OCPP security - Neural network for detecting malicious traffic

Adrian Gabriel Morosan
University Politehnica of Bucharest, ROMANIA
Splaiul Independentei 313
Bucharest
morosan.ag@gmail.com

Florin Pop
University Politehnica of Bucharest, ROMANIA
Splaiul Independentei 313
Bucharest
florin.pop@cs.pub.ro

ABSTRACT

Because the electric mobility has its focus on eco-friendly means of transport, a distributed platform designed for a smart city environment that can manage the electrical charging stations is vital. One of the major problems of distributed systems and cloud is security. The purpose of this article is to determine the malicious traffic using a backpropagation neural network. The main focus of the paper is to present a composite network that is able to detect faulted, random and normal types of traffic.

CCS Concepts

•**Computer systems organization** → **Neural networks;**
•**Networks** → *Network mobility;* Network security;

Keywords

electric mobility; security; OCPP; backpropagation; neural networks

1. INTRODUCTION

In the actual context where pollution is one of the most challenging problems, electrical vehicles could play a crucial role in decarbonizing transport and in keeping the environment clean [1]. Due to increasing need for electrical vehicles, the smart grid will be harder to monitor and manage. For this purpose, the usage of open protocols in communication between entities will address this issue [2, 3]. OCPP (Open Charge Point Protocol) is a solution - a standard communication between charging stations and the central server. The current version of this protocol is 1.6, there is a lot of work involved in releasing a new version - 2.0, which is currently in beta stage. OCPP is a standard de facto, a protocol which does not reach its full maturity. Rather than wait for this technology to fully grow and then retroactively try to develop security and privacy safeguards, it would be much more convenient to consider some privacy and security issues now, while it is still young and malleable [4].

RACS '17, September 20-23, 2017, Krakow, Poland

ACM ISBN 978-1-4503-5027-3/17/09· · · $15.00
https://doi.org/10.1145/3129676.3129693

The overall suggestion for OCPP implementation is the usage of TLS/SSL to secure communication, more precisely, HTTPS and WebSocket Secure (WSS). However, introducing TLS increases the overhead and since most charging stations use cellular network this means additional cost. Even with the usage of secured connection, basic requirements such as end-to-end security and non-repudiation are not always provided. These two security aspects are fulfilled only for one communication link, but not across multiple links.

Since the secure layers are only a suggestion and a recommended best practice we need to make sure that the stations' traffic is genuine. One way to ensure this is by using authentication of stations to the central server, but similar with the secure layers, this is not implemented by all providers, both on stations and central servers. A solution for the problem would be a system that differentiates the malicious traffic from normal traffic.

Machine learning is used very often in finding similarities and for classification rather than identifying outliers. For this, the proposed solution splits the traffic into two classes: normal and not normal. The latter one is mapped to erroneous responses from the server because, in most of the cases, when a station is compromised it sends multiple similar or inconsistent requests to the server resulting in an error or invalid response.

For a system that detects there is a high cost of errors. It is very important to have very small rates of both false positive and false negative. In the case of false negative, a benign traffic is considered malicious resulting in no additions to database and isolation for that station. From a business perspective, this could mean there are no data regarding a transaction or the impossibility for a client to authorize a card and start a transaction. On the other hand, false positive means that a malicious traffic can communicate with central server as a normal one, being able to start transaction, populating in this way database with fake data [5].

2. RELATED WORK

In the literature, there are previous usages of machine learning techniques for detecting malicious traffic especially for detecting Dos and DDoS with no previous usage on detecting malicious OCPP traffic. Ashis Pradhan et al [7] made a comparison between Support Vector Machine and Artificial Neural Network (multilayer perceptron) for detecting malicious traffic (TCP, UDP, ICMP). The results show similar accuracy (about 87%) in favor for SVM [7]. Vasilis Maglaris et al proposed a solution for detecting DDoS attacks using multi-layer perceptrons to classify the traffic

into three categories: DDoS source, DDoS victim and normal, protecting from both incoming and outgoing DDoS attacks [8]. The same neural network (MLP) was used to detect DDOS attacks in IoT, the system being able to classify with an accuracy of 99.4% various DDoS/Dos attacks [9].

Furthermore, Patrick Onotu et al [11] proposed an approach to shellcode recognition by using multi-layer perceptron with back-propagation. The proposed solution was capable of classifying few types of data with high precision and accuracy. Another solution in detecting intrusion detection is given by Yousef Abuadlla et al [12]. The main difference between their work and the previous is that they are more focused on aggregated flow statistics of network traffic. Their method consists of two stages: significant changes detection in traffic and the second classifies the type of attack.

There are several reasons why neural networks are suitable for malicious station detection. One of them is the speed. Since we need to scan every request from each station, we need a way to make sure that the time for scanning a request is no longer than the time for processing a request through a normal workflow [6]. The most important advantage of the neural network is that this system learns through previous experience by using training set and it can also learn through usage comparing the expected results with the ones returned by the neural network. For example, we can retrain the network by comparing only the negative results with the response from the server.

On the other hand, there are few reasons why using neural network could be difficult. The main issue is the "black box" nature of these systems. After the network has achieved an accepted level of precision, the weights and transfer functions are frozen. Furthermore, in order to have a precise neural network, big amount of data is needed for training sets. These data is critical because it is the basis of the network and in most of the cases it is very difficult to collect the data and to make sure this is statistically accurate [6].

3. PROPOSED SOLUTION

The proposed model is a back-propagation neural network used to classify the traffic (a pair of requests/responses) into two classes: malicious traffic and benign. The difference between the two classes is the value from the output that delimits the classes using a threshold.

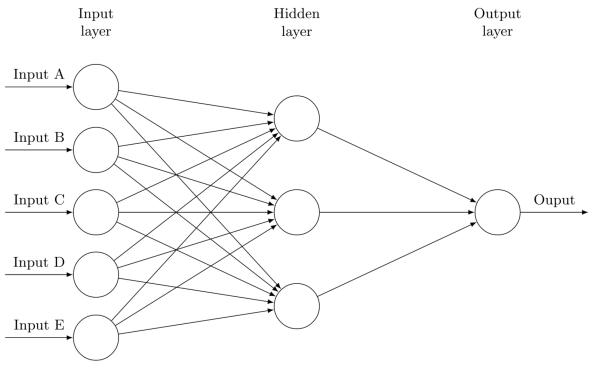

Figure 1: Proposed neural network.

The characteristics of the networks are:

- 5 neurons on input level [0 - 1];
- 3 neurons on hidden layer [0 - 1];
- 1 neuron on output layer [0 - 1];
- μ (learning rate) - 0.05;
- threshold - used to differentiate the two classes from the output;
- MAX_TIME - represents the maximum time between two consecutive requests from the same station. This time is maximum when there are only Heartbeat requests (default value for hearbeat interval on most of the stations) - 15 minutes.

in Figure 1 the 5 neurons on the input level have the following meaning:

- A - time difference between the current request and the previous request from the same station normalized by MAX_TIME;
- B - time difference between the current request and the previous request from the same station having the same type normalized by MAX_TIME;
- C, D, E - the last three types of request sent by the station, each of them corresponding to a 0 to 1 value, equally distributed.

For the testing phase, three types of stations were used: faulted station (FAULTED) which transmits multiple similar requests, random generated station (RANDOM) that sends random requests but in normal amount and normal functioning station (NORMAL) that respects the OCPP flow and also sends normal amount of data in terms of traffic and time. The main focus of the proposed method is to have a composite method to detect both random and faulted traffic using a single neural network.

In Figure 2 the traffic of the three station is presented, one for each type. As we can see, both RANDOM and NORMAL have similar, almost linear data traffic in comparison with FAULTED that has a burst of traffic in the first part of the day. The FAULTED traffic is represented by repetitive sending of StatusNotification requests for each of its three connectors, it is very unstable, changing its connectors' statuses very often. The reason for a steady traffic after 08:00 for FAULTED station is a hard reset sent from the server by the system administrator who detected manually an abnormal traffic coming from that station.

3.1 Message format

The messages collected in rows have the following format:

- Timestamp (2017-05-12 13:47:40)
- Sending point (station or server)
- Receiving point
- Message content:
  - Message type (2 - request, 3 - response, 4 - response with error)
  - Call id ("4174")
  - Procedure name ("Authorize")
  - Request (json format) [10]

The only difference between request and response is the fact that the response does not contain the procedure name since the request already has that call id.

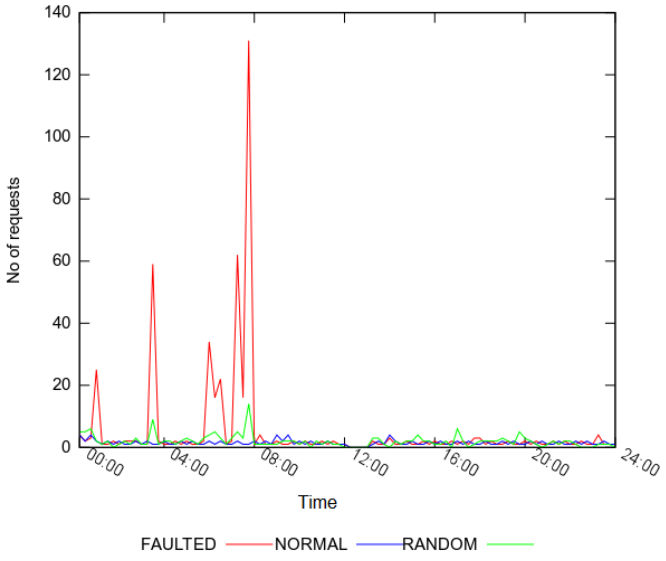

Figure 2: Traffic of the three types of station.

3.2 Learning sets

For the training set, there have been gathered logs for 20 days from 10 stations. In addition, a random bot that sends random, but OCPP compliant messages to the server was generated. These messages were infiltrated into the testing sets.

All the stations were located in the same country, so there was no problem with the clock synchronization.

3.3 Classification process

In the process of classification were used few automatic procedures. First of them is identifying the erroneous or invalid responses from the server for a specific request. Several requests initiated by the station have side effects, meaning that the responses of the same consecutive requests are different, the second one being an error one. For example, if a station initiates a request to start a transaction the server responds with details about this new transaction, but if it is sent the same request again the server will return an error since there is already one in action. We used this property to classify some requests as being invalid.

The second type of automated classification is the similarity of requests. In most of the cases, malfunctioning stations send repeatedly similar requests to the server and the reason for this problem is that the server responds with error messages that is not supported or not understood by the station.

4. EXPERIMENTAL METHODOLOGY

The learning sets are split into two parts, each of them focusing on detecting random traffic, on one side and faulted traffic, on the other side. We firstly run the testing sets for Faulted traffic recognition then the ones for Random traffic recognition. If the first set of tests was collected from real life usage of stations, the second ones were entirely generated, based on OCPP normal flow.

The files contain besides the regular station's traffic, malfunctioning station traffic, also. The latter is used as a negative learning set. In most of the cases, the malfunction makes the station to send similar and repeatedly messages. Another learning set that was used is from an emulated station that randomly sends messages to the server.

4.1 Faulted traffic learning sets

For the first part, it was used traffic generated by 10 stations, 9 of them are normal and 1 of them have time intervals in which it is faulted. The malfunctioning of stations was that they keep sending similar requests having the same type with slightly different pieces of information. There was used a 24-hours logs file containing requests and responses between stations and the central server.

The main problem for this was to classify the pairs of requests/responses as malfunctioning or normal. For this initial classification we used the following characteristics:

- Similarity between two consecutive pairs of request/response;
- The messages type from the server (if it was an error message or not).

The similarity between two consecutive pairs was the average of similarity between requests and responses, actually, we compare the current pair with the previous pair. The edit distance between requests and responses as strings was computed and then multiplied the result with 10 for a better representation on 0 to 1 scale since there is a lot of common information from the JSON representation.

The second characteristic is based solely on the response from the server. OCPP protocol indicates two types of responses from the server: CallResult (id = 3) and CallError (id = 4). The latter one means that there was an error processing the request from the station and it is much easier to detect. In case there is the same OCPP version implementation on both station and the central server, this type of response should be very rarely or not at all. CallResult can either mean that the transaction, for example, was started successfully or not, in the second case the id for a transaction is set to -1. Similar to StartTRansaction is Authorize which returns Occupied, Expired and Invalid for a certain idToken, in case it is not accepted. These type of responses are totally valid and can occur anytime during the communication, but in case there are many successive ones, something is very probably to be malfunctioning. We used these fields that indicate a not accepted/successful behavior to determine if the traffic is compromised. By the addition of the two characteristics, we decide if a pair of request/response is valid or not.

Table 1: No of requests per type and station type.

	Random	**Faulted**	**Normal**
Authorize	206	0	1
MeterValues	199	0	0
Heartbeat	133	45	130
StartTransaction	255	0	8
StopTransaction	239	0	8
StatusNotification	122	11592	18
BootNotification	177	30	0

In Figure 3 we plotted the data from Table 1 from above, thresholding the number of StatusNotification requests to

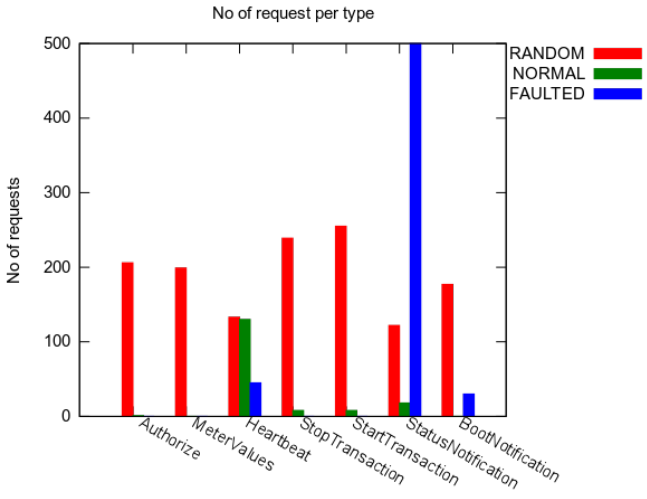

Figure 3: No of requests per type and station type.

500, for a better visualisation. As we can see in Table 1, the number of requests sent of each type by RANDOM is steady, about 150-200. NORMAL station sends the most Heartbeat requests and few StartTransction and StopTransaction for each of the 8 chargings that were done.

On the other hand, FAULTED station sends an enormous amount of StatusNotification requests because it had a malfunctioning with its connectors, making the station to send repeatedly this type of request. For all three types of stations, the closest values are for HeartBeat and this is almost because it is predefined, set by default on all three stations to 15 minutes.

4.2 Random traffic learning sets

The second part is focused on detecting malicious traffic based on sequences of types of requests. For the testing, there were used only 7 types of requests generated by the station, the rest of them are not sent on a daily basis.

$$RequestType = \{Authorize, BootNotification, Heartbeat, MeterValues, StartTransaction, StopTransaction, StatusNotification\} \quad (1)$$

There are multiple OCPP workflows that could be resulted based on OCPP messages, but several of them could not be realized by a normal OCPP compliant traffic. The second part of learning sets is focused on detecting station-server flows that are not OCPP compliant. In Table 2 are shown the sequential pairs of requests/responses that could not be produced.

For example, StartTransaction and StopTransaction are preceded or succeeded by Authorize or StatusNotification. On the other hand, Heartbeat request can occur anytime and should not be treated as a malicious sequence. Similar BootNotification is always succeeded by StatusNotification through which the station announces the state of its connectors. Most of the OCPP requests do not have side effects, but StartTransaction and StopTransaction. The station cannot send two consecutive StartTransactions and receive succesful response for both of them, because for the second one it will receive that the connector or card is occupied, similar for StopTransaction.

Table 2: Non OCPP compliant consecutive request types.

First	Second
Authorize	MeterValues
MeterValues	StartTransaction
StopTransaction	MeterValues
BootNotification	MeterValues
StartTransaction	StartTransaction
StopTransaction	StopTransaction
MeterValues	MeterValues
StartTransaction	StopTransaction
StopTransaction	StartTransaction
BootNotification	RequestType - StatusNotification

In this second learning set, the difference in time between the current pair and the previous one is not relevant so in order to have a full range of time we iterate through 0 to 1 for neurons A and B. We also iterate through RequestType and add the pairs of request from the Table 2 to have a larger and more meaningful sequence of requests.

5. EXPERIMENTAL RESULTS

In order to conduct the experiments, there were used 10 electrical charging stations from two different providers, all of them having implemented JSON over WebSocket version of OCPP 1.5. The logs were collected for a six months period but selected only the most relevant 30 days to the learning process. For conducting the testing, we chose three stations, one for each type: FAULTED, RANDOM and NORMAL and were run in three ways: one using only the first learning sets (FAULT), one using only the second learning sets (RAND) and one using both of them (FAULT + RAND). In the first part of the testing, we were focused on detecting the learning curve and best accuracy for all three types of station using the three approaches. In case of RANDOM and FAULTED stations, we chose threshold value of 0.5 and 0.7 for NORMAL. The reason for choosing a different threshold for NORMAL traffic is that we wanted to detect increasing precision during multiple epochs, because in all of the cases the traffic was correctly detected. We calculated the accuracy with the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

- TP - true positive (correctly identified)
- FP - false positive (incorrectly identified)
- TN - true negative (correctly rejected)
- FN - false negative (incorrectly rejected)

In addition to the testing from above, we also conducted some testing on the importance of the size of the hidden layer and learning rate with no major breakthroughs. For the first, we iterated the number of the hidden layer from 2 to 10 with the best results between 3 and 5 in terms of accuracy and for the second, we iterated the learning between 0 and 0.3 with the best results between 0.2.

5.1 Faulted station detection accuracy

As we can see in Figure 4, the first type of learning set realized an accuracy of 98.54%, significantly higher than the second approach - 30.76%. The main reason for this is that the first learning set is trying to detect especially the faulted traffic on one hand, and the fact that, even faulted a station implementing OCPP respects its flow. The addition of the two types of learning sets resulted in an important improvement of detection - 91.29%. A very important fact for the learning curve is that the first few days do not contain malicious traffic, the malfunctioning is starting to happen after two days.

Table 3: Best accuracy for FAULTED station detection

FAULT	RAND	FAULT + RAND
98.54%	30.76%	91.29%

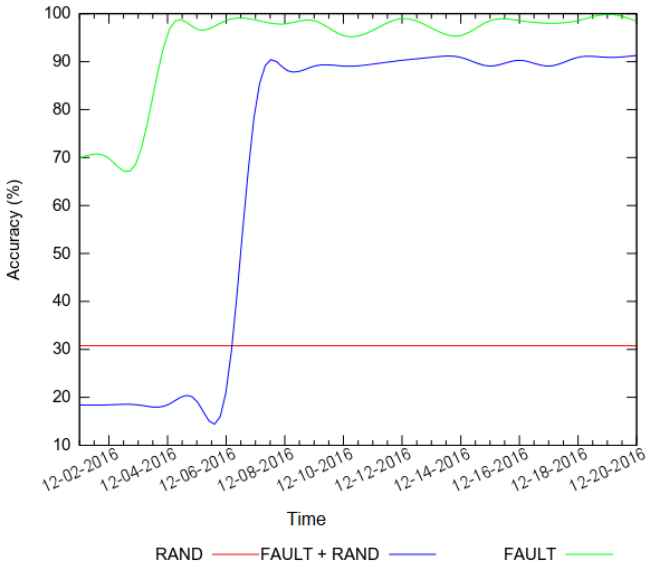

Figure 4: The learning curves for FAULTED station.

5.2 Random station detection accuracy

The second set of tests were conducted on RANDOM station. As we expected there is a tremendous difference in terms of accuracy between first (65.77%) and the second type (87.16%) of testing and this is mostly because the random station sends requests at a steady pace, just like a normal one, as a result the first type of learning sets is not very helpful. On the other hand, the second approach is quite accurate because it detects the non-compliant OCPP sequences of requests. In this case, the combination of the two approaches gives the best results in terms of accuracy - 90.90%.

Table 4: Best accuracy for RANDOM station detection.

FAULT	RAND	FAULT + RAND
65.77%	87.16%	90.90%

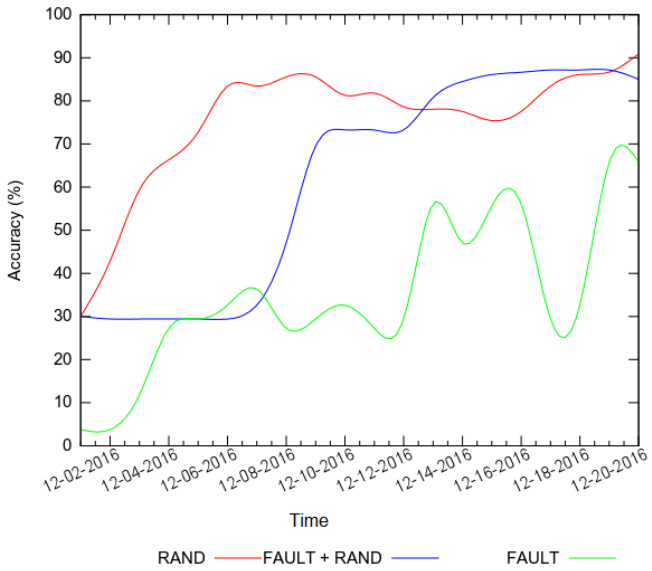

Figure 5: The learning curves for RANDOM station.

5.3 Normal station detection accuracy

The last testing was conducted on the NORMAL station. The results are similar using all three approaches and this is because the NORMAL station has a steady traffic and also respects the OCPP workflow. Anyway, even for this station, the results were best for the combination of the two approaches.

Table 5: Best accuracy for NORMAL station detection.

FAULT	RAND	FAULT + RAND
98.58%	97.16%	99.29%

5.4 Results

To sum up, for NORMAL and RANDOM stations the best results were realized using the combination of the two approaches. In case of FAULTED stations there is a slight difference in favor of FAULT approach than using FAULT + RAND. The reason for a lower accuracy in detecting FAULTED station may be the fact that this type of station is not faulted from the beginning till the end but only in the first part of the day, in the rest of the time is having a normal behavior. For this type of station, we only assumed that Heartbeat traffic is benign, the rest of it being malicious. In the case of generated learning sets for random traffic detection, the best results were obtained when trained with ratio 0.33 of negative/positive examples. On the other hand, the major improvements in detecting faulted traffic were obtained when were used especially the alternations between logs from days with massive faulted traffic and days with low faulted traffic.

Table 6: Best accuracy for the composite solution.

NORMAL	RANDOM	FAULTED
99.29%	91.29%	90.90%

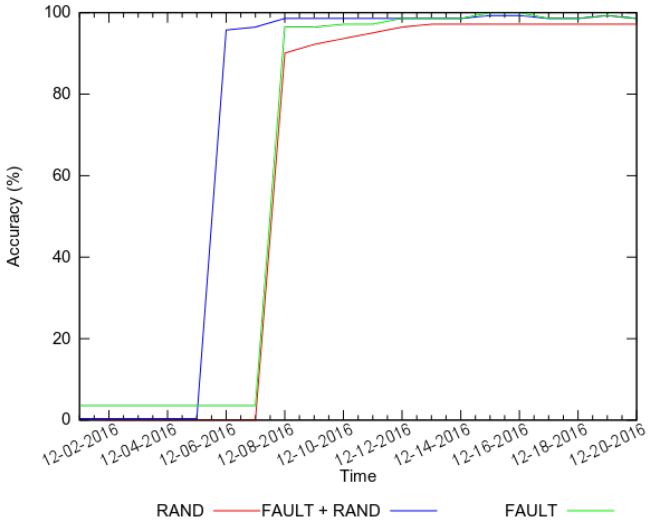

Figure 6: The learning curves for NORMAL station.

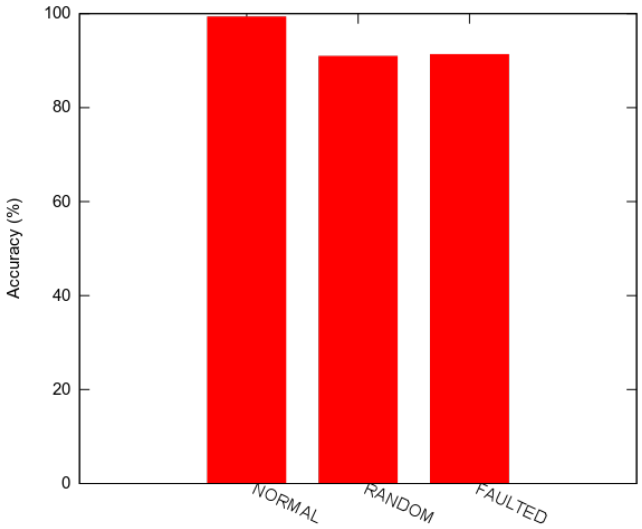

Figure 7: Accuracy for all three types of station.

6. CONCLUSION

The smart charging technology is a trending topic in the research field, raising important problems in the near future. With the increasing share of electric vehicles and renewable energy sources, the existence of a flexible and secure grid in this area becomes necessary. In this paper, we presented some of the related work in using a neural network in detecting malicious behavior, like Dos and DDoS and also proposed a backpropagation neural network in detecting faulted and random OCPP traffic. The entire system is based solely on the meaning of information that is passed, and not on length, throughput, latency, etc.

7. FUTURE WORK

In the previous work, we proposed a flexible architecture for integrating JSON over WebSocket with SOAP over HTTP versions. For the future work, we are working on a system that is based on the current paper and detects in real-time malicious traffic of the stations preventing the system to collapse in case of a major attack or stations' malfunctioning. Besides the integration of the current system, we are working on expanding the current neural network in a way that could also be used as a load-balancing software that controls and populate distributed queues.

Acknowledgment

The research presented in this paper is supported by projects: *DataWay*: Real-time Data Processing Platform for Smart Cities: Making sense of Big Data - PN-II-RU-TE-2014-4-2731; *MobiWay*: Mobility Beyond Individualism: an Integrated Platform for Intelligent Transportation Systems of Tomorrow - PN-II-PT-PCCA-2013-4-0321.

We would like to thank the reviewers for their time and expertise, constructive comments and valuable insight.

8. REFERENCES

[1] Adrian-Gabriel Morosan, Florin Pop, and Aurel-Florin Arion. "Electric mobility in green and smart cities." In International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, pp. 173-184. Springer International Publishing, 2016.

[2] Marcel Antal, Claudia Pop, Tudor Cioara, Ionut Anghel, Ioan Salomie, and Florin Pop. "A system of systems approach for data centers optimization and integration into smart energy grids." Future Generation Computer Systems (2017).

[3] Cristian Chilipirea, Andreea-Cristina Petre, Loredana-Marsilia Groza, Ciprian Dobre, and Florin Pop. "An integrated architecture for future studies in data processing for smart cities." Microprocessors and Microsystems (2017).

[4] Fabian van den Broek, Erik Poll, Barbara Vieira. "Securing the information infrastructure for EV charging". December 2015

[5] Robin Sommer, Vern Paxson. "Outside the Closed World: On Using Machine Learning For Network Intrusion Detection". May 16 - 19, 2010

[6] James Cannady. "Artificial Neural Networks for Misuse Detection". 1998

[7] Ashis Pradhan. "Network Traffic Classification using Support Vector Machine and Artificial Neural Network". 20 June 2015

[8] Christos Siaterlis, Vasilis Maglaris. "Detecting incoming and outgoing DDoS attacks at the edge using a single set of network characteristics". 15 August 2005

[9] Elike Hodo, Xavier Bellekens, Andrew Hamilton. "Threat analysis of IoT networks Using Artificial Neural Network Intrusion Detection System". 17 November 2016

[10] P. Rademakers, F. Bodewes. "Implementing JSON over websockets". July 2016

[11] Patrick Onotu, David Day, Marcos A Rodrigues. "Accurate Shellcode Recognition from Network Traffic Data using Artificial Neural Nets". 25 June 2015

[12] Yousef Abuadlla, Goran Kvascev, Slavko Gajin, Zoran Jovanovic. "Flow-Based Anomaly Intrusion Detection System Using Two Neural Network Stages". June 2016