Recognizing Manipulated Electronic Control Units

2015-01-0202

Published 04/14/2015

Armin Wasicek
University of California

Andre Weimerskirch
University of Michigan

CITATION: Wasicek, A. and Weimerskirch, A., "Recognizing Manipulated Electronic Control Units," SAE Technical Paper 2015-01-0202, 2015, doi:10.4271/2015-01-0202.

Abstract

Combatting the modification of automotive control systems is a current and future challenge for OEMs and suppliers. ‘Chip-tuning’ is a manifestation of manipulation of a vehicle's original setup and calibration. With the increase in automotive functions implemented in software and corresponding business models, chip tuning will become a major concern. Recognizing and reporting of tuned control units in a vehicle is required for technical as well as legal reasons.

This work approaches the problem by capturing the behavior of relevant control units within a machine learning system called a recognition module. The recognition module continuously monitors vehicle's sensor data. It comprises a set of classifiers that have been trained on the intended behavior of a control unit before the vehicle is delivered. When the vehicle is on the road, the recognition module uses the classifier together with current data to ascertain that the behavior of the vehicle is as intended.

A proof-of-concept implementation uses the TORCS racing simulator to generate traces of the engine's behavior. The recognition module extracts features from these traces and feeds them to an artificial neural network (ANN). After training on different tracks, the ANN successfully distinguishes traces originating from the original vehicles as well as traces taken from modified vehicles.

The results show that assessing a vehicle's behavior is feasible and contributes to protect its integrity against modifications. Additionally, the availability of a vehicle's behavioral model can trigger even more interesting applications.

Introduction

Automotive systems are becoming a big target for malicious manipulation. Considering the ongoing massive changes in the automotive industry, it is obvious that software will be a major stakeholder to fuel future innovations in the automotive sector. Threats originate not only from the hobbyist hacker or car tuner, but the emerging automotive technologies will be an attractive target for economically motivated hackers. Cybersecurity means will be required to meet the challenges of a new automotive market. For instance, as the revenue from car sales is decreasing, Original Equipment Manufacturers (OEM) have to look for new business opportunities. Therefore, after market sales and subscription services are getting increasingly important.

These new business models together with their dependency on software, however, make manipulations of the vehicle's computer systems more attractive for economically motivated hackers. For example, chip tuning has been a niche market and thus not a major concern for OEMs up to now. Hobbyist hackers have manipulated the parameters of their vehicle's engine control units to gain, for instance, more horsepower. Consider an automotive software application that can enable higher horsepower. Illegally increasing the engine's power by circumventing this application might be a high motivation for an attacker. Commercializing such a hack can threaten the app provider's business.

Moreover, from an OEM's point of view it is required that the vehicle is operated within its technical specification. Failures and accidents that happen during this phase can harm then reputation of an OEM (e.g., bad news coverage in the media). If a vehicle becomes known to be exploitable, prospective customers might refrain from actually buying this vehicle model. Moreover, driving a vehicle outside the limits of its specification and certification, enable a great potential of malfunction and defects with catastrophic consequences that are not present in the originally configured vehicle.

This paper proposes a system architecture to securely and safely monitor a vehicle's telemetry data and engine parameters in order to recognize anomalous and therefore potentially manipulated behavior. Particularly, the paper elaborates on:

- A security architecture for automotive systems
- Monitoring of an engine's behavior during operation

- Machine learning algorithms facilitating the recognition of malicious behavior
- Demonstration in a case study using a car racing simulation

The paper is organized as follows: The next section elaborates on the chip tuning attack model. Next, an architecture to recognize chip tuning attacks is presented. The subsequent sections present the proposed monitoring and detection mechanisms and their application in a case study. Finally, related work is described and a conclusion is drawn.

Attacks on Electronic Control Units

Electronic Control Units (ECU) form the backbone of a vehicle's computing infrastructure. Most commonly, ECUs perform tasks related to the control of the vehicle's physical dynamics. A modern car deploys up to 100 ECUs [5]. The percentage relating the cost of electronics and the production cost of the entire vehicle highlights the rising importance of ECUs in vehicles: it increased from 19% in 2004 to 40% in 2010 and it will eventually reach 50% in 2020 [2].

Traditional applications are, for instance, engine control, antilock braking system (ABS) and electronic stability program (ESP). Future applications will include driver assistance systems (e.g., Autonomous Emergency Braking), advanced energy management in hybrid and electrical vehicles, as well as information exchange (i.e., vehicle-to-vehicle, vehicle-to-infrastructure). These emerging services pose opportunities to deliver new, valuable services to drivers and passengers, but also represent new risks that have to be addressed.

Protecting the integrity of the vehicle's E/E platform is a major effort for cybersecurity in the automotive domain. Integrity and availability directly relate to a system's dependability, thus, a failure in these properties might propagate to the safety domain. Assuming that an attacker employs economic thinking, availability is a lesser concern than integrity, because a failure in availability caused by a manipulation usually results in denial-of-service, which serves neither the attacker nor the vehicle holder.

Chip Tuning

This paper focuses on chip tuning. Chip tuning has accompanied the introduction of ECUs in vehicles right from the beginning. The basic observations made in this study translate, however, easily to other automotive systems.

Chip tuning attacks aim at changing the behavior of a vehicle's control algorithms by modifying the vehicle's software. Control algorithms are increasingly implemented through digital control systems, thus, as programs stored in a vehicle's ECU. The critical parts of a control program are the parameters of the control algorithm. These parameters adjust a control algorithm to the intended behavior in a specific engine. Usually, these parameters are stored in a table within an ECU's flash memory.

Definition 1: Chip tuning refers to any attempt to change the behavior of a vehicle with respect to control. The car tuner intrudes ECUs by illegally accessing its flash memory and programmatically resetting parameters or software, or the attacker adds hardware components that act in an abnormal manner.

An exemplary chip tuning attack might be executed as follows: The tuner reads out the ECU's flash memory to produce a raw binary dump of the control application's binary image. This can be done, for instance, using the On-Board Diagnostics (OBD) interface [6] of an ECU. Next, the image is disassembled to reveal the structure of the code. The next step is to locate and modify the desired parameters. This is often done in a try-and-error fashion. The modified image is then reprogrammed in the original ECU's flash memory. Various companies offer chip tuning as a service. These companies assist vehicle owners or they work on behalf of the owners to carry out chip tuning. For example, a popular tuning attack is to increase the engine's horsepower by manipulating the engine's ECU.

System Architecture for Intrusion Detection

Figure 1 describes the proposed system architecture. There is a central gateway (CGW) that connects to the OBD2 port, telematics and infotainment units, as well as one or several communication busses (e.g. CAN) with connected ECUs. We assume that data on the in-vehicle communication bus is not encrypted but ideally it is authenticated. Such authentication stops obvious packet injection attacks and reduces the attack surface to attackers that successfully extracted cryptographic keys or successfully compromised an ECU. We also expect that in-vehicle authentication reduces error rates of the intrusion detection and prevention system (IPS). Ideally ECUs (including telematics and infotainment) are equipped with firewalls/ packet filters that only allow packets to pass that were white-listed.

The IPS is installed at least in the CGW but additional IPS could be deployed in dedicated or even all ECUs. A recognition module in the IPS monitors the in-vehicle communication packets and also the ECU status it is installed in. The IPS has means to react to detected misbehavior. We suggest that the IPS regularly pings all essential ECUs to notify them of a normal status. If ECUs do not receive pings anymore for a certain amount of time periods, they can conclude that an anomaly was detected by the IPS. The IPS is able to provide reports of detected abnormal behavior to a central server that analyzes the reports and potentially updates the IPS' detection rules (e.g. as part of a firmware update).

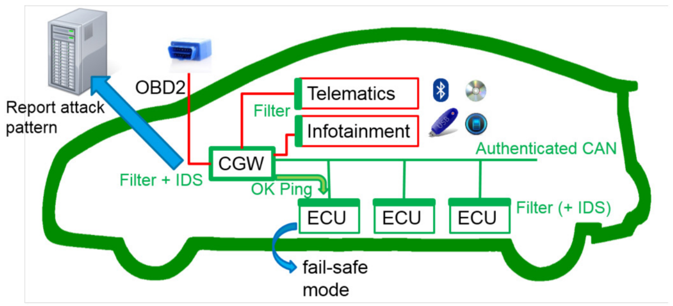

Figure 1. System architecture

Recognizing Manipulated Behavior

Manipulations of an ECU might be detected in two ways:

a. **Internally** by verifying the ECU's flash memory through checksums and other integrity protection mechanisms
b. **Externally** through monitoring the ECU's behavior and comparing to a reference of the intended/specified behavior

Internal verification is a very worthwhile approach to protect an ECU's integrity. It is, however, often not sufficient, because integrity protection mechanisms can fail. Researchers have demonstrated that flash protection mechanisms can be overridden [7]. Therefore, we advocate an external approach by monitoring an ECU to *complement* the internal verification.

Challenges

A manipulated engine does not always exhibit manipulated behavior. For example, even through the controller might be manipulated it might not become visible at all times. Manipulation in the control setting is not a binary property, on the contrary to cryptography where a key can be either secret or disclosed. Consider a manipulated engine controller that maximizes the engine's horsepower. In an urban driving situation involving frequent stops, the engine will most likely not exhibit the manipulated behavior, because quick accelerations or similar maneuvers might not be possible. Although, an intrusion is present in the ECU, it is not activated and consequently it cannot be observed.

Figure 2 depicts this relationship between ‘specified’ and ‘modified’ behavior. The state space of the ECU is partitioned in safe and unsafe states. Unsafe states refer to states that might have catastrophic consequences like a crash or irreversible damage. Usually, fault tolerance mediates between safe and unsafe system states. Regarding security the boundaries get blurred. Clearly, if a system operates within its specified behavior, it is safe and secure. If a manipulation is triggered, the system will transition to a modified behavior. This modified behavior might still be in the safe partition, but it could also cause unsafe system states. For instance, chip tuning aims to modify a system, while still being safe. However, if the car tuner goes over the top or works imprecise, the engine might get damaged. Recovery mechanisms can repair the system (e.g., by reflashing the ECU). A prerequisite for recovery is that the manipulation has been recognized.

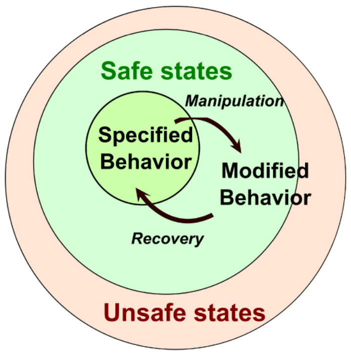

Figure 2. Diagram showing the interaction between safety and security

Learning an Engine's Characteristics

Machine learning is the field of learning statistical models from data. As every engine can have its own characteristics and the characteristics are even subject to change over time, there is no single equation that covers the entire variety of engine behaviors. Therefore, this work employs machine learning techniques to build a statistical model of the engine's behavior. Such model can be built in a controlled environment, for instance, when the vehicle is on the test track or when it is under maintenance. Training data is collected and used to calibrate the statistical model. During operation the current behavior of the vehicle is constantly checked against the model of its accepted behavior. The IPS' recognition module then determines, if the actual behavior deviates from the accepted, trained behavior. Such a deviation is then related to potential manipulations. Note that the IPS runs in the CGW and monitors the engine ECU externally.

Recognition Module

The recognition module encompasses all programs and data collection and processing routines that are required to determine manipulated behavior of an engine. Sensors provide the recognition module with current telemetry data that the recognition module processes to generate a report to any interested party (e.g., the car holder, the maintenance technician, or the OEM).

In order to recognize manipulated behavior, the recognition module aims to detect anomalous behavior in the telemetry data of the engine. By solving the problem of recognizing manipulated behavior through anomaly detection, we implicitly account for the circumstance that a manipulated engine will not always exhibit manipulated behavior, but only under specific conditions.

Figure 3 depicts the basic workflow of the recognition module. At each analysis step a portion of the data is used to compute a feature vector (i.e., an individual). The insight exploited in this work is that the trained model will reconstruct some small number of individuals poorly and these can be considered as anomaly or outliers. Ranking data according to the magnitude of the reconstruction error and thus computing an anomaly score measures outlyingness. The recognition module searches for bundles of anomalies in the telemetry data stream and reports a potential manipulation, if a certain threshold is exceeded. The manipulation detector block performs a plausibility check by fusing the result of the anomaly detection with other relevant data such as a diagnostic data. For instance, if the engine is faulty, it might produce a high number of anomalies that might lead to false positives. The result of the recognition module is reported to the user. Thus, the recognition module implements an open loop control.

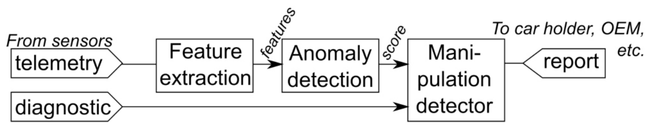

Figure 3. Recognition module process flow

Features

Features represent characteristic forms or shapes of an object under observation. Features are measurable and quantifiable. An individual is a vector of measurements of individuals.

Feature selection

In order to characterize the behavior of a regular combustion engine the three parameters speed, RPM, and torque were selected. These three parameters are deemed to be able to characterize vehicles and their behavior in different traffic situations (e.g., going uphill, in turns, etc) sufficiently well to draw a differentiated picture of the engine behavior. All three parameters were measured at the same instant and carry therefore the same timestamp. Optionally acceleration as a fourth feature could be computed from this data.

This selection of features is aimed to distinguish more powerful engine configurations from less powerful ones. In order to recognize other manipulations (e.g., towards fuel consumption), another set of features will most likely be selected.

Feature extraction

The selected features are collected in the form of time series data. Time series data cannot be directly used for the intended machine learning approach. First, the time series is chopped in sequences that are used as atomic input units. Then features that can be fed to the neural network are extracted from each sequence. Following the feature extraction approach described in [9], following statistical moments features are computed to characterize each sequence and to make different sequences comparable:

- **Mean value** of all values in the sequence
- **Standard deviations** or variance measures amount of variation or dispersion from the average
- **Variance** is the square of the standard deviation
- **Skewness**: degree of asymmetry of values around the mean value
- **Kurtosis:** relative peakness or flatness

The features are computed on the original (primary) sequence as well as on a shifted sequence. Figure 4 depicts a flow diagram of the feature extraction method after [9].

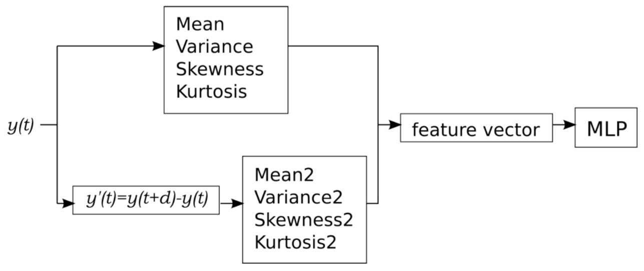

Figure 4. Feature extraction of time series data

Feature scaling

Unity-based normalization is applied to all features *X* that brings them in the range *[0,1]*:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Normalizing values eliminates effects of different magnitudes of single features. This facilitates the comparison of different features in a feature vector.

Anomaly Detection Algorithm

Technically, the anticipated anomaly detection algorithm has to solve a one-class classification problem. Since it is not known at design time in which ways the vehicle might be tuned, only normal data can be used to train the model. Thus, only one label is available. This is also called semi-supervised anomaly detection. The survey in [11] lists several possible methods to perform this task. The proposed one-class classification approach is based on a particular variant of artificial neural networks (ANN), namely a bottleneck type network. Traditionally, these ANNs were used for data compression, the paper in [10] suggested to use them on one-class classification problems [12]. The hidden layer generalizes all trained features, thus, it stores the typical behavior of an engine.

As depicted in Figure 5, such a network consists of an input layer and an output layer of equal sizes, with an intermediate layer of smaller size in-between. For example, using the three selected parameters speed, RPM, and torque for the feature vector and extracting five features for each parameter, the resulting ANN encompasses 30 input and output units. Typically, six units are used in the hidden layer [12].

The training of a bottleneck ANN works as follows: First, a neural network is trained on the normal training data to learn the different normal classes. Second, each test instance is provided as an input to the neural network. Then outputs are combined using a Root-Mean-Square (RMS) function to compute an anomaly score for each feature vector.

$$Score = \| x - \hat{x} \|^2$$

The anomaly score is a measure of similarity to the trained, normal features. If it is close to that of the training set, it is assumed that the feature vector under consideration is normal, otherwise it is an anomaly.

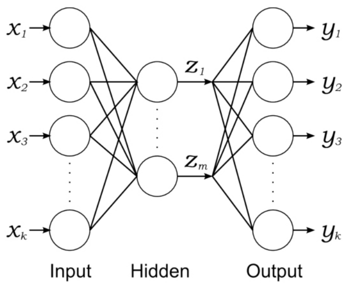

Figure 5. Bottleneck Neural Network

Case Study: Racing Simulation

The case study uses the TORCS racing simulator [8] to simulate vehicles with different engine configurations. TORCS provides a realistic physics environment, car models, racetracks, and an interface to implement robots that are computer programs steering vehicles. The simulator has a logging interface that was used to output time-stamped engine data, namely speed, RPM, and torque. This datasets were then imported and preprocessed in Matlab and feed into the recognition module.

Simulation configuration

TORCS was configured with the standard parameters of version 1.3.6. The simulation was executed from the command line to shorten experiment time.

Cars

The car model 'p406' simulating a Peugeot 406 is the baseline for our experiments. A car in TORCS consists of a configuration contained in an XML file. All different kinds of vehicle parameters are configurable, e.g., chassis size, aero dynamical properties, wheel rigidity, axles diameter, etc. For the purpose of emulating a manipulated engine, the 'Engine' parameter was modified. An RPM/ torque curve characterizes the performance of an engine. Figure **6** depicts the engine characteristics of the employed car model. It shows two curves, blue for the original engine, red for the manipulated engine. The manipulated engine increased its torque by values between 10 Nm and 30 N. The same robot was used for each vehicle configuration.

Tracks

The simulation was executed on a set of 41 racing tracks. The standard as well as the expansion pack was used for the experiments. Some tracks were discarded, because the robot did not manage to successfully complete every track. The tracks are a representative sample to accommodate for many different driving situations and maneuvers. Figure 7 depicts an overview of the driving performance on all tracks with the two engine configurations. It shows that the employed manipulation resulted in lower track times, thus, increased the overall speed performance.

Recognition of Manipulated Behavior

An ANN was trained according to the parameters described in Table 1 using the MATLAB Neural Network Toolbox. Figure 8 shows the training performance for each of the three datasets (i.e., 5 s, 1 s, 0.5 s sampling periods) for six nodes in the hidden layer. The ANNs are converging after a few iterations.

The trained ANN was then validated using traces from the manipulated engine. Figure 10 shows the performance of different numbers of nodes in the hidden layer.

The Manipulation detector contains a threshold parameter for the anomaly score to determine, if a current sample represents an anomaly or not. Depending on the selected threshold different anomaly rates are resulting. Figure 9 shows the relation between the anomaly rate for original and manipulated behavior. Receiver Operating Characteristic (ROC) curves is a standard means used to represent the performance of a particular recognition mechanism. It shows the relation between true positives and false positives under a varying parameter that is this case the threshold in the Manipulation detector. The figure reveals that the True positive rate is larger than the False positive rate, thus, the recognition works. Figure 10 compares the ROC curve for different numbers of hidden units in the ANN. We read out of the graph that eight seems to be the best choice.

The ROC results look promising, but leave definitely room for optimization. Generally spoken, unsupervised learning will not give as good results as supervised learning. Since the proposed system does not rely on single triggered anomalies, but rather looks at unusual high rates of detected anomalies, the impact of the fair accuracy of this diagnostic test is not deciding for the overall recognition.

Consequently, an attacker who tries to conceal his activities has to aim at keeping the potential number of detected anomalies low. If the attacker manages to manipulated the ECU in a way that the resulting anomalous behavior cannot be verified, the attack remains undetected. Particular care has to be taken when choosing the parameters for the recognition module to minimize the difference between specified and undetected behavior.

Table 1. Neural Network parameters

ANN Structure	Feedforward	Inputs: 30	Hidden: 8	Outputs : 30
Training function	Levenberg-Marquardt backpropagation			
Performance function	Mean-square-error			
Regularization	Automated Regularization (trainbr)			

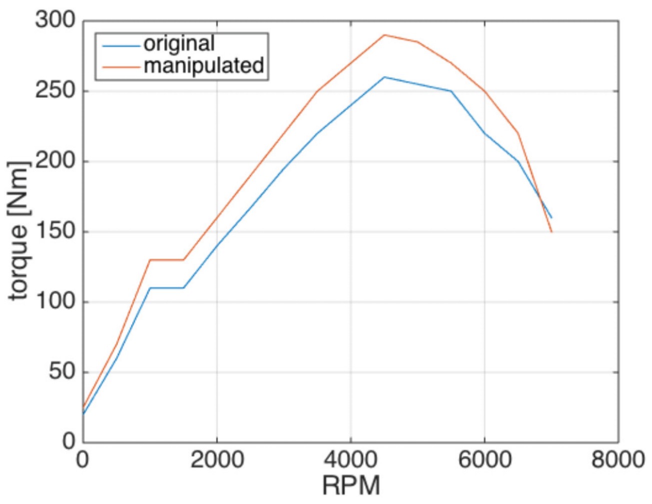

Figure 6. Engine characteristics

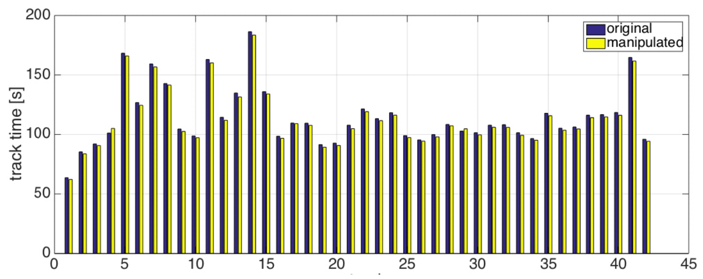

Figure 7. Speed of manipulated engine on different tracks

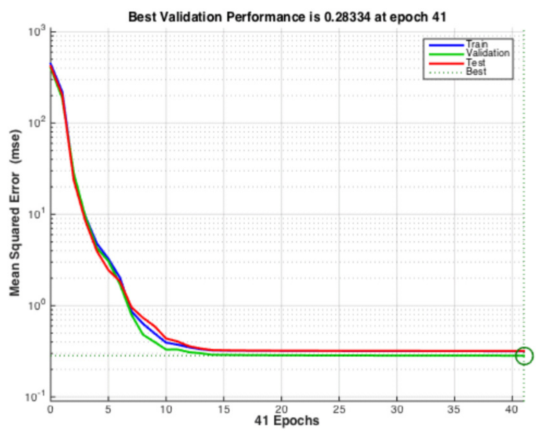

Figure 8. Typical Neural Network training performance

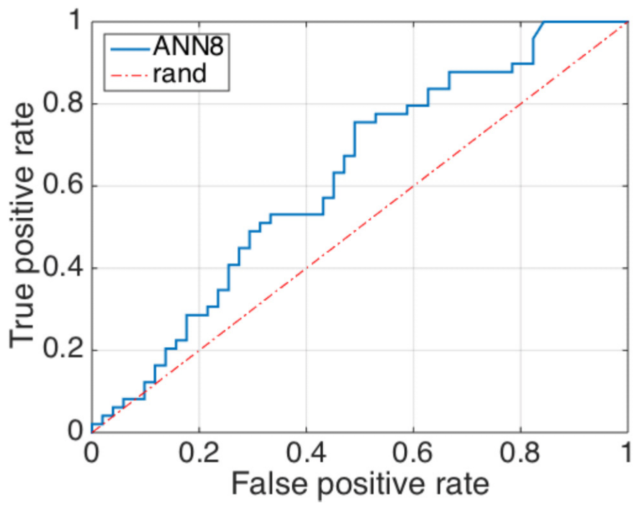

Figure 9. ROC of the anomaly detector with 8 hidden units

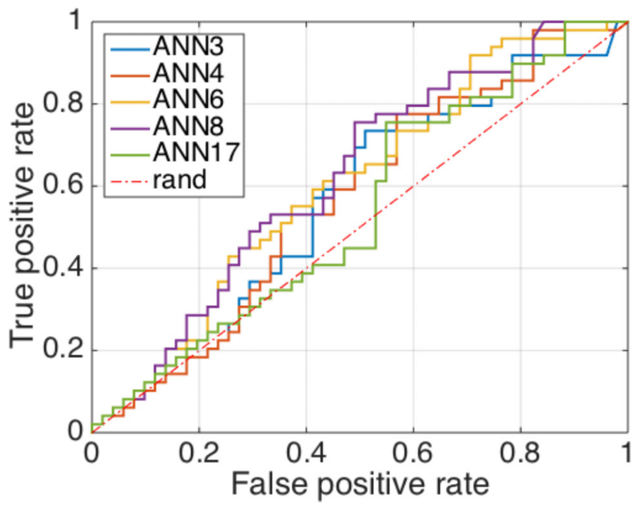

Figure 10. ROC for a varying number of hidden units

Related Work

Automotive security is a research field emerging at a high speed. Several surveys and overview papers are available [15] [17] [18]. Automotive intrusion detection systems (IDS) have been proposed in [16]. An approach to IDS using the entropy of messages on the automotive bus is in [19]. First commercial IDS are appearing on the market[1]. Chip tuning and IP protection in automotive environments has been extensively discussed in the literature [15].

Diagnostics of machinery is a large application field for machine learning [1]. ANNs have been used to diagnose faults in combustion engines [10]. Other variants of bottleneck ANNs are Replicator Neural Networks (RNN) [13], and autoencoder neural networks [14], which have been used in [20] to analyze (electrical) engine behavior.

Conclusion

This paper presents a method based on Neural Networks to differentiate between original and manipulated behavior of an engine. Studies have shown that the selected approach is feasible. The results look promising, however, there is still some room for improving the anomaly detection algorithms. Future work has to include adaptive learning to accommodate for ageing and repair effects of the engine. Moreover, faulty behavior has to be considered as well. Fusing the recognition module with diagnostic data can enable a powerful approach to assess the state of the vehicle and the applications will go far beyond that of recognizing manipulations.

References

1. Kankar, P.K., Sharma, S.C., Harsha, S.P., “Fault Diagnosis of Ball Bearings using Machine Learning Methods”, In Elsevier, Expert Systems with Applications 38. pages 1876-1886, 2011
2. Wallentowitz, H., Freialdenhoven, A., Olschewski, I., “Strategien in der Automobilindustrie: Technologietrends und Marktentwicklungen.” Teubner Verlag / GWV Fachverlage GmbH, 2009 doi:10.1007/978-3-8348-9311-6
3. Wasicek, A., “Copy protection for automotive electronic control units using authenticity heartbeat signals.” 10th IEEE International Conference on Industrial Informatics (INDIN), 2012, doi:10.1109/INDIN.2012.6301060.
4. Wasicek, A, El-Salloum, C., Kopetz, H., “Authentication in Time-Triggered Systems Using Time-Delayed Release of Keys”, In 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), pp. 31-39, 2011 doi:10.1109/ISORC.2011.14
5. Broy, M., Kruger, I.H., Pretschner, A., Salzmann, C., “Engineering Automotive Software,” Proceedings of the IEEE, vol.95, no.2, pp.356,373, Feb. 2007 doi:10.1109/JPROC.2006.888386.
6. SAE Surface Vehicle Standard, “E/E Diagnostic Test Modes”, SAE Standard J1979, Rev. Sep. 2010.
7. Skorobogatov. Sergei P.. Copy Protection in Modern Microcontrollers. Online. http://www.cl.cam.ac.uk/sps32/mculock.html.
8. Wymann B., Espi E.é, Guionneau C., Dimitrakakis C., Coulom R., Sumner A.. TORCS: The Open Racing Car Simulator, v1.3.6, 2014.
9. Nanopoulos, A., Alcock, R., Manolopoulos, Y., “Feature-based Classification of Time-series Data,” International Journal of Computer Research, Vol. 10, pp 49-61, 2001
10. Wu, J-D., Liu, C.H., “An expert system for fault diagnosis in internal combustion engines using wavelet packet transform and neural network,” In Expert Systems with Applications, Vol. 36(3), pp. 4278-4286, April, 2009 doi:10.1016/j.eswa.2008.03.008.

11. Chandola, V., Banerjee, A., Kumar, V., “Anomaly Detection : A Survey”, ACM Computing Surveys, Vol. 41(3), Article 15, July 2009 doi:10.1145/1541880.1541882.
12. Manevitz, L., Yousef, M., “One-class document classification via Neural Networks”, In Elsevier, Neurocomputing, Vol. 70(7-9), pp. 1466-1481, 2007 doi:10.1016/j.neucom.2006.05.013
13. Hawkins, S., He, H., Williams, G., Baxter, R., “Outlier Detection Using Replicator Neural Networks”, Springer LCNS 2454, Data Warehousing and Knowledge Discovery, pp 170-180, 2002
14. Markou, M., Singh, S., “Novelty detection: a review-part 2: neural network based approaches”, In Elsevier, Signal Processing, Vol. 83(12), pp. 2499-2521, 2003 doi:10.1016/j.sigpro.2003.07.019
15. Wasicek, A., “Protection of Intellectual Property Rights in Automotive Control Units,” *SAE Int. J. Passeng. Cars - Electron. Electr. Syst.* 7(1):201-212, 2014, doi:10.4271/2014-01-0338.
16. Hoppe, T., Kiltz, S., Dittmann, J., “Security threats to automotive CAN networks - practical examples and selected short-term countermeasures,” Proceedings of the 27th international conference on Computer Safety, Reliability, and Security (SAFECOMP), 2008, pages 235-248, doi:10.1007/978-3-540-87698-4_21
17. Studnia, I., Nicomette, V., Alata, E., Deswarte, Y., Kaâniche, M., Laarouchi, Y., “Survey on Security Threats and Protection Mechanisms in Embedded Automotive Networks,” The 2nd Workshop on Open Resilient human-aware Cyber-physical Systems (WORCS-2013), 2013
18. 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 Proc. 20th USENIX Security, San Francisco, CA, 2011
19. Muter, M., Asaj, N., “Entropy-based anomaly detection for in-vehicle networks”, Porc. of IEEE Intelligent Vehicles Symposium (IV), pp. 1110 - 1115, 2011 doi:10.1109/IVS.2011.5940552
20. Petsche, T., Marcantonio, A., Darken, C., Hanson, S.J., Kuhn, G.M., Santoso, I., “A Neural Network Autoassociator for Induction Motor Failure Prediction”, In NIPS, MIT Press, pp. 924--930, 1996

Contact Information

The corresponding author is Armin Wasicek (arminw@berkeley.edu).

Acknowledgments

This research was in part supported by a Marie Curie IOF Action within the 7th Framework Programme under the funding ID PIOF-GA-2012-326604 (*MODESEC*). The responsibility for the content rests with the authors.

Definitions/Abbreviations

ABS	- Anti-lock Braking System
ANN	- Artificial Neural Network
CAN	- Controller Area Network
CGW	- Central Gateway
ECU	- Electronic Control Unit
ESP	- Electronic Stability Program
IDS	- Intrusion Detection System
IPS	- Intrusion Detection and Prevention System
OBD	- On-board Diagnostics
OEM	- Original Equipment Manufacturer
RMS	- Root-Mean-Square
ROC	- Receiver Operating Characteristic
RPM	- Revolution per Minute
XML	- eXtensible Markup Language

ISSN 0148-7191

http://papers.sae.org/2015-01-0202