CyberSecurity for Autonomous Vehicles Against Malware Attacks in Smart-Cities

Sana Aurangzeb (sanaaurangzeb@numl.edu.pk)
National University of Modern Languages
Muhammad Aleem (m.aleem@nu.edu.pk)
National University of Computer and Emerging Sciences
Muhammad Taimoor Khan (m.khan@gre.ac.uk)
University of Greenwich
Haris Anwar (harisanwar64@gmail.com)
National University of Computer and Emerging Sciences
Muhammad Shaoor Siddique (shaoorsiddique.mrf@gmail.com)
National University of Computer and Emerging Sciences

Research Article

Keywords:

DOI: https://doi.org/

CyberSecurity for Autonomous Vehicles Against Malware Attacks in Smart-Cities

Sana Aurangzeb, Muhammad Aleem, Muhammad Taimoor Khan*, Haris Anwar, Muhammad Shaoor Siddique
sanaaurangzeb@numl.edu.pk,m.aleem@nu.edu.pk *m.khan@gre.ac.uk, harisanwar64@gmail.com,
shaoorsiddique.mrf@gmail.com

***Abstract*—Smart autonomous vehicles (AVs) are networks of cyber physical systems (CPS) in which they wirelessly communicate with other CPS 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 (ITS). 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.**

***Index Terms*—Malware detection; security; smart cities; autonomous systems;**

I. Introduction

RECENTLY autonomous vehicular systems (AVs) have seen a gigantic growth in a wide variety of aspects with the development of smart cities to build the Intelligent Transport Systems (ITS). For instance, the dramatic use of embedded systems and wireless communication (e.g., 4G LTE and 5G) in modern internet of vehicles which ultimately improve users safety and comfort. However, growing interest in the development of connected autonomous vehicles (CAVs) and ITS has introduced new security challenges and vulnerabilities in AVs 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 cyber-criminals have successfully demonstrated practical but remote attacks to key functions of automotive vehicles (as depicted in Figure 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 sensitive IT systems [6]. Previously, ransomware attack infected either personal computers, public or private organizations, health sectors, Internet of Things, smartphones and smart industrial systems. Now, ransomware attack is targeting smart-vehicles and smart cities that could result in the loss of human lives and financial instability. 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 AVs [7], [8]. For instance, by exploiting known vulnerabilities in the design and implementation of onboard communication systems, embedded software, and application software [9]–[11] as sketched in Table I. Moreover, a report in [2], [12], [13] has shown that an AVs 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 AVs 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 AVs 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 AVs 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 AVs are among the most significant security issues for the intelligent vehicles [8], [17]. Specifically, the most damaging cyber threats, are emerging as the vehicles connect to the Internet, provide onboard

S. Aurangzeb is with the Department of Computer Science, National University of Modern Languages Islamabad, Pakistan e-mail: (sanaaurangzeb@numl.edu.pk).

M. Aleem and M.A. Islam are with the National University of Computer and Emerging Sciences, Islamabad, Pakistan .

Corresponding Author: M. T. Khan (Member, IEEE) is with the School of Computing and Mathematical Sciences, University of Greenwich, London, UK e-mail: m.khan@gre.ac.uk

TABLE I
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 [14]
Driver distraction	Misusing vehicle features to distract the driver by arbitrarily turning on the in-vehicle audio and tuning its volume [12], [14]
Locking vehicle	Locking vehicle features resulting in jackware [15]
Externally connected devices	Modifying files on the vehicle and on users brought-in devices connected to the vehicle [12]
Computational resources of the vehicle	Consuming computational resources (such as memory space and CPU cycles) to disrupt vehicle actual operations [13]
Sensitive and private data	Stealing private and sensitive data [16]
Passengers safety	Threatening passengers lives by disabling vehicle safety functions
CAVs	Using the compromised vehicle to send misleading, false, and bogus data to CAVs

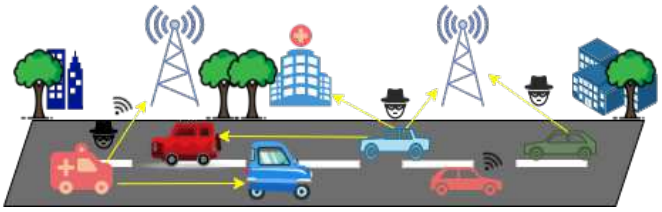


Fig. 1. Typical V2V, V2X Cyber Threat Scenario in Smart Autonomous Vehicles

Wi-Fi hotspot services, communicate with other vehicles and ITS infrastructures, and support advanced applications such as over-the-air (OTA) ECU firmware update [7]. 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 AVs, it is important to detect a malware in real-time to timely protect any physical and financial damage and loss of human lives. Current approaches to detect such malware either employ static analysis or dynamic analysis techniques. 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 [18] but such approaches typically require more computational resources which is not the case in autonomous vehicles. 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) [19]. 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: Section II provides background of autonomous vehicles, while Section III sketches state of the art about rigorous malware detection techniques. Section IV explains our malware detection methodology, while Section V presents experimental setup, experiment results and critical discussion. Finally, we conclude in Section VI.

II. BACKGROUND

Modern smart Autonomous vehicles (AVs) will strikingly change the worldwide transport industry and smart environments. AVs 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 ITS technology preventing injuries and deaths in combat [20]. 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 ITS. With the increasing trend of Internet of Things (IoT), ITS aims to improve the efficiency and safety of AV networks [21]. ITS in societies that are converting into smart cities becomes more vulnerable to cyber-threat and cyber-terrorism [22]. Different types of ITS 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 AVs 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 [23].

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 Figure 2. Although ECU act as a brain for AVs 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 [24] is now testing Windows OS [25] and Tesla patent seems to be working on windows operating system [26]

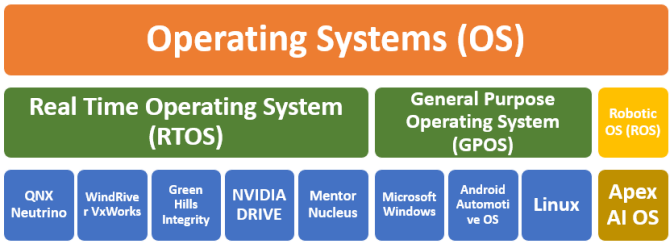


Fig. 2. Types of Operating Systems (OS) used in Smart Autonomous Vehicles

III. 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 [27] 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 [28], authors have used Bayesian Network (BN) model to analyse cyber risk in AVs 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 AVs 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 [29], the authors presented a case study of CryptoLuck Ransomware to highlight the importance of behavioral-based Ransomware detection. In [30], authors statically analyzed process monitoring on file events, processor usage, and I/O rates. In [31], authors suggested that static detection technique as used by [32], can help in evading anti-virus (AV). In [33], authors performed ransomware behavioral analysis on windows platform of 14 strains of ransomware. They observed the individual behavioral pattern of ransomware. In [34], authors presented an automated detection and analysis of ransomware to monitor dynamic behavior by generating API calls and control flow graph (CFG). Authors in [35], 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 [31]. 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 [36]. However, none of the existing dynamic and ML malware detection techniques use hardware performance counter for malware classification specifically in autonomous vehicles. Although, however, [37] employs a dynamic approach to classify malware based on their hardware performance counters and [38] have used hardware performance counter for ransomware classification on Windows platform.

It has been observed from the literature work that most of the techniques [31] can either only observe System/API calls [33], [34], [39], file operations [35], processor usage [30], or registry activities [40]. Some of the studies are based on static analysis [29] 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 [41]–[44] have shown that there is a lack of behavioral analysis that use hybrid technique to classify malware in AVs 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 AVs. 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 AVs. 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 AVs.

IV. Methodology

Autonomous vehicles (AVs) have become a core constituent of the smart transportation system [45]. The computation power of AVs is 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 AVs, 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 [46]) 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 AV 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 AVs i.e., the hybrid mechanisms Combined Hybrid Analyzer (CHA) is shown in Figure 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

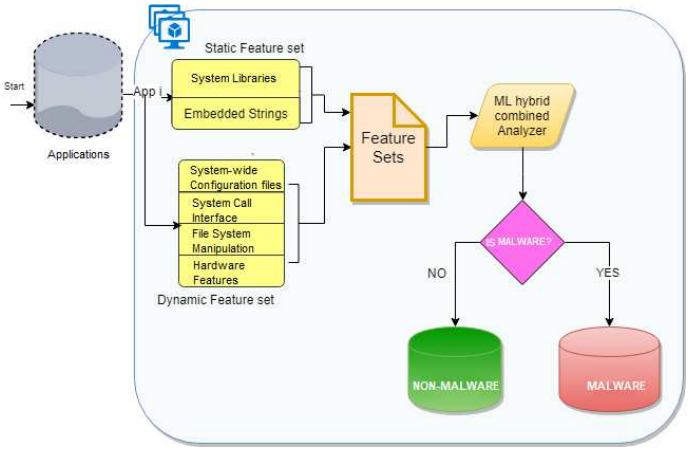


Fig. 3. Combined Hybrid Analyzer (CHA)

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). 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 Section IV-B, IV-C and IV-D. The machine learning model i.e., J48, Naive Bayes (NB), 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. For IoT related malware detection 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 II) for their live communication, which is certainly a key source of threat to such services and eventually to the vehicles [47], [48]. Furthermore, as reported in [49], 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

TABLE II
MICROSOFT WINDOWS BASED SERVICES FOR AUTOMOTIVE VEHICLES

Automotive Brand	Goal	Windows based Services
Porsche Holding	Mobile-first and Virtual Workplace	Office 365 Teams Microsoft's Cloud Services
Brimborg	Online Stream-based Services for Rentals	Microsoft Dynamic 365
Mercedes-Benz	Connected Cars Platform	Microsoft's Cloud-based containerized platform Azure Monitor
Moovit	Real-time in-city and out-city transits Mobility-as-a-service	Azure Maps
Daimler	Detroit Connect platform Virtual Technician Remote Updates Remote Analytics	Azure Microsoft's cloud computing service

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 [50].

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

TABLE III
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)

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

A. Dataset

As discussed in the previous section, we have used a dataset of 1000 malware applications of different families (e.g., crypto, petya, locker) downloaded from Virusshare.com repository [51]. 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

B. 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 [31], [38], however, more features have also been added in this research i.e., hardware performance counters [37], [38], DLLs [52], and strings [16], [31], [53]. We have extracted a total of 1713 features and 10985 features during static and dynamic analysis, respectively. Cuckoo Sandbox is selected in a Linux platform for automated dynamic analysis of Windows 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, a 20% testing and 80% training partition was employed. A list of features extracted are shown in Tables V and VI as sketched in Appendix A.

C. 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 [54]. 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 10985 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.

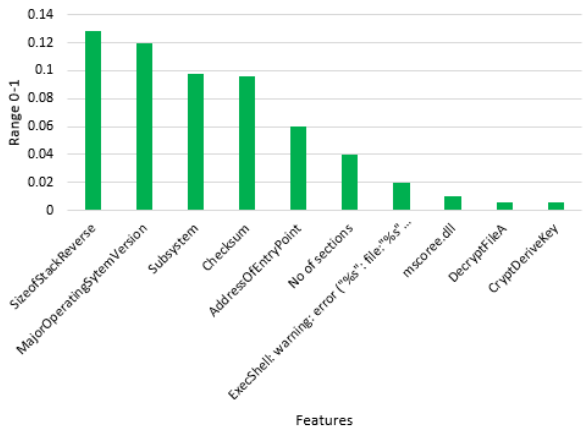


Fig. 4. Top-10 ranked static features

D. 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) [55], Random Forest (RF) [56], [57], and Decision Tree (J48) [58], [59] which are more suitable for categorical and mixed data. The area under the ROC Curve [60] 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. Figure 5a shows the ROC Curve for the CHA. As depicted in Figure 5a, the RF 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 is the best performing classification model as compared to the other two models.

V. Experimental Setup, Results and Discussion

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

TABLE IV
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

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

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.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

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

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

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.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

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.

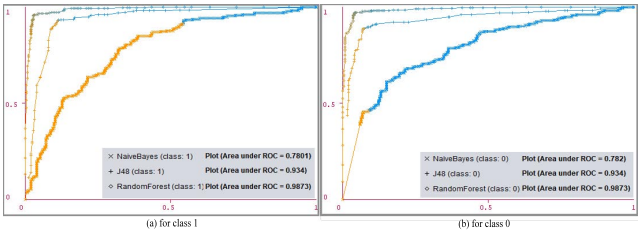
$$FMeasure = 2 \times \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

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 [61]. Figure 6 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 three classifiers of the CHA approach are explained in Figure 6. Results depict that RF generated 32.7% and 5.5% improvement in precision as compared to NB and J48. The values of precision for RF, NB, and J48 are 0.96, 0.723, and 0.91, respectively. RF attained the highest values of precision and recall.

VI. Conclusion

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 AVs 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 AVs. In this paper, we performed a comprehensive



(a) Combined Hierarchy Analyzer (CHA)

Fig. 5. ROC curve for CHA

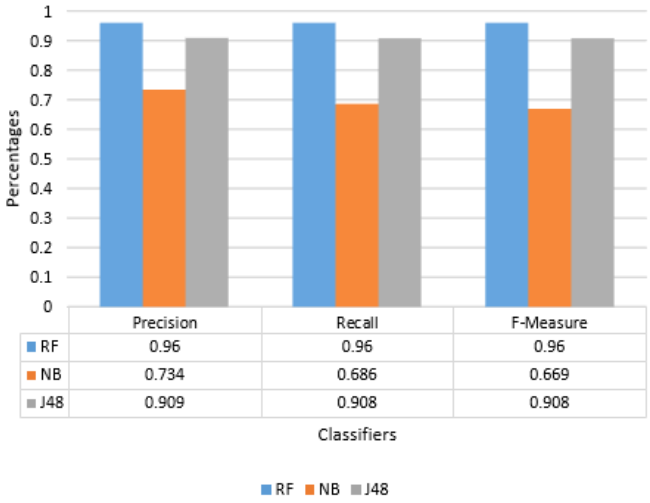


Fig. 6. Precision, Recall and F-measure of CHA

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 AV systems in general and automotive CPS 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.

REFERENCES

[1] M. Cheah, S. A. Shaikh, J. Bryans, and P. Wooderson, "Building an automotive security assurance case using systematic security evaluations," *Computers & Security*, vol. 77, pp. 360–379, 2018.

[2] Q. Luo and J. Liu, "Wireless telematics systems in emerging intelligent and connected vehicles: Threats and solutions," *IEEE Wireless Communications*, vol. 25, no. 6, pp. 113–119, 2018.

[3] B. Canis, "Issues in autonomous vehicle testing and deployment," Congressional Research Service, Tech. Rep., 2019.

[4] O. Solon, "Team of hackers take remote control of tesla model s from 12 miles away," *The Guardian*, vol. 20, 2016.

[5] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, 2015.

[6] S. Malik and W. Sun, "Analysis and simulation of cyber attacks against connected and autonomous vehicles," in *2020 International Conference on Connected and Autonomous Driving (MetroCAD)*. IEEE, 2020, pp. 62–70.

[7] V. K. Kukkala, S. Pasricha, and T. Bradley, "Sedan: Security-aware design of time-critical automotive networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 9017–9030, 2020.

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.

TABLE V
LIST OF EXTRACTED FEATURES (1)

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

TABLE VI
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\%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 Perfor-mance 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

[8] A. Skatkov, A. Bryukhovetskiy, D. Moiseev, and V. Shevchenko, "Detecting vulnerabilities of information resources of unmanned vehicles method based on dynamic evaluation of markov sequences properties," in *Journal of Physics: Conference Series*, vol. 1515, no. 2. IOP Publishing, 2020, p. 022033.

[9] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*, vol. 4. San Francisco, 2011, pp. 447–462.

[10] M. Hamad and V. Prevelakis, "Savta: A hybrid vehicular threat model: Overview and case study," *Information*, vol. 11, no. 5, p. 273, 2020.

[11] M. Dunn, "Toyota's killer firmware: Bad design and its consequences," *EDN Network*, vol. 28, 2013.

[12] S. Ornes, "How to hack a self-driving car," *Physics World*, vol. 33, no. 8, p. 37, 2020.

[13] M. Dibaei, X. Zheng, K. Jiang, R. Abbas, S. Liu, Y. Zhang, Y. Xiang, and S. Yu, "Attacks and defences on intelligent connected vehicles: A survey," *Digital Communications and Networks*, 2020.

[14] H. Olufowobi and G. Bloom, "Connected cars: Automotive cybersecurity and privacy for smart cities," in *Smart cities cybersecurity and privacy*. Elsevier, 2019, pp. 227–240.

[15] S. Cobb, "Rot: Ransomware of things," 2017.

[16] Z. Zhang, P. Qi, and W. Wang, "Dynamic malware analysis with feature engineering and feature learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 1210–1217.

[17] C. David and S. Fry, "Automotive security best practices," *Recommendations for security and privacy in the era of the next-generation car. Online available: https://www. mcafee. com/enterprise/enus/assets/white-papers/wp-automotive-security. pdf*, 2016.

[18] M. T. Khan, D. Serpanos, and H. Shrobe, "Armet: Behavior-based secure and resilient industrial control systems," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 129–143, 2017.

[19] T. Hoppe, S. Kiltz, and J. Dittmann, "Applying intrusion detection to automotive it-early insights and remaining challenges," *Journal of Information Assurance and Security (JIAS)*, vol. 4, no. 6, pp. 226–235, 2009.

[20] Y. Wiseman, "Autonomous vehicles," in *Encyclopedia of Information Science and Technology, Fifth Edition*. IGI Global, 2021, pp. 1–11.

[21] F. Zhou, Q. Yang, T. Zhong, D. Chen, and N. Zhang, "Variational graph neural networks for road traffic prediction in intelligent transportation systems," *IEEE Transactions on Industrial Informatics*, 2020.

[22] B. Han, B. Wu, Q. Nguyen, R. Camargo, and I. Arancibia, "The threat of cyber-terrorism & security in intelligent transportation systems architecture."

[23] K. Ç. Bayindir, M. A. Gözüküçük, and A. Teke, "A comprehensive overview of hybrid electric vehicle: Powertrain configurations, powertrain control techniques and electronic control units," *Energy conversion and Management*, vol. 52, no. 2, pp. 1305–1313, 2011.

[24] P. Liu, L. Dong, X. Shao, M. Lin, Y. Gu, and X. Hou, "Research on the development trend of vehicle operating system in china," in *The 2nd International Conference on Computing and Data Science*, 2021, pp. 1–6.

[25] Z. Gittins and M. Soltys, "Malware persistence mechanisms," *Procedia Computer Science*, vol. 176, pp. 88–97, 2020.

[26] Patent shows new tesla windows operating systesm. https://www.greencarreports.com/news/1120662_patent-shows-new-tesla-windows-operating-system.

[27] W. Niu, X. Zhang, X. Du, T. Hu, X. Xie, and N. Guizani, "Detecting malware on x86-based iot devices in autonomous driving," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 80–87, 2019.

[28] B. Sheehan, F. Murphy, M. Mullins, and C. Ryan, "Connected and autonomous vehicles: A cyber-risk classification framework," *Transportation research part A: policy and practice*, vol. 124, pp. 523–536, 2019.

[29] D. Nieuwenhuizen, "A behavioural-based approach to ransomware detection," *Whitepaper. MWR Labs Whitepaper*, 2017.

[30] S. Song, B. Kim, and S. Lee, "The effective ransomware prevention technique using process monitoring on android platform," *Mobile Information Systems*, vol. 2016, 2016.

[31] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection," *arXiv preprint arXiv:1609.03020*, 2016.

[32] U. Sternfeld, "Operation kofer: Mutating ransomware enters the fray," 2015.

[33] N. Hampton, Z. Baig, and S. Zeadally, "Ransomware behavioural analysis on windows platforms," *Journal of Information Security and Applications*, vol. 40, pp. 44–51, 2018.

[34] Z.-G. Chen, H.-S. Kang, S.-N. Yin, and S.-R. Kim, "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, 2017, pp. 196–201.

[35] A. Kharraz, S. Arshad, C. Mulliner, W. K. Robertson, and E. Kirda, "Unveil: A large-scale, automated approach to detecting ransomware." in *USENIX Security Symposium*, 2016, pp. 757–772.

[36] F. Beneventi, A. Bartolini, C. Cavazzoni, and L. Benini, "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, 2017, pp. 1038–1043.

[37] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 559–570.

[38] S. Aurangzeb, R. N. B. Rais, M. Aleem, M. A. Islam, and M. A. Iqbal, "On the classification of microsoft-windows ransomware using hardware profile," *PeerJ Computer Science*, vol. 7, p. e361, 2021.

[39] D. Maiorca, F. Mercaldo, G. Giacinto, C. A. Visaggio, and F. Martinelli, "R-packdroid: Api package-based characterization and detection of mobile ransomware," in *Proceedings of the Symposium on Applied Computing*. ACM, 2017, pp. 1718–1723.

[40] P. Zavarsky, D. Lindskog *et al.*, "Experimental analysis of ransomware on windows and android platforms: Evolution and characterization," *Procedia Computer Science*, vol. 94, pp. 465–472, 2016.

[41] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "A 0-day aware crypto-ransomware early behavioral detection framework," in *International Conference of Reliable Information and Communication Technology*. Springer, 2017, pp. 758–766.

[42] N. Andronio, S. Zanero, and F. Maggi, "Heldroid: Dissecting and detecting mobile ransomware," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2015, pp. 382–404.

[43] Ö. Aslan and R. Samet, "Investigation of possibilities to detect malware using existing tools," in *14th ACS/IEEE International Conference on Computer Systems and Applications AICCSA*, 2017.

[44] G. Kaur, R. Dhir, and M. Singh, "Anatomy of ransomware malware: detection, analysis and reporting," *International Journal of Security and Networks*, vol. 12, no. 3, pp. 188–197, 2017.

[45] A. Ferdowsi, U. Challita, and W. Saad, "Deep learning for reliable mobile edge analytics in intelligent transportation systems: An overview," *ieee vehicular technology magazine*, vol. 14, no. 1, pp. 62–70, 2019.

[46] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, 2019.

[47] R. Coppola and M. Morisio, "Connected car: Technologies, issues, future trends," *ACM Computing Surveys*, vol. 49, no. 3, Oct. 2016.

[48] Automotive Future. https://download.microsoft.com/download/5/0/4/5040df6f-00f1-4e91-abef-082236e7be6e/PSFK_Microsoft_FutureOfAutomotive.pdf.

[49] Microsoft Connected Vehicle Platform helps Automakers Transform Cars. https://blogs.microsoft.com/blog/2017/01/05/microsoft-connected-vehicle-platform-helps-automakers-transform-cars/.

[50] As Location Data Grows, Microsoft Partners with Mapping Companies to Build next World Graph. https://blogs.microsoft.com/blog/2016/12/14/location-data-grows-microsoft-partners-mapping-companies-build-next-world-graph/#sm.0001ido1tzj63f0dztc1xdwcoy8hd.

[51] Y. VirusShare, "Virusshare. com–because sharing is caring," 2019.

[52] A. Arabo, R. Dijoux, T. Poulain, and G. Chevalier, "Detecting ransomware using process behavior analysis," *Procedia Computer Science*, vol. 168, pp. 289–296, 2020.

[53] J. Hwang, J. Kim, S. Lee, and K. Kim, "Two-stage ransomware detection using dynamic analysis and machine learning techniques," *Wireless Personal Communications*, vol. 112, no. 4, pp. 2597–2609, 2020.

[54] C. E. Shannon, "A mathematical theory of communication, part ii," *Bell Syst. Tech. J.*, vol. 27, pp. 623–656, 1948.

[55] K. S. Jones, *Readings in information retrieval*. Morgan Kaufmann, 1997.

[56] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[57] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[58] R. Kohavi, "Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid." in *KDD*, vol. 96. Citeseer, 1996, pp. 202–207.
[59] S. L. Salzberg, "C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993," 1994.
[60] D. J. Hand and R. J. Till, "A simple generalisation of the area under the roc curve for multiple class classification problems," *Machine learning*, vol. 45, no. 2, pp. 171–186, 2001.
[61] H. Sayadi, N. Patel, S. M. PD, A. Sasan, S. Rafatirad, and H. Homayoun, "Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.