IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 26, 2021, accepted November 14, 2021, date of publication November 23, 2021,
date of current version December 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3130495

Vehicle Security: A Survey of Security Issues and Vulnerabilities, Malware Attacks and Defenses

ABDULRAHMAN ABU ELKHAIL[1], (Member, IEEE), RAFI UD DAULA REFAT[1], (Graduate Student Member, IEEE), RICARDO HABRE[1], AZEEM HAFEEZ[1], ANYS BACHA[2], (Member, IEEE), AND HAFIZ MALIK[1], (Senior Member, IEEE)

[1]Electrical and Computer Engineering Department, University of Michigan–Dearborn, Dearborn, MI 48128, USA
[2]Computer and Information Science Department, University of Michigan–Dearborn, Dearborn, MI 48128, USA

Corresponding author: Anys Bacha (bacha@umich.edu)

This work was supported in part by the National Science Foundation under Grant CNS-1947580 and Grant CNS-2035770, and in part by the Deputyship for Research and Innovation, Ministry of Education in the Kingdom of Saudi Arabia (KSA), under Project DRI-KSU-934.

ABSTRACT Recent years have led the path to the evolution of automotive technology and with these new developments, modern vehicles are getting increasingly astute and offering growing quantities of innovative applications that cover various functionalities. These functionalities are controlled by hundreds of Electronic Control Units (ECUs) which are connected to each other via the Control Area Network (CAN) bus. Although ECUs are designed to offer various amenities that are associated with modern vehicles including comfort, such features expose new attack surfaces that can be harnessed by attackers. This trend is exacerbated by the fact that many of these ECUs rely on wireless communication for interacting with the outside world. Therefore, making them vulnerable to common threats such as malware injection that can compromise the overall security of modern vehicles. In this paper, we provide a detailed description of the architecture associated with intelligent vehicles, and identify various security issues and vulnerabilities that impact such systems. We provide an overview of different malware types and the vectors of attacks they leverage for infecting modern vehicles. This work also presents a detailed survey of available defenses against such attacks including: signature, behavior, heuristic, cloud, and machine learning-based detection measures. Furthermore, this paper intends to assist researchers in becoming familiar with the available defenses and how they can be applied to secure intelligent vehicles against emerging malware threats that can compromise the security of today's vehicles. It also provides future directions for researchers who are interested in developing new defenses that can safeguard intelligent vehicles systems against malware attacks.

INDEX TERMS Vehicle security, vulnerabilities, security issues, malware, intelligent vehicle, malware detection, intrusion detection system, defense system, cybersecurity.

I. INTRODUCTION

Vehicle systems have seen a great transformation since the previous decade in many aspects going from vehicle control to telematics and advanced driver help frameworks. Vehicular systems have seen plenty of additions and increased their complexity of using the ECUs to provide many improvements in terms of functionality and comfort [1]. With the increase in usage of ECUs in vehicle systems, functionalities have improved, but they have also exposed vehicles to be more susceptible to cyberthreat, making them more gullible to cyber attacks. For instance, with physical access to a vehicle, an attacker can inject malicious messages into the CAN bus, modify and read an ECU via vulnerable interfaces such as CD players, USB and OBD-II [2]. To prove the fact, some researchers have sent out fake messages using the in-vehicle networks to different ECU's, peruse ECU memory and ECU security keys, peruse and alter ECU programming and control a wide scope of vehicle capacities at ease [3]. Such attacks can cause severe repercussions on the vehicle system tasks and also bring great danger to the safety of the drivers.

On the other hand, with the development of wireless technologies such as Bluetooth, Wi-Fi, Cellular, LTE, and 5G, vehicles can no longer be considered as closed systems,

The associate editor coordinating the review of this manuscript and approving it for publication was Yan Huo.

as they are increasingly equipped with functionalities that interact with the environment through these technologies, which can be exposed to attacks over-the-air (OTA) [4]. For example, security keys have been used by vehicle key fobs in order to hack a live system [5]. Radio signals are another way for hackers to breach security and in a few instances, researchers were able to transmit radio signals from a key fob to the car without disrupting any security keys, allowing attackers to simply unlock doors and steal or burglarize the vehicle [6]. Another very common way for Hobbyists to mishandle systems is by tampering with the tire pressure monitoring systems (TPMS) where one can set false readings to send out bogus warnings, causing confusion to the driver [7]. Also, the authors of [1]–[3], [8] were able to inject malicious firmware into a vehicle's OTA system while performing an ECU firmware update. Additionally, researchers could hack into the steering and brakes of two cars [9]. In another report, a team of hackers was able to hack a Tesla Model S remotely from a distance of 12 miles [10]. Other work by Miller and Valasek [11] was shown to hack and stop a Jeep Cherokee running on a highway remotely, which led to a recall of 1.4 million vehicles. Another example is provided by Cai *et al.* [12], which revealed multiple vulnerabilities in numerous BMW models including the ability to compromise ECUs connected through CAN over a wireless connection. Such a reality concerning vehicle attacks makes automotive security one of the most critical issues.

Many attacks that previously could take place through physical access only, can now be easily carried out remotely with the help of wireless technologies. Therefore, allowing attackers to breach into the vehicle systems with the possibility of extending such attacks to multiple vehicles through daisy chaining. One severe threat to intelligent vehicles is malware which is a malicious software designed to obtain unauthorized access to data or disrupt computer operations. Malware can infect intelligent vehicles through a variety of vulnerabilities, including wireless communication with roadside networks, vehicle-based Wi-Fi hotspots, and internet connectivity. Another common vector of attack is concerned with malware-infected consumer electronic devices such as cell phones, iPods, and laptops that can be physically or wirelessly connected to the vehicle and in turn used to exchange files between vehicles. Vulnerabilities in onboard communication systems, software, and hardware designs, [2], [8], [13] can also be abused by malware to infect a vehicle. Malware can cause a wide range of disturbances and harm to the vehicle system once it is inside the vehicle [1]–[3], [8]. Some examples of how malware affects the vehicle's normal operation are: Toying with the general features of the vehicle causing driver distraction, disrupting standard functions of the vehicle like messing with the in-car radio so that the driver cannot switch it on, locking the car's features, illegitimately occupying memory space and CPU cycles, mishandling of data and invading privacy, and disabling safety features of the vehicle. The aforementioned examples underscore that intelligent vehicular systems are a high priority that must be appropriately handled in order to effectively safeguard them.

In this paper, after offering a detailed description of the intelligent vehicle's architecture, this paper discusses the security issues and vulnerabilities that intelligent vehicles face. It also gives an overview of malware attacks and examines the many forms of malware that might infiltrate intelligent vehicles, as well as the malware's probable methods of infection. It also provides a comprehensive survey of available malware defense systems, categorizing them into five categories: signature-based malware detection techniques, behavior-based malware detection techniques, heuristic-based malware detection techniques, cloud-based malware detection techniques, and machine learning-based malware detection techniques. It also discusses the upsides and downsides of every defense system against malware attacks and the various strategies that are utilized in these defense systems. This paper aims to aid researchers in developing a broad understanding of malware protection systems that are available for protecting such systems. It also identifies potential research directions for researchers to pursue in order to increase the intelligent vehicle system's resistance against malware attacks. To the best of our knowledge, this is the first study that offers a detailed survey of the most recent existing malware defense systems and assesses the benefits and drawbacks of deploying such defenses onto intelligent vehicle systems.

Overall, this paper makes the following contributions:

- Provides an in-depth description of the intelligent vehicle system's architecture.
- Describes the most prevalent types of malware that might infiltrate the intelligent vehicle system.
- Identifies the issues and vulnerabilities that intelligent vehicles face in terms of security.
- Discusses all possible entry points for malware to infect the intelligent vehicle system.
- Presents a detailed survey of the most recent malware detection techniques in the last decade and discusses the upsides and downsides of applying such techniques to the intelligent vehicle system.
- Provides researchers with prospective study areas for improving the intelligence of vehicle systems and making them more resistant to malware attacks.

Overall, this paper represents an effort of understanding how malware attacks affect vehicle systems and the best practices undertaken for building safer and sturdier systems. The paper is divided into the said sections: Section II gives a detailed description of the architecture of intelligent vehicles. Section III identifies the security issues and vulnerabilities of intelligent vehicles. Section IV provides an overview of malware attacks and discusses the main kinds of malware that can infect intelligent vehicles, as well as the malware's possible ways of infection. Section V discusses existing malware defense techniques, as well as, their pros and cons. Section VI discusses research problems for researchers to address and provides future directions along with some recommendations

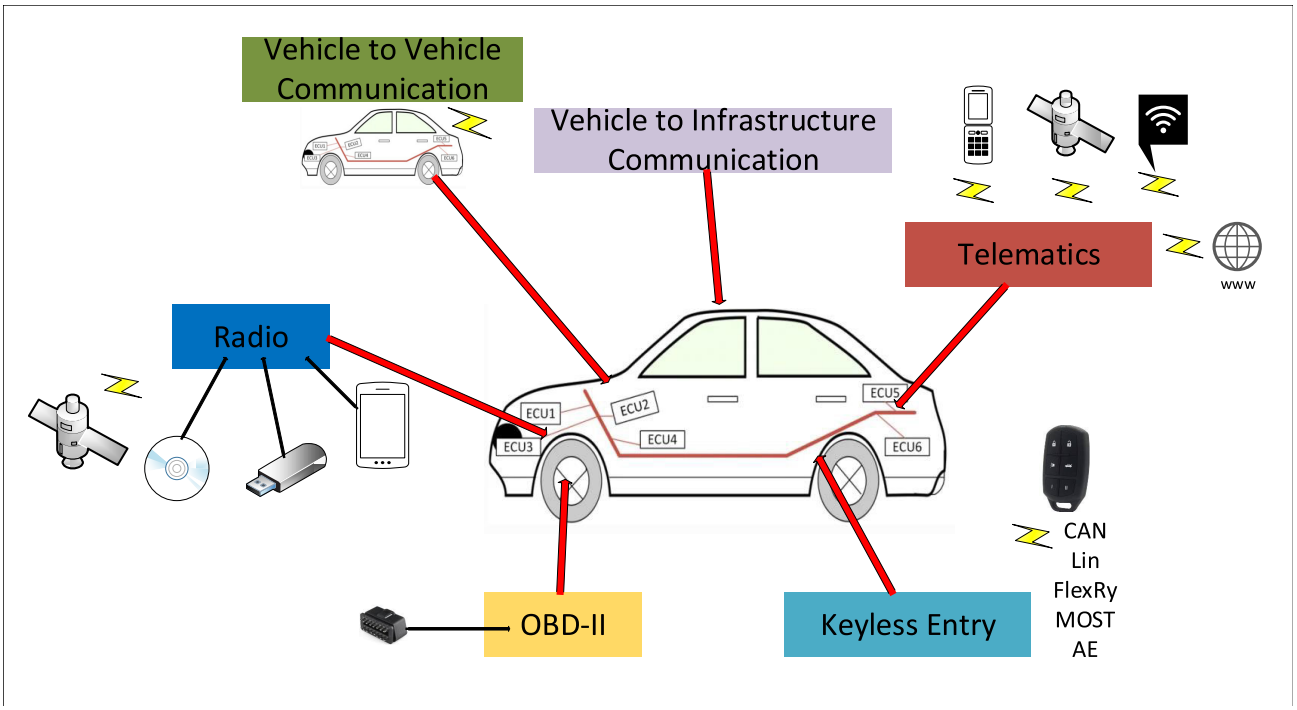


FIGURE 1. The intelligent vehicle architecture.

for developing a more effective malware defense system for intelligent vehicles; and Section VII concludes.

II. THE ARCHITECTURE OF THE INTELLIGENT VEHICLE

Driver assistance technology is the new future in the automotive industry. Automotive companies are leaning towards developing intelligent connected vehicles that are capable of assisting the driver by including safety features like lane-keeping assist, adaptive cruise control, brake collision, etc. However, such technologies require a high-speed communication protocol that is suitable for all advanced Electronic Control Units (ECUs). For this particular reason, the architecture of intelligent vehicles must be designed in a way such that all the different ECU modules are able to communicate with less complexity [14]. In other words, the architecture of intelligent vehicles requires a technological upgrade with respect to in-vehicle network architecture, computational platforms and sensors. The architecture of the intelligent vehicle is shown in Figure 1.

A. IN-VEHICLE NETWORK ARCHITECTURE

To have a better comprehension of the threats that ECUs face against hackers, it is worth having an understanding of the communication protocol between the ECUs that could serve as a potential entry point for the hacker [15]. The Controlled Area Network (CAN) bus was developed in 1983 by an automotive company called Bosch [16]. This protocol has now made it possible for different ECUs to communicate in a fast and reliable manner. The CAN bus has provided a durable and inexpensive solution that allows ECUs to communicate with each other using a single CAN interface instead of analog and digital inputs [16].

Each ECU transmits CAN frames to the receiver labeled by an arbitration ID. All connected ECUs receive the frames, but each ECU decides whether or not it can accept the frame depending on the arbitration ID. Previously used electronic architecture technologies weren't able to allow much space for different ECUs in intelligent vehicles. With the help of the CAN bus, intelligent vehicle manufactures are now able to fit many more ECUs while minimizing the complexity of wiring [16]. Figure 2 illustrates different ECUs and how they are connected to various electronic subsystems.

Each and every subsystem that we can see in Figure 2 has multiple ECUs that are responsible for controlling specific functionality in the vehicle [1]. Through a high-speed communication protocol (CAN), different ECUs in different subsystems are able to communicate with each other. Different subsystems use different types of subnetworks depending on the time sensitivity of each subsystem [15]. For instance, time-sensitive engine control, power-train, and safety subsystems use the high speed controlled area network (CAN) whereas fewer safety subsystems such as seats and windows motor control use a Local Interconnect Network (LIN) [14], [15]. The Automotive Ethernet (AE) and the Media Oriented System Transport (MOST) are used in the In-Vehicle Infotainment (IVI) subsystem to control car radio, navigation system, Bluetooth, etc., [14], [15], [17]. The MOST network is isolated from electromagnetic interference because it utilizes plastic optical fibers as its physical layer which stops problems like buzzing noises in the

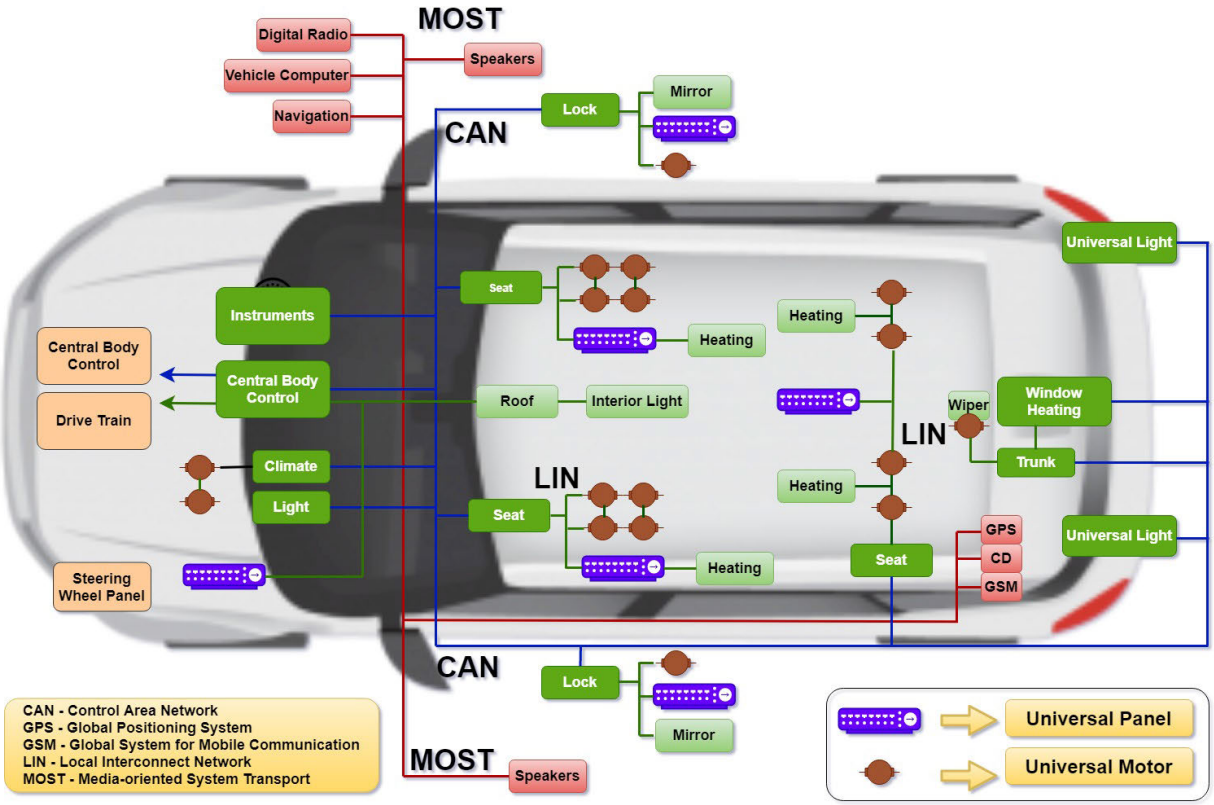


FIGURE 2. In-vehicle network architecture.

infotainment system [14]. The AE has a great advantage when it comes to bandwidth capacity since it can support up to 100 Mbps that is slated to increase to nearly 1 Gbps in the near future [14]. In general, the AE is considered to be approximately 100 times faster than the CAN protocol. Therefore, it would be a good choice to replace CAN with Ethernet; however, due to the fact that Ethernet's cost per ECU is higher than CAN, it will most likely not replace but rather get added on to it [18]. Flex-Ray is another in-vehicle network that has high transmission rates and is used to obtain a good control system. Flex-Ray supports drive-by-wire systems such as steer-by-wire and brake-by-wire which also requires great error management to perform as a great driver assistance system. The specification of the in-vehicle network buses is shown in Table 1.

Intelligent vehicles nowadays offer access to an in-vehicle network system to keep track of messages over this system through the On-Board Diagnostics (OBD-II) port in order to provide diagnostic reports. The intelligent vehicles are also provided with an entertainment system with either a USB connectivity option or a CD player. These options enable the users to synchronize and access entertainment content from their mobile devices and play or view them on the vehicle's entertainment systems. Besides, remote key entries and RFID car keys are other modern car technologies that have been largely applied to intelligent vehicles. These technologies can be used to access the vehicle functions such as door opening, flashlights and in some recent cases, are used to even access ignition functions. In addition, the technology of intelligent vehicles nowadays has tremendously shifted towards connecting the in-vehicle network subsystem to the outside world through WiFi, Bluetooth, and cellular networks such as LTE, 3G, 4G, and now 5G [20]. For example, a cell phone can now connect to the infotainment system of the vehicle wirelessly, using Bluetooth connectivity that allows the infotainment system to use apple car play and android auto through the connected phone. Furthermore, WiFi and 5G can be used to offer functionalities like Global Positioning System (GPS), digital radio and traffic messages. Additionally, the telematics unit allows the car to communicate with 3G, 4G, and now 5G networks. It can send and receive telematics data, communicate with back-end cloud servers, and allow access to the internet. Moreover, Dedicated Short Range Communications (DSRC) is an on-board vehicle unit that is developed to establish short-range communications between Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) as well. DSRC offers great autonomous technology services by allowing vehicles to exchange information either with each other or with the infrastructure such as roadside units that are surrounding the vehicle. DSRC utilizes radio frequency (RF) channels to achieve this communication [21].

TABLE 1. The in-vehicle network buses specification [19].

	AE	CAN	FlexRay	MOST	LIN
Bandwidth (Mb/s)	1000 (developing)	1 or 10 (CANFD)	20	150	0.02
Maximum number of nodes	Number of switch ports	30	22	64	16
Network length	15 m per link	40 m	24 m	1280 m	40 m
Messaging	IP based	Multi-master	Multi-master	Cyclic frames /streams	Master-slave
Cost	High	Low	Low	High	Very low
Safety-critical functionality	Proven outside of automotive applications	Yes	Yes	Yes	No
Availability	Growing	Many	Few	One	Many
Cabling	UTP	UTP	UTP	Optical,UTP	1-wire
Main applications	Infotainment, Backbone (future)	General bus	Safety-critical, X-by-wire	Infotainment	Seats, doors, Switches

CAN Bus: The CAN bus is a standard communication protocol that is currently the most commonly used in the in-vehicle networks. It is a broadcast-based protocol that provides up to 1Mb/s data rate on a single bus and it enables ECUs to exchange messages between them to control the process of vehicle components. The CAN protocol adopts Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) which prevents collisions on the bus when several ECUs compete for access to the bus since all nodes share the same bus. It also doesn't have addressing scheme like TCP/IP protocol. Therefore, when ECUs broadcast their CAN message frames on to the bus, each message frame is allocated a unique identifier, known as the CAN ID or the arbitration ID, which defines the priority and the content of the message frame. If two or more ECUs attempt to send messages simultaneously, the message with the smallest ID has the highest priority to transmit the message first. The CAN ID range starts from 0x000 to 0x7FF for the standard identifier field which has 11 bits.

In general, the transmitted messages frames on the CAN bus are divided into four major types: the remote frame, the overload frame, the data frame and the error frame. There into, the remote frame is used to enable the received ECU to request the data from specific ECU, the overload frame is utilized to inform that the source ECU cannot receive the data and the error frame is utilized to inform other ECUs regarding the happened error. The Data Frame is used to carry the data from the transmitter ECU to the receiver ECU. The Data Frame is composed of the start of frame (SOF) field which contains one dominant bit and informs a start of transmission to all ECUs, arbitration field which consists of 11 bits as an identifier and one bit as remote transmission request (RTR) and characterizes the priority and the type of the frame, control field which consists of two reserved bits and four bits as data length code (DLC), data field which includes the actual data in a range of 0 to 64 bytes. In addition, the cyclic redundancy check (CRC) field which consists of 15 bits as CRC and 1 bit as CRC delimiter and performs the data error detection, the ACK field which consists of one bit as ACK part and one bit ACK delimiter part, and end of frame which consists of 7 bits and indicates the end of the CAN frame by a recessive bit flag [22]. Figure 3 shows the structure of CAN frame.

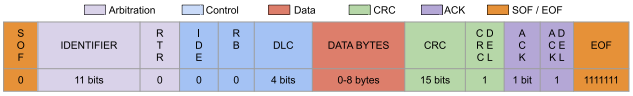


FIGURE 3. Structure of CAN frame.

B. INTELLIGENT VEHICLES' COMPUTATION PLATFORMS

The vehicle's computation platform plays an important role in high intelligent vehicle systems to make sure that the autonomous technology process is smooth, robust, and efficient. Millions of lines of code must get executed in order to accomplish different intelligent algorithms and autonomous functionalities. Generally, Digital Signal Processors (DSPs) and Micro-controller Units (MCUs) are used for signal processing to establish several vehicle functions.

Furthermore, DSPs are capable of establishing more complicated applications that demand high quality processing capacity and integration such as Advanced driver-assistance systems (ADAS) [14]. Moreover, a robust and advanced computation platform such as Graphics Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs) must be implemented to ensure the efficiency of the autonomous system. GPUs are a great way to perform various types of image processing which could improve obstacle detection algorithms, traffic signs, and all the ADAS functionalities. FPGAs are also useful for similar computations with less energy consumption [14], [23].

Looking at the software system of the computation platform, the automotive industry uses many open systems such as OSEK, JASPAR, and VDX. However, they fail to be reusable for the advanced ECUs. Automotive Open System Architecture (AUTOSAR) is also another open system that is developed to divide the associated hardware from the application software. This open system also requires additional development to further assist the artificial intelligence and machine learning algorithms [14], [24]. Software updates over-the-air (OTA) are also important and highly recommended to be implemented even after the vehicle is sold to the customer to keep the operating systems up to date and bring the latest features to the consumer.

C. SENSORS IN INTELLIGENT VEHICLES

As vehicles are becoming more technologically advanced in order to achieve fully autonomous self-driven cars, intelligent vehicles are using various types of sensors to achieve autonomous vision. Therefore, fusing these sensors together is an excellent way to ensure great autonomous stoutness. Some of the main physical sensors that are used include:

- **High-resolution Camera:** A high-resolution camera is used to detect various different shapes that help in self-driven car technology. Through different stages of image processing, and through the camera, the system is able to detect lines in the road that help the vehicle stay on course, as well as properly yield to other cars, pedestrians, and any surrounding traffic signs. However, cameras alone are insufficient for detecting distances between the intelligent vehicle and the objects that surround it, be it another car, an obstacle, or a traffic sign. A great solution for this is to fuse it with a LiDAR or a RADAR sensor.
- **LIDAR:** Light Detection and Ranging (LiDAR) sensor that uses light in the form of a pulsed laser to map out the surroundings of the intelligent vehicle at the speed of light, namely 300 000 km/s. With the use of LiDAR, intelligent vehicles are able to easily detect distances between all the objects surrounding them.
- **Ultrasonic:** This sensor is also known as sonar. It is considered to be an electronic device that utilizes echolocation to identify if an object is within range of the sensor [25]. It can detect any object in its range by transmitting and receiving ultrasound waves. It also has the ability to measure the distance from the vehicle to a target object by utilizing the time taken by the signal to return back to the ultrasonic sensor after emitting it. However, ultrasonic sensors have a visually impaired zone created due to nearness and common obstruction which may cause incorrect readings. Furthermore, materials with sonic wave dampening abilities like acoustic foam have the tendency to compromise the readings from ultrasonic sensors [26].
- **RADAR:** Millimeter-wave RADAR technology is very commonly used in intelligent vehicles. The RADAR is designed to obtain distances as far as 250 meters, making adaptive cruise control and collision avoidance very reliable [27]. A major advantage of RADAR lies in its capability to penetrate nontransparent materials such as dust, smoke, snow and fog [14]. RADAR is be able to detect distances irrespective of the weather condition of the operating environment. However, one disadvantage of RADAR is the low side view it has which puts a limitation on its horizontal view [28], [29]. One way to solve this issue is by implementing a monocular camera which helps in improving accuracy and precision [14], [30].
- **Intelligent Vision Systems:** The Intelligent vision system is a combination of various sensors to achieve reliable driving assistance. This system consists of the monocular visual system and the stereo vision system [14]. These visual sensors are responsible for observing the driver's attention towards the road and the environment that the vehicle is operating in [31]. AI technology and machine learning are essential for adapting to the driver's environment and reacting accordingly.

III. SECURITY ISSUES AND VULNERABILITIES OF INTELLIGENT VEHICLES

With the advancement of car innovation, intelligent vehicles are getting progressively clever and are developing a number of creative applications performing different functionalities. These functionalities are controlled by 70 to 100 ECUs that communicate with each other through the in-vehicle communication buses [1]. While increasing the utilization of about 100 ECUs improves functionality and comfort, it also introduces a new cyberthreat by making vehicles a target for attackers. Additionally, with the advancement of remote communication innovations, vehicles can never be considered as closed frameworks, as they are dynamically equipped with functionality that interacts with the outside world [2]. Despite the fact that remote communication technology brings many improvements in terms of functionality and luxury, nevertheless, communications with the outside world exposes vulnerabilities that can be abused by an attacker and lead to infection of the vehicle. In this section, we discuss vulnerabilities associated with intelligent vehicles, as well as the potential ways an attacker could use to gain access to a vehicle and deliver malicious payloads.

A. THE VULNERABILITIES OF INTELLIGENT VEHICLES

With the advancement of automotive technologies, intelligent vehicle systems are controlled through I/O access channels of the embedded ECUs. These access channels present commands and output to the users of intelligent vehicles. However, these channels are vulnerable to attack due to their lack of security features such as authentication scheme, access control and verification process. These access channels can be categorized into four major categories: direct physical access, indirect physical access, short range wireless access and long range wireless access.

1) DIRECT PHYSICAL ACCESS (V1)

Automotive vehicles have many direct physical interfaces that can become potential surfaces for an attacker to infect an intelligent vehicle and have a malicious effect. These surfaces can provide direct access to the ECUs and in-vehicle network busses of an intelligent vehicle. Such an interface is the On-Board Diagnostics system (OBD) which is usually used by service professionals for performing diagnosis and ECU programming during periodic maintenance inspections. The OBD system can provide direct access to the vehicle's ECUs and its internal network busses through the OBD-II port and the OBD dongle [32].

2) INDIRECT PHYSICAL ACCESS (V2)

The ECUs and in-vehicle network busses of intelligent vehicles can be accessed through indirect physical interfaces without the presence of the attacker. These interfaces can be used by the user to indirectly pass commands or receive communication from the targeted ECUs. Most intelligent vehicles nowadays offer indirect physical access through the entertainment system using physical sources such as CD, disc, USB and iPod. However, these interfaces are vulnerable to attack due to their lack of security features [2].

3) SHORT RANGE WIRELESS ACCESS (V3)

Since car technology and network system has tremendously improved, vehicles are now exposed to the outside world through either the short range wireless access or the long range wireless access [33]. The short range wireless access provides many advantages over direct and indirect physical access as it would inflict many operational complexities, in targeting precise locations, and the inability to control the time of compromise. This type of communication method works mostly on short ranges to attack the surface of automotive wireless systems like Bluetooth, Remote keyless entry, Dedicated Short Range Communications (DSRC) and Wi-Fi. For these architectures, hackers can put a wireless transmitter close to the car's receiver, depending on the channel distance [2].

4) LONG RANGE WIRELESS ACCESS (V4)

The long-distance digital access channels, which are divided into two types: broadcast channels and addressable channels, have been deployed in intelligent vehicles now. The broadcast channels, such as GPS, Traffic Message Channel, Satellite Radio, and Digital Radio, are indirect channels that receivers tune into as part of a media system that is connected to other important ECUs. However, because it is difficult to attribute and command multiple channels at once, these channels are subject to external surface attacks, which might allow an attacker to manipulate channels and their behavior. The addressable Channels, as opposed to broadcast channels, are direct channels that frequently employ cellular phone and data networks and may be accessed over arbitrary distances. However, this type of long-range wireless is vulnerable to attack by the remote transfer system that provides continuous connectivity through cellular voice and data networks [2].

B. ENTRY POINTS INTO INTELLIGENT VEHICLES

With the evolution of vehicle technologies, in the wrong hands, these advanced technologies can lead to severe situations. To some degree, Intrusion Detection Systems (IDS) can block the potential ways and access channels that an attacker uses to gain access to a vehicle. Yet, no protection technique is absolutely efficient; a protection technique can be effective today however may not remain so for long, since hackers are continually updating the entry points, and looking for new ones. Therefore, in this section, we discuss the potential ways and entry points that an attacker might use to gain access to a vehicle in order to deliver a malicious effect. Furthermore, the attacker's presence in the vehicle, which specifies whether or not the attacker should be present in the vehicle during the compromise process. The scale which captures the approximate scale of the attack and the cost which represents the estimated effort involved in developing the attack capability. All of the aforementioned factors are presented for each entry point as shown in Table 2. Some of the potential entry points that hackers may attempt to gain access to a vehicle include:

The OBD-II Port: The OBD-II port system in a vehicle is responsible for tracking and modulating the vehicle's performance by monitoring the mileage, speed and other important data [34]. The OBD-II port reports data acquired from its sensors that are presented in the vehicle's infrastructure and it's connected to the check engine light that emits once a problem gets reported. However, the OBD-II port may be vulnerable to malicious attacks since it lacks an authentication method such as voices, facial features, retinas, irises, and fingerprints that can be used to authenticate a vehicle's owner identification. Furthermore, the OBD-II port also lacks an access control mechanism that assures that it is only accessible by the vehicle's owner. In other words, the OBD-II port may be accessed not only by the vehicle's owner, but also by other users and parties. This vulnerability may be exploited by unauthorized users and parties to get access to the vehicle and carry out malicious actions within it. For example, the OBD-II port can be attached to a laptop in order to interrogate the car's ECU program and this allows easy access to an attacker to alter or delete or inject a malicious code into the ECUs [2]. As a demonstration, by using an ECOM cable and

TABLE 2. The entry points to the intelligent vehicles.

Entry Point	Vulnerability Class	Attacker's Presence	Scale	Control	Cost	References
OBD-II port	V1	Yes	Small	Full Control	Low	[2], [32], [34], [35]
OBD Dongle	V1	Yes	Small	Full Control	Low	[2], [32]
	V3	No	Medium	Full Control	Low-Medium	[32], [36]
Entertainment System via CD/USB	V2	Yes	Small	Full Control	Low-Medium	[2], [12], [37]
Infotainment System via CD/USB	V2	Yes	Small	Full Control	Low-Medium	[38]–[40]
Telematics System via Cellular	V4	No	Large	Full Control	Medium-High	[2], [41], [42]
Sensors	V1	Yes	Small	Full Control	Low	[2], [43], [44]
	V2	Yes	Small	Full Control	Low-Medium	
	V3	No	Medium	Full Control	Low-Medium	
	V4	No	Large	Full Control	Medium-High	
In-Vehicle Network Busses	V1	Yes	Small	Full Control	Low	[2], [12] [35], [45], [46]
	V2	Yes	Small	Full Control	Low-Medium	
	V3	No	Medium	Full Control	Low-Medium	
	V4	No	Large	Full Control	Medium-High	
Bluetooth	V3	No	Medium	Full Control	Low-Medium	[2], [37], [47]
RKE	V3	No	Medium	Full Control	Low-Medium	[48], [49]
Wi-FI	V3	No	Medium	Full Control	Low-Medium	[50]–[52]
DSRC	V3	No	Medium	Full Control	Low-Medium	[21], [53]
Cellular	V4	No	Large	Full Control	Medium-High	[54]–[56]
In-vehicle Applications	V3	No	Medium	Full Control	Low-Medium	[15], [57], [58]
	V4	No	Large	Full Control	Medium-High	

handmade connections to attach to the OBD-II port, Valasek and Miller [35] were able to transmit and receive messages over the CAN bus.

The OBD Dongle: The OBD dongles are used to access the reported data from the OBD-II port. This OBD dongle also allows access to the CAN bus of the vehicle, which poses a security threat to the ECUs that are connected through the CAN interface. This allows attackers to easily get access to the CAN bus through the OBD port and send bogus messages to all the connected ECUs [2]. Although the fact that the OBD dongle is a physical connection to the OBD port, modern cars are implementing Wi-Fi technology to access the OBD port through a computer. This allows the hacker to do a variety of tasks on the vehicle, such as locking and unlocking doors, turning on and off vehicles using push button start/stop, steering adjustments, and braking, among other things [36].

The Entertainment System: The entertainment system in intelligent vehicles is an indirect physical access interface to the vehicle ECUs. Most of the intelligent vehicles nowadays are provided with a form of entertainment system that has a USB connectivity option, disk option, iPod, or CD player. These options enable the users to synchronize and access entertainment content from their mobile devices, navigation systems, USB devices, or from CD and play/view on the vehicle entertainment system [37]. In the advanced systems, the entertainment system is not standalone but also has a CAN connection to ECUs of other systems in the vehicle. These systems enable the synchronized mobile device to

access more features on the vehicle apart from the media system which creates a threat to the vehicle [2]. For example, Cai *et al.* [12] demonstrated that attackers can create a backdoor in the BMW vehicle entertainment system via the USB port.

The Infotainment System: The infotainment system supplies the vehicle with information and entertainment such as emails, text messages, voice calls, personal contacts, and many forms of information that can be gotten by interfacing with a cell phone such as stream music, and watch videos [38]. The infotainment system, on the other hand, may be hacked using simple tools like a CD or USB flash drive. Such a tool might be contaminated with malicious codes and infiltrate the car's infotainment system and spread to other systems, such as those that control the vehicle's engine and brakes systems. As a demonstration, a research group demonstrated an attack by altering an audio file to broadcast malicious CAN messages to compromise different in-vehicle systems When played on the vehicle's media player [2]. Furthermore, researchers were able to get a permanent connection to Mazda's infotainment system by running a bash script on the vehicle's Linux working system [39]. Another research group was able to access the address book, conversation history and even location data remotely by connecting the infotainment system's root account [40].

The Telematics System: The Telematics system supplements infotainment systems by giving information about in-vehicular systems such as vehicle speed, acceleration, tire pressure, fuel efficiency, oil life, door locking, seat belts, transmission issues and engine failures [41]. Furthermore, the telematics unit in the intelligent vehicles allows the vehicles to communicate with 3G, 4G, and now 5G networks. This allows attackers to get access to the vehicle through 3G, 4G and now 5G and do a variety of harmful actions on the vehicle. For example, researchers previously exploited a car's telematics unit remotely without user interaction [2]. They also were able by using reverse-engineering techniques to gain access to the operating system of the telematics ECU. Additionally, work by Jo *et al.* [42] investigated security risks in Android OS-based telematics frameworks that allow drivers to access and lock vehicle doors remotely, as well as start and stop the vehicle engine

Sensors: As vehicles are turning out to be more innovative to accomplish fully autonomous self-driven vehicles, intelligent vehicles nowadays are utilizing different kinds of sensors to accomplish the autonomous vision. Hence, combining those sensors is an extraordinary method to guarantee incredible autonomous strength. However, because there are no adequate security mechanisms in place to restrict the usage of sensors by installed apps, vehicles are exposed to sensor based threats and attacks. For example, the sensors in intelligent vehicles can be hacked easily either remotely or physically. As a demonstration, Petit *et al.* have shown the efficacy of relay and spoofing attacks against LiDAR [43]. Furthermore, Liu *et al.* used ultrasonic sensor attacks like jamming and spoofing to test Tesla, Audi, Volkswagen, and Ford. They demonstrated that all of the cars they examined could be jammed and spoofed [44].

In-Vehicle Network Busses: Controller Area Network (CAN) lacks sufficient communication protection. Since it is a broadcast-based communication protocol and there are no sender and receiver addresses, every node receives the frame and it is not secured by any Message Authentication Code (MAC) or digital signature [45], [46]. This creates a threat to confidential data that could be either stolen or manipulated by sending false and fake frames to each and every node which causes unintended behaviors. For example, an attacker can easily access the CAN bus and inject a malicious message in the CAN bus either directly through the OBD-II port or indirectly through the CD player, disc, USB and iPod [2], [35]. Another example is provided by Cai *et al.* [12] revealed multiple vulnerabilities in numerous BMW models, including the ability to compromise ECUs connected through CAN over a wireless connection.

Bluetooth: Bluetooth is currently available in most intelligent vehicles and has a range of up to 10 meters. It is commonly used to connect cell phones to the vehicle's infotainment and telematics system to make calls, check calendars, and listen to music streaming. The Bluetooth, on the other hand, does not need pairing with the target device or even being discoverable. Almost every Bluetooth-enabled device is at risk. This can be exploited by hackers to get access to the vehicle, giving them full control of the vehicle and the ability to carry out harmful operations within it. For example, an attacker can link his or her smartphone with the intelligent vehicle's Bluetooth. Then the attacker can send a malicious code to get uploaded into the system. This could be problematic and implementing confirmation Bluetooth connectivity on the infotainment system should be considered to make it harder for a hacker to connect via Bluetooth [2], [47].

Remote Keyless Entry (RKE): This type of communication uses radio frequency communication in order to control various functionalities of the intelligent vehicles remotely such as open doors, control lights, activate alarms, and even start and lock the ignition of the vehicle. The remote keyless entry, on the other hand, is open to attacks since it doesn't have a security mechanism such as cryptographic to protect the confidentiality of radio signal that will be transmitted from the vehicle's key. This vulnerability can be exploited by hackers to get access to the vehicle without possessing the key. The attack operates by eavesdropping the signal transmitted when a driver presses his or her key fob to open their vehicle. With $30 cost of equipment, the signal may be cloned, allowing the hacker to have access to the vehicle in the future. The attack can be within 100 meters of the car to clone the key's signal and the hacker can steal the car in less than two minutes [87]. For example, Liu *et al.* [48] demonstrated that many attacks can be infected to the Hitag2 cipher which is used in many remote keyless entry systems. Another example is by Dibaei *et al.* [49] showed that two hackers were able to steal a Mercedes-Benz vehicle by manipulating the keyless entry system.

Wi-Fi: The intelligent vehicles are currently equipped with Wi-Fi and consequently, they can connect to the internet via Wi-Fi hotspots on the roadway within the same range of the vehicle. However, some of these wireless hotspots might put the vehicle at risk for a variety of reasons. For instance, these wireless hotspots may employ outdated encryption standards, putting the vehicle security at risk. One of the initial encryption standards for wireless networking devices, the Wireless Encryption Protocol (WEP), is deemed weak and vulnerable to hacking. Wi-Fi protected access (WPA) was supposed to take the place of WEP as the wireless networking standard, but it, too, was proven to have flaws. Furthermore, these wireless hotspots may expose vehicles to a rogue or fake Wi-Fi hotspot [51]. For example, in the case of the vehicle connect to a malicious hotspot, this allows the hacker to operate many activities on the vehicle such as transfer malicious code to the vehicle. As a demonstration, Nie *et al.* [50] were able to remotely hack a Tesla vehicle by exploiting the way that the secret key to an installed Wi-Fi was saved in plain text. Furthermore, Nakhila *et al.* [51] showed that by connecting to an illegitimate Wi-Fi access point, an attacker may eavesdrop on Wi-Fi activity. Vanhoef *et al.* also looked at the possibility of Denial of Service attacks against Wi-Fi Protected Access [52].

DSRC: DSRC is an on-board vehicle unit that is developed to operate short-range communications between Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I). DSRC offers great autonomous technology services by allowing vehicles to exchange information either with each other or with the infrastructure such as roadside units that are surrounding the vehicle. DSRC utilizes radio frequency (RF) channels to achieve this communication. However, this concept could create an entry point for attacks to enter the DSRC system and cause serious damage by transmitting fake information. This can trick the vehicle's system and cause catastrophic consequences if the hacker was successful. Therefore, serious safety measures have to be taken into consideration to protect V2V and V2I communications [21], [53].

Cellular: The intelligent vehicles are currently equipped with cellular network technologies such as LTE,3G,4G and now 5G [20] and consequently, they can communicate to either another vehicle (V2V) or the infrastructure (V2I) at long distances on the scale of miles [54]. Cellular networks, on the other hand, are prone to eavesdropping and jamming attacks [56]. Cichonski *et al.* demonstrated that LTE can be hacked easily by jamming attacks and eavesdropping attacks [56]. Other work by Muhammad and Safdar [55] demonstrated that the LTE and 5G-based vehicular networks are vulnerable to a huge number of attacks. This allows attackers to track vehicle whereabouts in order to get access to the vehicle and carry out harmful operations inside of it. For instance, Miller and Valasek [11] have been able to hack and stop a Jeep Cherokee running on a highway remotely through 4G.

In-Vehicle Applications: The new development of the vehicle industry has implemented a new system in the Human-Machine Interface (HMI) screen that supports smartphone applications such as Google Android Auto and Apple Car Play. However, those vehicle applications can cause security threats and can create a path to inject malicious attacks into the HMI and obtain unaccredited access to vehicle functions. There is a chance where that automobile application can be infected with an attack on the phone itself, thus creating a potential threat to the vehicle's functions if those infected apps are being used by the vehicle's HMI. Those automobile applications support wireless mobile telecommunication technologies such as 3G, 4G, 5G as well as WiFi and Bluetooth to communicate with the vehicle which makes intelligent vehicles to be an open system that causes a potential threat [15]. For example, an attacker can penetrate the application itself and utilize this to get to a vehicle. Researchers discovered several vulnerabilities in seven popular applications that permit attackers to gain entry to vehicles [57]. Furthermore, Symantec researchers explored fake malicious applications that are created to look legitimate as the Uber application [58].

IV. AN OVERVIEW OF MALWARE AND HOW ITS SPREAD

In this section, we first present an overview of malware and common malware types. Second, we discuss the main motivations of an attacker to spread malware to the vehicle systems. Finally, we present the potential ways for Malware to infect the vehicle systems.

A. AN OVERVIEW OF MALWARE AND ITS COMMON TYPES

Malware is a malicious code that embeds itself into a software program that intentionally meets the harmful purposes of the malicious attackers who target any computing device [59]–[61]. Malware can enter any device through different channels such as files and directories from removable media, downloaded applications and files, and through email attachments. Once the malware reaches the device, the execution of the malware is easy by going through the interacting user authorization privileges or by bypassing the PC's authentication strategies to run without the device victim's permissions. Once it's executed on the device, it can harm the infected device by compromising its functions, disturbing its operations, stealing data or evading access controls, gathering personal sensitive information without the victim's permission. It also can obtain unauthorized access to a network system to create destructive damage to its subsystems. Malware can be categorized into many categories based on the way in which they cause harm and proliferate systems. This section provides an overview of the most common sorts of malware, including virus, worm, trojan, spyware, rootkit, backdoor, botnet, adware, scareware, and ransomware [62].

- **Virus:** It is a type of malicious software that can replicate itself into other programs and only attach themselves to other files, data, and computers when it is activated [63]. Viruses cannot cause much harm unless the infected transporter program is executed. The virus usually runs with user involvement [64] and it can

spread from one program to another and from one PC to another [65].

- **Worm:** It is a malicious program that may infect any machine, spreads over computer networks, and takes advantage of system flaws to further its malicious purposes. It utilizes networking protocols to inspect its local network and grows once it comes upon possible victim systems [66]. Worms can easily spread and execute within a system and also have the ability to replicate itself in a PC to tamper with important documents and the information on it [67]. It also has the ability to encrypt data and deliver spam messages. Worms, unlike viruses, have their own containers via which they spread [68].
- **Trojan:** It is sometimes called a Trojan horse. It is malicious software that can look legitimate with a useful purpose while in fact, it is executing whatever task the hacker intended. It can compromise computer security by gaining unauthorized access to the compelling PC and extract user confidential information such as credit card information and user credentials and it can cause much damage by executing unknown and unwanted activities [69].
- **Spyware:** It is a malicious program that is installed on any electronic device without the user's knowledge and it continuously spies on the user activities without the user's permission [70]. Spyware presents its danger only if the device is connected to the internet since It can be used to steal sensitive data like credit card information, government and medical records without one's knowledge. Spyware collects this information and sends it to the hacker, who can easily misuse the obtained data [71].
- **Rootkit:** It is a collection of malicious software designed to allow hackers to access and change operating systems and kernel data structures for harmful purposes [72]. Rootkits also give access to other types of malware to enter into a system and conceal their presence on the computer [73].
- **Backdoor:** It is one form of malware that gets the infected PC to be remotely accessed without the user's permission by opening a backdoor in the victim PC [74].
- **Botnet:** It is malicious software that allows attackers to remotely manipulate a group of infected and controlled devices such as cellphones, PCs, tablets, and internet of things devices. It happens without the users being aware that their PCs have been infected by botnet malware [75]. It is typically used for sending unruly commands and spamming computer systems and performing denial of service attacks [76].
- **Adware:** It is malicious advertising-supported software that brings advertisements to the computer. It can infect any system when a user tries to download free applications and software such as free playing games [77]. The main sole purpose of this malware is to scrutinize the user's activities while they are networking [78].
- **Scareware:** It is malicious software that is designed to mislead users into purchasing and downloading unneeded and potentially harmful software and programs, such as fake antivirus protection, which have posed serious financial and privacy risks to the victims [79].
- **Ransomware:** It is a malicious program that allows the attacker to either lock the victim's computer or encrypt the victim's data, aiming to deny service to the victim and restrict the victim access to his data in return for ransom. The malware then demands a ransom payment from the victim in order to restore access, and decrypt the victim's data on the infected computer [80].

B. MOTIVATIONS FOR INSTALLING MALWARE ON VEHICLE SYSTEMS

There are a various number of motivations behind attackers choosing to spread malware across vehicles. Here are some of the few motivations:

- **Financial Gain (M1):** An attacker can restrict the driver's access to his vehicle by infecting the vehicle remotely with ransomware which can disable the vehicle's functionalities such as immobilize the motor, locking the in-vehicle radio and locking the doors. Such an attack could restrict the vehicle's functionalities in a way that the proprietor's car keys can no longer activate them. The attackers would then be able to demand payoff before these functionalities were re-enabled. As a demonstration for academic research purposes only, work by Wolf *et al.* [81] showed that vehicle ransomware can be easily created and deployed. Additionally, researchers from McAfee security [82] demonstrated that the ransomware can block the use of the vehicle until the ransom is paid. Furthermore, fraud can be a major route for hackers to bring in cash. Hacked vehicles could give access to stalkers to be able to track the vehicle identification number of any potential victim through GPS since all intelligent vehicles nowadays have GPS. So in an event that an attacker can track any vehicle, the attacker can begin assistance for anybody that needs to track someone can in exchange for money. As a result, the attacker can gain a lot of money by tracking hundreds of vehicles. It's an extraordinary business. As an example of that, according to a report from Boston 25 News [83], an attacker was able to track a vehicle for many years by hiding a GPS tracking device on the victim's car. Another way of hackers to bring cash is automated toll booth payments, it may create more points of entry for hackers to steal individual information, for example, visa or banking data. Hackers are hoping to put forth the greatest benefit for the base attempt since the intelligent vehicles are going to have a lot of payment systems in order to provide the comfort for the driver to pay via his vehicle when he goes to toll roads and parking lots [84].

- **Infringement on the Driver's Privacy (M2):** An attacker could infringe the privacy of drivers by infusing spyware into a vehicle. An attacker could steal and access sensitive and private data about the driver for example, where he is located, his driving propensities, his credentials, his visa and banking information, his telephone number and call history, the music he tunes in to, and considerably more. According to an IBM Security report [85], a third party was able to gain access to the personal information of 27.7 million Texas drivers.
- **Vandalism (M3):** The malware can make a wide scope of disruptions to a driver. Malware might deactivate the brakes or force the car to abruptly slow down while driving, resulting in an accident. Furthermore, malware can be used to lock up infotainment systems in a vehicle to a random radio station, tampering with the tire pressure monitoring system's displays or false messages that force the driver to make important decisions while driving, such as changing the audio level or displaying arbitrary messages or images on the head unit display. Any such disturbance could make the driver commit dangerous errors while driving, cause auto collisions, and harm a carmaker's reputation. A team of hackers was able to hack a Tesla Model S remotely from a 12-mile distance [10] for academic research purposes. The authors of [81] demonstrated that ransomware can be easily deployed and disabled the vehicle's braking system. Furthermore, a research group [86] were able to disable the braking system of a 2009 Chevy Impala, which can harm both the passengers and their properties. Additionally, the authors of [11] were able to remotely hack and halt a Jeep Cherokee running on a highway, resulting in a 1.4 million car recall.
- **Hobby and Fun (M4):** Several hackers target just PCs or cellphones for the sake of amusement or to demonstrate their security expertise. It is foreseen that numerous hackers will see the increasing populace of vehicles as profoundly intriguing targets. Hacking vehicles could create more prominent exposure than hacking purchased PCs or smartphones. To prove the fact, Miller and Valasek [11] were able to demonstrate that for academic research purposes they were able to hack and stop a Jeep Cherokee running on a highway.
- **Theft (M5):** The malware attack may be used by an attacker to unlock a car's doors, disable its alarms, and disable its immobilizer in order to steal the vehicle. As proof of the concept, a research group [87] was able to prove that for academic research purposes the ability to steal a Tesla vehicle in few seconds by injecting the malware through the firmware update into the key fob via Bluetooth.
- **Facilitation Extraneous (M6):** Attackers most of the time use intermediaries and different frameworks to attack their final target. For this reason, it is important to note that a few associations and frameworks may essentially be advantageous focuses on that empower and encourage the attacker's activities. Consider botnets, systems are compromised to enable them to then attack other systems [88]. In this way, an attacker might hack a victim's vehicle system in order to track the victim aiming to attack the victim's home and such. Furthermore, hackers might be less intrigued by the victim's vehicle's systems and more intrigued by the vehicle's connected devices such as cell phones, laptops, and tablets which can give them admittance to charge card data, passwords, and monetary information, and considerably more. In the event that they're ready to get into the victim's vehicle's systems and locate the victim's connected devices, the victim's data might be in danger. For example, [85], demonstrated that millions of drivers' devices have been accessed by a third party.

C. THE POTENTIAL WAYS FOR MALWARE TO INFECT THE VEHICLE SYSTEMS

In addition to the potential ways presented in Table 2 for an attack to infect the vehicle systems. There are other numerous factors that influence the way malware can enter a vehicle and exploit any vehicle network interface, physical or wireless. Some of the factors are: F1) weaknesses in the design of the software. F2) weaknesses in the hardware. F3) weaknesses in the in-vehicle applications. F4) weaknesses in the in-vehicle network system. F5) The driver's inability to protect document downloads into the vehicle when the driver accesses websites and downloads apps from external sources. F6) External information may be laced with weaknesses that can enter a vehicle, for example, a software update bundle that can be infected with malware before it gets stacked onto a vehicle. F7) weaknesses in the operating systems utilized on the vehicles. There are various methods in which malware can abuse these weaknesses to infect a vehicle as shown in Table 3:

- **Direct Access:** An attacker can infect the vehicular system with malware by getting direct access to the vehicle. For example, Valasek and Miller were able to hack a Jeep Cherokee's infotainment system using the cellular network from a laptop. Upon scanning the network for other vehicles with high vulnerability, 2,695 more vehicles were discovered, which possessed similar vulnerabilities that exposed the jeep to be hacked [11]. Computerizing the attack with a laptop having all the programming steps, the same laptop could be used to hack other vehicles directly.
- **Updates Over The Air (OTA):** Intelligent vehicles as of now have millions of lines of code and the intricacy of in-vehicle programming keeps on developing. In this way, remote OTA ECU firmware update turns out to be progressively significant and expected, which increases the chances of malware infecting vehicles from remote locations [89]. For instance, the authors of [87] were able to hack a Tesla vehicle OTA and steal it in few seconds by injecting a malicious firmware update into the key fob.

- **Web Browsers:** Web browsers in intelligent vehicles allow drivers to access the internet and download information and other applications into their vehicles from application stores provided by the vehicle maker. This proves as a way for the malware to be downloaded into the vehicle. For example, It has been demonstrated that over 80% of the malware comes from well-known sites [90], [91].
- **Aftermarket Equipment:** Aftermarket infotainment systems use fastened or implanted devices to provide network connectivity and also support in hosting third-party applications. Many aftermarket equipments are used widely to replace original factory installed hardware, making it a huge threat to vehicles. For example, infotainment system head units, which are mostly Linux, Android, or Windows-based devices that can be readily hacked to run malicious software. It has been demonstrated [85] that a third party was able to access the details of millions of drivers in Texas.
- **Removable Media USB Flash Drive:** Most intelligent vehicles have USB connections to attach their newly acquired devices. These connections permit the installed system on a vehicle, for example, the infotainment system, to get to information documents like music records on the removable media. However, these removable media can be contaminated with malware, which can then infiltrate into the vehicle's embedded systems in a variety of methods. Such a method includes storing the malware on the removable media under a benign name such as firmware update in order to trick the vehicle's embedded system and consequently, introduce and run the malware when the removable media is connected. Another way is by adding malware to music records and consequently, run the malware when the music record is played. A research group demonstrated an attack by modifying audio file to transmit malicious CAN messages to compromise various in-vehicle systems when played on the vehicle's media player [2].
- **Operating Systems:** While practices vary by the automaker, the bulk of software running in intelligent vehicles is not written by the automakers and some of it comes from free open-source software, such as Linux and Android and most of the intelligent vehicles nowadays use LINUX or Android operating system [92], [93]. Although the LINUX systems are proved to be less affected by malware than other operating systems like windows, and android since they are owned by limited repositories and operated by trusted distributors. Nevertheless, it has been demonstrated that the LINUX systems are not immune to malware and LINUX malware has been on the rise [94], [95] and what's more, Linux apps and users can be tricked into permitting malware to enter and execute [96].
- **Spam and Advertising:** Although adding more services to vehicles brings comfort for the driver it likewise adds greater security risks. With the appearance of internet services in intelligent vehicles that permit Internet access from a browser, it is achievable to convey another kind of spam dependent on geographical location and travel. For example, as you approach a fast-food restaurant, imagine a pop-up discount. Not only is this type of behavior likely to be unpleasant, but it may also cause drivers to get distracted. Additionally, those kinds of spam and advertising are well-known infection vectors for malware that can convey the malware to infect the vehicle systems [77], [78].
- **Third-Party Applications:** Intelligent vehicles have been allowing third parties to create applications for extended services. For instance, an application on a smartphone can be used to open or close vehicle doors. These applications can harm vehicle systems as they are open-ended and is accessible to everyone, making them an easy target to hackers. Smartphone applications are an easy target for hackers when compared to ECU's as applications provide many resources and are more flexible and offer more resources. Vehicle applications are also susceptible since certain third parties employ shoddy security methods and credentials are frequently stored in cleartext [50]. These applications may also store individual data, for example, GPS information, vehicle models, and other data. This situation has just been shown by the OnStar application that permitted a hacker to open a vehicle remotely [97].
- **Vehicle-to-Vehicle Communications (V2V) technology:** V2V technologies establish communications in vehicles on the road using Wi-Fi connections. V2V technology acts as a security layer to the vehicle while on the road and also assists in decreasing vehicle speed when it is very close to another vehicle. This technology can also be used to speak with street sign device's vehicle to infrastructure (V2I) [98]. The data obtained can be used to improve the driving experience and also safety. The possibility of this technology being exploited by malware will result in many connected vehicles being affected in an adverse way [99], [100].
- **Mobile Device to Vehicle:** Intelligent vehicles nowadays have gotten typical to connect smartphones to the vehicle, usually by Bluetooth. This association permits hands-free calling while the driver is driving, playing sound from the driver's smartphone on the vehicle's speaker framework, and different comforts [37]. It is additionally a potential vector for malware [101]. A widespread smartphone virus or other smartphone malware probably won't influence the smartphone's behavior at all but could stand by quietly for the smartphone to pair with a vehicle, at that point transfer malware to the vehicle [37].
- **Supply Chain:** Vehicles built with parts from various manufacturers and suppliers might be having clashes and wrong intentions with each other. This might cause malicious software to embed into the creation cycle. This malware is inactive until an external stimulus, for

TABLE 3. Various methods for malware to infect the vehicle systems.

Infection Method	Weaknesses Factors	Attacker's Motivations	References
Direct Access	F1-F7	M1-M6	[11]
Updates OTA	F3,F6	M1-M6	[87]
Web Browsers	F3,F5	M1-M6	[90], [91]
Aftermarket Equipments	F2,F4,F7	M1-M6	[85]
Removable Media USB Flash Drive	F2,F3	M1-M6	[2]
Operating Systems	F1,F7	M1-M6	[92]–[96]
Spam and Advertising	F1,F3,F5,F6	M1-M6	[77], [78]
Third-Party Applicationss	F3,F5	M1-M6	[50], [97]
V2V technology	F3,F5,F6	M1-M6	[98]–[100]
Mobile to Vehicle	F3,F5,F6	M1-M6	[37], [101]
Supply Chain	F1-F7	M1-M6	[102]
Home Base	F1-F7	M1-M6	[103]
WiFi Hotspot	F3,F5	M1-M6	[50]–[52]
Software Bugs	F1,F3,F7	M1-M6	[104]–[106]

example, a signal arriving over the vehicle's internet connection, causes it to release its fatal impacts. As a demonstration, the supply chain attack of Shadow Hammer with ASUS systems led to 57,000 users having a backdoored version of the live update utility [102].

- **Home Base:** Intelligent vehicles exchange information with the maker's PCs, including software updates, which are a compelling method to get malware into vehicles. This implies the well-being of the fleet is just on par with the security of the producer's corporate servers. On the off chance that similar attacks effectively completed routinely against retailers, banks, and sites are utilized on vehicle manufacturers, it could place the maker's whole fleet in danger [103].
- **WiFi Hotspot:** Intelligent vehicles are currently equipped with WiFi and consequently interface with close-by hotspots with recognizable names. For instance, on the off chance that the vehicle previously connected with a hotspot with the name free WiFi, at that point, the vehicle will probably interface with any hotspot with a similar name automatically. A hacker can use this feature to setup a malicious hotspot with a common name and will have the option to get within a close range of the vehicle to connect to it automatically, at which point the hotspot can transfer malware to the vehicle. These attacks can also spread to other vehicles just by turning ON the Wi-Fi in the infected vehicles which might to the creation of additional malicious hotspots. Vehicles moving next to each other on roads can act as transfer agents of malware almost like biological viruses transmitted between humans [50]–[52].
- **Software Bugs:** The software bug is an instance of software failing to behave as it was designed, usually caused by mistakes made during the process of writing the software [104]. Bugs can cause software-based systems to be unreliable, commit errors, or give access and control to unauthorized parties [104]. The larger and more complex the body of code, the more bugs it is probably going to contain [104]. Today's intelligent vehicles can contain over a hundred million lines of code and the intricacy of in-vehicle programming keeps on developing. In this way, it will increase the software bugs in the vehicles that hackers can exploit to infect the malware on the vehicles [105], [106].

Once the malware infects any subsystem on a vehicle, for example, an infotainment system, it will have the option to harm other subsystems in the vehicle, as many subsystems are connected internally creating a cross-framework

functionality. Malware can transmit signals that cause a vehicle's regular operation to be disrupted. It may also launch denial-of-service (DoS) attacks by flooding various subsystems and in-vehicle networks with bogus messages in order to bring down various subsystems [107]. In some cases, malware may simply impact vehicle system performance and make over-burden processes or making unauthorized access to ECUs and harasses the passengers [91]. In the case of spying, the malware conceals itself in the system, steals sensitive information about the driver, and delivers it to the attackers [91]. Identifying malware is important as there is an increase in the damage to a large surface area and plenty of potential entry points could be taken by the hacker if the situation was not seriously taken.

V. EXISTING DEFENSE TECHNIQUES AGAINST MALWARE

In the last decade, researchers have explored a wide range of malware defense solutions for computer and mobile systems. Those solutions can be categorized into signature-based, behavior-based, heuristic-based, cloud-based, and machine learning-based techniques [108]–[113]. In this section, we present a detailed review of the main factors of applying these defense systems to protect intelligent vehicles against malware. These factors include the used approach, the used data analysis method, the targeted operating system, the detection time and the detection response, the data source, the main advantages and disadvantages of each defense system. Figure 4 shows the taxonomy dimensions distributed into six classes. We also briefly describe these classes below.

1) **Technique**. We classify the existing malware detection techniques into five categories, i.e. signature-based malware detection techniques, behavior-based malware detection techniques, heuristic-based malware detection techniques, cloud-based malware detection techniques, and machine learning-based malware detection techniques. Each of these techniques has certain advantages and disadvantages, we discuss the benefits and drawbacks of each technique.
2) **Analysis Methods**. The whole detection process is accomplished with static, dynamic and hybrid analysis methods. The description of each method is presented below.
   Static Analysis. It's a malware analysis method that analyzes an executable code without actually executing the code itself. In static analysis, the low-level information from codes is extracted by disassembling the codes by using any disassembler tools. The main advantage of this method is revealing the code structure of the program without executing it. However, this method may fail in analyzing unknown malware. It may also fail to detect malware that employs obfuscation and evasion techniques in its code [114].
   Dynamic Analysis. It's a malware analysis method that entails running the malware and monitoring its behavior, interactions with the host system, and its impacts on the host environment. The infected files in this method are analyzed in a simulated environment such as an emulator, virtual machine and sandbox in order to make the environment invisible to the malware [115]. Although this method is efficient in detecting malware, nevertheless, it may fail to detect malware that uses obfuscation code and evasion techniques.
   Hybrid Analysis. It's a malware analysis method that combines both dynamic and static analysis. It examines almost all of the static features of any malware code then combine them with other behavioral features to better the overall analysis process. Despite this method can overcome the limitations of both static and dynamic analysis methods. However, it may result in a rise in the execution time's total overhead [116].
3) **Target Operating System (OS)**. It refers to the operating system analyzed by the system. It can be LINUX, Windows, or Android [92], [93].
4) **Detection Time**. It refers to the time between the analyzed event and the detection itself. It can be real-time (online) detection, which enables an automatic response such as blocking the attacker and killing the malware process, or non-real-time (offline) detection [117].
5) **Detection Response**. The relevant outcome of the system, which can be a passive response which is an event notification such as printing an alert message, or an active response which is an automatic reaction such as blocking the attacker or killing the malware process [117].
6) **Data Source**. It refers to the source of the input data analyzed by the system. It can be host logs which are data from the operating system and system applications or application logs which are data directly generated by applications, or network traffic which are data generated by the network layer [117].

A. SIGNATURE-BASED MALWARE DETECTION

The signature-based malware detection process occurs in two sequential phases. First, after identifying the malware, a unique representation or signature for each malware must be created. This process is generally achieved by using a combination of manual and automated analysis of the data obtained from networks and user devices. Second, every device restores the malware signatures. It can then detect if a file or data stream is infected by malware or not by scanning the contents of malware signatures and uniquely identifying each malware [91]. The signature-based detection technique is the most often used in commercial antivirus tools which create different unique signatures using productivity by looking at the disassembled codes of the malware binary. The binary executable files are disintegrated using various disassemblers and debuggers [118]. The features of the disassembled code are extracted and analyzed further. Then, these features are used to create the malware family's signature. The signature-based detection is simpler, faster and

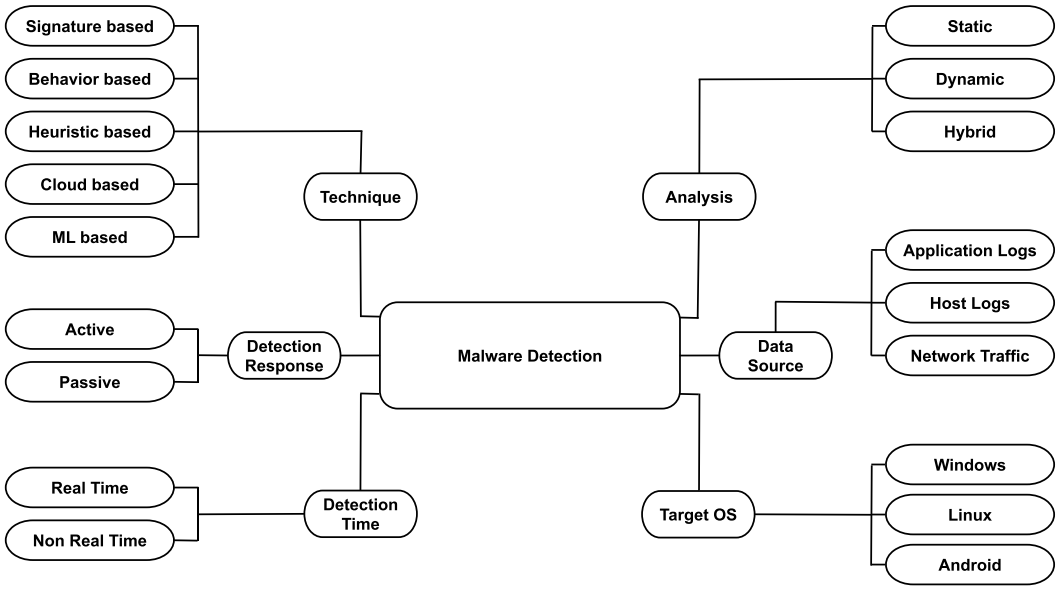


FIGURE 4. Malware detection system taxonomy.

safer to implement on intelligent vehicles when compared to other techniques. It's also efficient at detecting known malware. However, it is insufficient for detecting unknown malware and it is also subject to obfuscation and evasion techniques [119].

Researchers have proposed several approaches to detect malware based on the digital footprints of program files or applications like [120]–[135]. Table 4 shows a detailed comparison of the signature-based detection for several published articles in the last decade. These state-of-the-art approaches have used different log files (i.e., application logs, host logs, network traffic logs) to find the digital footprints. Most of the works can detect malware on windows operating system (OS). Researchers in [125], [130], [132], and [133] have demonstrated their work on android OS. Works by [124] and [135] remain the only two works that can detect Linux OS-based malware. Apart from the OS dependencies, the detection approaches differ in their way of analysis. Some researchers like [120], [121], [132], [133], [135] tried to detect malware by only considering the program bit file. That means detection has been done without executing the code, i.e., static analysis. For example, Shang *et al.* [120] proposed a novel malware detection method based on function call graph similarity. Other work by Shankarapani *et al.* [121] used API call sequences and assembly instructions to detect malware. The authors of [132], [133] have used control flow graph signatures to detect malware. Wan *et al.* [135] was able to detect malware based on using byte sequences of executable files. Although these approaches are efficient at detecting known malware and provide high accuracy, however, these approaches are insufficient for detecting unknown malware. Furthermore, these methods are incapable of detecting malware in real-time, making them unsuitable for use in intelligent cars.

Additionally, the researchers in [122]–[124], [126]–[130] have used the dynamic analysis (data acquired from running application) to detect malware. For instance, the authors of [122], [123] have used opcode sequences to detect malware. Similarly work in [129] and [130] have used Instruction sequences and application permissions in order to detect malware. Demme *et al.* [124] proposed a novel method to detect malware based on hardware performance counters. Despite these methods are effective at identifying known malware and have a high level of accuracy compared to static-based approaches, however, in addition to the high computational time required and hardware modifications needed, these methods are also insufficient for detecting new malware. Additionally, these approaches are incapable of identifying malware in real time, rendering them unsuitable for use in intelligent vehicles. Other works focus on a hybrid approach that performs both the static and the dynamic analysis [131], [134] in order to detect malware. For example, Fan *et al.* [131] used instruction sequences to detect malware. Similarly, work by Ojugo *et al.* [134] proposed a method to detect malware by using Boyer Moore string matching algorithm. These approaches could guarantee efficiency and accuracy higher than static and dynamic based approaches. However, these approaches are not capable of real-time malware detection, which makes them impractical for implementation in modern cars.

The signature-based detection technique is simpler and safer to implement compared to other detection techniques since it typically requires less processing power. However, it has numerous drawbacks when applied to defending intelligent cars. For example, signature-based detection is ineffective in detecting new malware (zero-day malware) for which no signatures have been generated. It's also vulnerable to obfuscation and evasion techniques [119]. Furthermore, the

TABLE 4. Survey of signature-based malware detection techniques.

Ref No	Approach	Analysis Method	Target OS	Detection Time	Detection Response	Data Source	Advantages	Disadvantages	Year
[120]	Function call graph similarity	Static	Windows	Non-real time	Passive	Applications logs	High detection rate	Invalid for implicit function-call	2010
[121]	Assembly instructions and API call sequences	Static	Windows	Non-real time	Passive	Applications logs	Low false positive rate & low detection time	Invalid to detect programs that use network functions or encryption algorithms	2011
[122]	Instruction opcodes frequency histogram	Dynamic	Windows	Non-real time	Passive	Applications logs	High accuracy & high precision	Small dataset & high computational time	2012
[123]	Opcode sequences	Dynamic	Windows	Non-real time	Passive	Applications logs	High accuracy & low false positive rate & low detection time	High number of selected features	2013
[124]	Hardware performance counters	Dynamic	Linux	Non-real time	Passive	Applications logs	High accuracy & low false positive rate	High cost & it needs hardware modifications	2013
[125]	Static & dynamic signatures extraction and encoding	Hybrid	Android	Non-real time	Passive	Applications logs	Variable size detectors & mutation mechanism	Low detection rate & high false positive rate	2014
[126]	N-grams-based	Dynamic	Windows	Non-real time	Passive	Applications logs	High detection rate	Didn't examine features selection	2015
[127]	Dynamic-Link Libraries (DLLs)-based	Dynamic	Windows	Non-real time	Passive	Applications logs	High accuracy & low execution time	Didn't examine features selection	2015
[128]	Network signatures and matching similarity	Dynamic	Windows	Non-real time	Passive	Network traffic	High accuracy	High computational time	2016
[129]	Instruction sequences and file property information	Dynamic	Windows	Non-real time	Passive	Host logs	High accuracy & low false positive rate	High number of selected features & dataset limitations	2016
[130]	Significant application permissions	Dynamic	Android	Non-real time	Passive	Applications logs	High detection rate	High number of selected permissions & high detection time	2016
[131]	Instruction sequences	Hybrid	Windows	Non-real time	Passive	Applications logs	High detection rate & low false positive rate	Didn't examine features selection	2016
[132]	Control flow graph signatures	Static	Android	Non-real time	Passive	Applications logs	High accuracy & low false positive rate	Low scalability	2017
[133]	Control flow graph signatures	Static	Android	Non-real time	Passive	Applications logs	High recall	Low scalability & high computational time	2018
[134]	Boyer Moore string matching algorithm	Hybrid	Windows	Non-real time	Passive	Host logs	High accuracy	High computational time	2019
[135]	Byte sequences of executable files	Static	Linux	Non-real time	Passive	Applications logs	High accuracy & low false positive rate	High cost & high overhead	2020

existing huge quantity of malware can result in an excessively big malware signature database for a resource-constrained in-vehicle device to store and analyze, which can increase considerably during a vehicle's lengthy lifespan [136]. A typical malware signature database now comprises over a million malicious signatures, resulting in tens of gigabytes of data [137]. As a result, when a car is manufactured, a huge malware signature database must be loaded. However, it will be difficult to anticipate how large a database should be put on a car when it is manufactured, so that it would be able to handle all potential new malware during the vehicle's long lifespan [136]. As a result, a vehicle's storage capacity may need to be increased over time. Additionally, as the number of malware signatures rises [138], the amount of processing power required to scan files for malware signatures will also increase. That is to say, the needed CPU capacity on a vehicle confronts the same problem as the required storage space for malware signatures. Furthermore, when new malware is detected and new malware signatures are created, the malware signature database on each vehicle must be updated on a regular basis. However, frequent malware signature updates to millions of vehicles will be difficult to handle and can be costly to vehicle owners.

B. BEHAVIOUR-BASED MALWARE DETECTION

The behavior-based malware detection technique is used to analyze the execution of a program in order to determine whether it is malicious or not [139]. This approach analyzes the execution of a program in a secure environment such as a virtual machine or a sandbox environment. This technique also uses monitoring tools in order to monitor and determine the behaviors of a program and decide if the program is malicious or benign based on its behaviors [140], [141]. This technique allows the vehicle to detect malware without relying on off-board systems, even with zero-day malware that has never been seen before [142]. The main purpose of this technique is to examine the behavior of any type of malware. Although the malware codes can be developed in different ways depend on the malware makers, however, the malware's behavior remains the same, consequently, the majority of new malware may be discovered using this technique [143]. This is the main advantage of this technique, however, some malware samples on the other hand do not run properly in a secured environment such as a virtual machine and sandbox environments. As a result, malware samples may be incorrectly classified as benign. Furthermore, this approach is insufficient for identifying all behaviors for a program and classifying them as malicious or benign. Additionally, the advanced code obfuscation and evasion techniques can simply prevent malware from being correctly evaluated [143].

Multiple bodies of work have adopted behavior-based malware detection technique as a solution against malware [144]–[159]. Table 5 presents a detailed comparison of the behavior-based detection solutions. These state-of-the-art approaches use the application's potential behavior in order to detect suspicious activities. Similar to the signature-based detection approaches, the majority of the presented solutions use the data file logs and have been demonstrated on Windows, Android, Linux OS. Another similarity between the behavior-based and signatures-based techniques is using the same data analysis methods (static, dynamic, and hybrid). For example, Sheen *et al.* [152] proposed a novel method for detecting malware based on static analysis of API calls and permissions. Similarly, the authors of [148], [149] have developed a method to detect malware based on hybrid analysis of API call sequences. Although the fact that these approaches have a high detection rate, nevertheless, cost efficiency, overhead, and detection time are the main drawbacks of these approaches. Because of these drawbacks, these approaches are unsuitable for intelligent vehicles.

In addition, multiple bodies of work examined the use of dynamic analysis for detecting malware [144]–[147], [150], [151], [153]–[159]. For instance, Nikolopoulos *et al.* [155] proposed a dynamic malware graph-based detection approach based on converting system calls to a temporal graph. Despite this approach provides a high detection rate, nevertheless, it has high time consumption and high complexity, which makes it unsuitable for use in intelligent vehicles. Other work by Marhusin *et al.* [157] proposed a malware n-grams-based detection method based on extraction of API sequences. This method has a low false-positive rate, on other hand, this method has high detection time and high complexity, which makes it unsuitable for use in modern cars. Similarly, the authors of [158], [159] proposed a dynamic malware detection approach based on analysis of API calls and permissions. Other work by Das *et al.* [154] proposed a dynamic hardware-based method for detecting malware based on system call patterns by using processor and field-programmable gate array (FPGA). In this method, the system calls first are extracted and the features are constructed. Then, the extracted features from the benign and malware samples are utilized to train the multilayer perceptron machine learning classifier. The evaluation results of this method showed that this method can detect malware in real-time and block their execution within the first 30% of their execution. Although this method [154] is the only solution that can detect malware in real-time and has an active detection reaction, while the remaining approaches are not capable of real-time detection. However, this solution [154] is highly complicated, not cost-effective, and not adaptable for intelligent vehicles since it requires hardware modifications to be made into the vehicle devices. As a result, the hardware changes that will be made to millions of vehicles will be difficult to handle and may be costly to vehicle owners and automakers as well.

The behavior-based detection technique has an advantage over the signature-based detection technique in detecting new malware generations (zero-day malware) that has never been seen before. The behavior-based detection technique, on other hand, is difficult and complex to implement compared to the signature-based detection technique since it typically requires higher processing power and more resources. Although the fact that the behavior-based detection technique

TABLE 5. Survey of behaviour-based malware detection techniques.

Ref No	Approach	Analysis Method	Target OS	Detection Time	Detection Response	Data Source	Advantages	Disadvantages	Year
[144]	Suspicious process behavior-based	Dynamic	Windows	Non-real time	Passive	Host logs	High simplicity	Low detection rate & small dataset	2010
[145]	MapReduce-based system calls	Dynamic	Windows	Non-real time	Passive	Applications logs	High detection rate & low false positive rate	Invalid to detect malware that doesn't use system calls	2011
[146]	Instruction traces	Dynamic	Windows	Non-real time	Passive	Applications logs	High accuracy & low false positive rate	High computational time & high complexity	2011
[147]	API Call-grams	Dynamic	Windows	Non-real time	Passive	Host logs	High accuracy & low false positive rate	High number of selected features	2012
[148]	API-based objective-oriented association	Hybrid	Windows	Non-real time	Passive	Applications logs	High detection rate	High computational time & high complexity	2013
[149]	API call sequences	Hybrid	Windows	Non-real time	Passive	Network traffic	High accuracy	High time consumption & high overhead	2013
[150]	System calls traces	Dynamic	Windows	Non-real time	Passive	Applications logs	High accuracy & low false positive rate	High complexity & high overhead	2013
[151]	N-grams based on the extraction of a distinct set of API sequences	Dynamic	Windows	Non-real time	Passive	Host logs	High accuracy	Didn't examine the selected features	2014
[152]	Permission & API call-based features	Static	Android	Non-real time	Passive	Applications logs	High scalability & low false positive rate	High complexity & high dimensional features	2015
[153]	Deep packet inspection & flow packet headers	Dynamic	Windows	Non-real time	Passive	Network traffic	High precision & low false positive rate	High complexity	2016
[154]	A hardware-enhanced architecture based on system calls patterns	Dynamic	Linux	Real time	Active	Applications logs	High detection rate & low false positive rate	High cost & high complexity & it needs hardware modifications	2016
[155]	Control flow graph-based using system-call groups	Dynamic	Android	Non-real time	Passive	Applications logs	High detection rate	High time consumption & high complexity	2016
[156]	System calls sequences	Dynamic	Android	Non-real time	Passive	Applications logs	High accuracy & low false positive rate	Didn't examine the selected features	2017
[157]	N-grams based on the extraction of API sequences	Dynamic	Windows	Non-real time	Passive	Host logs	High detection rate & low false positive rate	High detection time & high complexity & low scalability	2018
[158]	API call features & machine metrics	Dynamic	Windows	Non-real time	Passive	Host logs	High robustness & high simplicity	Low accuracy & high number of selected features	2019
[159]	Permission requests and API calls	Dynamic	Android	Non-real time	Passive	Applications logs	High F-measure	High number of selected features & permissions	2020

has the advantage of detecting most of new malware generations. However, it has a lot of drawbacks when applied to safeguarding intelligent vehicles. For instance, the behavior-based detection approach is insufficient for recognizing and categorizing all of a program's behaviors as malicious or benign. As a result, an abnormally high rate of false positives or false negatives may occur [144], [158]. Furthermore, complex code obfuscation and evasion techniques might simply prevent malware from being properly assessed [143]. Additionally, when compared the behavior-based detection technique to signature-based detection, the behavior-based detection approach is much more difficult to install and resource-intensive to execute on each vehicle. As a result, this technique might not be appropriate for resource-constrained in-vehicle devices that also require a lightweight solution. Furthermore, any behavior-based approach implemented on a vehicle today will almost certainly become obsolete over time and will need to be modified or replaced during the vehicle's long lifecycle [136].

C. HEURISTIC-BASED MALWARE DETECTION

The heuristic-based malware detection technique is used to examine program files for suspicious characteristics or emulate the execution of a program or chosen ports of the program to identify if it will perform malicious activities or not [160]. This technique is known for its complexity since it relies on previous experiences and other methods such as data mining, rule-based and machine learning to learn the characteristics of a program in order to assess whether it is malicious or not. It is also used by a lot of existing antivirus software [161]. It is also capable of detecting a wide range of known and unknown malware [162]. This methodology can also allow the vehicle to identify malware without relying on off-board systems, even with zero-day malware that has never been detected before [3]. Although this technique is capable of detecting a wide range of known and unknown malware with a high degree of accuracy, however, it fails to identify most new malware generations and sophisticated malware as well [160]. Furthermore, it is vulnerable to the advanced code obfuscation and evasion techniques that might simply prevent malware from being correctly detected [143].

Several researchers have proposed various heuristic-based malware detection techniques in the last decade [163]–[177]. A thorough comparison of heuristic-based detection solutions is included in Table 6. Some researchers like [165], [170], [172], [177] have relied on static analysis to detect malware. For example, the authors of [165], [172] have proposed a method for detecting malware based on control flow graphs and extracted opcodes from disassembled executable files. Work by Zaker *et al.* [170] used Dynamic Link Libraries (DLLs) to detect malware. Other recent work by Suryati *et al.* [177] relied on API calls network for detecting malware. These methods are effective at identifying known malware; however, they are insufficient for detecting unknown malware. These approaches are complex and prone to high false-positive rates. These methods are also incapable of identifying malware in real-time since they require high time for detecting malware, making them unsuitable for use in intelligent vehicles.

Additionally, researchers in [167], [174], [176] have relied on dynamic analysis for detecting malware. For instance, Shabtai *et al.* [167] proposed a dynamic method for detecting malware based on monitoring system opcode n-gram patterns. The authors of [174], [176] have proposed a dynamic graph-based method for detecting malware based on converting system calls to a graph. However, in addition to the high complexity and high computational time needed by these methods to detect malware, these methods are invalid to detect malware if malware can hide its malicious behaviors. They are therefore unfit for use in intelligent cars. Other researchers have used hybrid analysis for detecting malware [163], [164], [166], [168], [169], [171], [173], [175]. For example, the authors of [163], [164] have used API calls and opcode sequences to detect malware. The remaining works [166], [168], [169], [171], [173], [175] relied on the graph-based method, in which the classification is done on the basis of a graph. For instance, in [166], the authors have implemented a solution against malware based on opcode similarity, in case of malware attack, the commands are present in the code which should not be present in a normal set of code. Other work by Narayanan *et al.* [173] proposed a hybrid method for detecting malware through online learning. The online machine learning-based framework was used to learn the new malware features over time. This approach was able to detect both known and unknown malware in real-time. However, this method [173] has high complexity and requires high computational power, hence, is not feasible for intelligent vehicles due to the limited computing power of the ECUs to procedure such a complex process. Furthermore, the response time of this method, from data collection to detection, frequently results in a partially damaged vehicle system, putting drivers at risk.

The heuristic-based detection technique outperforms both signature-based detection and behavior-based detection techniques in detecting unknown malware. In contrast to signature-based detection and behavior-based detection techniques, the heuristic-based detection technique is more difficult and complex to execute since it generally needs more computing power and resources. Despite the fact that heuristic-based detection offers the benefit of detecting unknown malware. When it comes to protecting intelligent cars, however, it has a number of limitations. For example, this technique might fail to detect new malware generations, as well as sophisticated malware [160]. It's also vulnerable to complex code obfuscation and evasion techniques, which might prevent malware from being identified appropriately [143]. Additionally, this technique is known for its complexity because it depends on prior experiences and other approaches such as data mining and machine learning to learn the features of a program in order to determine whether it behaves maliciously or not [160]. As a result, this technique might not be suitable for resource-constrained in-vehicle

TABLE 6. Survey of heuristic-based malware detection techniques.

Ref No	Approach	Analysis Method	Target OS	Detection Time	Detection Response	Data Source	Advantages	Disadvantages	Year
[163]	A hierarchical associative classifier based on API calls' relationship	Hybrid	Windows	Non-real time	Passive	Host logs	High precision & high recall	High number of the selected feature & high computational time	2010
[164]	Opcode sequences frequency	Hybrid	Linux	Non-real time	Passive	Applications logs	High accuracy	High computational time & high complexity	2010
[165]	Control flow graph	Static	Windows	Non-real time	Passive	Host logs	High detection rate	High number of selected features & high computational time	2011
[166]	Opcode graph similarity	Hybrid	Windows	Non-real time	Passive	Host logs	High accuracy	High computational time & high complexity	2012
[167]	Opcode n-gram patterns	Dynamic	Windows	Non-real time	Passive	Host logs	High accuracy & low false positive rate	High number of selected features	2012
[168]	Hybrid features	Hybrid	Windows	Non-real time	Passive	Host logs	High accuracy & low false positive rate	High complexity & high computational time	2013
[169]	Frequent sub-graph mining	Hybrid	Windows	Non-real time	Passive	Host logs	High detection rate	High computational time & high complexity	2014
[170]	Dynamic-Link Libraries (DLLs)-based model	Static	Windows	Non-real time	Passive	Applications logs	High accuracy & low false positive rate	High number of the selected feature	2015
[171]	N-gram opcode features based	Hybrid	Android	Non-real time	Passive	Applications logs	High detection rate	High overhead	2016
[172]	Control flow graph using bi-normal separation	Static	Windows	Non-real time	Passive	Host logs	High accuracy & precision & recall	High number of selected features & small dataset	2016
[173]	Context-aware graph-based	Hybrid	Android	Real time	Passive	Applications logs	High scalability	High cost & high complexity	2017
[174]	System calls dependency graph-based	Dynamic	Windows	Non-real time	Passive	Applications logs	High accuracy & low detection time	Invalid if malware can hide its malicious behaviors	2018
[175]	Heterogeneous graph matching network based	Hybrid	Windows & Linux	Non-real time	Passive	Network traffic	High detection rate	High complexity & high overhead	2019
[176]	System calls graph signal-based	Dynamic	Android	Non-real time	Passive	Applications logs	High accuracy & low false positive rate	High computational time & high complexity	2020
[177]	API call network	Static	Windows	Non-real time	Passive	Network traffic	Improve security	High computational time & high complexity & low accuracy	2020

gadgets that also need to be light. Furthermore, any heuristic-based solution deployed on a vehicle today would almost definitely become obsolete over time, requiring modification or replacement at some point throughout the vehicle's lengthy lifespan [136].

D. CLOUD-BASED MALWARE DETECTION

Cloud computing has grown a lot in popularity in the last decade since it provides a lot of benefits, including easy access, on-demand storage, and reduced prices. Because the cloud became so popular in the last ten years, it has also been utilized recently to detect malware. The Cloud-based malware detection technique employs a variety of detection agents that are hosted on cloud servers and provides security as a service. Furthermore, a user can submit any type of file and obtain a report indicating whether the submitted file is malware or not [178]. The main advantage of the Cloud-based malware detection technique is that it can enhance the detection performance of PCs, mobile devices and vehicular systems with significantly huge malware databases and ponderous computing resources. Other advantages of this technique are Installations, configurations, setups are updated regularly. However, the cloud-based malware detection technique, on the other hand, has significant drawbacks. For example, the internet connection must constantly be fast and always available in order to work properly, but this is not always the case. Furthermore, in the cloud, real-time monitoring of all files is not possible. Additionally, this technique is vulnerable to obfuscation and evasion techniques [17].

Recently, several researchers have used cloud-based techniques to analyze and identify malware [178]–[194]. Table 7 shows a detailed comparison of cloud-based malware detection solutions. Researchers like in [178], [181], [189] have relied on static analysis to detect malware. For example, Ye *et al.* [178] have used file content and file relations features for detecting malware. Similarly, work by Li *et al.* [189] proposed a static method to detect malware based on n-gram string features. However, in addition to the high cost and high overhead of these methods, they are not up to the task of detecting unknown malware. These methods are also inappropriate for usage in intelligent cars since they are incapable of detecting malware in real-time because they need a long time to detect malware. In recent studies, dynamic analysis has been utilized in the cloud to detect malware [188], [190]–[194]. For instance, the authors of [188], [192] have proposed a dynamic method for detecting malware based on monitoring system calls. Similarly, work by Mishra *et al.* [194] proposed a dynamic method to detect malware based on n-gram features. The authors of [191], [193] have used hardware features and hardware performance counters to detect malware. However, in addition to the additional resources and sophisticated hardware changes that these approaches necessitate, they are unable to detect malware in real-time since they need a long time to identify malware. Unfortunately, because of these drawbacks, these approaches are unsuitable for intelligent vehicles.

In addition, several studies have looked into the use of hybrid analysis to detect malware [179], [180], [182]–[187]. For example, Jarabek *et al.* [180] have proposed a web-based method for detecting malware based on file scanning services. However, this method can't keep track of all files in the cloud in real-time. The authors of [182], [186] have proposed monitoring system parameters, such as API calls, file contents and permissions as features for detecting malware. However, these approaches might fail in detecting malware in the cloud if the malware can disguise its harmful activities. Other work by Yadav *et al.* [187] proposed a hybrid approach for detecting malware by utilizing fuzzy k-means and deep neural network in the cloud. However, this technique requires a large quantity of data for training, hence, this technique consumes enormous time for training, making it unsuitable for use in current intelligent cars.

The cloud-based malware detection technique has a number of advantages over conventional malware detection techniques, including quick access, on-demand storage, and lower pricing. The major benefit of using a cloud-based malware detection approach is that it may improve the detection performance of any system with large malware databases and a lot of processing power. Other benefits of this approach are installations and setups are all updated on a regular basis. However, it has a lot of drawbacks when it comes to protecting intelligent vehicles. For example, this technique is subject to sophisticated code obfuscation and evasion techniques, which may make malware difficult to detect in the cloud [17]. The other issue of this approach is real-time monitoring of all files in the cloud is not possible, making it impractical for implementation in intelligent vehicles. Additionally, this technique requires a reliable internet connection in order to work properly for security implementation, however, if for some reason the internet connection is lost, in that case, security can be compromised. As a result, this technique might not be safe enough for applying for intelligent vehicles. But with the advent of high-speed 5G technology [20], this technique might be safer to apply for intelligent vehicles.

E. MACHINE LEARNING-BASED MALWARE DETECTION

For many years, machine learning methods have been employed to identify malware [195]. Naive Bayes (NB), bayesian network (BN), logistic regression (LR), logistic model trees (LMT), C4.5 decision tree variant (J48), sequential minimal optimization (SMO), random forest tree (RF), multilayer perceptron (MLP), k-nearest neighbor (KNN), and support vector machine (SVM) are examples of well-known machine learning algorithms that have been used for many years in malware detection [195]. Although each algorithm has its own set of benefits and drawbacks, it is impossible to say that one is more effective than the other. However, one algorithm can outperform other algorithms in terms of the distribution of data, the amount of features, and the correlations between characteristics and attributes as well [195]. Deep Learning is a subfield of machine learning that evolved from artificial neural networks (ANN) that learn

TABLE 7. Survey of cloud-based malware detection techniques.

Ref No	Approach	Analysis Method	Target OS	Detection Time	Detection Response	Data Source	Advantages	Disadvantages	Year
[179]	Strings-based using web services and malware intrusion ontology	Hybrid	Windows & Linux	Non-real time	Passive	Host logs	High detection rate	High cost & high overhead	2010
[178]	File content and file relations features-based	Static	Windows	Non-real time	Passive	Host logs	High detection rate	High cost & high computational time	2011
[180]	Pre-existing web-based file scanning services	Hybrid	Android	Non-real time	Passive	Applications logs	High accuracy	High complexity & operating system needs to be modified	2012
[181]	Cloud application scan	Static	Android	Non-real time	Passive	Applications logs	High accuracy	High cost & high complexity	2013
[182]	API calls & permissions-based	Hybrid	Android	Non-real time	Passive	Applications logs	Improve security	Did not examine the accuracy and the overhead	2014
[183]	Reversible sketch-based	Hybrid	Windows & Linux	Non-real time	Passive	Applications logs	Improve security	High cost & high overhead & low detection rate	2015
[184]	Hybrid features-based	Hybrid	Android	Non-real time	Passive	Applications logs	High detection rate	High cost & high overhead	2016
[185]	Utilizing nash equilibrium property & Q-learning	Hybrid	Android	Non-real time	Passive	Applications logs	High detection rate	High response time & high overhead	2017
[186]	N-grams-based	Hybrid	Windows	Non-real time	Passive	Applications logs	High detection rate & low false positive rate	High response time & high cost & high complexity	2018
[187]	Utilizing fuzzy k-means & neural network	Hybrid	Windows	Non-real time	Passive	Applications logs	High detection rate & high precision	High computational time & high complexity & high overhead	2019
[188]	System calls-based	Dynamic	Windows & Linux	Non-real time	Passive	Applications logs	High accuracy & low false positive rate	High cost & high number of selected features	2020
[189]	N-gram string features	Static	Windows	Non-real time	Passive	Host logs	High accuracy & high F-1 score	High cost & high overhead	2021
[190]	Volatile memory dump features	Dynamic	Linux	Non-real time	Passive	Application logs	Detect unknown malware & high accuracy	High number of the selected feature & high computational time	2021
[191]	Hardware features	Dynamic	Linux	Non-real time	Passive	Application logs	Low false positive rate & high accuracy	High cost & it needs hardware modifications	2021
[192]	System calls-based	Dynamic	Windows	Non-real time	Passive	Application logs	High accuracy & detect unknown malware	High overhead & high computational time	2021
[193]	Hardware performance counters	Dynamic	Linux	Non-real time	Passive	Application logs	High accuracy & low false positive rate	High cost & it needs hardware modifications	2021
[194]	N-grams-based	Dynamic	Windows & Linux	Non-real time	Passive	Application logs	High accuracy	High overhead & high computational time	2021

from examples. It is a novel methodology that is extensively employed in image processing, voice control, intelligent vehicles, and recently in malware detection as well [196]. It seems highly effective and dramatically lowers feature space and is powerful to detect malware. However, it can be deceived by obfuscation and evasion attacks. Furthermore, building a hidden layer requires a lot of time, and adding more hidden layers seldom improves model performance [197].

In the last decade, researchers have proposed various machine learning-based malware detection techniques [198]–[221]. Table 8 and Table 9 show a detailed comparison of machine-based malware detection solutions. Some researchers have used machine learning for detecting malware based on static features [201], [204], [207], [212], [214], [217], [218], [220]. For example, the authors of [201], [204], [207], [212], [214], [218], [220] have used static features such as system calls, strings, byte sequences, DLLs, data flow, native opcodes and image features for detecting malware. However, these methods may fail to identify malware if the malware is able to hide its destructive activities and its contents. Furthermore, the time it takes for these methods to respond from data collection to detection usually results in a partially damaged system, making them unsuitable for use in intelligent vehicles. Other work by Sayadi *et al.* [217] proposed a novel method for detecting malware based on microarchitectural features. However, in addition to the high computational time and sophisticated hardware changes that are needed by this method to detect malware, this method is also incapable of identifying malware in real-time, making it inappropriate for intelligent cars.

Other researchers have relied on dynamic features for detecting malware [198], [202], [210], [215], [221]. For instance, the authors of [210], [215] have used dynamic features such as behavior features, API calls and opcode sequences for detecting malware. However, if malware is able to disguise its behaviors and contents, these approaches may fail to detect it. In addition, the time it takes these approaches to respond from data collection to detection generally results in a largely infected system, making them unsuitable for use in intelligent cars. Other work by Ghanei *et al.* [221] used hardware performance counters as features for detecting malware. However, in addition to the high detection time and complex hardware modifications required to detect malware, this approach is also incapable of detecting malware in real-time, making it unsuitable for modern cars. A large portion of existing machine learning-based malware detection techniques relied on hybrid features to detect malware [199], [200], [203], [205], [206], [208], [210], [211], [213], [215], [219]. For example, the authors of [199], [200], [203], [205], [206], [208], [210], [213], [215], [219] have used system calls, instructions, image features, API calls, data flow, network flow, API call sequences and permissions as features for detecting malware. However, these methods may be ineffective, if malware is able to conceal its harmful actions and contents, making them inappropriate for modern vehicles. Other work by Sayadi *et al.* [211] proposed a novel approach for detecting malware based on hardware performance counters. However, this approach is not adaptable for intelligent vehicles since it requires hardware modifications to be made into vehicle devices. As a result, the hardware modifications that will be required for millions of vehicles would be difficult to implement and might be costly to both vehicle owners and automakers.

The machine learning-based malware detection technique provides several advantages over traditional malware detection techniques, including the ability to detect both known and unknown malware, and improving the detection accuracy. However, it has a lot of limitations when applied to safeguarding intelligent vehicles. For instance, the machine learning-based malware detection technique can be deceived by complex code obfuscation and evasion techniques that make malware difficult to identify [197]. Furthermore, this technique needs an abundant amount of data for training. As a result, it takes a long time to train for this method, rendering it unsuitable for usage in today's intelligent vehicles. Additionally, most of the solutions that relied on this technique have been suggested and tested on datasets and are not suitable for real-time detection. The non-real-time detection approaches are inappropriate and ineffective for intelligent cars because if a vehicle is attacked with malware, the malware must be identified in real-time in order to ensure the safety of the driver and passengers.

F. INTRUSION DETECTION SYSTEM

The need for an efficient intrusion detection system (IDS) for modern vehicles is becoming one of the most essential security components as these vehicles are exposed to a huge number of threats. To this end, several IDSs to detect vehicle attacks have been explored in multiple bodies of work. For example, Lee *et al.* [107] and song *et al.* [222] proposed techniques for detecting an intrusion based on analysis of the CAN data time interval by monitoring the request time and response time of the CAN data traffic. Despite these techniques are lightweight, these techniques have limitations, especially when in-vehicle environments change frequently, as they require a lot of data updates. Müter *et al.* [223], [224] proposed IDS based on monitoring the state of the CAN bus traffic and the entropy of in-vehicle networks. Despite the fact that this technique does not need any hardware modifications, it is unable to detect irregular message incoming.

In addition, multiple bodies of work have adopted physical fingerprinting techniques for IDSs [228], [229], [235]. For instance, Avatefipour *et al.* [229] proposed a physical fingerprinting technique based on physical ECU features and the physical channel features to detect spoofing attacks. However, this technique can be failed when the channel length is increased which makes the physical ECU features are negligible. Other work by [228] proposed a clock-based intrusion detection system (CIDS) for fingerprinting each ECU based on using the clock skew characteristic of ECUs. Despite the efficiency of their technique, it is demonstrated that CIDS

TABLE 8. Survey of machine learning-based malware detection techniques: Part-1.

Ref No	Approach	ML ALG	Analysis Method	Target OS	Detection Time	Detection Response	Data Source	Advantages	Disadvantages	Year
[198]	Automated behavior-based	KNN, Naïve Bayes, J48, SVM, MLP	Dynamic	Windows	Non-real time	Passive	Host logs	High accuracy, recall and precision using J48	High number of selected features	2010
[199]	Image features-based	KNN	Hybrid	Windows	Non-real time	Passive	Host logs	High accuracy	High response time & high dimensional features	2011
[200]	Markov chain graph based on instructions and system calls	SVM	Hybrid	Windows	Non-real time	Passive	Host logs	High accuracy	High computational time & high cost & high complexity	2012
[201]	System calls-based using function call graph	SVM & KNN	Static	Windows	Non-real time	Passive	Host logs	High accuracy using KNN	High complexity & high computational time	2013
[202]	Random projections and neural networks	LR & one, two and three layers neural network	Dynamic	Windows & Linux	Non-real time	Passive	Applications logs	High accuracy using one layer neural network	High dimensional features & high computational time and power	2013
[203]	API calls-based	Deep Learning	Hybrid	Android	Non-real time	Passive	Applications logs	High accuracy using three layers & 150 neurons for each layer	High cost & high complexity & high dimensional features	2014
[204]	String, byte and DDLs features based	Deep Learning	Static	Windows	Non-real time	Passive	Applications logs	Low false positive rate & high accuracy	High number of selected features & high complexity	2015
[205]	API calls - based	MRMR –SVMS	Hybrid	Windows	Non-real time	Passive	Applications logs	High accuracy	High computational time & high overhead	2016
[206]	Available unlabeled data features-based	RF, J48, SVM	Hybrid	Windows	Non-real time	Passive	Applications logs	High accuracy using SVM & Random forest	High computational time & high complexity	2017
[207]	A data flow features-based	DNN, LR, SVM, MLP, Naïve Base	Static	Android	Non-real time	Passive	Applications logs	High accuracy using Deep Learning	High number of selected features & high complexity	2017
[208]	API Call Sequences-based	MDNB, MLP, BayesNet, J48, KNN, RF	Hybrid	Windows	Non-real time	Passive	Host logs	High accuracy using MDNB	High computational time & high complexity	2018

TABLE 9. Survey of machine learning-based malware detection techniques: Part-2.

Ref No	Approach	ML ALG	Analysis Method	Target OS	Detection Time	Detection Response	Data Source	Advantages	Disadvantages	Year
[209]	Network flow features-based	SVM, BayesNet, Adaboost, C4.5, Grading	Hybrid	Android	Real time	Active	Network traffic	High accuracy & only six network flow features were used	High cost & high complexity	2018
[210]	API calls	RF	Dynamic	Android	Non-real time	Passive	Applications logs	High accuracy	High cost & high complexity	2018
[211]	Hardware performance counters	J48, JRip, MLP	Hybrid	Linux	Non-real time	Passive	Applications logs	High detection rate	High cost & it needs hardware modifications	2019
[212]	Native opcodes	LR, RF, SVM, KNN, ANN	Static	Android	Non-real time	Passive	Applications logs	High accuracy	High overhead & high computational time	2019
[213]	API calls & permissions	RF & SVM	Hybrid	Android	Non-real time	Passive	Applications logs	High accuracy	High complexity & high computational time	2019
[214]	Control flow graphs	C4.5, DNN	Static	Android	Non-real time	Passive	Applications logs	High accuracy	Low scalability & high computational time	2019
[215]	API-pair graph-based	Their Model, CNN, DNN, RNN, LSTM	Hybrid	Windows	Non-real time	Passive	Host logs	High accuracy using their model	High cost & high complexity	2020
[216]	Opcode sequences-based	Their Model, SGD, KNN, LR, RF, SVM	Dynamic	Android	Non-real time	Passive	Applications logs	High accuracy using their model	High execution time & high overhead	2020
[217]	Micro-architectural features-based	LR, JRIP, J48, KNN	Static	Linux	Non-real time	Passive	Applications logs	High accuracy	High computational time & it needs hardware modifications	2020
[218]	Image features-based	CNN	Static	Android	Non-real time	Passive	Applications logs	High accuracy & low false positive rate	High computational time	2021
[219]	Image features-based	ANN	Hybrid	Windows	Non-real time	Passive	Applications logs	Improve security	Low accuracy & high false positive rate	2021
[220]	API calls & permissions	RNN	Static	Android	Non-real time	Passive	Applications logs	High accuracy	High computational time	2021
[221]	Hardware performance counters	CNN, DNN, LSTM	Dynamic	Windows	Non-real time	Passive	Applications logs	High accuracy	High computational time & it needs hardware modifications	2021

may be defeated by a spoofing attacker who can observe the clock skew and adjust his transmission accordingly [236].

Additionally, several message authentication techniques have been explored by researchers to safeguard vehicles against attacks [225]–[227], [237]. For example, Oguma *et al.* [237] proposed a novel security architecture by adding a master ECU to the network in order to verify other ECUs in the same way as a verification server does. Groza *et al.* [227] proposed a broadcast authentication technique based on time synchronization and key chains. Similarly, work by Lin *et al.* [226] proposed a message authentication technique by sending extra messages which prompts a higher burden on the CAN bus and hence a reduction of the available bandwidth of the CAN bus. Other work by Herrewege *et al.* [225] proposed a message authentication system for the CAN bus by adding the Hash-based Message Authentication Code (HMAC) field to the CAN data frame. Although these approaches improve security, they are inefficient and unsuitable solutions for vehicles since they need additional resources and sophisticated hardware modifications to be made in the CAN protocol.

Several methods were recently proposed to detect intrusions on the CAN bus based on machine learning techniques [113], [230]–[234], [238], [239]. For instance, Theissler [231] proposed a novel IDS to detect an anomaly on CAN bus based on multivariate time series. In order to identify both known and unknown fault types in various driving circumstances, an ensemble anomaly detector consisting of two-class and one-class classifiers was created. However, this method has drawbacks, particularly when the in-vehicle environment changes often; these drawbacks might include the constant requirement for calibration and data updates. Other work by Barletta *et al.* [233] proposed an IDS based on a combination of an unsupervised Kohonen Self-Organizing Map (SOM) network and k-means algorithm. The CAN IDs, timestamp, DLC and data field were used as features in order to identify attack messages sent on the CAN bus. Minawi *et al.* [232] also suggested an IDS that uses machine learning and includes crucial warning capabilities to safeguard vehicle operations. The key features utilized to evaluate whether the communication was benign or malicious were the CAN ID and the Data field. Furthermore, Martinelli *et al.* [230] suggested an IDS based on the eight data bytes of a CAN packet as the main features for determining whether a message is benign or malicious. Another study by Hossain *et al.* [234] presented an IDS using LSTM deep learning model-based. For an in-vehicle CAN bus network attack, the CAN ID, DLC, and data field were exploited as features. Hanselmann *et al.* [238] developed an IDS based on unsupervised neural network architecture to identify intrusions and abnormalities on the CAN bus, where the CAN IDs and timestamps were utilized as features. Additionally, the authors of [113], [239] proposed a graph-based IDS by converting the CAN bus messages into a temporal graph, then the machine learning techniques have been used to identify attack messages sent on the CAN bus. Although the aforementioned methods improve the vehicle's security, nevertheless, these methods are not feasible for a vehicular network due to the limited computing power of the ECUs to procedure a complex process.

Table 10 shows a detailed comparison of the IDS-based solutions. We observe that some of these solutions can detect any anomalies on CAN bus by using machine learning technology through different features such as CAN IDs, CAN bus data field, DLC, timestamp, entropy and graph features [107], [113], [224], [230]–[234]. The main benefits of these solutions are that they provide high accuracy and low false positive rates. However, in addition to the high complexity and high computational time required, these solutions lack the ability to detect critical attacks such as malware since these solutions rely on the data link layer and can't detect an attack such as malware which relies on the application layer. Other IDS approaches like [228], [229] can detect any intrusions on CAN bus by using physical fingerprinting technique. Although such approaches provide some degree of security, nevertheless, these approaches are unable to identify malware attacks that rely on the application layer because they rely on the physical layer. Other IDS methods such as [225]–[227] can detect any anomalies on in-vehicle network by adding a message authentication system field to the CAN bus data frame. Despite these methods provide high detection rate and improve the vehicle's security, however, in addition to the additional resources required and sophisticated hardware modifications needed, these methods lack the ability to detect malware attack since they rely on the data link layer and not rely on the application layer. In summary, the aforementioned IDS solutions can't detect malware attacks at application level and may can detect malware attacks at either the data link layer or the physical layer after the actual damage has likely been occurred. Therefore, in addition to the need for an efficient IDS for intelligent vehicles at data link and physical layers, an efficient malware defense system for modern cars at application layer is also needed.

VI. OPEN ISSUES AND FUTURE DIRECTIONS

In the previous section, we review malware detection approaches that have been proposed in the last decade based on the method used, the analysis method used, the target operating system, the detection and the response times, the data source, the main benefits and drawbacks of each method. In this section, we first discuss the limitations of applying these approaches in securing and protecting the intelligent vehicles against malware. Second, we discuss the security requirements that are needed in order to provide a successful and secure intelligent vehicle system. Finally, we summarize and discuss open research problems for the scientific community to address in order to meet the security requirements that are needed for a successful and secure intelligent vehicle system, and offer some recommendations for developing a more successful detection schema against malware for intelligent vehicles.

TABLE 10. Survey of intrusion detection systems.

Ref No	Approach	Features	Detection Time	Detection Response	Advantages	Disadvantages	Year
[224]	IDS based on monitoring CAN bus data traffic	Entropy	Non-real time	Passive	High accuracy	It can't detect irregular message incoming and malware	2011
[225]	Message authentication system for CAN bus	Hash-based Message Authentication Code (HMAC)	Non-real time	Passive	High detection rate and low false positive rate	High cost, high overhead, it can't detect malware and it is not adaptable with current vehicles since sophisticated hardware modifications are needed	2011
[226]	Message authentication technique for in-vehicle network	Extra messages	Non-real time	Passive	High detection rate and low false positive rate	High cost, high overhead, it can't detect malware and it utilizes CAN bus bandwidth	2012
[227]	Broadcast message authentication method for in-vehicle network	Time synchronization and key chains	Non-real time	Passive	High detection rate and low false positive rate	High cost, high overhead and it can't detect malware	2013
[228]	Clock-based intrusion detection system (CIDS)	ECU's clock skew	Non-real time	Passive	Low false positive rate	It can't detect spoofing attacks and malware	2016
[229]	IDS based on physical fingerprinting	Physical ECU and physical channel	Non-real time	Passive	High accuracy	It fails to detect spoofing attacks when the channel length is increased and it can't detect malware	2017
[107]	IDS based on analysis of the CAN data time interval	Timestamp	Non-real time	Passive	High detection rate	It can't detect malware and it requires a lot of data updates	2017
[230]	IDS based on machine learning technology	CAN bus data field	Non-real time	Passive	High accuracy	It can't detect malware and it requires high computational time	2017
[231]	IDS based on multivariate time	Timestamp	Non-real time	Passive	High accuracy and low false positive rate	High computational time, it can't detect malware and it requires data updates continuously	2017
[232]	IDS based on machine learning approach	CAN IDs and CAN bus data field	Non-real time	Passive	High accuracy	It can't detect malware and it requires high computational time	2020
[233]	IDS based on a combination of an unsupervised Kohonen SOM network and k-means algorithm	CAN IDs, CAN bus data field, CAN IDs and timestamp	Non-real time	Passive	High accuracy and low false positive rate	High cost, high complexity, high overhead and it can't detect malware	2020
[234]	IDS using LSTM deep learning model-based	CAN IDs, DLC and CAN data field	Non-real time	Passive	High accuracy	It can't detect malware and it requires high computational time	2020
[113]	Graph-based IDS by converting CAN bus messages into a temporal graph	Graph properties-based	Non-real time	Passive	High detection rate	High complexity, high overhead, it requires high computational time and it can't detect malware	2021

A. EXISTING TECHNIQUES LIMITATIONS IN SECURING INTELLIGENT VEHICLES AGAINST MALWARE

Despite the fact that malware detection techniques are improving day over day, the following limitations of applying these malware detection techniques to intelligent vehicles remain an unresolved issues.

- All present approaches [120]–[135], [144]–[159], [163]–[194], [198]–[221] are vulnerable to various types of obfuscation and evasion techniques as new malware generations utilize various sorts of obfuscation and evasion techniques to disguise themselves. For example, some kinds of malware employ throttled execution in order to evade detection [240], [241]. Malware can use this technique on vehicles to throttle its execution across multiple ECUs in order to evade detection. Other forms of malware take advantage of multi-core processors, as well as other capabilities like hyper-threading in order to spread malware activity across several cores to evade detection, as well as speed up execution to outrun any preventative measures taken by a victim or system administrator [242], [243]. Malware also can use this technique on vehicles to spread its activity across multiple ECUs' threads in order to evade detection. Other sorts of malware can add dummy instructions to their code to make it look different [244], or use instruction substitution to change their code by substituting equivalent instructions for some of them [245], or use code transposition to reorder the sequence of instructions in their code [246], or use subroutine reordering to obfuscate their code by randomly rearranging their subroutines [247]. Consequently, malware can evade detection and avoid itself from being properly analyzed by employing such techniques. As a result, these approaches [120]–[135], [144]–[159], [163]–[194], [198]–[221] are unsuitable for use in intelligent vehicles due to concerns about passengers safety.
- All of the current approaches [120]–[135], [144]–[159], [163]–[194], [198]–[221] might fail to detect new malware generations, as well as sophisticated malware. As a result, these approaches are inappropriate for use in intelligent vehicles due to concerns regarding driver safety and passengers as well. Furthermore, with the exception of cloud-based approaches, all approaches cannot be used for intelligent vehicles since they need to be updated regularly in order to handle any potential new malware during the vehicle's long lifespan [136]. Besides, updating them on a regular basis on millions of vehicles would be difficult to handle and can be costly for both vehicle owners and automakers. Cloud-based approaches have an edge over other approaches since all installations and configurations are updated on a regular basis in the cloud. Therefore, we believe cloud-based malware detection will be a feasible solution for safeguarding intelligent vehicles against malware attacks in the future especially with the advent of high speed 5G technology [20].
- Malware detection in real-time is really challenge. The majority of malware detection approaches in the last decade [120]–[135], [144]–[159], [163]–[194], [198]–[221] have been proposed and validated to detect malware using datasets and are not suitable for real-time detection. The issue with these non-real-time approaches is that they are unsuitable for intelligent vehicles because if the vehicle is infected with malware, the malware must be detected in real-time in order to ensure the safety of the drivers and passengers.
- There is no well-known and widely recognized dataset that can be used to assess the effectiveness of malware detection methods [120]–[135], [144]–[159], [163]–[194], [198]–[221]. Despite the fact that each malware detection technique has its own set of advantages and disadvantages, however, it is difficult to say that one is more effective than the other. This is due to the fact that each malware detection technique uses different malware and dataset.
- According to our findings, we observe that there are only two malware detection methods [154], [173] that can detect malware in real time. However, these methods [154], [173] need a lot of computational resources, which make them infeasible for intelligent vehicles due to the limited computational resources of the ECUs and CAN bus. Furthermore, these methods [154], [173] are not cost-efficient and are not adaptable for intelligent vehicles since they need a sophisticated hardware modifications. As a result, these methods may not be suitable for resource-constrained in-vehicle devices that also need to be lightweight.
- All present IDS approaches [107], [113], [224]–[234] cannot identify malware attacks at the application level, but they may detect malware attacks at the data link layer or physical layer after the actual damage has likely happened. As a result, in addition to the need for an effective IDS for intelligent vehicles at the data link and physical layers, modern cars also require an effective defense system at the application layer in order to safeguard them against malware.

B. SECURITY REQUIREMENTS TO SECURING INTELLIGENT VEHICLES

In this section, we discuss four essential requirements for securing intelligent vehicles. These are critical security criteria for every communication system. These requirements are authentication, integrity, privacy, and availability. Each requirement is presented below along with its description.

1) AUTHENTICATION

It means that the access to any information or vehicle's data must be given to the only authorized users and parties. By giving authorization to specific users and parties to access any information or vehicle's data, malware attacks and unauthorized manipulations can be prevented from happening. In this way, vehicle's network system can be more protected by only

giving authorization to a certain users and parties. The key management and distribution must be efficient and accurate in order to meet this requirement [248].

2) INTEGRITY

It is referred to the validity of data between the sender and the recipient of a communication system. The most basic criterion of communication system integrity is that the data received is correct and not tampered with intentionally. It is important to check the honesty of the message that is being sent in the vehicle's network system. The message has to get validated to make sure that it hasn't been manipulated or corrupted by a malware, or some other factors such as noise and fading. Error detection and correction codes must be developed to ensure the integrity of any communication system [248].

3) PRIVACY

Intelligent vehicles tend to share information with each other (such as Vehicle-to-Vehicle communication) and between the surrounding infrastructure (Vehicle-to-Infrastructure communication) [249]. Therefore, privacy plays a big factor in this role to protect vehicle's information from being used to do unauthorized behaviors such as using the information to spy on vehicles and access its private data [38].

4) AVAILABILITY

It is referred to the fact that authorized users have access to the systems and resources they need. Improving the chances of all targeted vehicles receiving information is critical in vehicular networks. Continuous availability is tough to accomplish under normal working settings, and it gets more and more challenging when updates and patches are required at various points. It is critical that network activities continue and that the cars remain unaffected. The availability of services at all times is critical. As a result, the needed redundancy for this purpose must be appropriately implemented [250].

C. RECOMMENDATIONS AND FUTURE DIRECTIONS

One of the biggest challenges that automakers face is finding solutions against malware attacks and creating a full immunity system to combat this threat. Although the existing defenses are some of the most effective approaches of building structural defenses against malware attacks, there are still some challenges and issues that need further investigation and study. There are additional potential solutions that could be implemented to provide a great protection and immunity against malware attacks. Some additional potential solutions and directions that will enhance intelligent vehicles' security that need to be addressed to meet the security requirements to securing intelligent vehicles are presented below.

1) AUTHENTICATION SYSTEM USING Li-Fi TECHNOLOGY

A lightweight cryptographic authentication system if implemented would boost security in intelligent vehicles. This would provide a secure, efficient and flexible method that is able to handle complicated transportation circumstances [251]. The main idea of creating a lightweight cryptographic authentication system has been in key extraction, key establishment and key distribution. Major milestones have been achieved in protocols such as key extraction using wireless fading channels [252], key establishment using keyless cryptography technology [253] and key distribution using the Light fidelity (Li-Fi) [254]. It has been proven that Li-Fi technology can accomplish high-speed wireless communication of over 3 Gb/s compared to Wi-Fi. Furthermore, Li-Fi technology further provides security by avoiding interception and eavesdropping. For these reasons, there has been increased interest in integrating Li-Fi technology in intelligent vehicles design to be used for authentication system in intelligent vehicles [254]. Alongside with implementing authentication system, security criteria must be met in order to provide a successful and secure protection to the vehicle's system.

2) FIREWALL SYSTEM

Although malware attacks can be destructive to intelligent vehicles with its different entry points, there are many ways that can be implemented to defend against malware attacks. Intelligent vehicle's system tends to receive updates more often. Therefore, the liability of the source that is sending that information must be checked to make sure malware doesn't get injected in the intelligent vehicle's network. A network security device such as firewall should be implemented to monitor and block unwanted data [255]. The firewall's main purpose is to filter any data that enters the system and rejects malware attack vectors that have been recognized as a threat. Alongside with applying a network security device, security requirements need to be satisfied in order to provide a successful and secure protection to the vehicle's system.

3) DEEP LEARNING USING OFFLOADING COMPUTATION MECHANISM

Intelligent deep learning such as neural networks technology is a great way to detect vulnerabilities and eliminate malware attacks in intelligent vehicle systems. Because the fact that this technology is more accurate and performs better than machine learning technology in malware detection, it is worth considering this advanced technological approach for intelligent vehicle systems [256]. Deep learning, on the other hand, requires a lot of computing resources and capabilities in the vehicle's ECUs, which leads to memory overloading for deep learning implementation in ECUs owing to the vehicle's ECUs' limited computation resources. However, the offloading computation mechanism was found to be a possible solution to solve the limited computation resources of the vehicle's ECUs by transferring the resource intensive computational tasks to a separate processor such as an external platform, a hardware accelerator, a cluster, grid, or cloud server at the network edge [257]. The future of intelligent vehicles is quite promising with deep learning using offloading computation mechanism towards faster and secure vehicle system.

4) SOFTWARE DEFINED SECURITY

Intelligent vehicles need to be able to detect malware attacks efficiently and effectively. Therefore, the software defined security system can be a reliable solution to detect and eliminate malware threats and further improves network security for intelligent vehicles by forwarding the security threats characteristics and traffic parameters for forensic analysis. The software defined security is referred to the use of software defined platforms to automate threat detection and mitigation. This can be accomplished by adopting an open flow protocol, Network Function Virtualization (NFV) and Software-Defined Networking (SDN) that uses multi-layered open virtual switch with programmatic extension principle that allows automation of threat detection and elimination on a bigger scale [258]. This form of dynamic solution to threats will provide security for intelligent vehicles against malware attacks.

5) CLOUD-BASED SOLUTION USING 5G TECHNOLOGY

It is another potential future route for intelligent vehicles since it offers several advantages, such as simple access, on-demand storage, and lower pricing. Furthermore, installations, settings, and setups are all updated on a regular basis with this method. It also can improve the malware detection performance of the intelligent vehicle's system with large malware datasets and ponderous computing resources. It also can fix the resources allocation issues of intelligent vehicle's system by storing the data acquired at each ECU in cloud, the training and testing can be performed also on cloud to see whether the data is authentic or not. This solution of sending data to the cloud would have been impractical few years ago since the internet connection was not fast and always available, but with the advent of high speed 5G [20], it is now practical to store data in cloud. The future of intelligent vehicles looks bright, thanks to cloud solutions that leverage 5G technology to create a quicker and more secure vehicle system.

VII. CONCLUSION

In this paper, we first present a great depth description of the architecture of intelligent vehicles. We also identify the security issues and vulnerabilities of intelligent vehicles in order to illustrate the lack of protection against malware attacks. Furthermore, this paper discusses the most common types of malware that might infiltrate intelligent vehicles to show how each type of malware could be different than another. Additionally, different entry points for malware to infect intelligent vehicles were covered in this paper to emphasize the importance of protecting those aspects. A comprehensive survey of malware detection techniques is also discussed and further categorized into five categories, i.e. signature-based malware detection techniques, behavior-based malware detection techniques, heuristic-based malware detection techniques, cloud-based malware detection techniques, and machine learning-based malware detection techniques. Each of these techniques has certain advantages and disadvantages, we discussed the advantages and disadvantages of each technique. Finally, a future direction is provided to further improve the immunity for the system of intelligent vehicles to protect it against malware attacks.

REFERENCES

[1] R. N. Charette, "This car runs on code," *IEEE Spectr.*, vol. 46, no. 3, p. 3, Feb. 2009.

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohn, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. USENIX Secur. Symp.*, vol. 4, San Francisco, CA, USA, 2011, pp. 447–462.

[3] L. Delgrossi and T. Zhang, *Vehicle Safety Communications: Protocols, Security, and Privacy*, vol. 103. U.K.: J. Inf. Secur. Appl., 2012.

[4] M. Ring, D. Frkat, and M. Schmiedecker, "Cybersecurity evaluation of automotive E/E architectures," in *Proc. ACM Comput. Sci. Cars Symp. (CSCS)*, 2018, pp. 1–7.

[5] L. O'Carroll, "Scientist banned from revealing codes used to start luxury cars," *Guardian*, vol. 27, Jul. 2013. [Online]. Available: http://www.theguardian.com/technology/2013/jul/26/scientist-banned-revealing-codes-cars

[6] A. Francillon, B. Danev, and S. Capkun, "Relay attacks on passive keyless entry and start systems in modern cars," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*. Zürich, Switzerland: Eidgenössische Technische Hochschule Zürich, Department of Computer Science, 2011, pp. 1–16.

[7] I. Rouf, R. D. Miller, H. A. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study," in *Proc. USENIX Secur. Symp.*, vol. 10, 2010, pp. 1–16.

[8] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaaniche, and Y. Laarouchi, "Survey on security threats and protection mechanisms in embedded automotive networks," in *Proc. 43rd Annu. IEEE/IFIP Conf. Dependable Syst. Netw. Workshop (DSN-W)*, Jun. 2013, pp. 1–12.

[9] A. Greenberg, "Hackers reveal nasty new car attacks–with me behind the wheel," (Video), Forbes, Mumbai, India, Tech. Rep., Jul. 2013. [Online]. Available: https://www.forbes.com/sites/andygreenberg/2013/07/24/hackers-reveal-nasty-newcar-attacks-with-me-behind-the-wheel video/#6677f02228c7

[10] O. Solon, "Team of hackers take remote control of Tesla models from 12 miles away," *Guardian*, vol. 20, Sep. 2016. [Online]. Available: https://www.theguardian.com

[11] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, p. 91, Aug. 2015.

[12] Z. Cai, A. Wang, W. Zhang, M. Gruffke, and H. Schweppe, "0-days & mitigations: Roadways to exploit and secure connected BMW cars," *Black Hat USA*, vol. 2019, p. 39, Aug. 2019.

[13] M. Dunn, "Toyota's killer firmware: Bad design and its consequences," *EDN Netw.*, vol. 28, Oct. 2013. [Online]. Available: http://www.edn.com/design/automotive/4423428/Toyota-s-killer-firmware-Bad-design-and-its-consequences

[14] M. Dibaei, X. Zheng, K. Jiang, R. Abbas, S. Liu, Y. Zhang, Y. Xiang, and S. Yu, "Attacks and defences on intelligent connected vehicles: A survey," *Digit. Commun. Netw.*, vol. 6, no. 4, pp. 399–421, Nov. 2020.

[15] M. H. Eiza and Q. Ni, "Driving with sharks: Rethinking connected vehicles with vehicle cybersecurity," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 45–51, Jun. 2017.

[16] *CAN-Bus Specifications Report*, Robert Bosch, Stuttgart, Germany, 1983.

[17] T. Zhang, H. Antunes, and S. Aggarwal, "Defending connected vehicles against malware: Challenges and a solution framework," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 10–21, Feb. 2014.

[18] M. Sauerwald, "Can bus Ethernet or FPD-link: Which is best for automotive communications," *Analog Appl. J.*, vol. 1Q, pp. 20–22, 2014. Accessed: Feb. 4, 2016. [Online]. Available: http://www.ti.com/lit/an/slyt560/slyt560.pdf

[19] D. G. Yang, K. Jiang, D. Zhao, C. Yu, Z. Cao, S. Xie, Z. Xiao, X. Jiao, S. Wang, and K. Zhang, "Intelligent and connected vehicles: Current status and future perspectives," *Sci. China Technol. Sci.*, vol. 61, no. 10, pp. 1446–1471, Oct. 2018.

[20] M. H. Eiza, Q. Ni, and Q. Shi, "Secure and privacy-aware cloud-assisted video reporting service in 5G-enabled vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 7868–7881, Oct. 2016.

[21] X. Wu, S. Subramanian, R. Guha, R. G. White, J. Li, K. W. Lu, A. Bucceri, and T. Zhang, "Vehicular communications using DSRC: Challenges, enhancements, and evolution," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 9, pp. 399–408, Sep. 2013.

[22] S. Corrigan, "Introduction to the controller area network (CAN)," USA, Appl. Rep. SLOA101, Aug. 2002, pp. 1–17.
[23] S. Asano, T. Maruyama, and Y. Yamaguchi, "Performance comparison of FPGA, GPU and CPU in image processing," in *Proc. Int. Conf. Field Program. Log. Appl.*, Aug. 2009, pp. 126–131.
[24] S. Bunzel, "AUTOSAR—The standardized software architecture," *Informatik-Spektrum*, vol. 34, no. 1, pp. 79–83, Feb. 2011.
[25] S. Adarsh, S. M. Kaleemuddin, D. Bose, and K. I. Ramachandran, "Performance comparison of infrared and ultrasonic sensors for obstacles of different materials in vehicle/robot navigation applications," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 149, Sep. 2016, Art. no. 012141.
[26] B. S. Lim, S. L. Keoh, and V. L. L. Thing, "Autonomous vehicle ultrasonic sensor vulnerability and impact assessment," in *Proc. IEEE 4th World Forum Internet Things (WF-IoT)*, Feb. 2018, pp. 231–236.
[27] W. Menzel and A. Moebius, "Antenna concepts for millimeter-wave automotive radar sensors," *Proc. IEEE*, vol. 100, no. 7, pp. 2372–2379, Jul. 2012.
[28] D. G. Johnson, "Development of a high resolution MMW radar employing an antenna with combined frequency and mechanical scanning," in *Proc. IEEE Radar Conf.*, May 2008, pp. 1–5.
[29] X. Wang, L. Xu, H. Sun, J. Xin, and N. Zheng, "Bionic vision inspired on-road obstacle detection and tracking using radar and visual information," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 39–44.
[30] Y. Fang, I. Masaki, and B. Horn, "Depth-based target segmentation for intelligent vehicles: Fusion of radar and binocular stereo," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 3, pp. 196–202, Sep. 2002.
[31] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives," *Robot. Auto. Syst.*, vol. 32, no. 1, pp. 1–16, 2000.
[32] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy*, 2010, pp. 447–462.
[33] F. Sommer, J. Dürrwang, and R. Kriesten, "Survey and classification of automotive security attacks," *Information*, vol. 10, no. 4, p. 148, Apr. 2019.
[34] S. Bharati, P. Podder, M. R. H. Mondal, and M. R. Alam Robel, "Threats and countermeasures of cyber security in direct and remote vehicle communication systems," 2020, *arXiv:2006.08723*.
[35] C. Valasek and C. Miller, "Adventures in automotive networks and control units," in *Proc. DEF CON*, 2013, pp. 260–264.
[36] D. Klinedinst and C. King, "On board diagnostics: Risks and vulnerabilities of the connected vehicle," Softw. Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep., 2016, vol. 10.
[37] H. Onishi, K. Wu, K. Yoshida, and T. Kato, "Approaches for vehicle cyber-security in the US," *Int. J. Automot. Eng.*, vol. 8, no. 1, pp. 1–6, 2017.
[38] C. Bernardini, M. R. Asghar, and B. Crispo, "Security and privacy in vehicular communications: Challenges and opportunities," *Veh. Commun.*, vol. 10, pp. 13–28, Oct. 2017.
[39] (2018). *Bleepingcomputer Report*. [Online]. Available: https://www.bleepingcomputer.com/news/security/volkswagen-and-audi-cars-vulnerable-to-remote-hacking
[40] (2016). *Latimes Report*. [Online]. Available: https://www.latimes.com/business/la-fi-hy-mystery-car-stealing-device-20161207-story.html/
[41] K. Jaisingh, K. El-Khatib, and R. Akalu, "Paving the way for intelligent transport systems (ITS): Privacy implications of vehicle infotainment and telematics systems," in *Proc. 6th ACM Symp. Develop. Anal. Intell. Veh. Netw. Appl.*, Nov. 2016, pp. 25–31.
[42] H. J. Jo, W. Choi, S. Y. Na, S. Woo, and D. H. Lee, "Vulnerabilities of Android OS-based telematics system," *Wireless Pers. Commun.*, vol. 92, no. 4, pp. 1511–1530, 2017.
[43] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and LiDAR," in *Proc. Black Hat Eur.*, vol. 11, 2015, p. 2015.
[44] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," *Defcon*, vol. 24, no. 8, p. 109, 2016.
[45] P. Kleberger, T. Olovsson, and E. Jonsson, "Security aspects of the in-vehicle network in the connected car," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 528–533.
[46] D. K. Nilsson and U. E. Larson, "Simulated attacks on can buses: Vehicle virus," in *Proc. IASTED Int. Conf. Commun. Syst. Netw. (AsiaCSN)*, 2008, pp. 66–72.
[47] X. Li, Y. Yu, G. Sun, and K. Chen, "Connected vehicles' security from the perspective of the in-vehicle network," *IEEE Netw.*, vol. 32, no. 3, pp. 58–63, May 2018.
[48] H.-L. Liu, J.-S. Ma, S.-Y. Zhu, Z.-J. Lu, and Z.-L. Liu, "Practical contactless attacks on Hitag2-based immobilizer and RKE systems," in *Proc. Int. Conf. Comput., Commun. Netw. Technol.*, 2018, pp. 505–512.
[49] M. Dibaei, X. Zheng, K. Jiang, S. Maric, R. Abbas, S. Liu, Y. Zhang, Y. Deng, S. Wen, J. Zhang, Y. Xiang, and S. Yu, "An overview of attacks and defences on intelligent connected vehicles," 2019, *arXiv:1907.07455*.
[50] S. Nie, L. Liu, and Y. Du, "Free-fall: Hacking Tesla from wireless to CAN bus," *Briefing, Black Hat USA*, vol. 25, pp. 1–16, Jul. 2017.
[51] O. Nakhila, E. Dondyk, M. F. Amjad, and C. Zou, "User-side Wi-Fi evil twin attack detection using SSL/TCP protocols," in *Proc. 12th Annu. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2015, pp. 239–244.
[52] M. Vanhoef and F. Piessens, "Denial-of-service attacks against the 4-way Wi-Fi handshake," in *Proc. 9th Int. Conf. Netw. Commun. Secur.* Chennai, India: Academy & Industry Research Collaboration Center, 2017, pp. 1–10.
[53] W. Whyte, J. Petit, V. Kumar, J. Moring, and R. Roy, "Threat and countermeasures analysis for WAVE service advertisement," in *Proc. IEEE 18th Int. Conf. Intell. Transp. Syst.*, Sep. 2015, pp. 1061–1068.
[54] I. Ivanov, C. Maple, T. Watson, and S. Lee, "Cyber security standards and issues in V2X communications for internet of vehicles," in *Proc. Living Internet Things, Cybersecur. (IoT)*, London, U.K., 2018, pp. 1–6.
[55] M. Muhammad and G. A. Safdar, "Survey on existing authentication issues for cellular-assisted V2X communication," *Veh. Commun.*, vol. 12, pp. 50–65, Apr. 2018.
[56] (Dec. 2017). *Guide to LTE Security*. Accessed: Aug. 25, 2020. [Online]. Available: https://nvlpubs. nist.gov/nistpubs/SpecialPublications/NIST.SP.800-187.p%dfp
[57] (2016). *Autoblog Report*. [Online]. Available: https://www.autoblog.com/2016/02/25/nissanconnect-ev-leaf-app-hacking-followup/?guc counter=1
[58] (2017). *Threatpost Report*. [Online]. Available: https://threatpost.com/hyundai-patches-leaky-blue-link-mobile-app/125182/
[59] U. Bayer, A. Moser, C. Kruegel, and E. Kirda, "Dynamic analysis of malicious code," *J. Comput. Virol.*, vol. 2, no. 1, pp. 67–77, 2006.
[60] J. Aycock, *Computer Viruses and Malware*, vol. 22. France: Springer, 2006.
[61] M. Christodorescu, S. Jha, S. A. Seshia, D. Song, and R. E. Bryant, "Semantics-aware malware detection," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2005, pp. 32–46.
[62] P. Szor, *The Art of Computer Virus Research and Defense*. London, U.K.: Pearson, 2005.
[63] P. J. Denning, "Computer viruses," Nasa, USA, Tech. Rep., 1988.
[64] G. Bonfante, M. Kaczmarek, and J.-Y. Marion, "On abstract computer virology from a recursion theoretic perspective," *J. Comput. Virol.*, vol. 1, nos. 3–4, pp. 45–54, Mar. 2006.
[65] M. Karresand, "Separating Trojan horses, viruses, and worms—A proposed taxonomy of software weapons," in *Proc. IEEE Syst., Man Cybern. Soc. Inf. Assurance Workshop*, Jun. 2003, pp. 127–134.
[66] J. Markoff, "Worm infects millions of computers worldwide," *New York Times*, vol. 23, Jan. 2009. [Online]. Available: https://www.nytimes.com/2009/01/23/technology/internet/23worm.html
[67] D. M. Kienzle and M. C. Elder, "Recent worms: A survey and trends," in *Proc. ACM Workshop Rapid Malcode*, 2003, pp. 1–10.
[68] X. Wang, W. Yu, A. Champion, X. Fu, and D. Xuan, "Detecting worms via mining dynamic program execution," in *Proc. 3rd Int. Conf. Secur. Privacy Commun. Netw. Workshops (SecureComm)*, 2007, pp. 412–421.
[69] S. Kiltz, A. Lang, and J. Dittmann, "Malware: Specialized Trojan horse," in *Cyber Warfare and Cyber Terrorism*. Hershey, PA, USA: IGI Global, 2007, pp. 154–160.
[70] J. Edwards, "System, method and computer program product for preventing spyware/malware from installing a registry," U.S. Patent 11 010 993, Feb. 23, 2006.
[71] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Comput. Surv.*, vol. 50, no. 3, pp. 1–40, Oct. 2017.
[72] S. Embleton, S. Sparks, and C. C. Zou, "SMM rootkit: A new breed of OS independent malware," *Secur. Commun. Netw.*, vol. 6, no. 12, pp. 1590–1605, 2013.
[73] A. Zaki and B. Humphrey, "Unveiling the kernel: Rootkit discovery using selective automated kernel memory differencing," in *Proc. VIRUS Bull. Conf.*, 2014, pp. 239–256.
[74] A. Javed and M. Akhlaq, "Patterns in malware designed for data espionage and backdoor creation," in *Proc. 12th Int. Bhurban Conf. Appl. Sci. Technol. (IBCAST)*, Jan. 2015, pp. 338–342.

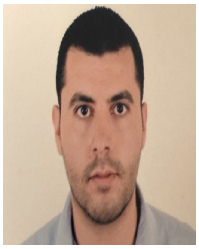
[75] M. Chowdhury, A. Rahman, and R. Islam, "Malware analysis and detection using data mining and machine learning classification," in *Proc. Int. Conf. Appl. Techn. Cyber Secur. Intell.* Jakarta, Indonesia: Springer, 2017, pp. 266–274.
[76] B. Stone-Gross, M. Cova, B. Gilbert, R. Kemmerer, C. Kruegel, and G. Vigna, "Analysis of a botnet takeover," *IEEE Security Privacy*, vol. 9, no. 1, pp. 64–72, Jan./Feb. 2011.
[77] G. A. N. Mohamed and N. B. Ithnin, "Survey on representation techniques for malware detection system," *Amer. J. Appl. Sci.*, vol. 14, no. 11, pp. 1049–1069, Nov. 2017.
[78] V. P. Laxmi and M. Gaur, "Survey on malware detection methods, 3rd hackers workshop on computer and internet security," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 998–1022, Apr./Jun. 2015.
[79] C. Seifert, J. W. Stokes, C. Colcernian, J. C. Platt, and L. Lu, "Robust scareware image detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 2920–2924.
[80] R. Richardson and M. M. North, "Ransomware: Evolution, mitigation and prevention," *Int. Manage. Rev.*, vol. 13, no. 1, p. 10, 2017.
[81] M. Wolf, R. Lambert, T. Enderle, and A. Schmidt, "Wanna drive? Feasible attack paths and effective protection against ransomware in modern vehicles," in *Proc. Embedded Secur. Cars Conf. (ESCAR) Eur.*, 2017, pp. 1–14.
[82] *The McAfee Security Report*. Accessed: Mar. 27, 2018. [Online]. Available: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/todays-connected-cars-vulnerable-hacking-malware
[83] *Boston 25 News Report*. Accessed: May 1, 2018. [Online]. Available: https://www.boston25news.com/news/discovery-of-hidden-gps-tracker-leads-to-mass-supreme-court-case/742286360
[84] Y. Wiseman, "Vehicle identification by OCR, RFID and Bluetooth for toll roads," *Int. J. Control Autom.*, vol. 11, no. 9, pp. 67–76, Sep. 2018.
[85] *The IBM Security Report*. Accessed: Nov. 12, 2020. [Online]. Available: https://www.zdnet.com/article/info-of-27-7-million-texas-drivers-exposed-in-vertafore-data-breach
[86] *Washington and California Universities Report*. [Online]. Available: https://www.wired.com/2015/10/car-hacking-tool-turns-repair-shops-malware-brothels
[87] *The University of Leuven Research Report*. Accessed: Nov. 23, 2020. [Online]. Available: https://www.washingtonpost.com/technology/2020/11/23/tesla-modelx-hack
[88] A. Arora, S. K. Yadav, and K. Sharma, "Denial-of-service (DoS) attack and botnet: Network analysis, research tactics, and mitigation," in *Handbook of Research on Network Forensics and Analysis Techniques*. Hershey, PA, USA: IGI Global, 2018, pp. 117–141.
[89] D. Ibdah, N. Lachtar, A. A. Elkhail, A. Bacha, and H. Malik, "Dark firmware: A systematic approach to exploring application security risks in the presence of untrusted firmware," in *Proc. 23rd Int. Symp. Res. Attacks, Intrusions Defenses (RAID)*, 2020, pp. 413–426.
[90] R. Rehman, G. Hazarika, and G. Chetia, "Malware threats and mitigation strategies: A survey," *J. Theor. Appl. Inf. Technol.*, vol. 29, no. 2, pp. 69–73, 2011.
[91] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Comput. Surv.*, vol. 44, no. 2, pp. 1–42, 2008.
[92] P. Sivakumar, R. S. S. Devi, A. N. Lakshmi, B. VinothKumar, and B. Vinod, "Automotive grade Linux software architecture for automotive infotainment system," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, Feb. 2020, pp. 391–395.
[93] N. H. V. Reddy, K. Thaiyalnayaki, and S. Sivasakthiselvan, "A study on real time vehicle surveillance and tracking system for Android application," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Jul. 2020, pp. 1599–1602.
[94] *The Meet Linux Viruses Report*. Accessed: Feb. 15, 2018. [Online]. Available: http://www.unixmen.com/meet-linux-viruses//
[95] *Malware Affecting Linux Web Servers Major Trend*. Accessed: Apr. 25, 2013. [Online]. Available: http://www.bnamericas.com/news/technology/malware-affecting-linux-web-servers-major-trend-in-2013-eset
[96] O. H. Alhazmi, Y. K. Malaiya, and I. Ray, "Measuring, analyzing and predicting security vulnerabilities in software systems," *Comput. Secur.*, vol. 26, no. 3, pp. 219–228, 2007.
[97] (2015). *Technology News Report*. [Online]. Available: https://www.reuters.com/article/us-gm-hacking/researcher-says-can-hack-gms-onstar-app-open-vehicle-start-engine
[98] K. C. Dey, A. Rayamajhi, M. Chowdhury, P. Bhavsar, and J. Martin, "Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication in a heterogeneous wireless network—Performance evaluation," *Transp. Res. C, Emerg. Technol.*, vol. 68, pp. 168–184, Jul. 2016.
[99] W. Ahmed and M. Elhadef, "Securing intelligent vehicular ad hoc networks: A survey," in *Advances in Computer Science and Ubiquitous Computing*. Tamil Nadu, India: Springer, 2017, pp. 6–14.
[100] V. H. La and A. R. Cavalli, "Security attacks and solutions in vehicular ad hoc networks: A survey," *Int. J. AdHoc Netw. Syst.*, vol. 4, no. 2, pp. 1–20, 2014.
[101] P. Cope, J. Campbell, and T. Hayajneh, "An investigation of Bluetooth security vulnerabilities," in *Proc. IEEE 7th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2017, pp. 1–7.
[102] Kaspersky. (2019). *Operation Shadowhammer*. [Online]. Available: https://securelist.com/operation-shadowhammer
[103] Upstream Security. (2021). *Upstream Security Global Automotive Cybersecurity Report*. [Online]. Available: https://info.upstream.auto/hubfs/Security_Report/Security_Report_2021/Upstream_Security-Global_Automotive_Cybersecurity_Report_2021.pdf
[104] A. A. Elkhail and T. Cerny, "On relating code smells to security vulnerabilities," in *Proc. IEEE IEEE 5th Int. Conf. Big Data Secur. Cloud (BigDataSecurity) Int. Conf. High Perform. Smart Comput. (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2019, pp. 7–12.
[105] H. Perl, S. Dechand, M. Smith, D. Arp, F. Yamaguchi, K. Rieck, S. Fahl, and Y. Acar, "VCCFinder: Finding potential vulnerabilities in open-source projects to assist code audits," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 426–437.
[106] Y. Shin, A. Meneely, L. Williams, and J. A. Osborne, "Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities," *IEEE Trans. Softw. Eng.*, vol. 37, no. 6, pp. 772–787, Dec. 2011.
[107] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame," in *Proc. 15th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2017, pp. 5709–5757.
[108] V. Kumar, J. Srivastava, and A. Lazarevic, *Managing Cyber Threats: Issues, Approaches, and Challenges*, vol. 5. Germany: Springer, 2006.
[109] N. Idika and A. P. Mathur, "A survey of malware detection techniques," *Purdue Univ.*, vol. 48, no. 2, pp. 1–48, 2007.
[110] G. Serazzi and S. Zanero, "Computer virus propagation models," in *Proc. Int. Workshop Modeling, Anal., Simulation Comput. Telecommun. Syst.* Orlando, FL, USA: Springer, 2003, pp. 26–50.
[111] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 42–57, 2013.
[112] Ö. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020.
[113] R. U. D. Refat, A. A. Elkhail, A. Hafeez, and H. Malik, "Detecting can bus intrusion by applying machine learning method to graph based features," in *Proc. SAI Intell. Syst. Conf.* London, U.K.: Springer, 2021, pp. 730–748.
[114] M. Siddiqui, "Data mining methods for malware detection," Univ. Central Florida, Orlando, FL, USA, Tech. Rep., 2008.
[115] P. Vinod, R. Jaipur, V. Laxmi, and M. Gaur, "Survey on malware detection methods," in *Proc. 3rd Hackers Workshop Comput. Internet Secur. (IITKHACK)*, 2009, pp. 74–79.
[116] S. Roy, S. Nag, I. K. Maitra, and S. K. Bandyopadhyay, "International journal of advanced research in computer science and software engineering," *Int. J.*, vol. 3, no. 6, pp. 1706–1746, 2013.
[117] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, 2013.
[118] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "RiskRanker: Scalable and accurate zero-day Android malware detection," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services*, 2012, pp. 281–294.
[119] P. Khodamoradi, M. Fazlali, F. Mardukhi, and M. Nosrati, "Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms," in *Proc. 18th CSI Int. Symp. Comput. Architecture Digit. Syst. (CADS)*, Oct. 2015, pp. 1–6.
[120] S. Shang, N. Zheng, J. Xu, M. Xu, and H. Zhang, "Detecting malware variants via function-call graph similarity," in *Proc. 5th Int. Conf. Malicious Unwanted Softw.*, Oct. 2010, pp. 113–120.
[121] M. K. Shankarapani, S. Ramamoorthy, R. S. Movva, and S. Mukkamala, "Malware detection using assembly and API call sequences," *J. Comput. Virol.*, vol. 7, no. 2, pp. 107–119, 2011.

[122] B. B. Rad, M. Masrom, and S. Ibrahim, "Opcodes histogram for classifying metamorphic portable executables malware," in *Proc. Int. Conf. E-Learn. E-Technol. Educ. (ICEEE)*, Sep. 2012, pp. 209–213.
[123] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Inf. Sci.*, vol. 231, pp. 64–82, May 2013.
[124] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," *ACM SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 559–570, 2013.
[125] B. Wu, T. Lu, K. Zheng, D. Zhang, and X. Lin, "Smartphone malware detection model based on artificial immune system," *China Commun.*, vol. 11, no. 13, pp. 86–92, 2014.
[126] M. E. Boujnouni, M. Jedra, and N. Zahid, "New malware detection framework based on N-grams and support vector domain description," in *Proc. 11th Int. Conf. Inf. Assurance Secur. (IAS)*, Dec. 2015, pp. 123–128.
[127] Y. Duan, X. Fu, B. Luo, Z. Wang, J. Shi, and X. Du, "Detective: Automatically identify and analyze malware processes in forensic scenarios via DLLs," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 5691–5696.
[128] Z. Li, L. Sun, Q. Yan, W. Srisa-An, and Z. Chen, "DroidClassifier: Efficient adaptive mining of application-layer header for classifying Android malware," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.* Washington, DC, USA: Springer, 2016, pp. 597–616.
[129] J. B. Fraley and M. Figueroa, "Polymorphic malware detection using topological feature extraction with data mining," in *Proc. SoutheastCon*, 2016, pp. 1–7.
[130] L. Sun, Z. Li, Q. Yan, W. Srisa-an, and Y. Pan, "SigPID: Significant permission identification for Android malware detection," in *Proc. 11th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2016, pp. 1–8.
[131] Y. Fan, Y. Ye, and L. Chen, "Malicious sequential pattern mining for automatic malware detection," *Expert Syst. Appl.*, vol. 52, pp. 16–25, Jun. 2016.
[132] S. Alam, Z. Qu, R. Riley, Y. Chen, and V. Rastogi, "DroidNative: Automating and optimizing detection of Android native code malware variants," *Comput. Secur.*, vol. 65, pp. 230–246, Mar. 2017.
[133] A. Narayanan, M. Chandramohan, L. Chen, and Y. Liu, "A multi-view context-aware approach to Android malware detection and malicious code localization," *Empirical Softw. Eng.*, vol. 23, no. 3, pp. 1222–1274, Jun. 2018.
[134] A. Ojugo and A. Eboka, "Signature-based malware detection using approximate Boyer Moore string matching algorithm," *Int. J. Math. Sci. Comput.*, vol. 5, no. 3, pp. 49–62, Jul. 2019.
[135] T.-L. Wan, T. Ban, Y.-T. Lee, S.-M. Cheng, R. Isawa, T. Takahashi, and D. Inoue, "IoT-malware detection based on byte sequences of executable files," in *Proc. 15th Asia Joint Conf. Inf. Secur. (AsiaJCIS)*, Aug. 2020, pp. 143–150.
[136] I. Markit. (2017). *Vehicles Getting Older: Average Age of Light Cars and Trucks in U.S.* [Online]. Available: http://news.ihsmarkit.com/press-release/automotive/vehicles-getting-olderaverage-age-light-cars-and-trucks-us-rises-again-2017
[137] X. Hu, T.-C. Chiueh, and K. G. Shin, "Large-scale malware indexing using function-call graphs," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 611–620.
[138] S. Cook. (2021). *Malware Statistics and Facts for 2021*. [Online]. Available: https://www.comparitech.com/antivirus/malware-statistics-facts/
[139] G. Jacob, H. Debar, and E. Filiol, "Behavioral detection of malware: From a survey towards an established taxonomy," *J. Comput. Virol.*, vol. 4, no. 3, pp. 251–266, Aug. 2008.
[140] N. Lachtar, A. A. Elkhail, A. Bacha, and H. Malik, "A cross-stack approach towards defending against cryptojacking," *IEEE Comput. Archit. Lett.*, vol. 19, no. 2, pp. 126–129, Jul. 2020.
[141] N. Lachtar, A. A. Elkhail, A. Bacha, and H. Malik, "An application agnostic defense against the dark arts of cryptojacking," in *Proc. 51st Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2021, pp. 314–325.
[142] T. Zhang and L. Delgrossi, *Vehicle Safety Communications: Protocols, Security, and Privacy*, vol. 103. Hoboken, NJ, USA: Wiley, 2012.
[143] O. Aslan and R. Samet, "Investigation of possibilities to detect malware using existing tools," in *Proc. IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct. 2017, pp. 1277–1284.
[144] Y. Fukushima, A. Sakai, Y. Hori, and K. Sakurai, "A behavior based malware detection scheme for avoiding false positive," in *Proc. 6th IEEE Workshop Secure Netw. Protocols*, Oct. 2010, pp. 79–84.
[145] S.-T. Liu, H.-C. Huang, and Y.-M. Chen, "A system call analysis method with mapreduce for malware detection," in *Proc. IEEE 17th Int. Conf. Parallel Distrib. Syst.*, Dec. 2011, pp. 631–637.
[146] B. Anderson, D. Quist, J. Neil, C. Storlie, and T. Lane, "Graph-based malware detection using dynamic analysis," *J. Comput. Virol.*, vol. 7, no. 4, pp. 247–258, 2011.
[147] P. Faruki, V. Laxmi, M. S. Gaur, and P. Vinod, "Behavioural detection with API call-grams to identify malicious PE files," in *Proc. SecurIT*, 2012, pp. 85–91.
[148] Y. Ding, X. Yuan, K. Tang, X. Xiao, and Y. Zhang, "A fast malware detection algorithm based on objective-oriented association mining," *Comput. Secur.*, vol. 39, pp. 315–324, Nov. 2013.
[149] M. Eskandari, Z. Khorshidpour, and S. Hashemi, "HDM-Analyser: A hybrid analysis approach based on data mining techniques for malware detection," *J. Comput. Virol. Hacking Techn.*, vol. 9, no. 2, pp. 77–93, May 2013.
[150] Y. Park, D. S. Reeves, and M. Stamp, "Deriving common malware behavior through graph clustering," *Comput. Secur.*, vol. 39, pp. 419–430, Nov. 2013.
[151] D. Uppal, R. Sinha, V. Mehra, and V. Jain, "Malware detection and classification based on extraction of API sequences," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2014, pp. 2337–2342.
[152] S. Sheen, R. Anitha, and V. Natarajan, "Android based malware detection using a multifeature collaborative decision fusion approach," *Neurocomputing*, vol. 151, pp. 905–912, Mar. 2015.
[153] A. Boukhtouta, S. A. Mokhov, N.-E. Lakhdari, M. Debbabi, and J. Paquet, "Network malware classification comparison using DPI and flow packet headers," *J. Comput. Virol. Hacking Techn.*, vol. 12, no. 2, pp. 69–100, May 2016.
[154] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics-based online malware detection: Towards efficient real-time protection against malware," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 289–302, Feb. 2016.
[155] S. D. Nikolopoulos and I. Polenakis, "A graph-based model for malware detection and classification using system-call groups," *J. Comput. Virol. Hacking Techn.*, vol. 13, no. 1, pp. 29–46, Feb. 2017.
[156] S. Chaba, R. Kumar, R. Pant, and M. Dave, "Malware detection approach for Android systems using system call logs," 2017, *arXiv:1709.08805*.
[157] F. Marhusin and C. J. Lokan, "A preemptive behaviour-based malware detection through analysis of API calls sequence inspired by human immune system," *Int. J. Eng. Technol.*, vol. 7, nos. 4–15, pp. 113–119, 2018.
[158] M. Rhode, L. Tuson, P. Burnap, and K. Jones, "LAB to SOC: Robust features for dynamic malware detection," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw., Ind. Track*, Jun. 2019, pp. 13–16.
[159] M. Alazab, M. Alazab, A. Shalaginov, A. Mesleh, and A. Awajan, "Intelligent mobile malware detection using permission requests and API calls," *Future Gener. Comput. Syst.*, vol. 107, pp. 509–521, Jun. 2020.
[160] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *Proc. 5th Conf. Inf. Knowl. Technol.*, May 2013, pp. 113–120.
[161] F. Adkins, L. Jones, M. Carlisle, and J. Upchurch, "Heuristic malware detection via basic block comparison," in *Proc. 8th Int. Conf. Malicious Unwanted Softw. Amer. (MALWARE)*, Oct. 2013, pp. 11–18.
[162] K. Alzarooni, "Malware variant detection," Ph.D. dissertation, Dept. Comput. Sci., Univ. College London, London, U.K., 2012.
[163] Y. Ye, T. Li, K. Huang, Q. Jiang, and Y. Chen, "Hierarchical associative classifier (HAC) for malware detection from the large and imbalanced gray list," *J. Intell. Inf. Syst.*, vol. 35, no. 1, pp. 1–20, Aug. 2010.
[164] I. Santos, F. Brezo, J. Nieves, Y. K. Penya, B. Sanz, C. Laorden, and P. G. Bringas, "Idea: Opcode-sequence-based malware detection," in *Proc. Int. Symp. Eng. Secure Softw. Syst.* Springer, 2010, pp. 35–43.
[165] Z. Zhao, "A virus detection scheme based on features of control flow graph," in *Proc. 2nd Int. Conf. Artif. Intell., Manage. Sci. Electron. Commerce (AIMSEC)*, Aug. 2011, pp. 943–947.
[166] N. Runwal, R. M. Low, and M. Stamp, "Opcode graph similarity and metamorphic detection," *J. Comput. Virol.*, vol. 8, nos. 1–2, pp. 37–52, 2012.
[167] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici, "Detecting unknown malicious code by applying classification techniques on opcode patterns," *Secur. Informat.*, vol. 1, no. 1, pp. 1–22, Dec. 2012.
[168] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 646–656, 2013.

[169] M. Eskandari and H. Raesi, "Frequent sub-graph mining for intelligent malware detection," *Secur. Commun. Netw.*, vol. 7, no. 11, pp. 1872–1886, Nov. 2014.
[170] M. Zakeri, F. F. Daneshgar, and M. Abbaspour, "A static heuristic approach to detecting malware targets," *Secur. Commun. Netw.*, vol. 8, no. 17, pp. 3015–3027, Nov. 2015.
[171] B. Kang, S. Y. Yerima, S. Sezer, and K. McLaughlin, "N-gram opcode analysis for Android malware detection," 2016, *arXiv:1612.01445*.
[172] A. Kapoor and S. Dhavale, "Control flow graph based multiclass malware detection using bi-normal separation," *Defence Sci. J.*, vol. 66, no. 2, p. 138, Mar. 2016.
[173] A. Narayanan, M. Chandramohan, L. Chen, and Y. Liu, "Context-aware, adaptive, and scalable Android malware detection through online learning," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 1, no. 3, pp. 157–175, Jun. 2017.
[174] Y. Ding, X. Xia, S. Chen, and Y. Li, "A malware detection method based on family behavior graph," *Comput. Secur.*, vol. 73, pp. 73–86, Mar. 2018.
[175] S. Wang and P. S. Yu, "Heterogeneous graph matching networks: Application to unknown malware detection," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 5401–5408.
[176] R. Surendran, T. Thomas, and S. Emmanuel, "GSDroid: Graph signal based compact feature representation for Android malware detection," *Expert Syst. Appl.*, vol. 159, Nov. 2020, Art. no. 113581.
[177] O. T. Suryati and A. Budiono, "Impact analysis of malware based on call network API with heuristic detection method," *Int. J. Adv. Data Inf. Syst.*, vol. 1, no. 1, pp. 1–8, Apr. 2020.
[178] Y. Ye, T. Li, S. Zhu, W. Zhuang, E. Tas, U. Gupta, and M. Abdulhayoglu, "Combining file content and file relations for cloud based malware detection," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 222–230.
[179] C. A. Martínez, G. I. Echeverri, and A. G. C. Sanz, "Malware detection based on cloud computing integrating intrusion ontology representation," in *Proc. IEEE Latin-Amer. Conf. Commun.*, Sep. 2010, pp. 1–6.
[180] C. Jarabek, D. Barrera, and J. Aycock, "ThinAV: Truly lightweight mobile cloud-based anti-malware," in *Proc. 28th Annu. Comput. Secur. Appl. Conf.*, 2012, pp. 209–218.
[181] S. Kim, S. W. Ko, and D. H. Lee, "Cloud-based malware analysis system for mobile applications," *Information*, vol. 16, no. 6, pp. 4357–4364, 2013.
[182] N. Penning, M. Hoffman, J. Nikolai, and Y. Wang, "Mobile malware security challeges and cloud-based detection," in *Proc. Int. Conf. Collaboration Technol. Syst. (CTS)*, May 2014, pp. 181–188.
[183] H. Sun, X. Wang, J. Su, and P. Chen, "RScam: Cloud-based anti-malware via reversible sketch," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.* Dallas, TX, USA: Springer, 2015, pp. 157–174.
[184] H. Zhang, Y. Cole, L. Ge, S. Wei, W. Yu, C. Lu, G. Chen, D. Shen, E. Blasch, and K. D. Pham, "ScanMe mobile: A cloud-based Android malware analysis service," *ACM SIGAPP Appl. Comput. Rev.*, vol. 16, no. 1, pp. 36–49, Apr. 2016.
[185] L. Xiao, Y. Li, X. Huang, and X. Du, "Cloud-based malware detection game for mobile devices with offloading," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2742–2750, Oct. 2017.
[186] Y. Kucuk, N. Patil, Z. Shu, and G. Yan, "BigBing: Privacy-preserving cloud-based malware classification service," in *Proc. IEEE Symp. Privacy-Aware Comput. (PAC)*, Sep. 2018, pp. 43–54.
[187] R. M. Yadav, "Effective analysis of malware detection in cloud computing," *Comput. Secur.*, vol. 83, pp. 14–21, Jun. 2019.
[188] R. Patil, H. Dudeja, and C. Modi, "Designing in-VM-assisted lightweight agent-based malware detection framework for securing virtual machines in cloud computing," *Int. J. Inf. Secur.*, vol. 19, no. 2, pp. 147–162, Apr. 2020.
[189] S. Li, Y. Li, W. Han, X. Du, M. Guizani, and Z. Tian, "Malicious mining code detection based on ensemble learning in cloud computing environment," *Simul. Model. Pract. Theory*, vol. 113, Dec. 2021, Art. no. 102391.
[190] T. Panker and N. Nissim, "Leveraging malicious behavior traces from volatile memory using machine learning methods for trusted unknown malware detection in Linux cloud environments," *Knowl.-Based Syst.*, vol. 226, Aug. 2021, Art. no. 107095.
[191] D. Tian, Q. Ying, X. Jia, R. Ma, C. Hu, and W. Liu, "MDCHD: A novel malware detection method in cloud using hardware trace and deep learning," *Comput. Netw.*, vol. 198, Oct. 2021, Art. no. 108394.
[192] O. Aslan, M. Ozkan-Okay, and D. Gupta, "Intelligent behavior-based malware detection system on cloud computing environment," *IEEE Access*, vol. 9, pp. 83252–83271, 2021.
[193] J. C. Kimmel, A. D. Mcdole, M. Abdelsalam, M. Gupta, and R. Sandhu, "Recurrent neural networks based online behavioural malware detection techniques for cloud infrastructure," *IEEE Access*, vol. 9, pp. 68066–68080, 2021.
[194] P. Mishra, P. Aggarwal, A. Vidyarthi, P. Singh, B. Khan, H. H. Alhelou, and P. Siano, "VMShield: Memory introspection-based malware detection to secure cloud-based services against stealthy attacks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 10, pp. 6754–6764, Oct. 2021.
[195] E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: A survey," *J. Inf. Secur.*, vol. 5, no. 2, pp. 56–64, 2014.
[196] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," 2016, *arXiv:1606.04435*.
[197] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2018, pp. 533–537.
[198] I. Firdausi, C. Lim, A. Erwin, and A. S. Nugroho, "Analysis of machine learning techniques used in behavior-based malware detection," in *Proc. 2nd Int. Conf. Adv. Comput., Control, Telecommun. Technol.*, Dec. 2010, pp. 201–203.
[199] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. 8th Int. Symp. Vis. Cyber Secur.*, 2011, pp. 1–7.
[200] B. Anderson, C. Storlie, and T. Lane, "Improving malware classification: Bridging the static/dynamic gap," in *Proc. 5th ACM workshop Secur. Artif. Intell.*, 2012, pp. 3–14.
[201] D. Kong and G. Yan, "Discriminant malware distance learning on structural information for automated malware classification," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2013, pp. 1357–1365.
[202] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 3422–3426.
[203] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-Sec: Deep learning in Android malware detection," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 371–372.
[204] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2015, pp. 11–20.
[205] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, and J. Yearwood, "Hybrids of support vector machine wrapper and filter based framework for malware detection," *Future Gener. Comput. Syst.*, vol. 55, pp. 376–390, Feb. 2016.
[206] S. Huda, S. Miah, M. M. Hassan, R. Islam, J. Yearwood, M. Alrubaian, and A. Almogren, "Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data," *Inf. Sci.*, vol. 379, pp. 211–228, Feb. 2017.
[207] D. Zhu, H. Jin, Y. Yang, D. Wu, and W. Chen, "DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017, pp. 438–443.
[208] M. A. Jerlin and K. Marimuthu, "A new malware detection system using machine learning techniques for API call sequences," *J. Appl. Secur. Res.*, vol. 13, no. 1, pp. 45–62, Jan. 2018.
[209] Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, and B. Yang, "Machine learning based mobile malware detection using highly imbalanced network traffic," *Inf. Sci.*, vols. 433–434, pp. 346–364, Apr. 2018.
[210] J. Jung, H. Kim, D. Shin, M. Lee, H. Lee, S.-J. Cho, and K. Suh, "Android malware detection based on useful API calls and machine learning," in *Proc. IEEE 1st Int. Conf. Artif. Intell. Knowl. Eng. (AIKE)*, Sep. 2018, pp. 175–178.
[211] H. Sayadi, H. M. Makrani, S. M. P. Dinakarrao, T. Mohsenin, A. Sasan, S. Rafatirad, and H. Homayoun, "2SMaRT: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 728–733.
[212] N. Lachtar, D. Ibdah, and A. Bacha, "The case for native instructions in the detection of mobile ransomware," *IEEE Lett. Comput. Soc.*, vol. 2, no. 2, pp. 16–19, Jun. 2019.

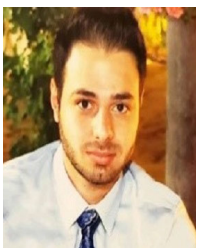
[213] W.-C. Kuo, T.-P. Liu, and C.-C. Wang, "Study on Android hybrid malware detection based on machine learning," in *Proc. IEEE 4th Int. Conf. Comput. Commun. Syst. (ICCCS)*, Feb. 2019, pp. 31–35.
[214] Z. Ma, H. Ge, Y. Liu, M. Zhao, and J. Ma, "A combination method for Android malware detection based on control flow graphs and machine learning algorithms," *IEEE Access*, vol. 7, pp. 21235–21245, 2019.
[215] S. Yang, S. Li, W. Chen, and Y. Liu, "A real-time and adaptive-learning malware detection method based on API-pair graph," *IEEE Access*, vol. 8, pp. 208120–208135, 2020.
[216] A. Pektaş and T. Acarman, "Learning to detect Android malware via opcode sequences," *Neurocomputing*, vol. 396, pp. 599–608, Jul. 2020.
[217] H. Sayadi, Y. Gao, H. M. Makrani, T. Mohsenin, A. Sasan, S. Rafatirad, J. Lin, and H. Homayoun, "StealthMiner: Specialized time series machine learning for run-time stealthy malware detection based on microarchitectural features," in *Proc. Great Lakes Symp. VLSI*, Sep. 2020, pp. 175–180.
[218] N. Lachtar, D. Ibdah, and A. Bacha, "Toward mobile malware detection through convolutional neural networks," *IEEE Embedded Syst. Lett.*, vol. 13, no. 3, pp. 134–137, Sep. 2021.
[219] X. Huang, L. Ma, W. Yang, and Y. Zhong, "A method for Windows malware detection based on deep learning," *J. Signal Process. Syst.*, vol. 93, nos. 2–3, pp. 265–273, Mar. 2021.
[220] M. Almahmoud, D. Alzu'bi, and Q. Yaseen, "ReDroidDet: Android malware detection based on recurrent neural network," *Proc. Comput. Sci.*, vol. 184, pp. 841–846, Jan. 2021.
[221] H. Ghanei, F. Manavi, and A. Hamzeh, "A novel method for malware detection based on hardware events using deep neural networks," *J. Comput. Virol. Hacking Techn.*, vol. 17, pp. 319–331, May 2021.
[222] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2016, pp. 63–68.
[223] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *Proc. 6th Int. Conf. Inf. Assurance Secur.*, Aug. 2010, pp. 92–98.
[224] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 1110–1115.
[225] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "CANAuth—A simple, backward compatible broadcast authentication protocol for can bus," in *Proc. ECRYPT Workshop Lightweight Cryptogr.*, 2011, pp. 1–7.
[226] C.-W. Lin and A. Sangiovanni-Vincentelli, "Cyber-security for the controller area network (CAN) communication protocol," in *Proc. Int. Conf. Cyber Secur.*, Dec. 2012, pp. 1–7.
[227] B. Groza and S. Murvay, "Efficient protocols for secure broadcast in controller area networks," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2034–2042, Nov. 2013.
[228] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *Proc. 25th USENIX Secur. Symp. (USENIX Security)*, 2016, pp. 911–927.
[229] O. Avatefipour, A. Hafeez, M. Tayyab, and H. Malik, "Linking received packet to the transmitter through physical-fingerprinting of controller area network," in *Proc. IEEE Workshop Inf. Forensics Secur. (WIFS)*, Dec. 2017, pp. 1–6.
[230] F. Martinelli, F. Mercaldo, V. Nardone, and A. Santone, "Car hacking identification through fuzzy logic algorithms," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Jul. 2017, pp. 1–7.
[231] A. Theissler, "Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection," *Knowl.-Based Syst.*, vol. 123, pp. 163–173, May 2017.
[232] O. Minawi, J. Whelan, A. Almehmadi, and K. El-Khatib, "Machine learning-based intrusion detection system for controller area networks," in *Proc. 10th ACM Symp. Design Anal. Intell. Veh. Netw. Appl.*, Nov. 2020, pp. 41–47.
[233] V. S. Barletta, D. Caivano, A. Nannavecchia, and M. Scalera, "Intrusion detection for in-vehicle communication networks: An unsupervised Kohonen SOM approach," *Future Internet*, vol. 12, no. 7, p. 119, Jul. 2020.
[234] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "LSTM-based intrusion detection system for in-vehicle can bus communications," *IEEE Access*, vol. 8, pp. 185489–185502, 2020.
[235] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "VoltageIDS: Low-level communication characteristics for automotive intrusion detection system," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 8, pp. 2114–2129, Aug. 2018.
[236] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran, "Cloaking the clock: Emulating clock skew in controller area networks," in *Proc. ACM/IEEE 9th Int. Conf. Cyber-Physical Syst. (ICCPS)*, Apr. 2018, pp. 32–42.
[237] H. Oguma, A. Yoshioka, M. Nishikawa, R. Shigetomi, A. Otsuka, and H. Imai, "New attestation based security architecture for in-vehicle communication," in *Proc. IEEE Global Telecommun. Conf. (IEEE GLOBECOM)*, Nov./Dec. 2008, pp. 1–6.
[238] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "CANet: An unsupervised intrusion detection system for high dimensional CAN bus data," *IEEE Access*, vol. 8, pp. 58194–58205, 2020.
[239] R. Islam, R. U. D. Refat, S. M. Yerram, and H. Malik, "Graph-based intrusion detection system for controller area networks," *IEEE Trans. Intell. Transp. Syst.*, early access, Oct. 1, 2020, doi: 10.1109/TITS.2020.3025685.
[240] H. L. Bijmans, T. M. Booij, and C. Doerr, "Inadvertently making cyber criminals rich: A comprehensive study of cryptojacking campaigns at internet scale," in *Proc. 28th USENIX Secur. Symp. (USENIX Security)*, 2019, pp. 1627–1644.
[241] G. Hong, Z. Yang, S. Yang, L. Zhang, Y. Nan, Z. Zhang, M. Yang, Y. Zhang, Z. Qian, and H. Duan, "How you get shot in the back: A systematical study about cryptojacking in the real world," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 1701–1713.
[242] D. Olenick. (2021). *How Conti Ransomware Works*. [Online]. Available: https://www.bankinfosecurity.com/how-conti-ransomware-works-a-15763
[243] M. Loman. (2019). *How the Most Damaging Ransomware Evades IT Security*. [Online]. Available: https://news.sophos.com/en-us/2019/11/14/how-the-most-damaging-ransomware-evades-it-security
[244] S. Hosseinzadeh, S. Rauti, S. Laurén, J.-M. Mäkelä, J. Holvitie, S. Hyrynsalmi, and V. Leppänen, "Diversification and obfuscation techniques for software security: A systematic literature review," *Inf. Softw. Technol.*, vol. 104, pp. 72–93, Dec. 2018.
[245] C. Barría, D. Cordero, C. Cubillos, and M. Palma, "Proposed classification of malware, based on obfuscation," in *Proc. 6th Int. Conf. Comput. Commun. Control (ICCCC)*, May 2016, pp. 37–44.
[246] F. Martinelli, F. Mercaldo, V. Nardone, A. Santone, A. K. Sangaiah, and A. Cimitile, "Evaluating model checking for cyber threats code obfuscation identification," *J. Parallel Distrib. Comput.*, vol. 119, pp. 203–218, Sep. 2018.
[247] H. Xu, Y. Zhou, J. Ming, and M. Lyu, "Layered obfuscation: A taxonomy of software obfuscation techniques for layered security," *Cybersecurity*, vol. 3, no. 1, pp. 1–18, Dec. 2020.
[248] E. Stavrou and A. Pitsillides, "A survey on secure multipath routing protocols in WSNs," *Comput. Netw.*, vol. 54, no. 13, pp. 2215–2238, 2010.
[249] Q. G. K. Safi, S. Luo, C. Wei, L. Pan, and G. Yan, "Cloud-based security and privacy-aware information dissemination over ubiquitous VANETs," *Comput. Standards Interfaces*, vol. 56, pp. 107–115, Feb. 2018.
[250] F. A. Silva, A. Boukerche, T. R. M. B. Silva, L. B. Ruiz, and A. A. F. Loureiro, "Geo-localized content availability in VANETs," *Ad Hoc Netw.*, vol. 36, pp. 425–434, Jan. 2016.
[251] F. Wang, Y. Xu, H. Zhang, Y. Zhang, and L. Zhu, "2FLIP: A two-factor lightweight privacy-preserving authentication scheme for VANET," *IEEE Trans. Veh. Technol.*, vol. 65, no. 2, pp. 896–911, Feb. 2016.
[252] S. Jana, S. N. Premnath, M. Clark, S. K. Kasera, N. Patwari, and S. V. Krishnamurthy, "On the effectiveness of secret key extraction from wireless signal strength in real environments," in *Proc. 15th Annu. Int. Conf. Mobile Comput. Netw.*, 2009, pp. 321–332.
[253] B. Alpern and F. B. Schneider, "Key exchange using 'keyless cryptography,'" *Inf. Process. Lett.*, vol. 16, no. 2, pp. 79–81, Feb. 1983.
[254] N. A. Abdulsalam, R. A. Hajri, Z. A. Abri, Z. A. Lawati, and M. M. Bait-Suwailam, "Design and implementation of a vehicle to vehicle communication system using Li-Fi technology," in *Proc. Int. Conf. Inf. Commun. Technol. Res. (ICTRC)*, May 2015, pp. 136–139.
[255] M. D. Pesé, K. Schmidt, and H. Zweck, "Hardware/software co-design of an automotive embedded firewall," SAE Tech. Paper 2017-01-1659, 2017.
[256] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D. Gan, "Cloud-based cyber-physical intrusion detection for vehicles using deep learning," *IEEE Access*, vol. 6, pp. 3491–3508, 2017.

[257] G. Loukas, Y. Yoon, G. Sakellari, T. Vuong, and R. Heartfield, "Computation offloading of a vehicle's continuous intrusion detection workload for energy efficiency and performance," *Simul. Model. Pract. Theory*, vol. 73, pp. 83–94, Apr. 2017.
[258] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos, "SDSecurity: A software defined security experimental framework," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, Jun. 2015, pp. 1871–1876.



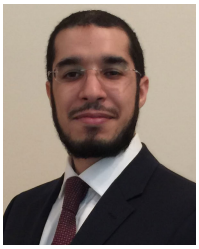
ABDULRAHMAN ABU ELKHAIL (Member, IEEE) received the B.S. degree in computer engineering from Yarmouk University, Irbid, Jordan, and the M.S. degree in computer engineering from the King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Michigan–Dearborn, USA. Before commencing his Ph.D., he worked in the industry for five years. He has several publications in referred reputable journals and conference proceedings and two U.S. patents. His areas of interests include automotive cybersecurity, network security and privacy, system and security, mobile and wireless communications, WSN and ad hoc networks, the IoT, and performance evaluation.

RAFI UD DAULA REFAT (Graduate Student Member, IEEE) received the B.Sc. degree in computer science and engineering from the Rajshahi University of Engineering & Technology. He is currently pursuing the Ph.D. degree with the Department of Electrical, Electronics and Computer Engineering, University of Michigan–Dearborn, Dearborn, MI, USA. Prior to starting his Ph.D., he worked in the software industry for nearly four years. His research interests focus on automotive network security, machine learning, data science, and speaker verification. He has publications in IEEE and Springer journals so far on automotive network security. He was awarded the RUET Academic Excellence Award, in 2013.



RICARDO HABRE is currently a Graduate Student in electrical engineering from the University of Michigan–Dearborn. His areas of interests include vehicle cybersecurity, DSP, AI/machine learning, and embedded systems.

AZEEM HAFEEZ is currently working as a Faculty with the Department of Electrical and Computer Engineering (ECE), University of Michigan–Dearborn, where he is also working with the Information Systems, Security, and Forensics (ISSF) Laboratory. Besides research and teaching, he is also an Advisor of senior design projects and Intelligent Systems Club at the University of Michigan–Dearborn. His areas of interests include vehicle cybersecurity, DSP, AI/machine learning/pattern recognition, data science, and embedded systems.



ANYS BACHA (Member, IEEE) is currently an Assistant Professor with the University of Michigan–Dearborn, where he leads the Security and Systems Lab, which focuses on advancing the state-of-the-art in mobile and computer systems to address important challenges in security, applied machine learning, and energy efficiency. His research contributions have been published in top tier venues, where his work received various prestigious awards. Furthermore, his industry impact is demonstrated through several U.S. and World patents. Prior to joining academia, he spent over 13 years in the industry, where he worked in different research and development roles on a variety of subsystems spanning the hardware, firmware, and operating systems layers. He led multiple interdisciplinary efforts that include driving architectural changes into next generation Intel processors that are necessary to meet the demands of emerging workloads. During his tenure at Hewlett-Packard, he led a group of engineers on a multi-million dollar scalable computing project that broke world records in performance, in 2015 and 2014.

HAFIZ MALIK (Senior Member, IEEE) is currently an Associate Professor with the Department of Electrical and Computer Engineering (ECE), University of Michigan–Dearborn. He has published more than 100 papers in leading journals, conferences, and workshops. His research interests include automotive cybersecurity, the IoT security, sensor security, multimedia forensics, steganography/steganalysis, information hiding, pattern recognition, and information fusion is funded by the National Science Foundation, National Academies, Ford Motor Company, and other agencies. Since 2015, he has been a member of the MCity Working Group on Cybersecurity. He is a Founding Member of the Cybersecurity Center for Research, Education, and Outreach at UM-Dearborn. He is a Member Leadership Circle of the Dearborn Artificial Intelligence Research Center, UM-Dearborn. He is also a member of the Scientific and Industrial Advisory Board (SIAB), National Center for Cyber Security, Pakistan.