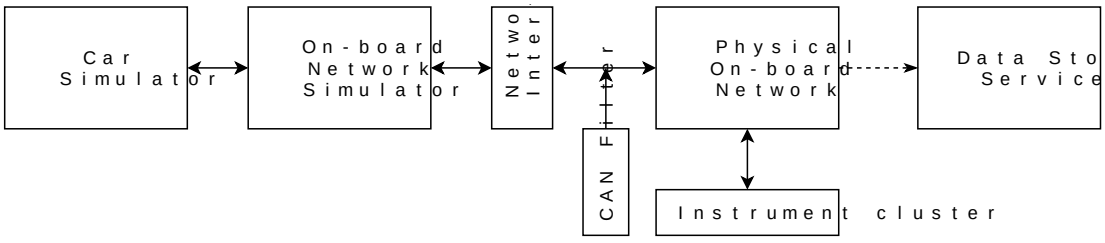

Figure 3 Logical setup of the demonstrator

In the next section, the components of the logical s

illustrated in Figure 2.

4.2. Implementation

In this section, implementation details of a testbed described. The proposed automotive testbed includ simulator, a physical network, a real car's instrume Figure 2 provides an overview of the testbed architectu and its CAN bus network is realised within a virtual en simulator. The network and car simulations is run c separate machine, respectively. The Vector hardwa layer, where an additional physical component requir proposed testbed, a real car's instrument cluster a hardware simulator to form the physical CAN bus networ the testbed's operational capability to perform re simulation setup creates a realistic framework of ph physical bus and with simulated ECUs via the virtua driving behaviour, a full set of driving wheel, ped simulator.

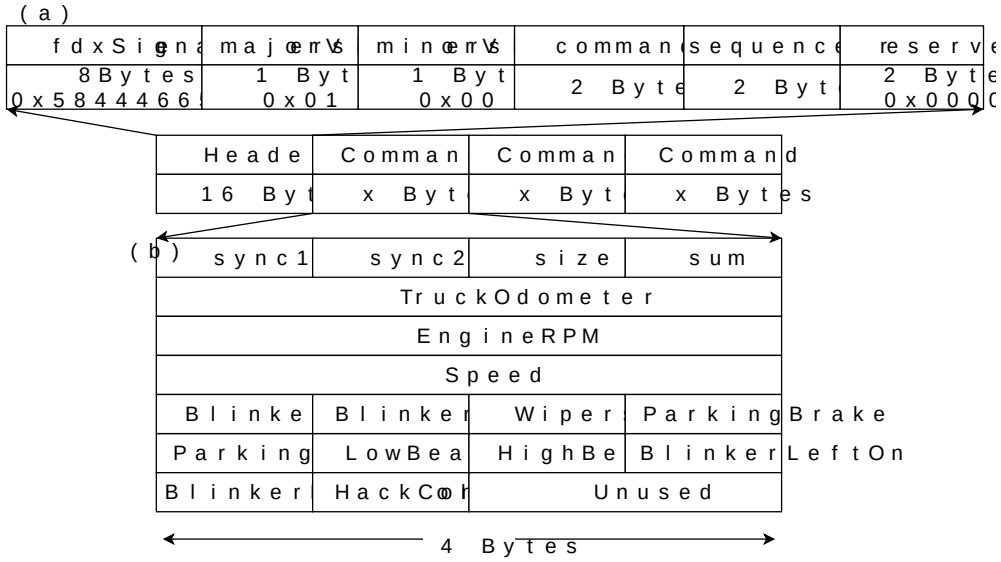

Figure 4: UDP datagram layout consisting two major sections: a) of a FDX signature, b) 28 bytes of datagram command

CARLA is used as the car simulator to generate car ph simulator. These include engine information such as gear, acceleration throttle, and brake status. Als Vector CANoe is employed. CARLA generates realisti CANoe by Fast Data eXchange (FDX).

CANoe FDX is a UDP-based protocol (User Datagram P
real-time exchange of data between CANoe and other
This protocol enables other 3rd party sofware both r
environment variables, and bus signals. To this end, e
simulated car information including Engine RPM, Spee
are sent by CARLA. UDP is a commonly used protocol wh
transmission model which means often there must be a
CARLA [45]. As there is no pre-defned UDP stack availabl
implemented prior to use. The transmitted data by C
reciprocal transmission of the UDP on a specifc port.
sends a datagram to CANoe frst and therefore has to kno
by CANoe for the FDX protocol. CANoe evaluates all i
sender's IP address and port number and always respon
data. UDP clients in python is an convenient way of sending
Therefore, UDP client is the specifcation of a designated so
in CARLA python code in order to communicate with CANo
the port number and the IP address of the CAN simulat
in Figure 4, a new command section including the selecte
entire process is implemented in CANoe. An environment
continually send the telemetry data while the game is
Receiving these FDX messages requires confguring
receive the incoming data in an appropriate order. The
by a FDX description fle and added to the CANoe confgur
eXtensible Markup Language (XML) fle. It specifes the
data groups that is being sent reciprocally by CARLA.
two data groups: groupID '1' and '2'. Each data group
data. In the proposed testbed, groupID '1' and '2' are
data to CANoe, see Figure 5. The FDX option must also be enabled.
be confgured using the edit option within the CANoe e
several parameters given to the data group members su
size and type.
A separate machine is used to confgure the simulated
virtual ECUs. To maintain timing in CARLA and CANo
are deployed. The simulation confguration is then l
(VN8914) with the communication module (VN8972) to e
The simulated CAN messages are transmitted to the ph
the instrument cluster through the hacking device th
The hacking device is a CAN flter that can manipula
bus and one or more ECUs. To realise the CAN flter, the A
is used in the testbed. It features two CAN interfaces
other is connected to the instrument cluster. We imp
MCU so that it is acted as a mileage corrector. The CAN
In mode 1, it is inactivated. This means the CAN flt
receives from the interface to the CAN Bus to the other

cluster. In modes 2, 3 and 4, the CAN filter is activate
of the CAN frames to reduce the incremental rate of th

- In mode 2, the incremental rate of the odometer increases by only 1 mile.
- In mode 3, the incremental rate of the odometer increases by only 5 miles.
- In mode 4, the incremental rate of the odometer does not increase.

One can switch between the modes by quickly flashing th
four times. The CAN filter starts with mode 1. If it
into mode 4, it switches it back to mode 1.

The filtered CAN messages are then sent to the real ins
the IDS.

5. IDS Integration

In this section we show how the testbed employing a
Section 4 can be used for testing a potential IDS implementa
in Section 3 where, following CAN packet log analysis t
a classifer is trained offline using captured CAN log
testbed.

5.1. Cluster Detection and Offline Training

Analysis of the packet flow generated by the CARLA simul
broadcast per second, with a mean broadcast interval
CAN packets were identified. These, as pure gross data
cluster which was used for the subsequent analysis.
training data, the log from a 30-minute journey was
snapshots. The classifer was then trained offline usi
The fitted model and the scaling objects are saved usi
can be loaded into our intrusion detection script and

5.2. Online Testing

The online testing using the trained classifier and t
run our Python intrusion detection script on a laptop
Light HS v2 logger.

The script reads the CAN messages via the CAN channel
requires drivers (e.g. Kvaser Drivers for Windows)
python-can library to connect to the CAN channel. Python-can
is broadcast, including the ID and eight data fields.

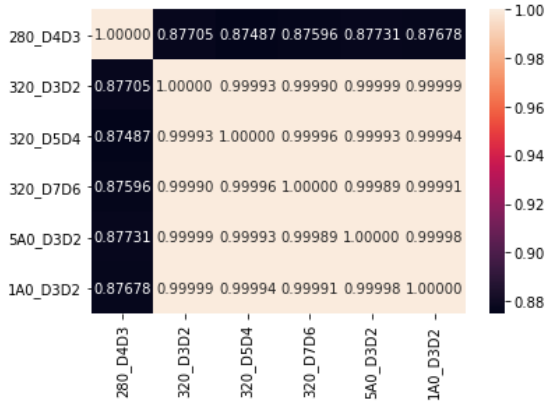

Figure 5. Correlation matrix of CAN data fields from a correlat

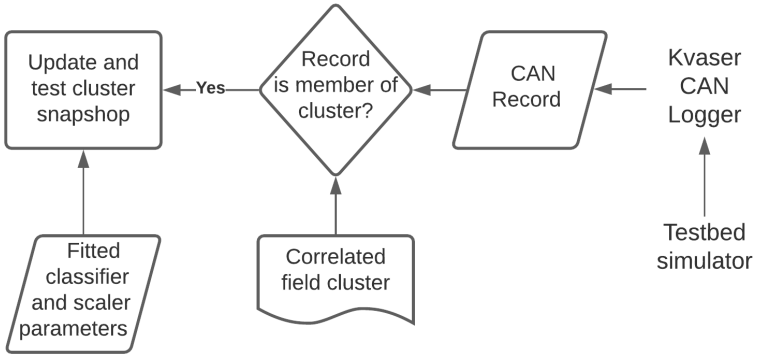

Figure 6. Overview of the online process for the IDS

As each packet is detected, the script checks the ID f
packet ID is in the cluster list, its relevant data fe
resulting converted integer feld values are updated
is then fed to the classifer, which uses the pickled
during the training stage. The snapshot is then clas
If the prediction is an anomaly, the CAN packet is con

5.3. Experiment Results

Prior to testing on the testbed, we tested the classi
against output generated by replaying previously ca
The data for this used diferent journeys to those use
30-minute journey representing one of the operationa
flewas fed into the trained IDS to make predictions if
2, 3 or 4. The evaluation is facilitated by two count
in the IDS to count howmany times it predicts “ inact

values of the counters in each run, we obtained ... For the results in Table 2, "accuracy" is 0.763, "recall" is 0.694, "precision" is 0.978, and "F1" is 0.813.

Table 2
Results from testing the IDS via replay of CAN logs on a vir

CAN Filter Mode	Predicting Activation	Predicting Inactivation
Mode 1 (inactivated)	25155	556672
Mode 2 (activated)	406834	156727
Mode 3 (activated)	382645	172793
Mode 4 (activated)	378270	177498

In comparison, we evaluated the IDS online by plug-
messages between the instrument cluster and the CAN fi
We ran the test four times, each time running 30 minut
set in one of the four modes. In spite of the high thro
was able to process each in real-time. Clearly, an im
diferent hardware compared to the laptop we used, th
evaluation of alternative implementations. Simila
described in Table 3.

Table 3
Results from live-testing the Online IDS on the testbed

CAN Filter Mode	Predicting Activation	Predicting Inactivation
Mode 1 (inactivated)	359386	143963
Mode 2 (activated)	392678	177223
Mode 3 (activated)	426183	143715
Mode 4 (activated)	376455	193490

The testbed detection (Table 3), which was processed in real-time, in
CAN replay (Table 2), showed more false positives in classifyin
"accuracy" is 0.652, "recall" is 0.691, "precision" is 0.760, and "F1" is 0.723. The accuracy reduce
60% on the testbed, compared with 76% using replay on the vi
repeated the testing using additional journeys and t

The reasons for the reduced detection accuracy on t
The log was replayed at the fastest available replay
so, perhaps a diference in the speed of transmission
calculations on the laptop running the IDS script.
observed on the testbed car operation that was not c
log used for the virtual CAN testing. It is thus possi
more training data might be required to improve the li
A further cause of poor prediction might stem from the
more research is needed to refne and test this. Wha
the results indicate the need for further explorati
highlight the beneft of using the testbed for validati

driving behaviours and experiences might be diferen
were collected for training. The approach adopted, wh
data derivations are unknown, clearly cannot give a co
what the data represents. Also, without a more thorc
whether convoluted manipulations or conditional ca
the data. These would reduce the detection reliabili

6. Conclusion

In this paper, we presented a novel approach for buil
comprises both a full-scale in-vehicle network and a c
realistic driving scenarios. The in-vehicle network
to resemble a real CAN Bus within a vehicle to a high
fexible confguration of the network in terms of (1) r
enrich simulations and (2) allowing plug-n-playing
this end, we realise driving scenarios by employing
customisable with respect to driving environments (
as input to the CAN Bus to generate realistic CAN traf
generating datasets and testing machine learning-b
This was illustrated by the example of training and
a classifer in real-time using the testbed revealed
performed by running the trained classifer ofine agai
reasons for the poorer detection and the circumstan
rates, need investigating - something the testbed c
diverse driving scenarios. It is conjectured that th
experiences, and those captured when collecting dat
to the reduction of the implemented IDS accuracy when
is compared against running a virtual simulation. Th
the testbed for validating the trained model agains
the live-journey replicating testbed can uncover iss
in ofine testing.

While this paper only focused on the IDS as anomaly d
methods such as potentially against nearest neighbou
will be bench-marked against each other to further i
their accuracy. Additionally, more physical compone
to include more abuse cases. Also, the data exchange w
study the efect of anomaly in the driving behavior.

References

[1K]. Koscher, A. Czeskis, F. Roesner, S. Patel, et a
modern automobile, in: 2010 IEEE Symposium on Sec

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, et
analyses of automotive attack surfaces., in: USEN
[3] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, Su
mechanisms in embedded automotive networks, in: I
[4] C. Miller, C. Valasek, Remote exploitation of an u
USA 2015 (2015) 91.
[5] C. Miller C., C. Valasek, Car hacking: for poories
[6] T. Hoppe, S. Kiltz, J. Dittmann, Security threats
examples and selected short-term countermeasure
[7] U. E. Larson, D. K. Nilsson, Securing vehicles agai
4th annual workshop on Cyber security and informat
[8] P. R. Thom, C. A. MacCarley, A spy under the hood: Co
Risk Management Magazine 55 (2008) 22.
[9] M. Wolf, A. Weimerskirch, C. Paar, Security in auto
Embedded Security in Cars, Bochum, 2004.
[10] M. Wolf, A. Weimerskirch, T. Wollinger, State of th
EURASIP Journal on Embedded Systems 2007 (2007) 1-
[11] Y. Zhao, Telematics: safe and fun driving, IEEE In
[12] A. Taylor, N. Japkowicz, S. Leblanc, Frequency-bas
CAN bus, World Congress on Industrial Control Syst
[13] A. Taylor, S. Leblanc, N. Japkowicz, Anomaly Detec
Data with Long Short-Term Memory Networks, 2016 IE
[14] I. Studnia, E. Alata, V. Nicomette, M. Ka, I. Studni
A language-based intrusion detection approach f
IEEE (Ed.), 21st IEEE PRDC on Dependable Computing
[15] M.-J. Kang, J.-W. Kang, Intrusion Detection Sys
In-Vehicle Network Security, PLoS ONE 11 (2016).
[16] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fe
network intrusion detection: Techniques, systems
[17] P. Borazjani, C. Everett, D. McCoy, Octane: An exte
in: Proceedings of the Embedded Security in Cars C
[18] J. Daily, R. Gamble, S. Moftt, C. Raines, P. Harris,
J. Johnson, Towards a cyber assurance testbed for h
International Journal of Commercial Vehicles 9 (2
[19] P. S. Oruganti, M. Appel, Q. Ahmed, Hardware-in-
systems cybersecurity evaluation testbed, in: Au
[20] T. Toyama, T. Yoshida, H. Oguma, T. Matsumoto, Pa
testbed with adaptability, Black Hat Europ, Tech.
[21] K. Zheng, L. Pan, H. Chen, R. Di Pietro, L. Batten, A
vehicle systems, in: 2017 IEEE Trustcom/BigDataS
[22] A. Marchetto, P. Pantazopoulos, A. Varádi, Cvs: D
implications of a real-world v2i prototype testbe
[23] C. Gay, T. Toyama, H. Oguma, Resistant automotiver
[24] H. N. Nguyen, S. Tavakoli, S. A. Shaikh, O. Maynard,

security: Experience of testing in the real-world
[25] A. J. Tomlinson, J. Bryans, S. Shaikh, Towards viab
automotive controller area network, in: ACM-CSCS
[26] S. Marksteiner, N. Marko, A. Smulders, S. Karagiann
S. Kraxberger, A process to facilitate automated a
[27] T. Hayward, A. Tomlinson, J. Bryans, Adding cybe
simulator, in: IEEE 19th QRS-C, 2019.
[28] T. K. T. N. (KTN), Automotive Cyber Security: An IE
of risk perspectives for connected vehicles, Tech
[29] C. Valasek, C. Miller, Adventures in Automotive Ne
White Paper (2013) 99.
[30] S. Fröschle, A. Stühring, Analyzing the capabiliti
Computer Science 10492 LNCS (2017) 464–482. arXiv:978
[31] D. S. Fowler, J. Bryans, M. Cheah, P. Wooderson, S. A
automotive cybersecurity tests, a can fuzz testin
[32] J. Singh, M. J. Nene, A Survey on Machine Learning
Systems, IJARCCE 2 (2013) 4349–4355.
[33] D. Bhattacharyya, J. Kalita, Network Anomaly Dete
[34] B. Groza, P. S. Murvay, Efcient Intrusion Detection
Networks, IEEE Transactions on Information Forens
[35] S. C. HPL, Introduction to the controller area netw
(2002) 1–17.
[36] W. Voss, A comprehensible guide to controller area
[37] M. Müter, N. Asaj, Entropy-based anomaly detecti
IEEE Intelligent Vehicles Symposium (IV), IEEE, 2
[38] N. Nowdehi, A. Lautenbach, T. Olovsson, In-vehi
evaluation based on industrial criteria, in: Vehi
[39] Vector, Canoe user manual, https://www.vector.co
[40] CAPL Documentation, Vector Informatik GmbH, 2020. URL https://kb.vector.com/
[41] CARLA - Open-source simulator for autonomous driving, 2021. URL https://carla.org
[42] A. Tomlinson, J. Bryans, S. A. Shaikh, Using interna
area network attacks, Computers & Electrical Engi
[43] L. A. Maglaras, A Novel Distributed Intrusion De
Networks, International Journal of Advanced Comp
[44] C. Chio, D. Freeman, Machine Learning and Security
[45] A. Huber, Fast Data Exchange with CANoe, 2020. URL https://assets.vector.co
[46] Kvaser, Kvaser, 2021. URL https://www.kvaser.com/
[47] Python-CAN, python-can, 2020. URL https://python-can.readthedoc