***Existing security measures and their effectiveness:***

Over the past decade, scholars have delved into a broad spectrum of strategies aimed at safeguarding computer and mobile systems from malware. These strategies encompass signature-based, behavior-based, heuristic-based, cloud-based, and machine learning-based methodologies. In this segment, we offer an exhaustive examination of the key elements involved in implementing these protective measures to shield smart vehicles from malicious software. These elements encompass the chosen approach, data analysis techniques employed, targeted operating system, detection speed and response, data sources, as well as the primary merits and drawbacks associated with each defense mechanism. We divided the taxonomy dimensions into six categories. Furthermore, we provide concise descriptions of these categories below.

1. *Methodologies*: We categorize the prevailing methods for detecting malware into five distinct groups: signature-based, behavior-based, heuristic-based, cloud-based, and machine learning-based techniques. Each of these methodologies presents its own set of advantages and disadvantages, which we thoroughly examine to understand the strengths and limitations of each approach.
2. *Analysis Approaches*: The entirety of the detection procedure employs static, dynamic, and hybrid analysis methods. Below, we delineate each method:

Static Analysis: This methodology involves analyzing executable code without executing it. In static analysis, low-level information from codes is extracted through disassembly using disassembler tools. The primary advantage of this approach is its ability to unveil the code structure of the program without actual execution. Nonetheless, it may falter when confronted with analyzing unknown malware or detecting malware that employs obfuscation and evasion techniques within its code.

*Dynamic Analysis*: This malware analysis method involves executing the malware and observing its behavior, interactions with the host system, and effects on the host environment. The infected files are scrutinized within a simulated environment, such as an emulator, virtual machine, or sandbox, to render the environment imperceptible to malware. While this approach is effective in detecting malware, it may still fall short in identifying malware that utilizes obfuscated code and evasion techniques.

*Hybrid Analysis*: This malware analysis method integrates both dynamic and static analysis techniques. It comprehensively assesses the static attributes of malware code and augments them with behavioral characteristics to enhance the overall analysis process. While this method can effectively address the limitations of both static and dynamic analysis approaches, it may incur an increase in total execution time overhead.

1. *Target Operating System (OS)*: This term denotes the operating system under scrutiny by the system. It may include LINUX, Windows, or Android, depending on the context of the analysis.
2. *Detection Time*: This term denotes the duration between the occurrence of the analyzed event and the actual detection of the threat. Detection time can be categorized into real-time (online) detection, where automatic responses such as blocking the attacker and terminating the malware process are enabled immediately, or non-real-time (offline) detection, which occurs after the event has transpired.
3. *Detection Response*: This refers to the consequential action taken by the system upon detecting a threat. It can be passive, involving an event notification such as displaying an alert message, or active, involving an automatic reaction such as blocking the attacker or terminating the malware process.
4. *Data Source*: This indicates the origin of the input data analyzed by the system. It may include hosting logs, which comprise data from the operating system and system applications; application logs, which consist of data directly generated by applications; or network traffic, which encompasses data generated by the network layer.

***Detection Techniques:***

***Malware detection system taxonomy Signature-Based Malware Detection:***

The signature-based approach to malware detection, widely employed in commercial antivirus solutions, consists of two primary phases. Initially, a distinct signature is generated for each malware instance, obtained through a combination of manual and automated analysis of data sourced from networks and user devices. Subsequently, these signatures are stored on devices for identifying malware in files or data streams. This method involves dissecting and examining malware binary codes and is straightforward, rapid, and secure, particularly for smart vehicles. While it excels at detecting known malware, it falls short in identifying novel threats due to susceptibility to evasion. In contrast, novel malware detection methods focus on digital traces in various log files and utilize different analysis techniques, such as static analysis, function call graph similarity, and API call sequences. While accurate for known malware, these approaches are limited in detecting unknown threats and are unsuitable for real-time applications, such as those in intelligent cars.

Additionally, scientists have investigated several advanced methods for detecting malware, with a primary focus on the Windows operating system. These approaches involve examining program binary files without executing the code (static analysis) and utilizing distinct markers such as control flow graph signatures and byte sequences from executable files.

**Behavior-Based Malware Detection:** This approach examines a program's actions to determine whether or not it is malicious. It accomplishes this in a secure environment, such as a virtual computer, and doesn't rely on external frameworks for new, undiscovered malware. Many use this tactic to combat malware, observing typical behaviors and employing information indicators on various operating systems. Despite having a high locating rate, it is inappropriate for sophisticated vehicles due to its large costs and complexity. Although it is successful in identifying new malware, it struggles to accurately arrange all conceivable behaviors, which may lead to false positives or negatives. In contrast to recognition based on signatures, it is more difficult to implement and take seriously for in-vehicle gadgets, posing challenges for long-distance vehicles use.

**Machine Learning Based Malware Detection:** Artificial intelligence has long been a crucial tool in the fight against malware, utilizing algorithms with different specialties like logistic regression, Bayesian networks, and Naive Bayes. A number of variables, including feature relationships and data distribution, affect how effective these algorithms are. An increasingly powerful technique is Deep Learning, a subtype of artificial neural networks, particularly in image processing, speech recognition, and, more recently, malware detection. Its vulnerability to avoidance and evasion strategies, as well as the lengthy process of creating hidden layers, highlight the necessity for careful implementation.

**Heuristic-Based Malware Detection:** The heuristic-based method for discovering malware involves examining program files for suspicious attributes or simulating program execution to identify potential malicious activities. This approach, renowned for its complexity, relies on past experiences and utilizes techniques such as data mining, rule-based systems, and AI to understand program characteristics. Widely used in antivirus software, it is capable of detecting various known and unknown malware, including zero-day threats. However, it struggles with identifying the most new and sophisticated malware and is susceptible to advanced techniques such as code obfuscation and evasion. Researchers have suggested static analysis techniques, such as control flow charts, and dynamic methods using DLLs or API call patterns. Although effective for known malware, these approaches are intricate, have high false positive rates, and are not suitable for continuous detection due to their time-consuming nature. Despite its strength in identifying unknown malware, the heuristic-based method is challenging and resource-intensive compared to signature-based and behavior-based strategies. It may not be suitable for resource-constrained in-vehicle devices due to their complexity and potential obsolescence over time.

**Cloud-Based Malware Detection:** Distributed computing has gained popularity due to its convenient accessibility, on-demand capacity, and cost-effectiveness. Recently, it has been employed in malware detection through the Cloud-based approach, utilizing agents located on cloud servers. This method enables users to submit files for analysis and receive reports on their malware status. While it enhances detection performance with extensive databases and computational resources, it encounters drawbacks such as reliance on a stable and fast internet connection, vulnerability to continuous file monitoring, and susceptibility to obfuscation and evasion techniques.

Researchers have explored cloud-based strategies for malware analysis, utilizing static analysis with features such as document content and relationships, as well as dynamic analysis through system call monitoring. However, these approaches incur significant costs and time delays, rendering them inadequate for continuous detection, particularly in smart vehicles. Despite the advantages of rapid access and updated installations, the cloud-based approach is hindered by the need for a reliable internet connection and vulnerability to advanced evasion techniques, raising concerns about its safety in smart vehicles. The emergence of high-speed 5G technology might enhance its feasibility in this specific context.

**Post-Protection Methods:**

The effectiveness of post-protection techniques such as attack identification and secure patch countermeasures can enhance the security of automobile CAN. This might be accomplished, for instance, by using an attestation service or a data recording system, which are both regarded as crucial components of a digital inquiry approach. Secure patches that restore the compromised ECU's firmware to its uninfected state are started as soon as the compromised ECU is found. The following is a discussion on secure patching and attack identification for automobile CAN.

**Attack Identification:**

Although a lot of work has gone into keeping malicious actors out of the automotive CAN, it is impossible to stop every kind of unidentified assault. Finding a new assault and a compromised ECU is therefore important, and this can be done via data logging CAN traffic. When an attack takes advantage of a stealth characteristic like Stuxnet, data recording by itself is insufficient to identify a compromised node completely; therefore, attestation could also be used to identify a compromised node.

I) **Data Logging:** For a data recording system, an information set consisting of CAN ID, packet arrival time, and domain ID (such as powertrain, comfort, and infotainment) should be well maintained. But because an attack attempt could readily taint this data set, it is insufficient to identify the source ECU from which a genuine automotive CAN assault was carried out. Stated differently, this set lacks unique data, such as digital fingerprints, such as electric CAN signal information. For instance, if a compromised ECU carried out message flooding assaults with the CAN ID of 0 × 00, details regarding the CAN ID and packet arrival time recorded in the logging system might be utilized to examine the characteristics (such as attack packet frequency) of the message flooding attacks. The logging system still has trouble figuring out which ECU has been used in message flooding attacks. One crucial and required step in a quick and effective response to an attack is identifying a hacked ECU. From there, a security patch or isolation technique can be applied to the affected ECU. A voltage-based ECU identification system that would use voltage measurements to fingerprint ECUs' CAN transceivers will be used, in order to satisfy this testing on two actual cars that their logging system can detect a corrupted ECU with a mere 0.2% false positive rate. To guard against forgery attacks, a trusted execution environment (TEE)-based logging system was recently proposed.

II) **Attestation:** The verifier is tasked with attestation of all ECUs on automotive CAN. to a particular device or another ECU to verify the integrity of the firmware that is placed on every ECU. As a result, an attestation code and a distinct attestation key ought to be installed in each ECU of a vehicle at production, together with a specific ECU for verification. When the ignition is turned on, the ECU firmware attestation procedure can be initiated, or it can run automatically anytime the ECU firmware has to be verified.

**Segmenting the network:**

One of the main safety measures is to implement network division by dividing the CAN system into subnetworks. This allows for control over access and limits the potential for assaults to spread. The standard procedure in business cars involves an Electronic Control Unit (ECU) on the door that controls connections between subnetworks. However, vulnerabilities arise if the door ECU is breached, as demonstrated in certain hacking scenarios.

**Encryption:**

The implementation of a lightweight encryption system is essential since the CAN convention exposes its correspondence to potential enemy eavesdropping in the absence of implicit encryption. Although there are producer-only techniques and encryption strategies based on business programming, publications point to vulnerabilities in commercially available car encryption frameworks. Difficulties with security Could encryption ever include the field for restricted information. That would be problematic for organisations with huge traffic volumes, but it could be handled by providing different CAN outlines for a single message. Furthermore, dynamic key trading is necessary due to the computing requirements of ECUs to prevent static key splitting during the course of a vehicle's life. However, dynamic key trading is unsuitable for security-critical continuous frameworks due to its execution difficulties, computational expenses, and idleness problems for asset-compelled ECUs.

**Secure Boot:**

The concept of Secure Boot is not unique to the automobile sector; in fact, the majority of home PC BIOSes support it. It is a Unified Extensible Firmware Interface (UEFI) method that restricts the bootable software on the machine to only those with legitimate signatures. During the manufacturing process, the manufacturer inserts databases containing keys and signatures inside the device. Before booting, the firmware would next be signed and verified by signature. Although malicious firmware cannot be booted thanks to the incredibly powerful security technique known as safe booting, there have been instances in the past when certain implementations' flaws have allowed for bypasses, therefore secure booting is not a foolproof solution.

**Safe Access Program:**

As the scientists worked on a 2010 Toyota Prius and a 2010 Portage, they investigated the diagnostics validation component. Break, exposing an ISO 14229-1-characterized Security Access system implanted within the Bound together Diagnostics Administration (UDS). For Electronic Control Units (ECUs) in the automotive industry, UDS serves as a standard diagnostics correspondence convention. It provides many services to collect information on a vehicle's utility and condition. Using a Test Reaction protocol, the Security Access administration requires a "seed" from the ECU, which is then returned to it after it has reached adulthood. An ECU key and a cryptographically secure capability are shared by both the analyzer and the ECU. However, the free standard requires interesting points like the seed age, keys, or competence. Remarkably, vulnerabilities were identified, such as predictable seeds and short reaction times, which let attackers to use beast power or replay attacks. The specialists were able to separate keys efficiently by identifying and locating security flaws in these auto frameworks.

***Secure Onboard Communications***

To address the authentication shortcomings in the Controller Area Network (CAN), the AUTOSAR community introduced the Secure Onboard Communications (SecOC) module. This module enhances CAN message authentication by adding signatures to in-vehicle communications. SecOC supports both symmetric and asymmetric cryptography, assuming that key management and exchange are already established. The approach involves appending a signature, along with a freshness value for uniqueness, to the protected data unit (PDU) within the CAN frame.

In symmetric mode, for instance, the sender computes a Message Authentication Code (MAC) over the input data and freshness value using a shared key, attaching it to the message. The receiver verifies the freshness value and recalculates the MAC, accepting the message if correct. However, adapting this system to a standard CAN framework presents challenges, such as truncating the 128-bit MAC into a 27-bit value, potentially vulnerable to brute force attacks.

Additionally, SecOC introduces measures to safeguard against replay attacks by including freshness values in the data packets. These values help to ensure that each message is unique and has not been intercepted and replayed by an attacker.

An alternative approach, the Plug and Secure Key Establishment (PnS), offers a low-cost symmetric key establishment protocol for CAN. It utilizes a physical property in the bus to establish a key between two devices whenever they communicate a frame simultaneously, enhancing security within the network.

***Firewall Gateway***

Enhancing security in in-vehicle communication systems may involve implementing a firewall mechanism within network gateways. If Message Authentication Codes (MACs) or digital signatures are utilized for authentication and validation between Electronic Control Units (ECUs), firewall policies can be derived from the permissions specified in each ECU's certificate. In cases where MACs or digital signatures are absent, firewall rules can be individually defined based on vehicular subnet permissions, allowing only messages from legitimate and authenticated ECUs to pass through and be transmitted across the in-vehicle communication system.

Another approach is to restrict the access level of different network types to specific segments of the communication system, preventing less critical networks, such as LIN or MOST, from sending messages to higher safety-critical systems, such as CAN or FlexRay. This helps in segregating and protecting critical communication channels from potentially less secure ones, thereby enhancing overall system security.

***Honeypots***

Securing in-vehicle communication systems is crucial due to the vulnerability introduced by wireless communication gateways. Attackers can exploit remote access to launch cyber-attacks directly on the in-vehicle network, which controls essential vehicle functions. Understanding attackers' behavior is essential for developing effective security solutions. Honeypots, designed to appear as vulnerable targets, serve as tools for the prevention and early detection of malicious attacks. In the automotive sector, realistic honeypots can be deployed to attract genuine attackers without disrupting normal vehicle operations. These honeypots collect data as the vehicle travels through predetermined areas, capturing information affecting the in-vehicle network. Analysis of this data identifies attack patterns, scenarios, and commands, aiding in the development of robust security measures to protect in-vehicle communication systems in future designs and implementations.

1. Jo, Hyo Jin, and Wonsuk Choi. “A Survey of Attacks on Controller Area Networks and Corresponding Countermeasures.” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, July 2022, pp. 6123–41. *IEEE Xplore*, https://doi.org/10.1109/TITS.2021.3078740.
2. Alkhateeb, Omar. *Controller Area Network Attacks and Defense Mechanisms: Survey Created By*. 2020. *DOI.org (Datacite)*, https://doi.org/10.13140/RG.2.2.25052.82561.
3. Sedar, Roshan, et al. “A Comprehensive Survey of V2X Cybersecurity Mechanisms and Future Research Paths.” *IEEE Open Journal of the Communications Society*, vol. 4, 2023, pp. 325–91. *IEEE Xplore*, https://doi.org/10.1109/OJCOMS.2023.3239115.
4. *A Survey and Comparative Analysis of Security Properties of CAN Authentication Protocols*. https://arxiv.org/html/2401.10736v1. Accessed 22 Mar. 2024.
5. “Cybersecurity Best Practices for the Safety of Modern Vehicles.” *Federal Register*, 9 Sept. 2022, https://www.federalregister.gov/documents/2022/09/09/2022-19507/cybersecurity-best-practices-for-the-safety-of-modern-vehicles.
6. Zhang, Haichun, et al. “A Cyber Security Evaluation Framework for In-Vehicle Electrical Control Units.” *IEEE Access*, vol. 9, 2021, pp. 149690–706. *IEEE Xplore*, https://doi.org/10.1109/ACCESS.2021.3124565.
7. Adly, Salah, et al. “Prevention of Controller Area Network (CAN) Attacks on Electric Autonomous Vehicles.” *Applied Sciences*, vol. 13, no. 16, Jan. 2023, p. 9374. *www.mdpi.com*, https://doi.org/10.3390/app13169374.
8. Dibaei, Mahdi, et al. “An Overview of Attacks and Defences on Intelligent Connected Vehicles.” *arXiv.Org*, 17 July 2019, https://arxiv.org/abs/1907.07455v1.
9. Lin, Chung-Wei, and Alberto Sangiovanni-Vincentelli. “Cyber-Security for the Controller Area Network (CAN) Communication Protocol.” *2012 International Conference on Cyber Security*, 2012, pp. 1–7. *IEEE Xplore*, https://doi.org/10.1109/CyberSecurity.2012.7.