Hindawi
Journal of Advanced Transportation
Volume 2020, Article ID 3035741, 9 pages
https://doi.org/10.1155/2020/3035741

Research Article

Malware Detection in Self-Driving Vehicles Using Machine Learning Algorithms

Seunghyun Park and Jin-Young Choi

Graduate School of Information Security, Korea University, Seoul 02841, Republic of Korea

Correspondence should be addressed to Jin-Young Choi; choi@formal.korea.ac.kr

Received 26 July 2019; Revised 19 October 2019; Accepted 19 November 2019; Published 17 January 2020

Guest Editor: Hsing-Chung Chen

The recent trend for vehicles to be connected to unspecified devices, vehicles, and infrastructure increases the potential for external threats to vehicle cybersecurity. Thus, intrusion detection is a key network security function in vehicles with open connectivity, such as self-driving and connected cars. Specifically, when a vehicle is connected to an external device through a smartphone inside the vehicle or when a vehicle communicates with external infrastructure, security technology is required to protect the software network inside the vehicle. Existing technology with this function includes vehicle gateways and intrusion detection systems. However, it is difficult to block malicious code based on application behaviors. In this study, we propose a machine learning-based data analysis method to accurately detect abnormal behaviors due to malware in large-scale network traffic in real time. First, we define a detection architecture, which is required by the intrusion detection module to detect and block malware attempting to affect the vehicle via a smartphone. Then, we propose an efficient algorithm for detecting malicious behaviors in a network environment and conduct experiments to verify algorithm accuracy and cost through comparisons with other algorithms.

1. Introduction

As automobiles become more intelligent, so do transportation systems [1]. New business requirements in the automotive market and advances in automotive communication technology are increasing the connectivity of automobiles. This greater connectivity portends the increased likelihood of future automobile cyberattacks [2]. Therefore, it is necessary to prepare countermeasures for various attack vectors to combat threats to vehicle cybersecurity.

For example, in 2015, Miller and Valasek [3] remotely hacked a traveling Jeep Cherokee to control the audio, windshield wipers, steering and braking, revealing that an unprepared cybersecurity system can threaten driver safety. Furthermore, in 2016 and 2017, Keen Security Lab [4] hacked a Tesla vehicle to demonstrate security threats and potential attacks related to connected vehicles. Typically, connected vehicles are a closed environment that only accepts remote control commands in an authorized communication path, such as a server built by the manufacturer or dedicated applications published by the manufacturer. In a closed environment, unauthorized commands are blocked. However, recent self-driving vehicles share their control signals and internal data with not only the controllers inside the vehicle, but also various unspecified vehicles, infrastructures, and smart devices outside the vehicle in real time. Thus, vehicle network protection should be prioritized in open environments.

The security of a self-driving vehicle is directly related to passenger safety; therefore, it is necessary to comprehensively consider the various attack vectors against vehicles based on the integrity, availability, and confidentiality of their cybersecurity [5]. When a connected vehicle's software is updated, it is essential to verify the integrity of the software. Attackers may use malicious applications to illegally steal privileges or gain access, repackage the software installed in the vehicle by injecting malicious code, and induce the installation of maliciously modified applications. This malicious software looks the same as the authorized software, but malicious code contained in the modified applications can collect the user's input to steal account information, activate abnormal service ports, or retain authorization for the attacker to access later. Such

malicious software can be even used as a medium for additional remote attacks through communication with the command and control server. Thus, it is important to protect vehicle software when either the vehicle is connected to an external device such as a smartphone via an interface inside the vehicle, or a communication channel is opened between the vehicle and surrounding infrastructure. Previous research has installed vehicle gateways which allow only authorized communication to the vehicles and introduced vehicle Intrusion Detection Systems (IDSs) to detect abnormal behaviors in the Controller Area Network (CAN) [6]. However, it is difficult for a gateway or IDS to block these actions in advance, as most malware and adware are behavior-based. In order to detect unknown threats, it is vital to introduce a technology that can detect abnormal behaviors and analyze anomalous indicators using data analysis technology.

In this study, we review the various security threats to self-driving vehicles imposed by malware in Android operating system (OS) and discuss a method for detecting such malware. In an embedded environment such as a vehicle, both response time and detection accuracy are key factors because resources are limited, and real-time responses are required. Therefore, we propose a machine learning-based detection model that can reduce analysis time and improve detection accuracy. The specific contributions of this research are as follows:

(i) We present a method for detecting adware and malware in a self-driving vehicle environment.

(ii) We define the intrusion detection module architecture required to detect malware and prevent it from affecting the vehicle through a smartphone.

(iii) We experimentally compare the detection accuracy and cost of different algorithms and present the most efficient algorithm.

First, we describe the security technology protecting the internal and external communication networks of self-driving vehicles. We then propose an architecture for an intrusion detection module that detects malicious behavior in the vehicle network based on machine learning. Then, we present an effective intrusion detection method and compare it with existing algorithms in experiments. Finally, we present the conclusions and future work.

2. Preliminaries

2.1. Vehicle-to-Device Communication. In the paradigm of vehicle-to-everything communication, communicating with a specific device is termed vehicle-to-device (V2D) communication [7]. Android-based smartphones are typical devices that communicate with a vehicle. Services that identify vehicle operational information or diagnose vehicle abnormalities via a smartphone are classified as performing V2D communication. Initially, to carry out these functions, vehicles were directly connected to an external device outside the vehicle through a universal serial bus connector or Bluetooth, and the data on the device were used. Because a direct wired connection from the vehicle to the device occurred only if the target vehicle was physically occupied, a hacker could not directly control multiple vehicles remotely, even if the vehicles were successfully stolen. Since then, vehicle manufacturers have installed telematics control units (TCUs) or connectivity control units (CCUs) in vehicles and implemented interfaces for remote control of vehicles that include communication functions. In addition, this service is not limited to the original equipment manufacturer. Global telecommunication companies or Internet of Things device manufacturers can also install Long-Term Evolution communication modules on the on-board diagnostics II terminal to collect and manage various data inside the vehicle. When the vehicle is connected to a server or smartphone through such a communication module, information from the vehicle can be transmitted externally. Similarly, it is also possible to control the vehicle by injecting commands to the vehicle from the outside. A connection to a smartphone or external communication device is used not only for convenience services such as music playback and navigation, but also for important functions for updating the vehicle software. If a connection is unauthorized or infected by malicious codes, it can be a serious security threat to the vehicle network. Therefore, security technology to protect the vehicle software and network is essential in V2D communication.

2.2. Android-Based Hacking Attacks. Malicious code is a widely used attack method at the application level that comes in various forms [8]. Various security threats such as leakage of private information, elevation of application privileges, and a denial-of-service (DoS) attack have been reported. The most common attack in the Android OS is the use of an application containing malicious code imported when a specific web page or email is loaded. Most malicious code is injected into the device without the user's awareness during the attack. When an application containing malicious code is executed on an Android OS, the code collects device and user information and sends it to a remote server. It also configures a backdoor by activating the service port to allow the attacker to reenter the device and elevate the privileges of available accounts. Subsequently, the malicious code can gain entire access to the infected device by rooting it. In particular, when an infected Android OS is connected to the inside of a self-driving vehicle, malicious code can be infiltrated directly into the vehicle to take control of the embedded OS or application software environment. For this reason, we need to detect malicious code from a self-driving vehicle.

2.3. Dataset. Recently, machine learning algorithms have been used to detect malicious code. This study proposes a machine learning-based intrusion detection module using the Android Adware and General Malware (AW&GM) dataset [9], which was developed by the Canadian Institute for Cybersecurity (CIC) in 2017. This publicly available dataset comprises Android sandboxes, Android adware, malware, and normal application traffic. It consists of traffic from 1,900 applications downloaded from Google Play (Android official application

Table 1: Three categories of the android adware and general malware dataset.

Class	Malware family	Number of apps
Adware	Airpush, dowgin, kemoge, mobidash, shuanet	250
General malware	AVpass, fakeAV, fakeflash/fakeplayer, GGtracker, penetho	150
Benign	Google play market (top free popular and top free new, 2015-2016)	1,500

market) and is used to classify normal and malicious code based on network traffic. This dataset is categorized with the following three classes (see Table 1).

2.4. Related Work for Protecting Vehicle Communication Networks. Kwon et al. [10] proposed a method for reconfiguring the electronic control units (ECUs) in a vehicle and deactivating attack packets to defend against network intrusion. In the proposed architecture, an IDS is introduced to detect cyberattacks in the network inside the vehicle, and a control module, called a mitigation manager, is applied to mitigate the damage from detected attacks. They then proposed an architecture to deliver commands to reconfigure ECUs, deactivate packets, reconfigure head units, delete packets in gateways at each domain, or switch domains into a secure mode. However, the framework and algorithms for the methodology were only proposed and not developed, and performance evaluations of the specific shape or architecture were insufficient. Therefore, a testbed and simulation environment should be prepared in order to verify the architecture appropriateness based on practical data such as detection accuracy, detection time, and resource utilization.

Han et al. [11] suggested an anomaly intrusion detection method for vehicular networks based on survival analysis. The method is based on an anomaly detection algorithm that detects a suspicious pattern within the usual pattern information. The method aims to detect three typical attack scenarios—flooding attacks, fuzzy attacks, and malfunction attacks—that attempt to manipulate and control using malicious packets. The authors noted that the proposed method can detect unknown attacks; however, they did not describe how to detect scenarios other than the three mentioned.

Zhang et al. [12] presented a cloud-assisted vehicle malware defense framework to defend vehicles against malware attacks. Such a service can help defend resource-constrained vehicle systems against malware by detecting new malware and updating onboard malware defense capabilities. Although the method is a cloud-based malware detection service, in-vehicle devices are also required to perform onboard threat defense functions. The premise of this service is that a single gateway should be able to control all external communication interfaces in the vehicle. If the vehicle cannot access the security cloud, it must find another way to inspect malware, however, no alternatives were explicitly suggested by the authors.

Table 2: User interaction scenarios analyzed in this study.

Category	User interaction scenario
Confidentiality	Information leakage (trip/location records, camera video/images, contact list, call/SMS history)
Integrity	Intentionally manipulated application installation (injection of malicious code disguised as a modified application)
Availability	Continuous resource consumption (large-scale traffic transmission) and system termination (intentionally resulting in various exception cases)

3. Machine Learning-Based Intrusion Detection Module

3.1. Malware Detection in Vehicle Networks. Study Group SG17 of the Telecommunication Standardization Sector, one of the International Telecommunication Unions that develops telecommunications standards, established the Intelligent Transport System (ITS) security investigation branch in order to standardize the ITS [13]. Specifically, X.itssec-4, which covers methodologies for IDSs for in-vehicle systems, defines the system structure and methods. Existing mechanisms for detecting unauthorized access into a CAN, injection of a malicious control message, and DoS attack include vehicle gateways and vehicle IDSs [14]. Attacks using adware and malware have various user interaction scenarios that can intrude into a vehicle through a smartphone (see Table 2).

Connected or self-driving vehicles are connected to external or public networks outside the vehicle via various interfaces. TCUs or CCUs are equipped with a modem and external communication interfaces to enable receipt of Global Positioning System signals and access to mobile networks. In-vehicle infotainment systems, which provide entertainment and information content, enable various applications by applying an embedded OS, such as QNX OS or Android OS. If security design is not considered in wired or wireless networks, these interfaces can be abused as a path for malware or malicious commands to enter the vehicle network (see Figure 1). In particular, the embedded OS environment can be controlled from the malware or malicious commands when these malicious processes bypass OS-level security logic or acquire root authority from self-privilege elevation. Therefore, in order to prevent malicious commands from gaining control of the embedded OS, this paper proposes a CAN gateway architecture that includes an intrusion detection module and detects malicious behaviors when Android OS-based devices are connected to the vehicle.

In this study, a machine learning-based intrusion detection module is installed in the vehicle IDS, which can detect intrusion into the CAN or any abnormalities, so that a head unit or ECU can be protected from malicious code. Such detection methods are implemented in the form of software-based computing modules to monitor malware injection or malicious code behaviors in the vehicle. The software can be installed as a component of the vehicle intrusion detection module or as an anti-virus agent in a head unit.

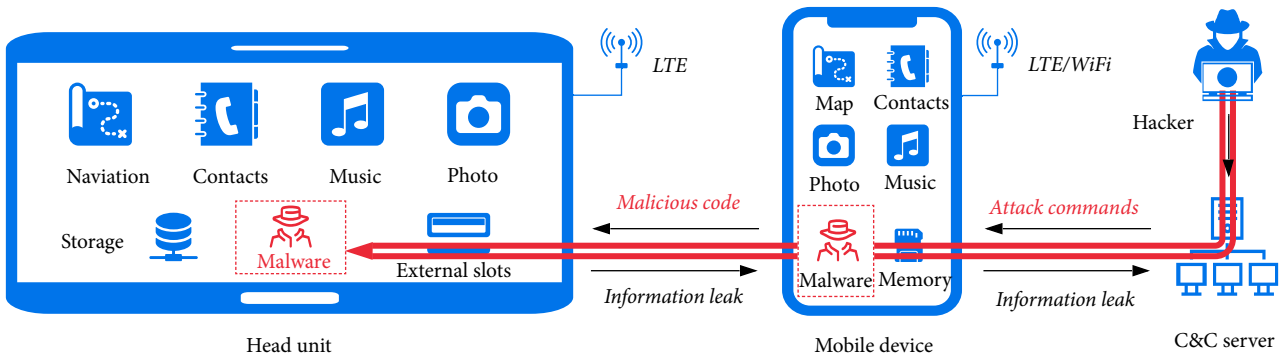

Figure 1: Schematic showing the head unit connected to an Android mobile device.

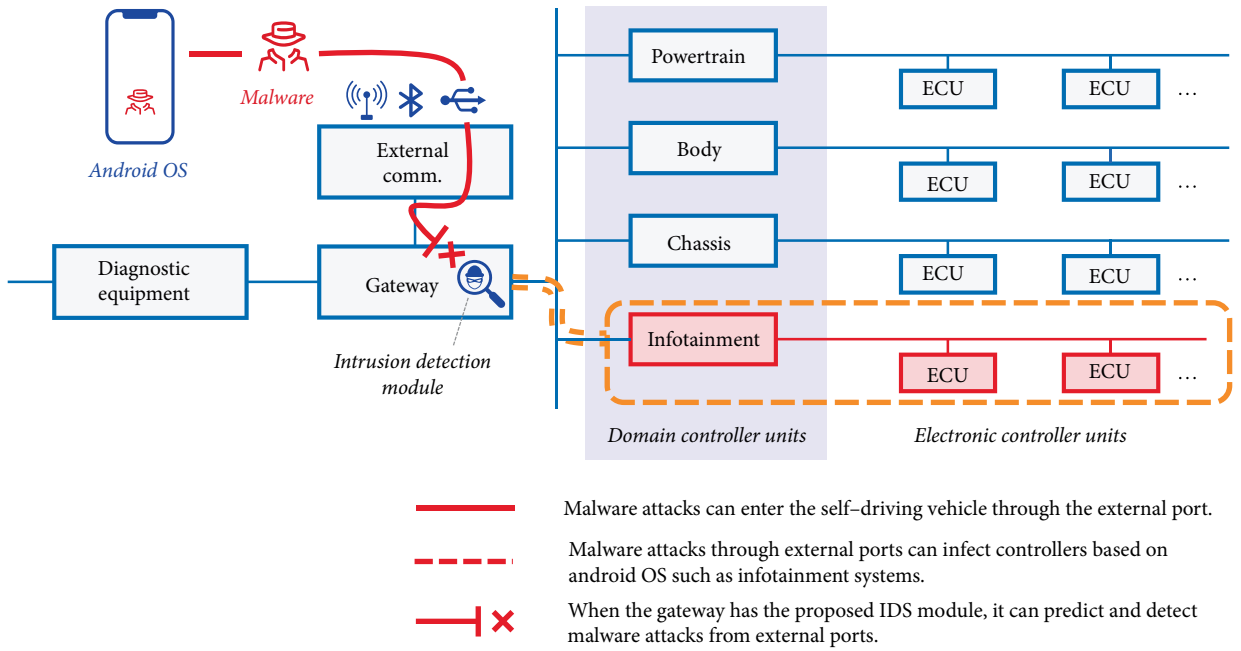

Figure 2: CAN topology with the vehicle gateway and intrusion detection module.

The proposed detection software consists of input, analysis, evaluation, and notification modules. The traffic injected through the CAN is processed through the input module and entered into the analysis module, which is equipped with a machine learning algorithm (see Figure 2). The analysis module evaluates intrusion or abnormal behaviors based on a learned model and provides intrusion behavior information to a user or control center in real time. This machine learning-based intrusion detection module can improve the model's accuracy by repeatedly learning, verifying, and evaluating message patterns. Furthermore, detection rules for malicious behaviors can be updated to the vehicle gateway and each controller to accurately detect malicious code.

3.2. Data Preprocessing for Malicious Code Analysis. The characteristics of 79 features included in the CIC AW&GM dataset are analyzed using the Waikato Environment for Knowledge Analysis [15]. Feature selection is needed to reduce the dimensionality of the data. First, ten-fold cross-validation is applied by employing correlation-based feature selection (CFS) and an entropy-based information gain (IG) method. Constructing a validated dataset for an efficient experimental environment is important in machine learning. In this paper, we propose the improved feature selection (IFS) method, which combines the higher values derived from correlation and IG methods.

The proposed learning algorithm uses the selected network traffic features to detect malware. Unlike existing feature selection methods, IFS finds both greedy features and the highest correlation. There are two broad categories that can be used to measure the correlation between two random variables, one based on classical linear correlation (i.e., CFS) and the other based on information theory (i.e., the IG method). First, a pair of variables is defined for the CFS method and the linear correlation coefficient is derived [16]. In addition, the IG method decides how important a given attribute of the feature vectors is [17]. These two vectors are combined in order to determine the final features from the dataset that are highly

Table 3: Feature selection results.

Category	Selected Features[1] from IFS	F1 score
CFS (5)	*min_flowpktl*, *max_flowpktl*, *max_idle*, *bVarianceDataBytes*, *Init_Win_bytes_forward*	0.796
IG (5)	*avgPacketSize*, *max_fpktl*, *max_flowiat*, *fPktsPerSecond*, *Init_Win_bytes_forward*	0.806

[1]The feature of *min_flowpktl* means minimum length of a flow; *max_flowpktl* means maximum length of a flow; *max_idle* means maximum time a flow was idle before becoming active; *bVarianceDataBytes* means variance of total bytes used in backward direction; *avgPacketSize* means average size of packet; *max_fpktl* means maximum size of packet in forward direction; *max_flowiat* means maximum inter-arrival time of packet; *fPktsPerSecond* means number of forward packets per second; *Init_Win_bytes_forward* means the total number of bytes sent in initial window in the forward direction, respectively. Especially the last item is included in both CFS and IG results.

Input: U is a universal set with all features.

Output: Ω^* is a subset with selected feature by IFS method.

1: Initialize $x_i, y_i, z_i \in U, (1 \le i \le N)$.

2: Get all $r(x_i, y_i)$ by linear correlation coefficient.

3: Sort $\left|r_{x_i,y_i}\right|$ values for $(1 \le i \le N)$.

4: Choose j sets for top x_i with high value of $|r|$, for relevant variable j and $(1 \le j \le N)$.

5: Get combination $x_i, y_i \in C_j$, where $C_j \subset U$ and $n(C_j) = j$.

6: Determine $C_j{}^*$, where the maximum of *F1 score* with x_i.

7: Get all $H(i)$ by *information gain*.

8: Get k is related to highly ranking variable.

9: Choose k sets for top z_i with high value of $H(i)$, relevant variable k and $(1 \le k \le N)$.

10: Get elements $z_i \in C_k$, where $C_k \subset U$ and $n(C_k) = k$.

11: Determine $C_k{}^*$, where maximum of *F1 score* with z_i.

12: Merge $\Omega^* = \{C_j{}^*\} \cup \{C_k{}^*\}$.

Algorithm 1: Improved feature selection

correlated and have a strong impact between classes (see Algorithm 1).

In the CFS stage, we derive the linear correlation coefficient, $r(x_i, y_i)$. In this paper, we determine the verified features x_i with high value of $|r|$. The number of elements in the set C_j with elements x_i is j. These elements are selected with a relevant variable from CFS. The final $C_j{}^*$ consists of a set with elements x_i calculating the highest F1 score (see Table 3 for CFS features). In the IG stage, we derive the IG ranking $H(i)$. The IG method finds the top 20% of z_i features, according to the χ^2 statistical distribution, from the 79 features. It finds that the statistic result is saturated at around k. The final C_k^* consists of a set with elements z_i calculating the highest F1 score (see Table 3 for IG features). The final feature selection is made by finding the union of the CFS and IG method feature sets. In this paper, each method selected five features; in total, nine features are used as input features (one feature was included in both feature sets). A total of 631,955 elements with these features were used for our model.

In-vehicle applications can be infected by Android malware via wireless or wired communication channels, as illustrated in Figure 1. Several studies suggesting IDS architectures have reported that malware can be detected in the network traffic of devices [18, 19]. This paper selected nine features using the IFS method and shows that malware can be detected from the network traffic using a machine learning-based IDS module.

Table 4: Feature scaling results.

Category	F1 score
MinMaxScaler	0.813
StandardScaler	0.810

Table 5: Types of applications and their ratio.

Category	Count	Ratio (%)
Benign	471,597	74.6
Adware	155,613	24.6
General malware	4,745	0.8

As the original data has unique characteristics and distributions, learning from these data may be slow or result in modeling errors. In the case of network traffic, it is essential to perform scaling because each feature has a uniquely defined data range and unit. Scaling is a data preprocessing task that helps prevent underflow and overflow when learning from experimental data. It is performed based on the nine selected features. The F1 score results after applying the MinMaxScaler and StandardScaler are described in Table 4. The MinMaxScaler scales all features to be exactly between zero and one. The StandardScaler, in contrast, does not limit the minimum and maximum values, but ensures that all features have an average of zero and a variance of one. Thus, all features have the same size. A comparison of their F1 score results of the two scaling methods indicates that MinMaxScaler is more advantageous due to the nature of the network traffic, which comprises a wide range of data. Therefore, in this study, the MinMaxScaler technique is applied to each algorithm.

Next, we analyzed algorithms that detect adware and malware typical in Android OS. In this study, these attack detection techniques are compared by applying six machine learning algorithms to the dataset. Furthermore, we analyzed the results of using a general machine learning algorithm, assuming that the computing power employed in the vehicle-embedded software can analyze traffic data using a general specification rather than a high-performance system. The dataset used in this study consists of three classes: benign, adware, and general malware. There is a strong imbalance between these classes (see Table 5).

When the data modeling results are evaluated with general accuracy, the evaluation result may suggest that its

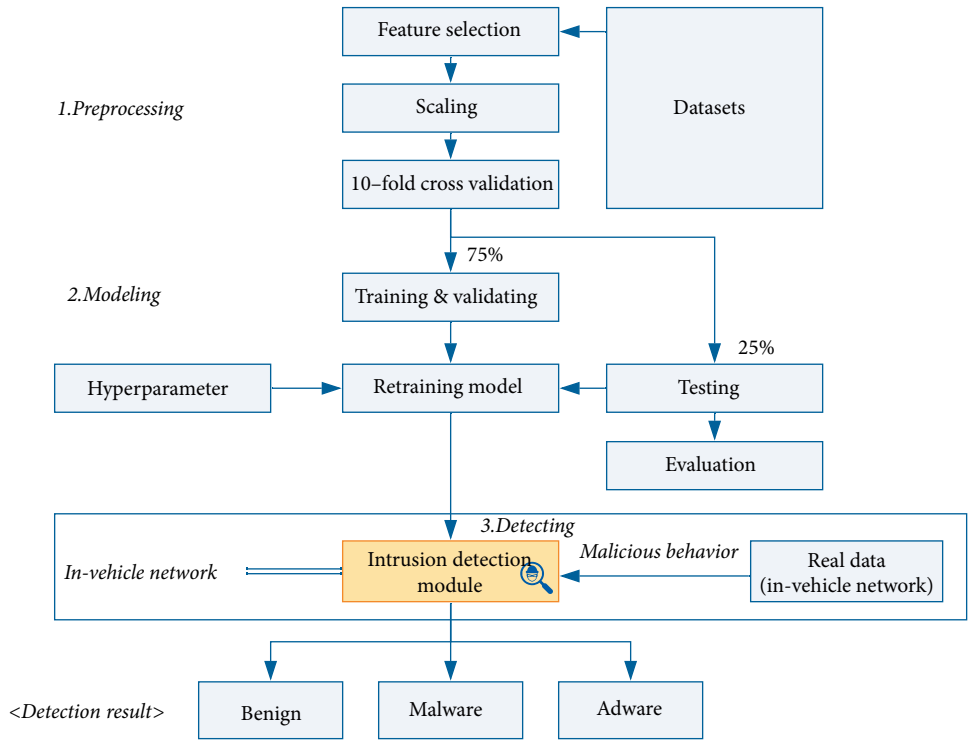

Figure 3: Intrusion detection module in a vehicle network.

performance is good even when it is not. For example, the overall accuracy can be high if the benign category, which has high importance in the dataset, is accurately predicted, even if general malware, which has low importance, is not accurately predicted. Therefore, the F1 score, which uses the harmonic mean based on recall and precision, is used to evaluate prediction accuracy.

In summary, the proposed machine learning-based intrusion detection module detects Android malware for a self-driving vehicle and labels its type (i.e., adware or general malware). The procedure, which is based on the detection of the network traffic deviation on Android OS, is divided into three phases, as shown in Figure 3. The first phase focuses on data preprocessing. Feature selection is performed to select the most relevant features from all measuring features in the dataset. The second phase consists of modeling. Using ten-fold cross-validation, this phase trains the machine learning model using 75% of the dataset and suggests the most suitable hyperparameters for the retraining model. In addition, this phase uses 25% of the dataset for testing and evaluating the proposed intrusion detection module. Therefore, a machine learning model tuned by hyperparameters is created using the training dataset, and a testing dataset is applied to evaluate the model. In the third phase, the intrusion detection module can detect malicious behaviors in real time when real data flows into the self-driving vehicle. Specifically, the proposed intrusion detection module should be included in the vehicle gateway shown in Figure 2.

4. Simulation Results

Six machine learning algorithms—the random forest (RF), decision tree (DT), *k*-nearest neighbors classifier (KC), gradient boosting classifier (GB), extra tree classifier (ET), and bagging classifier (BC) algorithms—are used to analyze the data and their results are compared to those of the proposed algorithm. In addition, we also present hyperparameters for each algorithm for comparative verification of the machine learning used to implement the malware detection module in a self-driving vehicle gateway. It is important to tune hyperparameters for result, performance, and cost optimization when analyzing data using a machine learning algorithm. Indeed, significant differences in the performance and accuracy of analysis results can occur depending on the configuration of the hyperparameters. We present the hyperparameters used in each experiment with the F1 score and elapsed time for each algorithm. These hyperparameters were derived by changing various experimental conditions repeatedly for each algorithm.

The nine input features are defined through the feature selection process and the output is defined using two classification scenarios to analyze the experimental results. In the first scenario (see Figure 4(a)), benign code, adware, and general malware are accurately detected, whereas the second scenario (see Figure 4(b)) is a binary classification scenario where only benign code and adware are detected because general malware accounts for only 0.8% of the dataset. It is meaningful to compare the results of the binary classification because its impact can be predicted through the first scenario.

In this paper, malware detection using machine learning is included to develop the IDS module included in self-driving vehicles. The F1 score used in machine learning calculates the accuracy, recall, and precision values for all cases to evaluate the model's performance. This general method, which took about 3.570 s to verify the dataset on average, is not suitable for real-time detection. We applied a faster F1 score evaluation

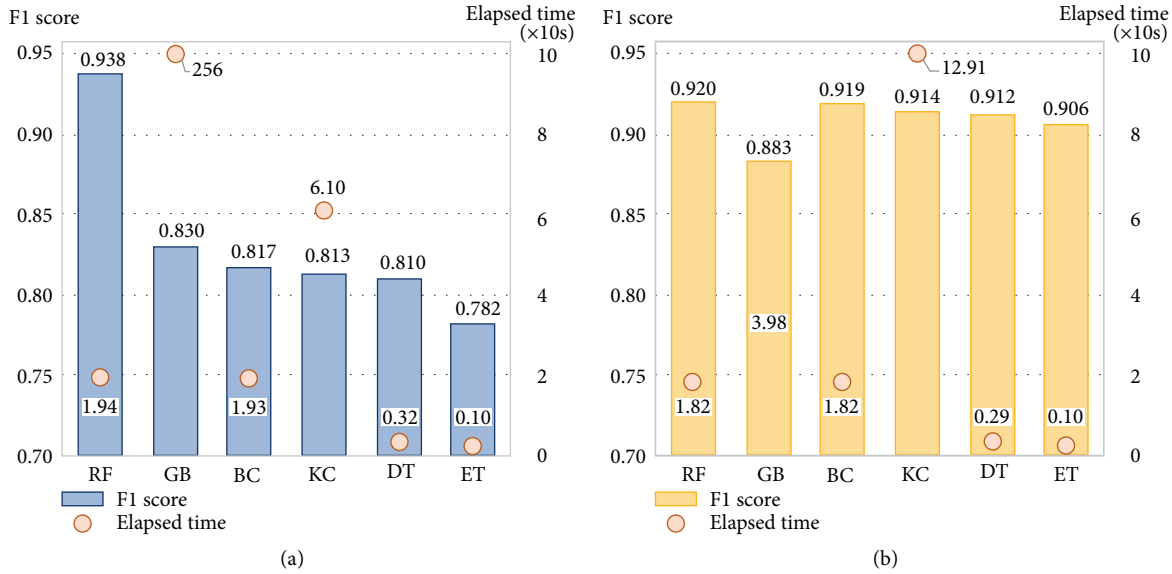

Figure 4: Simulation results. (a) Multiclass classification. (b) Binary classification.

Table 6: Comparison of the proposed score–function model with a basic scikit-learn method.

Evaluation factors	Basic model	Score-function model
Elapsed time (second)	3.570	0.049
Accuracy	0.929	0.929
Precision	0.849	0.849
Recall	0.761	0.761
F1 score	0.796	0.803

method because malware should be detected in real time on autonomous vehicles. In order to generate a class that calculates and returns a confusion matrix quickly, we proposed a new score function. Through this function, we obtained the F1 score directly when the model was training. The function stored the value of the computed confusion matrix and was reusable when the F1 score was called for performance evaluation. In this case, the elapsed average time was 0.049 s, which is acceptable for real-time detection. Abnormal behavior prediction can therefore determine within 0.049 s when new traffic occurred in the self-driving vehicle (see Table 6).

For the RF algorithm, the highest F1 score is obtained when the random state is 42, the number of estimators is 85, the maximum depth is 24, and the maximum number of features is 4. Although the RF's prediction accuracy is typically higher when using binary classification, in this case, it is higher under scenario 1. Overall, the RF algorithm had the highest prediction accuracy of the machine learning algorithms tested. For the DT algorithm, the highest F1 score was observed when the random state is 42 and the minimum leaf sample is 2. For KC, the highest F1 score was observed under the following conditions: uniform weights and 7 estimators. For both DT and KC, accuracy may decrease in datasets with large data imbalances. Moreover, although the KC algorithm exhibits higher prediction accuracy in binary classification scenarios, its learning time is more than twice that in scenario 1. For GB, the highest F1 score is observed when the random state is 42, the number of estimators is 50, the maximum depth is 15, and the maximum number of features is 5. Its prediction accuracy is generally high, but its learning time is the longest of all algorithms, at 2,556,517 ms; for comparison, the learning time of the second longest algorithm, KC, is 60,967 ms and that of the shortest, ET, is merely 976 ms. Therefore, although GB is suitable for binary classification, the learning time costs are too large for general classification. For ET, the highest F1 score was observed when the random state is 42, the splitter is random, and the number of estimators is 100. This algorithm shows the shortest learning time under both scenarios. Although the binary classification has high prediction accuracy and the shortest learning time (95 ms), making it very efficient, its prediction accuracy is significantly reduced in scenario 1. For BC, the highest F1 score is observed when the random state is 42 and the number of estimators is 10. Similar to ET, BC is efficient because of its high prediction accuracy in the binary classification scenario, but it has significantly reduced prediction accuracy under scenario 1.

In summary, the algorithm's overall prediction accuracy was 90% or greater with binary classification for all algorithms except GB. Therefore, in this case, an algorithm with a short learning time can be selected. In order to detect malware or adware in an embedded software environment such as a vehicle, high accuracy and a fast response time are very important. Therefore, the ET algorithm, with its learning time of 95 ms and prediction accuracy of 90.6% in binary classification scenarios, would be suitable. However, considering that the attack detection method in the Android OS is classification scenario 1, the RF algorithm, which has the highest prediction accuracy and a learning time of 19,401 ms, would be the most suitable.

We use the receiver operating characteristic (ROC) curve to evaluate the experimental results of each algorithm. The ROC curve, a widely used tool for binary classification, plots the method's sensitivity against its specificity. The area under

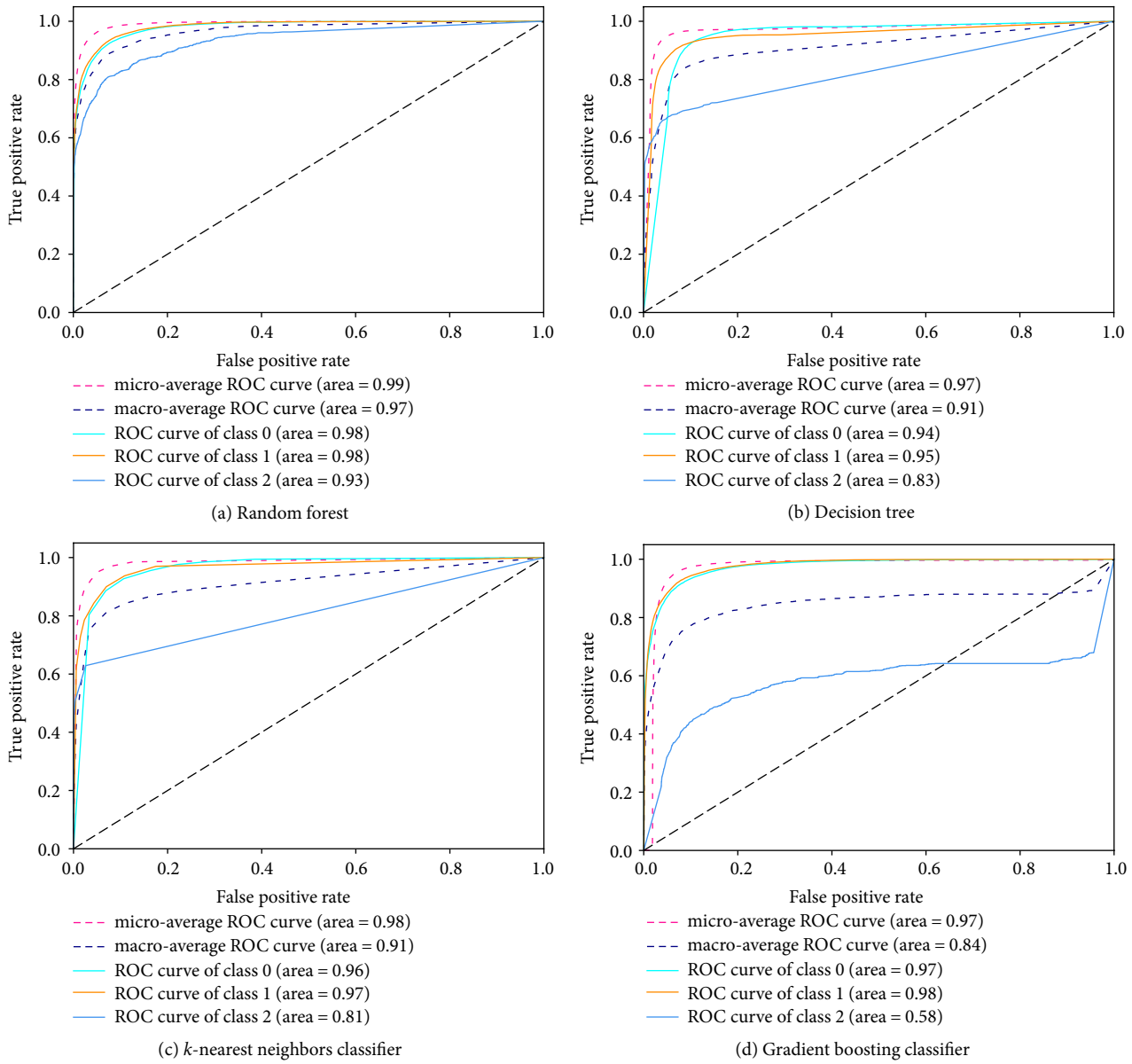

Figure 5: Simulation results with ROC curve and AUC, where "Area" denotes the AUC and classes 0, 1, and 2 correspond to benign, adware, and general malware, respectively. (a) RF. (b) DT. (c) KC. (d) GB.

the ROC curve (AUC), which represents the surface integral under the curve, is an indicator of the detection performance of each classifier. When the curve approaches the graph of $y = x$, the classifier is purely random and the AUC is near 0.5; likewise, detection performance is better when the curve at the top left area is far from the random classifier line. The AUC of the perfect classifier is 1.

We compared the performance of four algorithms: RF, DT, KC, and GB (see Figure 5). In the RF algorithm shown in Figure 5(a), the AUC of the macro-average obtained by calculating the measurement of each class is 0.97 and the micro-average for integrated classes is 0.99. For the imbalanced (class 2) malware, the AUC was 0.93, which is relatively good compared to other classifiers. In Figures 5(b) and 5(c), the DC and KC show similar detection results. However, when class 2 malware is detected, the DC is slightly better than the KC. Moreover, in Figure 5(d), the micro-average and macro-average detection results for GB are 0.97 and 0.84, respectively. However, the detection result of class 2 is low (0.58) due to class 2's scarcity in the dataset. That is, GB can perform well for binary classification, but is not suitable for multiclass classification. In conclusion, the ROC curve and AUC of each classifier show that the RF algorithm better detects malware than the other algorithms.

5. Conclusion

The increasing connectivity of vehicles has also increased security threats. Malicious code can flow into a vehicle's internal network when a device infected with malicious code is connected to the vehicle through an external communication channel. High accuracy and speed are key for detecting

malicious behaviors in the embedded environment of vehicles, where responses must be processed in real time. This study, therefore, analyzed security threats from adware and malware in the Android OS within a self-driving vehicle. Network traffic was analyzed to detect malicious behaviors at the network in the module. In addition, a machine learning-based intrusion detection module for malware detection was proposed. Finally, we proposed a machine learning algorithm that can detect Android malware for vehicles with high accuracy and in a short time. We compared the algorithm's detection accuracy and speed with proposed optimal hyperparameters to six machine learning algorithms. In addition, we also found that we can significantly reduce the elapsed time by using the novel score-function model for real-time detection. Our simulation we demonstrated that our algorithm is highly accurate (92.9%) and fast (0.049 s), making it suitable for real-time malware detection in a self-driving vehicle environment.

Data Availability

The CIC AW&GM dataset can be found in the official webpage of the institute https://www.unb.ca/cic/datasets/android-adware.html.

Conflicts of Interest

Seunghyun Park and Jin-Young Choi declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) [No. 2018-0-00532, Development of High-Assurance (≥EAL6) Secure Microkernel].

References

[1] J. Takahashi, "An overview of cyber security for connected vehicles," *IEICE Transactions on Information and Systems*, vol. E101-D, no. 11, pp. 2561–2575, 2018.

[2] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou, "A review on safety failures, security attacks, and available countermeasures for autonomous vehicles," *Ad Hoc Networks*, vol. 90, pp. 1–13, 2019.

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

[4] S. Nie, L. Liu, and Y. Du, "Free-fall: hacking tesla from wireless to CAN bus," *Black Hat USA*, pp. 1–16, 2017.

[5] C. Riggs, C.-E. Rigaud, R. Beard, T. Douglas, and K. Elish, "A survey on connected vehicles vulnerabilities and countermeasures," *Journal of Traffic and Logistics Engineering*, vol. 6, no. 1, pp. 11–16, 2018.

[6] T. Hoppe, S. Kiltz, and J. Dittmann, "Applying intrusion detection to automotive it—early insights and remaining challenges," *Journal of Information Assurance and Security*, vol. 4, no. 3, pp. 226–235, 2009.

[7] K. Bian, G. Zhang, and L. Song, "Toward secure crowd sensing in vehicle-to-everything networks," *IEEE Network*, vol. 32, no. 2, pp. 126–131, 2018.

[8] N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, "Machine learning aided android malware classification," *Computers & Electrical Engineering*, vol. 61, pp. 266–274, 2017.

[9] A. H. Lashkari, A. F. A. Kadir, H. Gonzalez, K. F. Mbah, and A. A. Ghorbani, "Towards a network-based framework for android malware detection and characterization," in *Proceedings of 2017 15th Annual Conference on Privacy, Security and Trust*, pp. 233–234, IEEE, Canada, 2017.

[10] H. Kwon, S. Lee, J. Choi, and B. Chung, "Mitigation mechanism against in-vehicle network intrusion by reconfiguring ECU and disabling attack packet," in *Proceedings of 2018 International Conference on Information Technology (InCIT)*, pp. 55–59, IEEE, Thailand, 2018.

[11] M. L. Han, B. I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Vehicular Communications*, vol. 14, pp. 52–63, 2018.

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

[13] E. Yagdereli, C. Gemci, and A. Z. Aktas, "A study on cyber-security on autonomous and unmanned vehicles," *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, vol. 12, no. 4, pp. 369–381, 2015.

[14] B. Groza and P.-S. Murvay, "Security solution for the controller area network: bringing authentication to in-vehicle networks," *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 40–47, 2018.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[16] M. Doshi and S. Chaturvedi, "Correlation based feature selection (CFS) technique to predict student performance," *International Journal of Computer Networks & Communications (IJCNC)*, vol. 6, no. 3, pp. 197–206, 2014.

[17] S. Lei, "A feature selection method based on information gain and genetic algorithm," in *Proceedings of 2012 International Conference on Computer Science and Electronics Engineering (ICCSEE)*, pp. 355–358, IEEE, China, 2012.

[18] A. Arora, S. Garg, and S. K. Peddoju, "Malware detection using network traffic analysis in android based mobile devices," in *Proceedings of 2014 Eighth International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST 2014)*, pp. 66–71, IEEE, United Kingdom, 2014.

[19] S. Iqbal, A. Haque, and M. Zulkernine, "Towards a security architecture for protecting connected vehicles from malware," in *Proceedings of 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Srping)*, pp. 1–5, IEEE, Malaysia, 2019.