Cluster Computing
https://doi.org/10.1007/s10586-023-04114-7

Cybersecurity for autonomous vehicles against malware attacks in smart-cities

Sana Aurangzeb[1,2] · Muhammad Aleem[2] · Muhammad Taimoor Khan[3] · Haris Anwar[2] · Muhammad Shaoor Siddique[2]

Received: 21 November 2022 / Revised: 19 July 2023 / Accepted: 27 July 2023

Abstract
Smart Autonomous Vehicles (AVSs) are networks of Cyber-Physical Systems (CPSs) in which they wirelessly communicate with other CPSs sub-systems (e.g., smart -vehicles and smart-devices) to efficiently and securely plan safe travel. Due to unreliable wireless communication among them, such vehicles are an easy target of malware attacks that may compromise vehicles' autonomy, increase inter-vehicle communication latency, and drain vehicles' power. Such compromises may result in traffic congestion, threaten the safety of passengers, and can result in financial loss. Therefore, real-time detection of such attacks is key to the safe smart transportation and Intelligent Transport Systems (ITSs). Current approaches either employ static analysis or dynamic analysis techniques to detect such attacks. However, these approaches may not detect malware in real-time because of zero-day attacks and huge computational resources. Therefore, we introduce a hybrid approach that combines the strength of both analyses to efficiently detect malware for the privacy of smart-cities.

Keywords Malware detection · Security · Smart cities · Autonomous systems

1 Introduction

Recently Autonomous Vehicular Systems (AVSs) have seen a gigantic growth in a wide variety of aspects with the development of smart cities to build the Intelligent Transport Systems (ITSs). For instance, the dramatic use of embedded systems and wireless communication (e.g., 4 G LTE and 5 G) in modern internet of vehicles which ultimately improve users safety and comfort. However, growing interest in the development of Connected Autonomous Vehicles (CAVs) and ITSs has introduced new security challenges and vulnerabilities in AVSs that has a great impact on the smart environments for smart-cities. However, classical computer security solutions are not applicable in automotive industry standards for in-vehicle, vehicle-to-vehicle (V2V) communication and vehicle to everything (V2X) communications mainly because of real-time performance requirements, constrained computational resources, and differences among heterogeneous networks and their configurations [1].

Various recent reports have sketched attempts where cybercriminals have successfully demonstrated practical but remote attacks to key functions of automotive vehicles (as depicted in Fig. 1) either through V2V or V2X that include disconnecting the engine and the brakes [2–5].

CryptoLocker, WannaCry, and Petya attacks are prominent one of the most widely used attacks against

✉ Muhammad Aleem
m.aleem@nu.edu.pk

✉ Muhammad Taimoor Khan
m.khan@greenwich.ac.uk

Sana Aurangzeb
sanaaurangzeb@numl.edu.pk

Haris Anwar
harisanwar64@gmail.com

Muhammad Shaoor Siddique
i212806@nu.edu.pk

[1] Department of Computer Science, National University of Modern Languages, Islamabad, Pakistan

[2] National University of Computer and Emerging Sciences, Islamabad, Pakistan

[3] School of Computing and Mathematical Sciences, University of Greenwich, London, UK

Published online: 03 October 2023

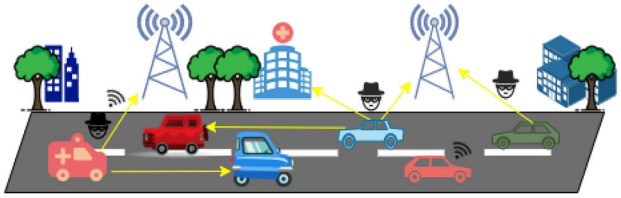

Fig. 1 Typical V2V, V2X cyber threat scenario in smart autonomous vehicles

sensitive IT systems [6]. In the past, ransomware attacks used to affect various entities such as personal computers, public or private organizations, health sectors, mobile phones, and other similar devices. However, the focus of ransomware attacks has now shifted towards smart vehicles and smart cities, posing a significant threat to both human lives and financial stability [7, 8]. Moreover, there have been attempts where researchers have shown that malware is one of the keys and emerging security threats that can be launched by exploiting the wireless communication system of AVSs [9, 10]. For instance, by exploiting known vulnerabilities in the design and implementation of onboard communication systems, embedded software, and application software [11–13] as depicted in Table 1. Moreover, a report in [2, 14, 15] has shown that an AVSs is not just a simple machine by hijacking the steering and brakes of a Ford Escape and a Toyota Prius. However, on the other hand, it is of utmost critical to understand that AVSs are now a network of computers that can be hacked by practicing classical cyber threat mechanisms. For instance, during the year 2015, approx. 1.5 million vehicles were subject to a recall by Daimler Chrysler mainly because cybercriminals could remotely take the control of a jeep's digital system over the Internet [3]. In another report [4], a team of cybercriminals remotely hijacked a Tesla Model S from a distance of approx. a dozen miles. In a more recent attempt [5], authors have identified 14 vulnerabilities in the infotainment system in several of BMW's series. Moreover, another Tesla S and Tesla X was targeted by cybercriminals in November 2019 via the Wi-Fi attack vector [6]. All of the above-mentioned incidents show that the security of AVSs is integral to their core functions in order to make smart transportation secure, therefore, it must be handled to protect the vehicles enabling them to operate safely.

The key to the afore-mentioned success of remote attacks on AVSs is information sharing by the vehicles over a wireless medium which increases the susceptibility of the vehicles to different security and malware attacks. Consequently, data exchange including input and output data as well as protecting Electronic Control Unit (ECUs) inside the AVSs are among the most significant security issues for the intelligent vehicles [10, 19]. Specifically, the most damaging cyber threats, are emerging as the vehicles connect to the Internet, provide onboard Wi-Fi hotspot services, communicate with other vehicles and ITSs infrastructures, and support advanced applications such as over-the-air (OTA) ECU firmware update [9]. As discussed above, many modern attacks do not require physical access to a vehicle instead can now be carried out remotely over wireless by exploiting communication vulnerabilities among vehicles and other connected network services. This allows attackers to compromise more vehicles with relative ease whereas later a compromised vehicle can also be used to attack other vehicles.

Considering the performance requirements of AVSs, it is important to detect a malware in real-time to timely protect any physical and financial damage and loss of human lives [20, 21]. Current approaches to detect such malware either employ static analysis or dynamic analysis techniques [22, 23]. Static analysis technique include:

Table 1 Various attacks to CAVs

Attack target	Attack description
Vehicle's actual behavioral disruption	Locking the in-vehicle radio so that the users cannot turn it on [16]
Driver distraction	Misusing vehicle features to distract the driver by arbitrarily turning on the in-vehicle audio and tuning its volume [14, 16]
Locking vehicle	Locking vehicle features resulting in jackware [17]
Externally connected devices	Modifying files on the vehicle and on users brought-in devices connected to the vehicle [14]
Computational resources of the vehicle	Consuming computational resources (such as memory space and CPU cycles) to disrupt vehicle actual operations [15]
Sensitive and private data	Stealing private and sensitive data [18]
Passengers safety	Threatening passengers lives by disabling vehicle safety functions
CAVs	Using the compromised vehicle to send misleading, false, and bogus data to CAVs

signature-based detection techniques that uses predefined patterns or signatures to identify known malware based on specific characteristics or sequences of code [24–28], *heuristic analysis* involves using predefined rules or algorithms to identify potentially malicious code by analyzing its structure, behavior, or attributes [26, 29–31]. *Code structure analysis* focuses on analyzing the structure and syntax of the code to identify suspicious or malicious patterns that may indicate the presence of malware [32–35], *String analysis* involves analyzing the strings or text within the code to detect hardcoded URLs, IP addresses, encryption keys, or other indicators of malicious behavior [36–40], *Metadata analysis* involves examining the file's metadata, such as file size, creation date, or digital signatures, to identify anomalies or signs of tampering [40–43], *Control flow analysis* technique analyzes the flow of instructions within the code to detect any unusual or malicious behavior, such as code obfuscation, anti-analysis techniques, or hidden functionality [44–47] whereas *Sandbox analysis* involves running the code or file in a controlled environment (sandbox) to observe its behavior, monitor system interactions, and detect any malicious activities or suspicious network communications [39, 41, 48, 49]. Just like static analysis, the most common dynamic analysis technique includes: *Behavior analysis* that involves monitoring the runtime behavior of the code or file to identify any suspicious or malicious activities, such as unauthorized system modifications, file system changes, or network communications [41, 44, 50], *API monitoring technique* focuses on monitoring the interactions between the code and Application Programming Interfaces (APIs) to detect any abnormal or malicious API calls that may indicate malicious intent[48, 51, 52]. *Network traffic analysis* involves capturing and analyzing the network traffic generated by the code or file during execution, looking for communication with known malicious servers, unusual data transfers, or suspicious network behavior [53–57]. *Dynamic code analysis* involves analyzing the code's behavior during runtime, including function calls, memory operations, and system calls, to identify any malicious or suspicious activities [41], *System call monitoring* focuses on monitoring the system calls made by the code or file to the operating system, detecting any unusual or unauthorized system calls that may indicate malicious behavior [58–60], *Sandboxing* involves executing the code or file in a controlled virtual environment (sandbox) to observe its behavior while isolating it from the host system, thus preventing potential harm to the system [39, 41, 48, 49], *Emulation and virtualization* emulates or virtualizes the target system environment to execute the code or file, allowing for the analysis of its behavior and interactions without directly affecting the host system [61–64]. These techniques, often used in combination, help in identifying and classifying malware, enhancing the security of systems and networks. The former techniques are good at detecting active malware, i.e., the malware that is directly targeting some unauthorized resource or feature of the vehicle, however, such techniques fail to detect any passive malware that exploits some system vulnerability through monitoring run-time data of the vehicle. The latter techniques are more robust and rigorous as they can detect any variant of malware through observing run-time behavior of systems [65] but such approaches typically require more computational resources which is not the case in autonomous vehicles. As autonomous vehicles may require less computational resources compared to other applications in a way that they have specialized hardware such as Application-Specific Integrated Circuits (ASICs) or Graphics Processing Units (GPUs), designed to efficiently handle the specific computations required for autonomous tasks. Furthermore, with the development of advanced algorithms and machine learning techniques specific to autonomous vehicle have enabled more efficient processing of data. Lastly, many autonomous vehicle systems leverage cloud computing capabilities to offload intensive computational tasks. By utilizing remote servers with powerful computing resources, the computational load on the vehicle itself can be reduced. This approach enables the vehicle to rely on the cloud for resource-demanding tasks like high-definition mapping, complex route planning, or deep learning-based processing such as malware detection [66–68]. Therefore, the proposed hybrid approach (utilizing both static and dynamic techniques) can help in detecting malware by leveraging the advantages of both approaches in a single model. Alternatively, some approaches attempted to install vehicle gateways that allow only authorised communication to the vehicles and introduced vehicle Intrusion Detection Systems (IDSs) to detect abnormal behaviors in the Controller Area Network (CAN) [69]. However, it is difficult for a gateway or IDS to block these actions in advance, as most malware and adware are behavior-based. Therefore, to detect unknown malware threats, it is vital to introduce a methodology that can detect suspicious behaviors and analyze anomalous indicators rigorously (i.e., negligible false alarms) and efficiently (i.e., in real-time).

The rest of the paper is structured as follows: Sect. 2 provides background of autonomous vehicles, while Sect. 3 sketches state of the art about rigorous malware detection techniques. Section 4 explains our malware detection methodology, while Sect. 5 presents experimental setup, experiment results and critical discussion. Finally, we conclude in Sect. 6.

2 Background and motivation

Modern smart AVSs will strikingly change the worldwide transport industry and smart environments. AVSs where improving the standard of smart living and road safety also require to wirelessly communicate with other vehicles and devices to efficiently and securely plan safe travel. The number of traffic accidents are reducing day by day. In Addition, people with disabilities can significantly taking advantage from smart cities and ITSs technology preventing injuries and deaths in combat [70]. However, due to unreliable wireless communication among them, such vehicles are an easy target of malware attacks that may compromise vehicles' autonomy, increase inter-vehicle communication latency, and drain vehicles' power. Such compromises may result in traffic congestion, threaten the safety of passengers, and can result in financial loss. Therefore, real-time detection of such attacks is key to the safe smart transportation and ITSs. With the increasing trend of Internet of Things (IoT), ITSs aims to improve the efficiency and safety of AVSsnetwork [71]. ITSs in societies that are converting into smart cities becomes more vulnerable to cyber-threat and cyber-terrorism [72]. Different types of ITSs are vulnerable to attacks. The success of remote attacks on autonomous vehicles is information sharing by the vehicles over a wireless medium which increases the susceptibility of the vehicles to different security and malicious attacks. Consequently, data exchange including input and output data as well as protecting ECUs inside the AVSs are among the most significant security issues for the intelligent vehicles. ECUs are the embedded system that monitors electrical systems or subsystems in a conventional vehicle for instance the energy conversion, the air conditioner, vehicle speed and the warnings on the instrument panel [73].

An AV is not just a massive car with four wheel but is made up of networked embedded computers that are responsible for performing different tasks in a smart and timely manner. Therefore, an AV is a diverse and complex environment that comprises of several types of Operating System (OS) installed among different vehicles as shown in Fig. 2. Although ECU act as a brain for AVSs and is considered as minicomputers yet they vary in size, purpose and the OS they run. Thus, we can divide ECUs into two categories: managed by realtime operating systems (RTOS) and general purpose operating system (GPOS). Other than that, Robotic operating system (ROS) is also used. ROS is not an operating system but is an open-source robotics framework having collection of software for robot software development. Tesla, a leading automotive car vehicle is a new energy innovation owns a self-developed OS [74] is now testing Windows OS [75] and Tesla patent seems to be working on windows operating system [76]

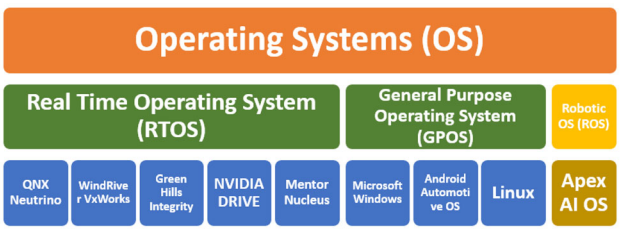

Fig. 2 Types of operating systems (OS) used in smart autonomous vehicles

Numerous research endeavors focus on utilizing Machine Learning (ML) techniques to identify malware that exploits the dynamic or runtime aspects of running applications. These efforts employ classification methods, considering diverse features like Windows API calls, Registry Key Operations, File System Operations, File Extension-based operations, Directory Operations, Dropped Files, and Strings to classify malware. In addition to static and dynamic approaches, contemporary practices involve utilizing Hardware Performance Counters, which accurately reflect the execution behavior of the application, to measure the performance of the software under investigation. [77]. However, none of the existing dynamic and ML malware detection techniques use hardware performance counter for malware classification specifically in autonomous vehicles. Although, however, [78] employs a dynamic approach to classify malware based on their hardware performance counters and [79] have used hardware performance counter for ransomware classification on Windows platform. There exists no such work that considers all these important aspects in a single methodology. We believe that collective consideration of all of the above stated aspects can significantly improve malware detection rates in AVSs. Therefore, this study encompasses efficient malware detection mechanisms in terms of a hybrid approach that utilizes static as well as dynamic analysis focuses on the use of hardware performance counters to analyze the runtime behavior to detect malware. Moreover, this work shows how accurately hardware performance counters are able to classify malware in AVSs.

3 Related work

Numerous static and dynamic analysis techniques have been presented by the scholarly community to detect and classify malware. Both of the techniques, static and dynamic have their own benefits and limitations. This section depicts state-of-the-art techniques that pertain to malware analysis.

In [80] authors have proposed the analysis of malware on X86-based IoT devices in an autonomous driving

approach features based on static analysis and using machine learning to solve problems of resource overhead for dynamic analysis. Paper [81], authors have used Bayesian Network (BN) model to analyse cyber risk in AVSs by introducing the variables and causal relationships derived from the Common Vulnerability Scoring Scheme (CVSS). The model is then applied on the GPS system of the connected AVSs without cyptographic authentication.

Beside other malware attacks, ransomware attacks are emerging and their analysis are used widely by the scholarly community now-a-days. In [82], the authors presented a case study of CryptoLuck Ransomware to highlight the importance of behavioral-based Ransomware detection. In [83], authors statically analyzed process monitoring on file events, processor usage, and I/O rates. In [84], authors suggested that static detection technique as used by [85], can help in evading anti-virus (AV). In [86], authors performed ransomware behavioral analysis on windows platform of 14 strains of ransomware. They observed the individual behavioral pattern of ransomware. In [87], authors presented an automated detection and analysis of ransomware to monitor dynamic behavior by generating API calls and Control Flow Graph (CFG). Authors in [88], developed a dynamic analysis system (UNVEIL), designed specifically for the detection of ransomware by automatically generating an artificial user environment.

There are several other research efforts which follow Machine Learning (ML) based approaches to detect malware exploiting the dynamic or runtime features of executing applications. Another proposed study of dynamic analysis using machine learning through monitoring file system activity of windows platform was conducted by [84]. They used classification technique by considering a wide range of features such as Windows API calls, Registry Key Operations, File System Operations, file operations performed per File Extension, Directory Operations, Dropped Files, and Strings to classify malware.

Other than static, dynamic and ML approaches, Hardware performance counters (represent the true execution behaviors of the application) are typically being employed nowadays to measure the performance of the under investigation software [77]. However, none of the existing dynamic and ML malware detection techniques use hardware performance counter for malware classification specifically in autonomous vehicles. Although, however, [78] employs a dynamic approach to classify malware based on their hardware performance counters and [79] have used hardware performance counter for ransomware classification on Windows platform.

It has been observed from the literature work that most of the techniques [84] can either only observe System/API calls [86, 87, 89], file operations [88], processor usage [83], or registry activities [90]. Some of the studies are based on static analysis [82] whereas other proposed techniques mainly focus on dynamic analysis for classification. A lot of solutions have been developed against malware and ransomware as well as ransomware classification among families that significantly improve the user's security. A few researches [91–94] have shown that there is a lack of behavioral analysis that use hybrid technique to classify malware in AVSs using API Calls, File operations, Registry keys, and Hardware performance counter based features (i.e., processor usage, cache-misses, memory usage, page faults, instructions, branches, etc.). So far, hardware-based features have been analyzed on malware and non-malware apps, but have not been considered for AVSs. There exists no such work that considers all these important aspects in a single methodology. We believe that collective consideration of all of the above-stated aspects can significantly improve malware detection rates in AVSs. Therefore, this study encompasses efficient malware detection mechanisms in terms of a hybrid approach that utilizes static as well as dynamic analysis focuses on the use of hardware performance counters to analyze the runtime behavior to detect malware. Moreover, this work shows how accurately hardware performance counters are able to classify malware in AVSs.

4 Methodology

Autonomous Vehicles (AVSs) have become a core constituent of the smart transportation system [95]. The computation power of AVSs gradually increasing and a large amount of information exchange is required with smart components of the transportation system. Information exchange with malicious counterparts in the smart systems could produce catastrophic results such as a change of drive-plan, sudden halt, and ignore obstacles on the roads. Generally, malware exploits different vulnerabilities of the computer system (i.e., hardware platform, operating system, and application software). However, considering the drastic implications of the malicious activity in AVSs, we should formulate a holistic approach considering handling precision, vehicle efficiency, and digital security.

With the static-analysis, malware detection can take place efficiently by merely matching the known application features such as signatures (before application execution) requiring few computational resources. Therefore, static analysis provides early detection to mitigate malicious activities during autonomous vehicle operation. However, the static analysis does not encompass the zero-day attacks and obfuscated (hidden or purposefully crafted features such as like packed or compressed programs or indirect addressing [96]) malicious applications. To address these

issues, a dynamic analysis based mechanism can be employed that exploits the run-time behavior (including system hardware, operating systems interactions, etc.) of the executing applications to classify and detect malicious behavior. However, the proficient detection capabilities of the dynamic analysis come along with the high-resource consumption (CPU, memory, energy-cost, etc.). Additionally, in the AVSs context, it would be too risky to rely directly on the dynamic analysis because of potentially high false-positive detection as compared to static analysis.

Therefore, this study encompasses efficient malware detection mechanisms in terms of a hybrid approach that utilizes static as well as dynamic analysis. Traditionally, the proposed models can be built using basic hybrid mechanisms, i.e., (i) a single hybrid approach where distinctive aspects related to both pre-/in-execution of the applications are obtained for analysis and detection. For the obligatory requirements such as efficient and thorough detection of malware with reduced false-positive rate, the hybrid-approach is appropriate and recommended.

The proposed security modules for AVSs i.e., the hybrid mechanisms Combined Hybrid Analyzer (CHA) is shown in Fig. 3. CHA adheres to a factual technical concept of using a hybridization concept for bringing together heterogeneous parameters (in terms of the execution requirements i.e., pre-/in-execution based parameter extraction). As discussed above, the utilization of this model has certain operational consequences that hinder its practical use.

Let's discuss the architecture of these models in detail. The proposed CHA model considers input applications and data to employ both pre-/in-execution feature extraction simultaneously. The specific features extracted can be divided into two categories, i.e., static-analysis based features (which can be extracted without application execution), and dynamic features are extracted during the execution of the application within an operating system. The static features include embedded command-strings and the usage of operating system manipulating libraries. The dynamic features (extracted during the execution) are the activity logs related to system-wide low-level configuration manipulations, invoking system call interface to gain privileged access, and manipulation of the operating system resources, file-system related activities, and hardware execution profiles (i.e., low-level hardware performance counters). After performing static and dynamic analysis of malware, the information extracted is in the form of raw data. This data can be converted into CSV (Comma-Separated Values) format using appropriate data processing techniques (such as parsing and extraction, data transformation, delimiter handling etc.,). We have extracted the relevant information that involves identifying specific patterns or structures within the data and extracting the desired fields or attributes. This can be done using specialized parsing libraries that can assist in this process. Furthermore, that data is normalized by handling missing values. Later, the file was converted in CSV format. Various programming languages, libraries, or data processing software like Python with pandas, R, or Excel can be utilized to facilitate the conversion of raw data into CSV format, where each attribute or feature corresponds to a column, and each record represents a row in the CSV file. These features are then combined in feature vectors to be used for both training and validation purposes. The Machine-Learning (ML) model training and validation strategies along with feature selection mechanisms are discussed in Sects. 4.2, 4.3 and 4.4. The machine learning model i.e., J48, Naive Bayes (NB), Gradient Boosting, XGBoost and Random Forest (RF) are used to classifying the applications into malware and non-malware classes. The reason of using these machine learning classifiers are that their results depict are better and efficient in terms of time and computational complexity. Moreover, classifiers like J48, which are more suitable for categorical and mixed data, Naïve Bayes which is often considered one of the fastest classifiers due to its simplicity and computational efficiency. It typically has faster training and prediction times compared to more complex models like random forests or gradient boosting. However, classifiers like Gradient Boosting and XGBoost are generally more computationally intensive and can take more time to train and make predictions, especially when dealing with AVSs which require quick response. These algorithms involve building an ensemble of weak models iteratively, which can be time-consuming compared to simpler algorithms like Naive Bayes, J48. Random Forest and Gradient Boosting (including XGBoost) are often recognized for their high accuracy in classification tasks like malware classification. For IoT related malware detection

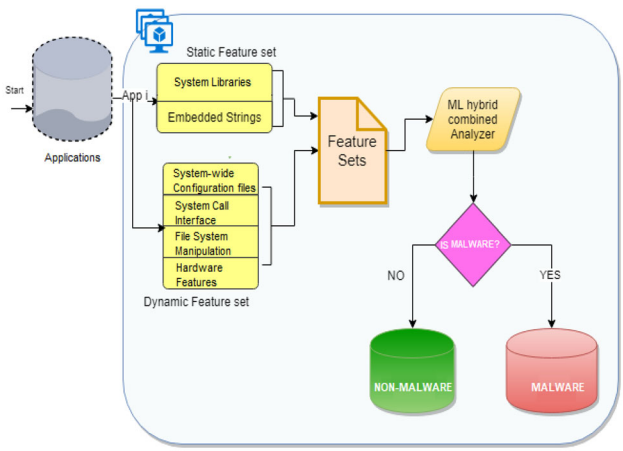

Fig. 3 Combined hybrid analyzer (CHA)

Table 2 Microsoft windows based services for automotive vehicles

Automotive brand	Goal	Windows based services
Porsche Holding	Mobile-first and Virtual Workplace	Office 365 [97, 98] Teams [99] Microsoft's Cloud Services [100]
Brimborg	Online Stream-based Services for Rentals	Microsoft Dynamic 365
Mercedes-Benz	Connected Cars Platform	Microsoft's Cloud-based containerized platform [100] Azure Monitor [101]
Moovit	Real-time in-city and out-city transits Mobility-as-a-service	Azure Maps [102]
Daimler	Detroit Connect platform Virtual Technician Remote Updates Remote Analytics	Azure [103–105] Microsoft's cloud computing service [104]

algorithms complexity should be lesser as IoTs have battery consumption problems.

For the initial investigation and proof of the concept, we have used a dataset of executable applications MS windows platform. We have chosen Windows based dataset for several reasons, for instance, most of the major initiatives in automotive vehicle industry use Windows based services (see Table 2) for their live communication, which is certainly a key source of threat to such services and eventually to the vehicles [106, 107]. Furthermore, as reported in [108], Microsoft services and platforms are helping automakers to create smart connected car solutions that seamlessly address their customers' unique needs, competitively differentiate their products and generate new and sustainable revenue streams. The Microsoft services do not only offer the right tools, but also allows them to keep their data, has a secure and compliant cloud platform, and operates at a truly global scale (given that most automotive brands operate in many countries). Importantly, 85% of Fortune 500 companies already rely on Microsoft's cloud for the afore-mentioned reasons. In principle, using such platforms, automakers and suppliers can benefit from the billions of dollars that Microsoft has already invested in the cloud services. For instance, Azure already offers more than 200 services in 38 worldwide regions, with robust measures for security and the global compliance and privacy regulations that are required to support connected cars, letting automakers focus on innovation rather than building out their own cloud-based infrastructure. Consequently, Microsoft aspires to empower automakers in their goals for fully autonomous driving, with elegant machine learning and artificial intelligence capabilities, as well as advanced mapping services. For instance, more recently Microsoft has partnered with TomTom, HERE and Esri, to create more intelligent location-based services across Microsoft [101].

Furthermore, pseudo-code for the proposed security modules for AVSs i.e., the hybrid mechanisms CHA is shown in Algorithm 1. Table 3 represents the notations used in pseudo-code for CHA.

Algorithm 1: Pseudo-code for CHA

input : Applications as i
output: Classification as $malware$ or $non-malware$

Process:
Extract i in Cuckoo Sandbox
Analyze i for static analysis
Get list of static $(f)_1$
Analyze i further for dynamic analysis
Get list of dynamic $(f)_2$
Set $(f)_1$ and $(f)_2$ for analyzer m
m predicted result of the i
Get output from predictive modeling
if *Yes* **then**
  label as $malware$;
end
else
  label as $non-malware$;
end

4.1 Dataset

We have used a dataset of 1000 malware applications of different families (e.g., crypto, petya, locker) downloaded from Virusshare.com repository [109]. Similarly, 1000 non-malware applications (freely available apps) are included resulting in a dataset of 2000 applications. We use a three-step ML-based mechanism: (i) feature extraction, (ii) feature selection, and (ii) application classification

4.2 Feature extraction

The choice of a good feature set is the initial phase of any data mining approach. A few of the extracted features are inspired by previous work [79, 84], however, more features have also been added in this research i.e., hardware performance counters [78, 79], DLLs [110], and strings [18, 84, 111]. We have extracted a total of 1713 features and 10,985 features during static and dynamic analysis, respectively. Cuckoo Sandbox is selected in a Linux platform for automated dynamic analysis of Windows

Table 3 Abbreviations used in pseudo-code for CHA

i	Application
f1	Feature set 1 against static analysis
f2	Feature set 2 against dynamic analysis
m	Machine learning algorithms (RF, J48, Naïve Bayes, Gradient Boosting, XGBoost)

executable malware. It automatically runs and analyzes files and collect comprehensive analysis results that outline what the malware does while running inside an isolated operating system. All processes and file changes are tracked and logged. Generated logs and behavioral analysis reports are recorded by Cuckoo. For validation, we have used the K-fold (k=10) cross-validation mechanism and compare the malware detection accuracy of different classifiers to make sure that the dataset is used uniformly without any biasness. This results in unbiased training and testing cycles producing the results on which we could conclude with confidence. For each cycle of the training/testing and validation, a 70% training and 20% testing and 10% validation partition was employed. A list of features extracted are shown in Tables 6 and 7 as sketched in "Appendix".

4.3 Feature selection

The reduced number of features increases ML model performance with minor or negligible effects on classification decisions. Moreover, feature selection minimizes the overfitting factors and the time required for training/testing increases the accuracy to generate simple interpreted models. For this, we employ the information gain criterion [112]. A specific method called *infogainAttributeEval* from Weka is applied for attribute selection. The value of information gain determines how important a given attribute of the feature vectors is by assigning weights to emphasise the effectiveness of the features. Therefore, the top 25 features out of 1713 selected after applying the feature selection infogain algorithm for static analysis, and top 47 features out of 10,985 were selected for dynamic analysis. Figure 4 depicts the top 10 static features formulated using the Info-gain method where X-Axis shows the rank of the feature.

4.4 Model selection and training

Considering the nature of the employed dataset (i.e., categorical and mixed data), this study has been conducted using the three well-known ML classifiers: Naive Bayes (NB) [113] which is often considered one of the fastest classifiers due to its simplicity and computational efficiency. It typically has faster training and prediction times compared to other complex models, Random Forest (RF)

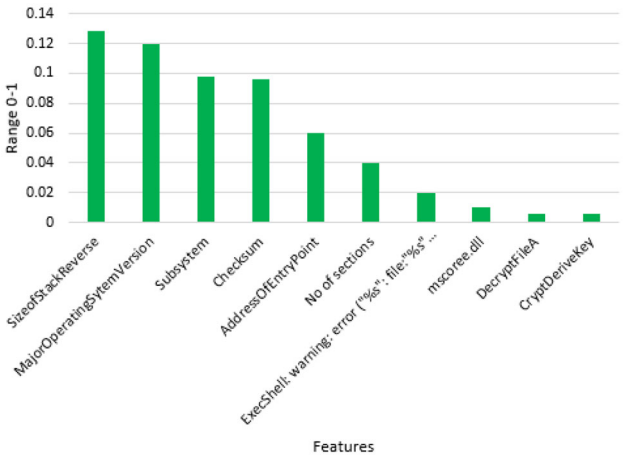

Fig. 4 Top-10 ranked static features

[114, 115], Decision Tree (J48) [116, 117] which are more suitable for categorical and mixed data Gradient Boosting and XGBoost. The area under the ROC Curve [118] is a common mechanism to calculate the performance of a certain ML classifier. A higher value (i.e., near to 1) reflects the better classification capability of the ML classifier. Table 4 shows the ROC values for the CHA. As shown in Table 4, the RF and XGBoost stands prominent as compared to other ML classifiers that have attained area under the ROC curve up to 0.9816 for both classes (i.e., malware and non-malware). This indicates that the RF and XGBoost are the best performing classification model as compared to the other two models.

4.5 Zero-day attack detection

Zero-day attacks refer to vulnerabilities or exploits that are unknown or not yet discovered by security researchers.

Table 4 Combined hierarchy analyzer (CHA) values of class 1 and class 0 for Area under ROC

Classifiers	Class 1	Class 0
Naive Bayes	0.7801	0.782
J48	0.934	0.934
Random Forest	0.9873	0.9873
Gradient Boosting	0.951	0.951
XGBoost	0.9812	0.9812

Hybrid methods for detecting zero-day attacks typically combine multiple techniques and approaches to enhance detection capabilities. While these methods cannot directly detect zero-day attacks that are unknown to the model, our proposed model can still provide some level of protection through the following mechanisms:

- Behavior-based Detection: The proposed hybrid method employ behavior-based detection techniques. This method focus on monitoring the behavior of software or systems to identify anomalies or suspicious activities by establishing a baseline of normal behavior, any deviations from the expected patterns can trigger alerts or raise suspicions, even for zero-day attacks. Behavioral analysis can help detect novel attack patterns or malicious activities that were not explicitly known to the model.
- Anomaly Detection: The proposed hybrid approach can incorporate anomaly detection techniques to identify deviations from normal behavior or expected patterns. By modeling the normal behavior of the system or application, any unusual activities can be flagged as potential threats. Anomaly detection can be effective in detecting previously unseen attack vectors or exploits, including zero-day attacks.

It's important to note that while hybrid methods provide enhanced detection capabilities, they cannot guarantee complete protection against all zero-day attacks. Zero-day attacks, by nature, exploit unknown vulnerabilities, and it takes time for security solutions to catch up. Nevertheless, employing a hybrid approach with multiple detection techniques significantly improves the overall security posture and helps mitigate the risks associated with zero-day attacks.

5 Experimental setup, results and discussion

We have performed experiments on a stand-alone machine having specifications shown in Table 5.

For performance evaluation of selected classifiers, we employed the following metrics.

5.1 Accuracy

We have used accuracy to evaluate the results. The accuracy is the fraction of the total number of correctly classified applications as malware or non-malware. Where TP, TN, FP, and FN stands for True Positive, True Negative, False Positive, and False Negative respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Table 5 System configuration

CPU	Intel core 2 duo 2.13GHz
System Type	32 bit
OS	Ubuntu 14.04 LTS
Data Mining Tool	WEKA 3.8
Platform	Windows XP and Windows 7
RAM	3GB
Sandbox	Cuckoo sandbox
Virtual Machine	VMWare

5.2 Precision

Precision denotes the proportion of the predicted correctly classified applications to the total of all applications that are correctly real positives.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

5.3 Recall

Recall is the fraction of the actual apps that are correctly classifies to the total number of the apps that are classified correctly or incorrectly.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

5.4 F-measure

F-measure is the harmonic mean of precision and recall. F measure represents the value that tells how much the model is capable of making fine distinctions.

$$FMeasure = 2 \times \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

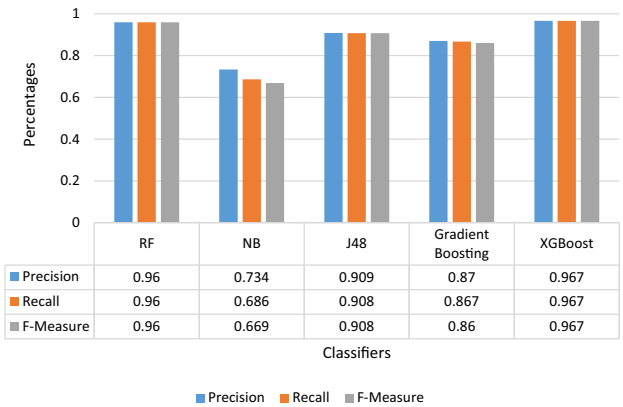

Fig. 5 Precision, recall and F-measure of CHA

Table 6 List of extracted features (1)

Features	Parameters	Classes	Description
Windows API Calls	API:VirtualProtectEx	Memory usage	To analyze the traces of invocations of native functions
	API:GetVolumeNameForVolumeMountPointW	System services	
	API:HttpOpenRequestA	HTTP information	
	API:HttpSendRequestA	Internet handle	
	API:timeGetTime	Process Handling	
	API:DeleteUrlCacheEntryW	disk R/W information	
	API:GetDiskFreeSpaceExW	System configuration settings	
	API:MessageBoxTimeoutA	Registry Key information and security	
	API:CreateDirectoryW	Sending messages to windows	
	API:InternetConnectW	File Path/File size information	
	API:listen	Socket Connection information	
	API:RegDeleteValueW	Anomaly Detector API	
	API:gethostbyname	Cryptography API: Next Generation	
	API:CryptDecodeObjectEx	Folder Paths	
	API:GetCursorPos	Thread execution	
	API:GetFileSize	Certificate store, e.g.,	
	API:FindWindowA	file-based or memory-based stores	
	API:socket	Addresses of exported functions	
	API:LdrGetProcedureAddress	Virtual addresses	
	API:CryptGenKey	Pointer resources	
	API:__anomaly	System time information	
	API:NtQueryDirectoryFile		
	API:InternetCloseHandle		
	API:WSASend		
	API:GetFileType		
	API:SearchPathW		
	API:RegQueryValueExW		
	API:SendNotifyMessageA		
	API:RegOpenKeyExA		
	API:CryptHashData		
	API:GetSystemMetrics		
	API:GetDiskFreeSpaceW		
	API:NtClose		
	API:FindWindowW		
	...		
File operations	FILES:DELETED:C:\WINDOWS\	File Read Operations	To analyze read, write, open and delete operations
	FILES:DELETED:C:\~\Temp\is-B4RA1.tmp\	File Write Operations	
	FILES:DELETED:C:\WINDOWS\system32\	File Delete Operations	
	FILES:OPENED:C:\WINDOWS\AppPatch\		
	FILES:OPENED:C:\SwSetup\SP63752\		
	FILES:READ:C:\~\Start Menu\		
	FILES:READ:?\PIPE\		
	FILES:READ:C:\~\Application Data\		
	FILES:WRITTEN:C:\Program Files\~\plugins\		
	FILES:WRITTEN:C:\~\Application Data\		
	FILES:WRITTEN:C:\		
	...		

Table 7 List of extracted features (2)

Features	Parameters	Classes	Description
Registry Operations	REG:DELETED:HKEY_CLASSES_ROOT\ REG:DELETED:HKEY_CURRENT_USER\∼\– O &O DiskImage Professional\ REG:DELETED:HKEY_LOCAL_MACHINE SOFTWARE\– Classes\.tar\ REG:OPENED:HKEY_LOCAL_MACHINE\∼ Installations\ REG:OPENED:HKEY_CURRENT_USER\∼\Disketch\ REG:OPENED:HKEY_LOCAL_MACHINE\∼\– Products 669F5A8189FAB114E826BA92DFB67647\ REG:READ:HKEY_LOCAL_MACHINE\∼\Abiosdsk\ REG:READ:HKEY_LOCAL_MACHINE\∼\– Installed Components\- {630b1da0-b465-11d1-9948-00c04f98bbc9}\ REG:READ:HKEY_LOCAL_MACHINE\∼\ql1080\ REG:WRITTEN:HKEY_LOCAL_MACHINE\∼\– CLSID {4C6EEFFD-CFF7-4D35-A8F5-52BAA2CC07FF}\ REG:WRITTEN:HKEY_LOCAL_MACHINE\∼\Boot file system\ REG:WRITTEN:HKEY_LOCAL_MACHINE\∼\– {B3D7DD5D-510B-477C-9521-2BCBCC91762C} \ProxyStubClsid\ REG:WRITTEN:HKEY_LOCAL_MACHINE\∼\– {58DA8D8F-9D6A-101B-AFC0-4210102A8DA7} \ProgID\ REG:WRITTEN:HKEY_LOCAL_MACHINE\∼\– shellex PropertySheetHandlers\– {B41DB860-8EE4-11D2-9906-E49FADC173CA}\ ...	Registry Read Operations Registry Write Operations Registry Delete Operations	To analyze read, write, open and delete operations
Embedded Strings	STR:setp32se.dll STR:SRP-3DES-EDE-CBC-SHA STR:No action was taken as BitLocker Drive Encryption is in raw access mode STR:Warning: Deleting a key that isn't empty: "%\s" STR:Click Uninstall to remove TrueCrypt from this system. STR:2http: crt.comodoca.com/COMODORSACodeSigningCA.crt0$ STR:CRYPTO: PrivateKey: Failed to import key ...	Crypto functions Imported libraries Network Information Strings	To analyze files having ASCII and Unicode strings in binary data for quick overview of malware capacity and ability
Dynamic Link Libraries	Kernel32.dll Advapi32.dll mscoree.dll ADVAPI32.dll WSock32.dll ...	Network communication Operating system or execution environment	To analyze required library functions
Hardware Performance Counters	Clock cycles Cache hits Cache misses Branch instructions Branch misses Retired instructions CPUs utilized Task clock Context switching CPU migrations Page faults ...		To analyze the true execution behaviours of applications

For evaluation, accuracy-related results are reported which can be defined as the fraction of the total number of correctly classified applications as malware or non-malware [119]. Figure 5 shows the accuracy results for the proposed model CHA for all three ML classifiers. It is evident from the results that the CHA have shown excellent accuracy indicating that a good-percentage of known malware can be identified using time-/cost-efficient and safer mechanism as compared to risking autonomous vehicle operations with dynamic analysis for all the potential applications.

Based on the values of the True Positive and True Negative, we have calculated precision, recall, and F-measure for CHA approach. The results of the precision and recall of classification using all the five classifiers of the CHA approach are explained in Fig. 5. Results depict that RF and XGBoost generated 32.7% and 5.5% improvement in precision as compared to NB and J48. The values of precision for RF, NB, Gradient Boosting, XGBoost and J48 are 0.96, 0.723, 0.87, 0.96 and 0.91, respectively. RF and XGBoost attained the highest values of precision and recall.

6 Conclusion and future directions

With the advancement in technology and use of smart connected vehicles, we can find examples where cybercriminals have already proven their intent by exploiting several vulnerabilities in the smart transportation systems of automotive ecosystem. we expect to see dramatic increase of cyber attacks against them. The vulnerabilities in the software of AVSs may prove far more dangerous than malware that may appear in personal computers and mobile devices. Malicious applications harm the lives of drivers, passengers as people who are not using AVSs. In this paper, we performed a comprehensive analysis of cybersecurity threat of malware targeting smart transportation systems of connected and autonomous vehicles by proposing hybrid model CHA. The experimentation discussed in the article provides a proof of concept for securing AVSs in general and automotive CPSs in particular, that is adaptive, lightweight, and promises accurate results.

For the future work, we plan to develop future of intelligent transportation system in smart cities that can efficiently detect high priority attacks based on IDS and evaluate their effectiveness using simulations. In addition, network feature analysis can be considered in future along with the communication protocols, such as encryption and authentication mechanisms that ensures that the vehicle's communication channels are protected from unauthorized access or tampering. Moreover, Implementing intrusion detection systems (IDS) within the vehicle's network infrastructure can help identify any unauthorized or malicious attempts to access or manipulate the vehicle's systems. IDS can detect patterns of known attacks or suspicious network traffic. Lastly, for future work, our proposed hybrid method can incorporate machine learning algorithms that adapt and learn from new data and emerging threats. By continuously updating the model with new information and training data, the system can improve its detection capabilities over time and become more adept at identifying previously unknown attack patterns, including zero-day attacks.

Appendix

List of extracted features

In this section we provide two tables that sketch the list of extracted features used in our malware analysis/experiment.

Author contributions SA, HA and MSS conducted experiments, SA, MTK and MA wrote the main manuscript text, SA and MA prepared Figs. 1, 2 and 3. All authors reviewed the manuscript

Funding This research was partially supported by the Higher Education Commission of Pakistan under the NRPU-2021 project"CyberMuhafiz" grant no. 15279.

Data availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Cheah, M., Shaikh, S.A., Bryans, J., Wooderson, P.: Building an automotive security assurance case using systematic security evaluations. Comput. Secur. **77**, 360–379 (2018)

2. Luo, Q., Liu, J.: Wireless telematics systems in emerging intelligent and connected vehicles: threats and solutions. IEEE Wirel. Commun. **25**(6), 113–119 (2018)
3. Canis, B.: Issues in autonomous vehicle testing and deployment. Tech. Rep, Congressional Research Service (2019)
4. Solon, O.: Team of hackers take remote control of Tesla Model S from 12 miles away. The Guardian **20** (2016)
5. Miller, C., Valasek, C.: Remote exploitation of an unaltered passenger vehicle. Black Hat USA **2015**(S91), 1–91 (2015)
6. Malik, S., Sun, W.: Analysis and simulation of cyber attacks against connected and autonomous vehicles. In: 2020 international conference on connected and autonomous driving (MetroCAD), pp. 62–70. IEEE (2020)
7. Al-Hawawreh, M., Sitnikova, E., Aboutorab, N.: Asynchronous peer-to-peer federated capability-based targeted ransomware detection model for industrial iot. IEEE Access **9**, 148738–148755 (2021)
8. Al-Hawawreh, M., Sitnikova, E.: Industrial internet of things based ransomware detection using stacked variational neural network. In: Proceedings of the 3rd international conference on big data and internet of things, pp. 126–130. (2019)
9. Kukkala, V.K., Pasricha, S., Bradley, T.: Sedan: security-aware design of time-critical automotive networks. IEEE Trans Veh. Technol. **69**(8), 9017–9030 (2020)
10. Skatkov, A., Bryukhovetskiy, A., Moiseev, D., Shevchenko, V.: Detecting vulnerabilities of information resources of unmanned vehicles method based on dynamic evaluation of Markov sequences properties. J. Phys.: Conf. Ser. **1515**(2), 022033 (2020)
11. Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T. et al.: Comprehensive experimental analyses of automotive attack surfaces. In: USENIX security symposium, vol. 4. San Francisco, pp. 447–462 (2011)
12. Hamad, M., Prevelakis, V.: Savta: a hybrid vehicular threat model: overview and case study. Information **11**(5), 273 (2020)
13. Dunn, M.: Toyota's killer firmware: bad design and its consequences, EDN Netw. **28** (2013)
14. Ornes, S.: How to hack a self-driving car. Phys. World **33**(8), 37 (2020)
15. Dibaei, M., Zheng, X., Jiang, K., Abbas, R., Liu, S., Zhang, Y., Xiang, Y., Yu, S.: Attacks and defences on intelligent connected vehicles: a survey. Digit. Commun. Netw. **6**, 399–421 (2020)
16. Olufowobi, H., Bloom, G.: Connected cars: automotive cybersecurity and privacy for smart cities. In: Smart cities cybersecurity and privacy, pp. 227–240. Elsevier, Amsterdam (2019)
17. Cobb, S.: Rot: ransomware of things. ESET (2017)
18. Zhang, Z., Qi, P., Wang, W.: Dynamic malware analysis with feature engineering and feature learning. In: Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 01, pp. 1210–1217. (2020)
19. David, C., Fry, S.: Automotive security best practices. In: Recommendations for security and privacy in the era of the next-generation car. https://www.mcafee.com/enterprise/enus/assets/white-papers/wp-automotive-security.pdf (2016). Accessed July 2021
20. Bhamare, D., Zolanvari, M., Erbad, A., Jain, R., Khan, K., Meskin, N.: Cybersecurity for industrial control systems: a survey. Comput. Secur. **89**, 101677 (2020)
21. Yaacoub, J.-P.A., Noura, H.N., Salman, O., Chehab, A.: Robotics cyber security: vulnerabilities, attacks, countermeasures, and recommendations. Int. J. Inf. Secur. **21**, 115–158 (2022)
22. da Costa, F.H., Medeiros, I., Menezes, T., da Silva, J.V., da Silva, I.L., Bonifácio, R., Narasimhan, K., Ribeiro, M.: Exploring the use of static and dynamic analysis to improve the performance of the mining sandbox approach for android malware identification. J. Syst. Softw. **183**, 111092 (2022)
23. Al Alsadi, A.A., Sameshima, K., Bleier, J., Yoshioka, K., Lindorfer, M., Van Eeten, M., Gañán, C.H.: No spring chicken: quantifying the lifespan of exploits in iot malware using static and dynamic analysis. In: Proceedings of the 2022 ACM on Asia conference on computer and communications security, pp. 309–321. (2022)
24. Ghillani, D., Gillani, D.H.: A perspective study on malware detection and protection, a review. Authorea Preprints, Authorea (2022)
25. Mohamed, K.F., Azer, M.A.: Malware detection techniques. In: 4th novel intelligent and leading emerging sciences conference (NILES), vol. 2022, pp. 349–353. IEEE (2022)
26. Kalyan, E.V.P., Adarsh, A.P., Reddy, S.S.L., Renjith, P.: Detection of malware using cnn. In: 2022 second international conference on computer science, engineering and applications (ICCSEA), pp. 1–6. IEEE (2022)
27. Bansal, V., Ghosh, M., Baliyan, N.: Efficient and effective static android malware detection using machine learning. In: International conference on information systems security, pp. 103–118. Springer (2022)
28. Muzaffar, A., Hassen, H.R., Lones, M.A., Zantout, H.: An in-depth review of machine learning based android malware detection. Comput. Secur. **121**, 102833 (2022)
29. Gopinath, M., Sethuraman, S.C.: A comprehensive survey on deep learning based malware detection techniques. Comput. Sci. Rev. **47**, 100529 (2023)
30. Bhagwat, S., Gupta, G.P.: Android malware detection using hybrid meta-heuristic feature selection and ensemble learning techniques. In: International conference on advances in computing and data sciences, pp. 145–156. Springer (2022)
31. Shah, I.A., Mehmood, A., Khan, A.N., Elhadef, M., Khan, A.R.: Heucrip: a malware detection approach for internet of battlefield things. Clust. Comput. **26**(2), 977–992 (2023)
32. Tang, J., Li, R., Jiang, Y., Gu, X., Li, Y.: Android malware obfuscation variants detection method based on multi-granularity opcode features. Future Gener. Comput. Syst. **129**, 141–151 (2022)
33. Kara, I., Aydos, M.: The rise of ransomware: forensic analysis for windows based ransomware attacks. Expert Syst. Appl. **190**, 116198 (2022)
34. Kok, S., Abdullah, A., Jhanjhi, N.: Early detection of crypto-ransomware using pre-encryption detection algorithm. J. King Saud Univ.-Comput. Inf. Sci. **34**(5), 1984–1999 (2022)
35. Yadav, P., Menon, N., Ravi, V., Vishvanathan, S., Pham, T.D.: A two-stage deep learning framework for image-based android malware detection and variant classification. Comput. Intell. **38**(5), 1748–1771 (2022)
36. Mimura, M., Ito, R.: Applying nlp techniques to malware detection in a practical environment. Int. J. Inf. Secur. **21**(2), 279–291 (2022)
37. Yamany, B., Elsayed, M.S., Jurcut, A.D., Abdelbaki, N., Azer, M.A.: A new scheme for ransomware classification and clustering using static features. Electronics **11**(20), 3307 (2022)
38. Mimura, M.: Evaluation of printable character-based malicious pe file-detection method. Internet Things **19**, 100521 (2022)
39. Elsersy, W.F., Feizollah, A., Anuar, N.B.: The rise of obfuscated android malware and impacts on detection methods. PeerJ Comput. Sci. **8**, e907 (2022)
40. Muralidharan, T., Cohen, A., Gerson, N., Nissim, N.: File packing from the malware perspective: techniques, analysis approaches, and directions for enhancements. ACM Comput. Surv. **55**(5), 1–45 (2022)
41. Alhaidari, F., Shaib, N.A., Alsafi, M., Alharbi, H., Alawami, M., Aljindan, R., Rahman, A., Zagrouba, R., et al.: Zevigilante: detecting zero-day malware using machine learning and

sandboxing analysis techniques. Comput. Intell. Neurosci. (2022). https://doi.org/10.1155/2022/1615528
42. Gera, T., Singh, J., Faruki, P., Thakur, D.: Efficacy of android security mechanisms on ransomware analysis and detection. In: AIP conference proceedings, vol. 2357, no. 1. AIP Publishing (2022)
43. Wang, L., Wang, H., He, R., Tao, R., Meng, G., Luo, X., Liu, X.: Malradar: demystifying android malware in the new era. Proc. ACM Meas. Anal. Comput. Syst. **6**(2), 1–27 (2022)
44. Qiang, W., Yang, L., Jin, H.: Efficient and robust malware detection based on control flow traces using deep neural networks. Comput. Secur. **122**, 102871 (2022)
45. Falana, O.J., Sodiya, A.S., Onashoga, S.A., Badmus, B.S.: Maldetect: an intelligent visualization approach for malware detection. J. King Saud Univ.-Comput. Inf. Sci. **34**(5), 1968–1983 (2022)
46. Obaidat, I., Sridhar, M., Pham, K.M., Phung, P.H.: Jadeite: a novel image-behavior-based approach for java malware detection using deep learning. Comput. Secur. **113**, 102547 (2022)
47. Romano, A., Lehmann, D., Pradel, M., Wang, W.: Wobfuscator: obfuscating javascript malware via opportunistic translation to webassembly. In: IEEE symposium on security and Privacy (SP), vol. 2022, pp. 1574–1589. IEEE (2022)
48. Kim, M., Cho, H., Yi, J.H.: Large-scale analysis on anti-analysis techniques in real-world malware. IEEE Access **10**, 75802–75815 (2022)
49. Liu, S., Feng, P., Wang, S., Sun, K., Cao, J.: Enhancing malware analysis sandboxes with emulated user behavior. Comput. Secur. **115**, 102613 (2022)
50. Maniriho, P., Mahmood, A.N., Chowdhury, M.J.M.: A study on malicious software behaviour analysis and detection techniques: taxonomy, current trends and challenges. Future Gener. Comput. Syst. **130**, 1–18 (2022)
51. Amer, E., Mohamed, A., Mohamed, S.E., Ashaf, M., Ehab, A., Shereef, O., Metwaie, H.: Using machine learning to identify android malware relying on api calling sequences and permissions. J. Comput. Commun. **1**(1), 38–47 (2022)
52. Yang, Y., Lin, Y., Li, Z., Zhao, L., Yao, M., Lai, Y., Li, P.: Goosebt: a programmable malware detection framework based on process, file, registry, and com monitoring. Comput. Commun. **204**, 24–32 (2023)
53. Malik, J., Kaushal, R.: Credroid: android malware detection by network traffic analysis. In: Proceedings of the 1st ACM workshop on privacy-aware mobile computing, pp. 28–36. (2016)
54. Zaman, M., Siddiqui, T., Amin, M.R., Hossain, M.S.: Malware detection in android by network traffic analysis. In: International conference on networking systems and security (NSysS), vol. 2015, pp. 1–5. IEEE (2015)
55. Wang, S., Chen, Z., Yan, Q., Yang, B., Peng, L., Jia, Z.: A mobile malware detection method using behavior features in network traffic. J. Netw. Comput. Appl. **133**, 15–25 (2019)
56. Prasse, P., Machlica, L., Pevnỳ, T., Havelka, J., Scheffer, T.: Malware detection by analysing network traffic with neural networks. In: IEEE security and privacy workshops (SPW), vol. 2017, pp. 205–210. IEEE (2017)
57. Wang, S., Chen, Z., Zhang, L., Yan, Q., Yang, B., Peng, L., Jia, Z.: Trafficav: an effective and explainable detection of mobile malware behavior using network traffic. In: IEEE/ACM 24th international symposium on quality of service (IWQoS), vol. 2016, pp. 1–6. IEEE (2016)
58. Malik, S., Khatter, K.: System call analysis of android malware families. Indian J. Sci. Technol. **9**(21), 1–13 (2016)
59. Canfora, G., Medvet, E., Mercaldo, F., Visaggio, C.A.: Detecting android malware using sequences of system calls. In: Proceedings of the 3rd international workshop on software development lifecycle for mobile, pp. 13–20 (2015)
60. Zhang, X., Mathur, A., Zhao, L., Rahmat, S., Niyaz, Q., Javaid, A., Yang, X.: An early detection of android malware using system calls based machine learning model. In: Proceedings of the 17th international conference on availability, reliability and security, pp. 1–9 (2022)
61. Dinaburg, A., Royal, P., Sharif, M., Lee, W.: Ether: malware analysis via hardware virtualization extensions. In: Proceedings of the 15th ACM conference on Computer and communications security, pp. 51–62 (2008)
62. Kirat, D., Vigna, G., Kruegel, C.: Barebox: efficient malware analysis on bare-metal. In: Proceedings of the 27th annual computer security applications conference, pp. 403–412 (2011)
63. Yan, L.-K., Jayachandra, M., Zhang, M., Yin, H.: V2e: combining hardware virtualization and softwareemulation for transparent and extensible malware analysis. In: Proceedings of the 8th ACM SIGPLAN/SIGOPS conference on virtual execution environments, pp. 227–238 (2012)
64. Pék, G., Bencsáth, B., Buttyán, L.: Nether: in-guest detection of out-of-the-guest malware analyzers. In: Proceedings of the fourth European workshop on system security, pp. 1–6 (2011)
65. Khan, M.T., Serpanos, D., Shrobe, H.: Armet: behavior-based secure and resilient industrial control systems. Proc. IEEE **106**(1), 129–143 (2017)
66. Scotece, D.: Edge computing for extreme reliability and scalability. Alma (2020)
67. Vuong, T.P.: Cyber-physical intrusion detection for robotic vehicles. Ph.D. dissertation, University of Greenwich (2017)
68. Vuong, T.P., Loukas, G., Gan, D.: Performance evaluation of cyber-physical intrusion detection on a robotic vehicle. In: 2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing, pp. 2106–2113. IEEE (2015)
69. Hoppe, T., Kiltz, S., Dittmann, J.: Applying intrusion detection to automotive it-early insights and remaining challenges. J. Inf. Assur. Secur. (JIAS) **4**(6), 226–235 (2009)
70. Wiseman, Y.: Autonomous vehicles. In: Encyclopedia of information science and technology, 5th edn., pp. 1–11. IGI Global, Hershey (2021)
71. Zhou, F., Yang, Q., Zhong, T., Chen, D., Zhang, N.: Variational graph neural networks for road traffic prediction in intelligent transportation systems. IEEE Trans. Ind. Inform. **17**, 2802–2812 (2020)
72. Han, B., Wu, B., Nguyen, Q., Camargo, R., Arancibia, I.: The threat of cyber-terrorism & security in intelligent transportation systems architecture
73. Bayindir, K.Ç., Gözüküçük, M.A., Teke, A.: A comprehensive overview of hybrid electric vehicle: powertrain configurations, powertrain control techniques and electronic control units. Energy Convers. Manage. **52**(2), 1305–1313 (2011)
74. Liu, P., Dong, L., Shao, X., Lin, M., Gu, Y., Hou, X.: Research on the development trend of vehicle operating system in china. In: The 2nd international conference on computing and data science, pp. 1–6 (2021)
75. Gittins, Z., Soltys, M.: Malware persistence mechanisms. Procedia Comput. Sci. **176**, 88–97 (2020)
76. Patent shows new tesla windows operating systesm. https://www.greencarreports.com/news/1120662_patent-shows-new-tesla-windows-operating-system. Accessed Aug 2021
77. Beneventi, F., Bartolini, A., Cavazzoni, C., Benini, L.: Continuous learning of hpc infrastructure models using big data analytics and in-memory processing tools. In: Proceedings of the conference on design, automation & test in Europe. European Design and Automation Association, pp. 1038–1043 (2017)
78. Demme, J., Maycock, M., Schmitz, J., Tang, A., Waksman, A., Sethumadhavan, S., Stolfo, S.: On the feasibility of online

malware detection with performance counters. In: ACM SIGARCH computer architecture news, vol. 41, no. 3. ACM, pp. 559–570 (2013)
79. Aurangzeb, S., Rais, R.N.B., Aleem, M., Islam, M.A., Iqbal, M.A.: On the classification of microsoft-windows ransomware using hardware profile. PeerJ Comput. Sci. **7**, e361 (2021)
80. Niu, W., Zhang, X., Du, X., Hu, T., Xie, X., Guizani, N.: Detecting malware on x86-based iot devices in autonomous driving. IEEE Wirel. Commun. **26**(4), 80–87 (2019)
81. Sheehan, B., Murphy, F., Mullins, M., Ryan, C.: Connected and autonomous vehicles: a cyber-risk classification framework. Transp. Res. A **124**, 523–536 (2019)
82. Nieuwenhuizen, D.: A behavioural-based approach to ransomware detection. Whitepaper, MWR Labs Whitepaper, p. 20 (2017)
83. Song, S., Kim, B., Lee, S.: The effective ransomware prevention technique using process monitoring on android platform. Mob. Inf. Syst. (2016). https://doi.org/10.1155/2016/2946735
84. Sgandurra, D., Muñoz-González, L., Mohsen, R., Lupu, E.C.: Automated dynamic analysis of ransomware: benefits, limitations and use for detection. (2016). arXiv preprint arXiv:1609.03020
85. Sternfeld, U.: Operation koffler: mutating ransomware enters the fray (2015)
86. Hampton, N., Baig, Z., Zeadally, S.: Ransomware behavioural analysis on windows platforms. J. Inf. Secur. Appl. **40**, 44–51 (2018)
87. Chen, Z.-G., Kang, H.-S., Yin, S.-N., Kim, S.-R.: Automatic ransomware detection and analysis based on dynamic api calls flow graph. In: Proceedings of the international conference on research in adaptive and convergent systems. ACM, pp. 196–201 (2017)
88. Kharraz, A., Arshad, S., Mulliner, C., Robertson, W.K., Kirda, E.: Unveil: a large-scale, automated approach to detecting ransomware. In: USENIX security symposium, pp. 757–772 (2016)
89. Maiorca, D., Mercaldo, F., Giacinto, G., Visaggio, C.A., Martinelli, F.: R-packdroid: Api package-based characterization and detection of mobile ransomware. In; Proceedings of the symposium on applied computing. ACM, pp. 1718–1723 (2017)
90. Zavarsky, P., Lindskog, D., et al.: Experimental analysis of ransomware on windows and android platforms: evolution and characterization. Procedia Comput. Sci. **94**, 465–472 (2016)
91. Al-rimy, B.A.S., Maarof, M.A., Shaid, S.Z.M.: A 0-day aware crypto-ransomware early behavioral detection framework. In: International conference of reliable information and communication technology, pp. 758–766. Springer (2017)
92. Andronio, N., Zanero, S., Maggi, F.: Heldroid: dissecting and detecting mobile ransomware. In: International workshop on recent advances in intrusion detection, pp. 382–404. Springer (2015)
93. Aslan, Ö., Samet, R.: Investigation of possibilities to detect malware using existing tools. In: 14th ACS/IEEE international conference on computer systems and applications AICCSA (2017)
94. Kaur, G., Dhir, R., Singh, M.: Anatomy of ransomware malware: detection, analysis and reporting. Int. J. Secur. Netw. **12**(3), 188–197 (2017)
95. Ferdowsi, A., Challita, U., Saad, W.: Deep learning for reliable mobile edge analytics in intelligent transportation systems: an overview. IEEE Veh. Technol. Mag. **14**(1), 62–70 (2019)
96. Ucci, D., Aniello, L., Baldoni, R.: Survey of machine learning techniques for malware analysis. Comput. Secur. **81**, 123–147 (2019)
97. Simon, H., Simon, H.: Profit driver: price. In: True profit! no company ever went broke turning a profit, pp. 123–150. Springer (2021)
98. NESTLER, M.: Smart use of digital tools. Italy (2020)
99. Watters, Y., Northey, W.F., Jr.: Online telesupervision: competence forged in a pandemic. J. Fam. Psychother. **31**(3–4), 157–177 (2020)
100. Continental, D., Ford, G., Hyundai, M.: Tesla—mercedes-benz—microsoft—autosar—vector consulting servicesl automotive software: where to from here?
101. Mercedes-Benz enhances drivers' experience with Azure OpenAI Service. https://azure.microsoft.com/en-us/blog/mercedes-benz-enhances-drivers-experience-with-azure-openai-service/. Accessed Aug 2021
102. Ashuri, T.: Shadowy knowledge infrastructures. Inf. Commun. Soc, pp. 1–17 (2023)
103. Cabigiosu, A.: Sustainable development and incumbents' open innovation strategies for a greener competence-destroying technology: The case of electric vehicles. Bus. Strat. Environ. **31**(5), 2315–2336 (2022)
104. Fehling, C., Leymann, F., Retter, R., Schumm, D., Schupeck, W.: An architectural pattern language of cloud-based applications. In: Proceedings of the 18th conference on pattern languages of programs, pp. 1–11 (2011)
105. Yaqoob, I., Ahmed, E., Hashem, I.A.T., Ahmed, A.I.A., Gani, A., Imran, M., Guizani, M.: Internet of things architecture: recent advances, taxonomy, requirements, and open challenges. IEEE Wirel. Commun. **24**(3), 10–16 (2017)
106. Coppola, R., Morisio, M.: Connected car: technologies, issues, future trends. ACM Comput. Surv. **49**(3), 1–36 (2016)
107. Automotive Future. https://download.microsoft.com/download/5/0/4/5040df6f-00f1-4e91-abef-082236e7be6e/PSFK_Microsoft_FutureOfAutomotive.pdf. Accessed Oct 2021
108. Microsoft connected vehicle platform helps automakers transform cars. https://blogs.microsoft.com/blog/2017/01/05/microsoft-connected-vehicle-platform-helps-automakers-transform-cars/. Accessed Sept 2021
109. VirusShare, V.: Virusshare. com–because sharing is caring (2019)
110. Arabo, A., Dijoux, R., Poulain, T., Chevalier, G.: Detecting ransomware using process behavior analysis. Procedia Comput. Sci. **168**, 289–296 (2020)
111. Hwang, J., Kim, J., Lee, S., Kim, K.: Two-stage ransomware detection using dynamic analysis and machine learning techniques. Wirel. Pers. Commun. **112**(4), 2597–2609 (2020)
112. Shannon, C.E.: A mathematical theory of communication, part ii. Bell Syst. Tech. J. **27**, 623–656 (1948)
113. Jones, K.S.: Readings in information retrieval. Morgan Kaufmann, Burlington (1997)
114. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
115. Liaw, A., Wiener, M., et al.: Classification and regression by randomforest. R news **2**(3), 18–22 (2002)
116. Kohavi, R.: Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In: KDD, vol. 96. Citeseer, pp. 202–207 (1996)
117. Salzberg, S.L.: C4. 5: Programs for machine learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc (1993)
118. Hand, D.J., Till, R.J.: A simple generalisation of the area under the roc curve for multiple class classification problems. Mach. Learn. **45**(2), 171–186 (2001)
119. Sayadi, H., Patel, N., S. M. PD, Sasan, A., Rafatirad, S., Homayoun, H.: Ensemble learning for effective run-time hardware-based malware detection: a comprehensive analysis and classification. In: 55th ACM/ESDA/IEEE design automation conference (DAC), vol. 2018, pp. 1–6. IEEE (2018)

Sana Aurangzeb received her MS degree in Computer Science from Capital University of Science and Technology in 2018. She is currently doing a Ph.D. at the National University of Computer and Emerging Sciences, Islamabad, Pakistan, and is currently a lecturer at the National University of Modern Languages, Islamabad. Her research focus is malware analysis and security services.

Muhammad Aleem received a Ph.D. degree in computer science from the Leopold-Franzens-University, Innsbruck, Austria in 2012. His research interests include parallel and distributed computing comprising programming environments, multi-/many-core computing, performance analysis, cloud computing, and big-data processing. He is currently working as a Professor at National University of Computer and Emerging Sciences, Islamabad, Pakistan.

Muhammad Taimoor Khan is an Associate Professor (Reader) of Computer Science at the University of Greenwich, UK. He is the Head of the Cyber Assurance Lab and a Centre of Sustainable Cyber Security member. His current research interests are the automatic detection and recovery of vulnerabilities/threats/risks in software through verification-based design-time and run-time security analysis of serious application software.

Haris Anwar is a data scientist with over seven years of invaluable experience in the industry with a Master's of Science in Data Science FAST-NUCES. His expertise spans diverse sectors, including medical, oil and gas, and cyber security.

Muhammad Shaoor Siddique is currently pursuing a Ph.D. from the National University of Computer and Emerging Sciences, Islamabad, Pakistan.