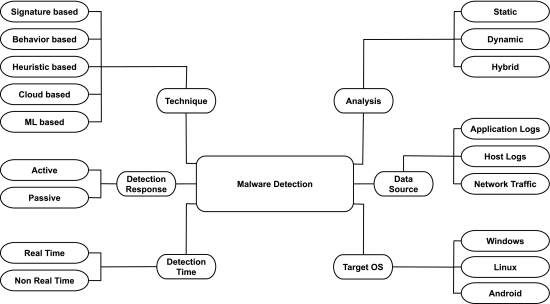
**Existing security measures and their effectiveness**

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. 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 1 shows the taxonomy dimensions distributed into six classes. We also briefly describe these classes below.

1. **Techniques**: 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.
   * **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. 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.
3. **Target Operating System (OS)**. It refers to the operating system analyzed by the system. It can be LINUX, Windows, or Android.
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.
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.
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.



**Malware detection system taxonomy**

* **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. 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. 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 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.

Researchers have proposed several approaches to detect malware based on the digital footprints of program files or applications. 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). Apart from the OS dependencies, the detection approaches differ in their way of analysis. Some researchers 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 proposed a novel malware detection method based on function call graph similarity. Other work by Shankarapani used API call sequences and assembly instructions to detect malware. The authors have used control flow graph signatures to detect malware. Wan 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.

* **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. 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. 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. 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. 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.

Multiple bodies of work have adopted behavior-based malware detection technique as a solution against malware. 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 proposed a novel method for detecting malware based on static analysis of API calls and permissions. Similarly, the authors 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. For instance, Nikolopoulos 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 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 proposed a dynamic malware detection approach based on analysis of API calls and permissions. Other work by Das 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 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 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 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. Furthermore, complex code obfuscation and evasion techniques might simply prevent malware from being properly assessed. 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.

* **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. 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. It is also capable of detecting a wide range of known and unknown malware. 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. 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. Furthermore, it is vulnerable to the advanced code obfuscation and evasion techniques that might simply prevent malware from being correctly detected.

Several researchers have proposed various heuristic-based malware detection techniques in the last decade. Some researchers have relied on static analysis to detect malware. For example, the authors have proposed a method for detecting malware based on control flow graphs and extracted opcodes from disassembled executable files. Work by Zaker used Dynamic Link Libraries (DLLs) to detect malware. Other recent work by Suryati 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 have relied on dynamic analysis for detecting malware. For instance, Shabtai proposed a dynamic method for detecting malware based on monitoring system opcode n-gram patterns. The authors 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. For example, the authors have used API calls and opcode sequences to detect malware. The remaining works relied on the graph-based method, in which the classification is done on the basis of a graph. For instance, 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 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 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. It’s also vulnerable to complex code obfuscation and evasion techniques, which might prevent malware from being identified appropriately. 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. As a result, this technique might not be suitable for resource-constrained in-vehicle 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.

* **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. 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.

Recently, several researchers have used cloud-based techniques to analyze and identify malware. Researchers have relied on static analysis to detect malware. For example, Ye have used file content and file relations features for detecting malware. Similarly, work by Li 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. For instance, the authors have proposed a dynamic method for detecting malware based on monitoring system calls. Similarly, work by Mishra proposed a dynamic method to detect malware based on n-gram features. The authors of 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. For example, Jarabek 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 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 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. 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, this technique might be safer to apply for intelligent vehicles.

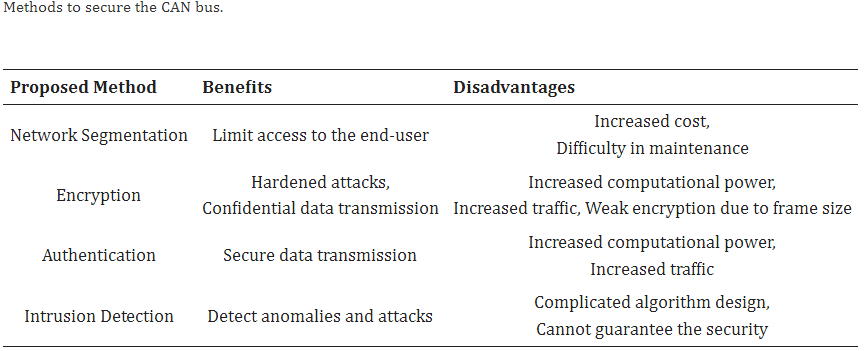
* **Machine Learning-Based Malware Detection:**

For many years, machine learning methods have been employed to identify malware. 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. 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 number of features, and the correlations between characteristics and attributes as well. Deep Learning is a subfield of machine learning that evolved from artificial neural networks (ANN) that learn 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. 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.

In the last decade, researchers have proposed various machine learning-based malware detection techniques. For example, the authors 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 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. For instance, the authors 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 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. For example, the authors 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 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. 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.



* **Network Segmentation**

The most straightforward protection mechanism is separating the CAN network into multiple subnetworks. The segmentation provides control over who can access particular subnetwork and reduce the damage of the attack by limiting its spread. The interconnection between subnetworks is controlled via a gateway ECU. This model currently exists in commercial vehicles. The method is simple to implement, but it is not effective if the gateway ECU is compromised or manipulated like the hacking exhibited in. Kammerer addressed this issue and proposed a star coupling router with security features. The paper ignored the security inside a subnetwork, but it is possible to implement a replay attack in a subnetwork and attack the other subnetworks bypassing the security check of the router.

Researchers at TU München proposed an automotive service bus architecture where their two-layer architecture was designed to prevent external attacks. The infotainment system and all vital functions were separated from each other. All components could send and receive messages, but by default, they could not send any data as the central ECU allows whom to write to the automotive service bus.

Network segmentation increases the security level, but it is not a sufficient method to protect the CAN. It also makes the maintenance of the system more difficult, along with the increased cost.

* **Encryption**

The CAN protocol uses a shared broadcast network without a built-in encryption mechanism. This allows an adversary to eavesdrop all the nodes and understand the communication. To prevent this data breach, a light-weight encryption system should be implemented. Although there are commercial software-based encryption methods (e.g., Trillium, CANcrypt) and manufacturers have proprietary encryption techniques implemented in cars, there have been reports claiming that encryption mechanisms in commercially available cars can be broken.

The limited data field is one of the problems for secure CAN encryption. This problem can be overcome by sending multiple CAN frames for a single message, and may solve the problem on low traffic networks, but it is not a solution for the currently rising traffic in automobile CAN networks. Another issue is the limited computational power of ECUs. If we consider the lifetime of a vehicle, it is possible to crack a static encryption key. Therefore, dynamic key exchange is required. However, this is harder to implement and is computationally expensive. The dynamic key can also cause latency on resource-constrained ECUs, and it is not acceptable for safety-critical real-time systems.

Doan and Ganesan implemented hardware-based AES-128 encryption on FPGA chips for the CAN system. The hardware implementation of the method decreases latency and increases throughput. However, the method changes the legacy ECU and is not backward compatible. Another study used physical unclonable functions (PUFs). This method can obtain the private key from the physical characteristics of the ECUs; thus, hiding the key is not a problem. Although the method solves the problem for generating encryption keys, it also requires modifying the ECU. Encryption hardens attacks and provides privacy; however, it is not sufficient to protect the CAN. Even the unbreakable encryption mechanism cannot prevent replay attacks.

* **Authentication:**

It is not possible to identify the sender of a CAN message. If an adversary has access to the network, they can send malicious messages and all the nodes accept them as authentic. This can be prevented via authentication.

VeCure authentication, which has an acceptable 50 us processing delay, is based on trust groups where high-trust groups share a symmetric secret key. The method has a major advantage with fewer key numbers, which corresponds in size to the number of trust groups rather than the ECU number. However, it sends an authentication message after every transmitted frame, which doubles the network traffic. Another drawback of the method is that it cannot protect the system if a node from the trust group is compromised. LiBrA-CAN, proposed by Groza et al., splits the authentication keys between groups of multiple nodes to improve efficiency. Although the method is quite successful, it requires high bandwidth and is not compatible with traditional CAN.

Nowdehi identified five criteria for an authentication method to be implemented commercially: cost-effectiveness, backward compatibility, support for vehicle repair and maintenance, sufficient implementation details, and acceptable overhead. They analysed ten authentication methods in the literature using them. Not surprisingly, none of the methods could pass all five criteria.

There are also off-the-shelf products providing hardware-based authentication like the S32K family from NXP. The S32K family has Cryptographic Service Engine (CSE), which has a Cipher-based Message Authentication Code (CMAC) to provide secure authentication, and is a hardware-based system that accelerates the process drastically. For instance, public-key authentication can be achieved in less than 100 us with hardware acceleration, while software authentication takes more than 10 ms, depending on the key size. However, the industry is currently concerned with the cost of ECUs. With the enhancement of hardware technology, it is possible to see more hardware-based methods to secure the CAN.

* **Intrusion Detection System (IDS)**

Implementing security features on a safety-critical real-time system is a difficult task. Strong cryptographic methods are not feasible due to the limited resources (memory, bandwidth, and computational power) and time constraints, from which research on intrusion detection system (IDS) for CAN has emerged. The main advantage of IDS is that it usually does not modify the current CAN controller and the bus traffic does not increase.

Intrusion detection methods can be categorised as signature-based (misuse) detection and anomaly-based detection. Signature-based detection checks for known attacks on the database; therefore, it requires regular updates for new attacks. Although it is quite successful in detecting known attacks, it fails to detect unknown attacks. Anomaly-based IDS analyses the behaviour of the network and recognises the deviation from normal behaviour. Accuracy is usually lower than that of the signature-based. In contrast to signature-based detection, anomaly-based IDS may detect unknown attacks.

There are different parameters that an IDS system can assess on the CAN. Deviation from the normal behaviour of these sensors is the sign of the intrusion, and different IDS solutions use these sensors to detect intrusions. These solutions can be categorised as time/frequency-based, physical system characteristic, specification-based, and feature-based.

A diagram of a system

Description automatically generated

* + **Time/Frequency-Based IDS:**

The automobiles have rigid safety rules and most of the ECUs transmit periodic signals. Any change in the frequency can be interpreted as abnormal behaviour, in other words, an intrusion. The basic IDS analyses the frequency of the CAN messages as presented.

Offset ratio and time interval-based IDS. As proposed by Lee et al., analyses the response time of the transmitted remote frame where the simple effective algorithm can detect attacks and type of attacks, however, the method increases bus traffic by injecting remote frames for analyses.

The time/frequency analysis provides useful information about the CAN. However, the vehicle’s situation (e.g., idle, running) and the priority scheme of the CAN may significantly change the timing information and affect the result of time/frequency-based IDS. The method also cannot detect attacks where the frequency is not changed, like a masquerade attack.

* + **Physical Characteristic Based IDS:**

The physical characteristic of the CAN network can be used to detect intrusions; hence each transceiver has a different signal shape even though they transmit the same data. This can be caused by random manufacturing variations, cabling, and aging.

Choi proposed VoltageIDS, which uses unique electrical characteristics of the CAN signal like a fingerprint. The different locations of the ECUs with different lengths of wire results in different resistance and the resistance changes the signal features. They analysed eight of the signal features like positive and negative slope values and voltage value at a dominant level. The method has zero false-positive rates and can differentiate between attacks and errors; however, it requires an oscilloscope to gather the network signal and has heavy signal processing.

The CAN does not have a shared master clock, and each ECU uses its own quartz crystal. Cho and Shin suggested the use of clock skew to detect intrusions. Although ECUs run the same frequency, they may have random drifting exceeding 2400 ms in a day. They fingerprinted the transmitter ECU via the clock skew and detected the intrusions. Although they could reach 97% of the anomaly detection with a false-positive rate of 0.55%, the method only worked for the periodic messages. However, this method can be tricked by mimicking the clock skew.

The physical characteristics of the CAN provides substantial information about ECUs. However, environmental factors like temperature and humidity and aging of the components can change the physical characteristics; therefore, the IDS may fail. They can also not detect the attacks from the software layer because the authenticated ECU will transmit the malicious messages, and the IDS does not find any changes to the signal characteristics. Similarly, the physical characteristic-based IDS requires heavy signal processing. As a result, it may cause latency or require expensive hardware.

* + **Specification Based IDS:**

Larson suggested specification-based attack detection and implemented specification rules based on the CAN Open protocol. This method has limited attack detection capability and requires all the ECUs to have detectors. The method also is not powerful enough to prevent attacks; hence there are protocol compliant attacks.

Studnia proposed a language-based intrusion detection and derived the language characteristic of the network from the ECUs’ specifications and generated the forbidden sequences. If one of these sequences occurs, an intrusion is detected.

* + **Feature-Based IDS:**

Feature-based system analysis examines the network parameters like busload, frequency, number of dropped messages, and other parameters like abnormal messages and payload. This is usually based on artificial intelligence techniques.

Generative adversarial nets (GAN) based IDS was proposed by Seo, who used the deep-learning model. The method is easy to expand and difficult to manipulate by an attacker, hence the detection mechanism has a black-box characteristic. Bloom filtering, proposed by Groza and Murvay, analysed the periodicity and payload of CAN messages. This method provides a memory-efficient analysis of data. Although both methods require heavy computation, they look promising in terms of tackling the CAN security problem.

Each method has a unique feature to suppress other methods, but also comes with a cost. For example, physical characteristic-based IDS can easily detect an inauthentic node, but it fails to detect an attack from a software layer. The best IDS system should be a hybrid system that takes advantage of different methods. Although IDS can mitigate a security problem, it cannot provide confidentiality. To have complete security, cryptography is required.

**Latest advancements in securing automotive networks**

**Survey of Security Solutions Based on Cryptography Techniques:**

A diagram of a crosswalk

Description automatically generated

Generally, there are two main components in the Cryptography approach.

First is known as Message Authentication Code (MAC) and another is called as cryptosystems having two fields symmetric and asymmetric. Further, Integrity and Authentication is ensured by the MAC while confidentiality is provided by the symmetric and asymmetric cryptosystems. Additionally, session keys can be utilized for providing authentication. For vehicle safety, the load on the CAN bus and latency issue in response time should be within specified limit. Additionally, error detection in data frame transmission is provided by the Cyclic Redundancy Code (CRC) at CAN bus. ECUs are having their own limitations in terms of computational capacity thereby lightweight encryption is one of the solutions for handling this issue since ECUs are core components inside the vehicle handling various functions simultaneously. The bus may be heavily loaded during key exchange and pre-loaded keys in the ECUs can tackle this situation in key distribution environment.

The Hardware Security Module (HSM) in ECUs can be effectively utilized for performing encryption and decryption in optimal time and compensating the issue of resource constrained ECUs.

Nilsson proposed a data authentication framework for modification and injection attacks. The proposed data authentication is known as delayed authentication since MAC is derived on a compound of successive messages and sent with other subsequent messages.

Herrewege explored the implementation issues of the message authentication protocol for CAN bus. After successful investigation, they find out various constraints which are related with backward compatible message authentication protocol and presented a new message authentication protocol in order to address the existing constraints.

Hazem designed a new protocol known as message source authentication protocol. The proposed authentication protocol performs well with minimum overhead. The implementation of the proposed protocol does not require neither any modifications in hardware for CAN network nor any changes in existing CAN message sets.

Groza utilizes symmetric primitives for designing the authentication protocol having two main mechanisms, namely mixing of MAC and splitting of keys. In the proposed protocol, authentication keys are split among multiple groups of nodes results in progressive authentication as compared with traditional approach of authentication for each node independently.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A white background with many text

Description automatically generated with medium confidence

**Survey of Security Solutions Based on Machine Learning Algorithms**

In today’s world of wireless networks, machine learning (ML) approaches are considered as the most promising choice for handling security-related issues. Researchers are proposing solutions utilizing ML to deal with vehicle security issues. Machine learning (ML) approaches have significant advantages as compared with other mechanisms, one of the main important features is to get optimal predictions about several types of attacks. The use of machine learning model in intrusion detection.

A diagram of a car data set

Description automatically generated(Fig-4 Intrusion Detection) General Flow Diagram reflecting series of steps for intrusion detection using machine learning model.

**Multi-Layered Security Framework**

The modern advanced vehicles have complex and sophisticated in-vehicle architecture. Additionally, these vehicles are equipped with highly sensitive sensors, different electronic devices, computer systems, etc. in order to secure these sophisticated systems, a coordinated, systemic integrated cybersecurity framework is needed to design the solutions and to minimize security risks. For the cyber security of in-vehicle networks, a multi-layered approach is the need of the hour. We need an integrated security approach since hackers invade vehicles through cyber and via the physical world. Further, Figure 13 illustrates the multi-layered security framework for in-vehicle networks with corresponding automotive protocols. In the context of scientific novelty, it is highlighted that the multi-layered security framework is proposed based on the findings of critical investigation of existing literature. Each layer addresses a specific type of security threat for in-vehicle communication network. It is highlighted that the multilayer security framework is proposed based on the understanding and knowledge aided by critical analysis and findings of existing approaches for the security of in-vehicle network. We found that the literature is lacking in terms of cohesive multi-layer security framework for in-vehicle network. In this framework, we addressed different security issues of in-vehicle network in specific layers including ECU-boot level security, ECU to ECU communication, domains/sub-domains communication, application software update or version issues, gateway security controller, and vehicle to outside services. However, it is clarified that in the proposed integrated multilayer security framework, each layer ensures specific functionality and address security threats. This is the reason we divided the framework into six layers staring from ECU-boot level security, ECU to ECU communication, domains/sub-domains communication, application software update/version issues, gateway security controller, and finally in-vehicle network to vehicle outside services security issues. In the Secure Gateway/Domain Controller layer, we are more concerned about security of gateway while at the secure external communication layer, we have a priority to protect the communication channels. In terms of in-general functional roles all automotive protocols are somehow or other play role in all layers due to complex structuring of modern in-vehicular system. However, if we make deep analysis based on maximum effectiveness of automotive protocols in terms of providing functionality to domain/sub-domain of in-vehicular system, then we can identify which specific automotive protocol is more suitable to a layer. Therefore, we used a protocol for a layer with more effective and suitable functionality consideration in mind while developing the framework.

**A diagram of a computer network

Description automatically generated with medium confidence**

Control Platform Layer This layer aims at enhancing the security of platform. This layer can be assumed as vehicle main nerve center for securing the in-vehicle network of smart intelligent vehicle against malicious threats. Control layer aspect deals with security solutions for protecting firmware of ECU, enhancing security at booting level, and also security of hardware modules (HM). The original equipment manufacturer (OEM) trusted server and trusted platform module (TPM) chip at ECU play a major role in this mechanism. The secure connection is effectively maintained between these two modules for the exchange of encrypted firmware update image file and data for enhancing security at booting level, ECU firmware, etc. The ECU firmware should be updated frequently to counter the diversified attacks in current advanced technological automotive era. Further, authenticity of ECU firmware update can be verified via signing/secure flashing mechanism.

Secure ECU to ECU Communication in In-Vehicle Network This layer ensures message delivery with integrity proof between ECUs. The hackers are targeting a large volume of sensitive information which is actually generated by The ECUs. The security mechanism should ensure the reliability, confidentiality, and integrity of this sensitive information. A hardware security module (HSM) chip should exist in the modern in-vehicle network that work as security controller. The security controller has three main modules, first module is responsible for ECU id authentication and ECU state verification, second module handles encryption/decryption of messages exchanged and third module control the secure flash storage. All messages from the ECUs should be forwarded to HSM chip (security controller). The security controller will analyze the messages and identify the destination ECU. Based on the state of the destination ECU and message security properties, it decides whether the message should be forwarded to the destination ECU or not.

Reliability and Privacy of Communication between Domains/Sub-Domains in In-Vehicle Network This layer ensures the reliability and privacy of communication between domains/sub domains in the vehicle. A vehicle domains/sub-domain reflects the grouping of functions and systems with respect to specified areas, for example telematics domain, infotainment domain, chassis domain, powertrain domain, sensor domain, and body domain, etc. This layer protects the domains/subdomains by using four security mechanisms as follows: # Message authentication system: Cryptographic certificate is used to ensure an authenticated sender and message integrity. This certificate is added to all messages in the network. # Encryption: Messages are encrypted inside the vehicles to guard against loss of data and identity theft since messages are distributed with several different ECUs. # Detection of intrusion: The corrective mechanism should use the cryptographic accelerators and security subsystems integrated with the microcontroller to guard against threats. # Validation at ECU level: In the vehicle network, ECU’s validity is checked first at the start of the engine and afterwards at specified time intervals

Application Software Reliability and Authenticity All application software should ensure reliability and authenticity qualities. The hackers are using software download/update features for attacks. The secure mechanism immediately updates the vehicle software whenever any security vulnerabilities is detected. The application software is generally developed by the third-party manufacturer and verified software file image is sent to original equipment manufacturer (OEM) trusted server for uploading. The software update manager module at OEM trusted server sends notification to vehicle owner for the availability of updated software version. After confirmation from the vehicle owner, the updated software version is installed in secure in-vehicle network storage area. In this entire mechanism, security of channels is a prime concern. There are various techniques exists for enhancing the security of channels, namely hash function, digital signatures, blockchain technologies, etc. Additionally, for verifying the trustworthiness of the encryption mechanism, modern advanced microcontrollers have features such as integrity testing in real time, etc.

Secure Gateway /Domain Controller This layer ensures the security of gateway/domain controller. All modern critical gateways must be equipped with next generation advanced security mechanisms such as advanced key management schemes/firewalls, intrusion detection schemes, etc. The main responsibility of the gateway is to maintain the secure configuration of the system. Context-aware routing is implemented by the gateway, in this routing mechanism, the gateway checks the validity of the messages, and all valid messages are permitted to transfer via the gateway to the corresponding destination through a different number of complex controls. The automotive manufacturers are working towards designing robust and secure gateway controller module. The cryptographic credentials are stored in the secure hardware extension (SHE) chip which is the part of secure gateway controller module. The main role of SHE chip is to protect the cryptographic credentials from the hackers. The public key infrastructures (PKI) of various communication service providers manages the cryptographic credentials inside the SHE chip

Secure External Communication This layer ensures secure communication from in-vehicle network to various services outside the vehicle. This layer provides authentication and message validation functionality for protecting message integrity and thereby protecting the communication channels from data manipulation and data theft. Due to these features, vehicle-to-everything communication, telematics, etc. are secured. All communications from in-vehicle communication stack Sensors 2022, 22, 6679 26 of 33 to roadside unit (RSU) must be established via trusted communication authority (TCA) for providing authentication and message validation. Certificates which are used during communication between in-vehicle communication stack and TCA, further between TCA to RSU should be changed frequently for maintaining the integrity of the sender. Additionally, the integrity of the message content can be protected with the help of digital signature.

**Open Issues and Future Directions:**

A diagram of a diagram

Description automatically generated

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.

* **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 issue.

* + All present approaches 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. 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. 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, or use instruction substitution to change their code by substituting equivalent instructions for some of them, or use code transposition to reorder the sequence of instructions in their code, or use subroutine reordering to obfuscate their code by randomly rearranging their subroutines. Consequently, malware can evade detection and avoid itself from being properly analyzed by employing such techniques. As a result, these approaches are unsuitable for use in intelligent vehicles due to concerns about passengers’ safety.
  + All of the current approaches 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. 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.
  + Malware detection in real-time is really challenge. The majority of malware detection approaches in the last decade 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. 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 that can detect malware in real time. However, these methods 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 are not cost-efficient and are not adaptable for intelligent vehicles since they need a sophisticated hardware modification. 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 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.
* **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.

* + **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.
  + **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.
  + **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). 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.
  + **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.
* **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.

* + **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. 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, key establishment using keyless cryptography technology and key distribution using the Light fidelity (Li-Fi). 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. Alongside with implementing authentication system, security criteria must be met in order to provide a successful and secure protection to the vehicle’s system.

* + **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. 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.

* + **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. 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. The future of intelligent vehicles is quite promising with deep learning using offloading computation mechanism towards faster and secure vehicle system.

* + **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. This form of dynamic solution to threats will provide security for intelligent vehicles against malware attacks.

* + **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, 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.

References:

[1] Scarfone, K., & Mell, P. (2007). Guide to intrusion detection and prevention systems (idps). NIST special publication, 800(2007), 94.

[2] Saxena, A., & Mueller, C. (2018). Intelligent Intrusion Detection in Computer Networks using Swarm Intelligence [J]. International Journal of Computer Applications, 179(16), 1-9.

[3] Azad, C., & Jha, V. K. (2013). Data mining in intrusion detection: a comparative study of methods, types and data sets. International Journal of Information Technology and Computer Science (IJITCS), 5(8), 75-90.

[4] Kazienko, P., & Dorosz, P. (2004). Intrusion detection systems (IDS) Part 2-Classification; methods; techniques.

[5] Spafford, E. H., & Zamboni, D. (2000). Intrusion detection using autonomous agents. Computer networks, 34(4), 547-570

[6] Denning, D. E. (1987). An intrusion-detection model. IEEE Transactions on software engineering, (2), 222-232.

[7] Lazarevic, A., Kumar, V., & Srivastava, J. (2005). Intrusion Detection: A Survey. Managing Cyber Threats, 19-78.

[8] Alsarhan, A., Alauthman, M., Alshdaifat, E., AlGhuwairi, A. R., & Al-Dubai, A. (2021). Machine Learning-driven optimization for SVM-based intrusion detection system in vehicular ad hoc networks. Journal of Ambient Intelligence and Humanized Computing, 1-10.

[9] Sharma, S., & Kaul, A. (2018). Hybrid fuzzy multicriteria decision making based multi cluster head dolphin swarm optimized IDS for VANET. Vehicular Communications, 12, 23-38.

[10] Ali Alheeti, K. M., Gruebler, A., & McDonaldMaier, K. (2016). Intelligent intrusion detection of grey hole and rushing attacks in self-driving vehicular networks. Computers, 5(3), 16.

[11] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, “Generating adversarial examples with adversarial networks,” arXiv preprint arXiv:1801.02610, 2018.

[12] Q. Wang, Y. Zhang, X. Lu, Z. Wang, Z. Qin, and K. Ren, “Real-time and spatio-temporal crowd-sourced social network data publishing with differential privacy,” IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 4, pp. 591–606, 2018

[13] N. Carlini, C. Liu, J. Kos, lfar Erlingsson, and D. Song, “The secret sharer: Measuring unintended neural network memorization & extracting secrets,” IACR Cryptology ePrint Archive, 2018

[14] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, “The architectural implications of autonomous driving: Constraints and acceleration,” in ACM SIGPLAN Notices, vol. 53, no. 2. ACM, 2018, pp. 751–766

[15] A. C. Berg and J. Malik, “Geometric blur for template matching,” in Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1. IEEE, 2001, pp. I–I.

[16] X. Chen, C. Liu, and D. Song, “Tree-to-tree neural networks for program translation,” in Advances in Neural Information Processing Systems, 2018, pp. 2547–2557.