Injection Attacks and Detection Strategy in Front-End Vehicle-to-Grid Communication

Sushil Poudel[§], J. Eileen Baugh[§], Abdulrahman Takiddin[†], Muhammad Ismail[§], and Shady S. Refaat[‡]
[§]Department of Computer Science, Tennessee Technological University, Cookeville, TN, USA
[†]Department of Electrical and Computer Engineering, Texas A&M University, TX, USA
[‡]Department of Engineering and Technology, University of Hertfordshire, Hertfordshire, UK
Emails: {spoudel43, jebaugh42, mismail}@tntech.edu, abdulrahman.takiddin@tamu.edu, s.khalil3@herts.ac.uk

***Abstract*—Public electric vehicle (EV) charging stations provide accessible charging options and play a vital role in addressing range anxiety and facilitating long-distance travel. However, the wide adoption of public charging stations poses serious security risks. This paper demonstrates for the first time an injection attack on the front-end vehicle-to-grid (V2G) communication based on the ISO 15118 protocol. Specifically, we developed a testbed that integrates V2Gdecoder, Parasite6, Open vSwitch, and MiniV2G to emulate traffic injections between the supply equipment communication controller (SECC) at a charging station and the EV's communication controller (EVCC). We showed that a malicious EV owner or infected supply equipment can inject harmful packets into the other side. This injection attack can modify the V2G messages to include runtime and denial-of-service instances, remote code executions, and other malware. To design a defense mechanism, we study the development of a machine learning-based system that can detect such injection attacks. We created a dataset of three cyber features that represent benign and malicious traffic between the SECC and EVCC. Then, we developed shallow and deep-learning supervised models that can detect injection attacks on front-end V2G traffic with detection rates up to 95% and false alarm rates down to 7%. Our experimental results highlight the potential of machine learning-based intrusion detection systems to effectively detect injection attacks on front-end V2G communications.**

***Index Terms*—Electric vehicle, charging station, V2G communication, injection attacks, cyber-security, intrusion detection.**

I. Introduction

The adoption of electric vehicles (EVs) and deployment of EV charging stations are rapidly increasing. In 2021, more than 1.4 million EVs were registered in the United States, an increase of more than 70% over the previous year [1]. As a result, charging stations are widely deployed in public places to fulfill EV charging demands. In the United States, there are currently more than 50 thousand public charging stations equipped with more than 138 thousand EV supply equipment (EVSE). Reports have demonstrated that roughly 65% of EVs rely on public EVSE as home charging may not always be feasible [2]. However, this wide adoption of public charging poses serious security risks. Specifically, public charging can lead to cyber-attacks on victim EVs and/or EVSE. One analogy is the juice jacking attack [3] that targets smartphone users charging their phones through public outlets whose USB ports have been compromised. Similar attacks can be launched on EVs and EVSE to inject malware and malicious traffic, hence, threatening the charging infrastructure, power grid, and transportation system. However, the feasibility of injection attacks on V2G communications and relevant defense mechanisms are not thoroughly studied in the literature.

This work was supported by NSF Award #2220346.

A. Background and Related Work

To initiate and complete a charging transaction, the EVSE at the charging station communicates with the EV to share charging information and preferences. This vehicle-to-grid (V2G) front-end communications follow the ISO 15118 communication standard in several EV models that are adopted in the United States and European Union [4]. The ISO 15118 standard allows for a bidirectional communication process that involves discovery, identification, authorization, and charging session configuration. Also, it includes features such as authentication, charging parameter negotiation, and energy transfer control. The messages exchanged between the EVSE and EV follow the Efficient XML Interchange (EXI) format that conforms to the ISO 15118 standard.

Recent studies show that the ISO 15118 standard could expose users to the risk of power theft by spoofing EVs [5]. Additionally, there are concerns about denial-of-service (DoS), session hijacking, and masquerading attacks, along with privacy risks [6]. Furthermore, the authentication and authorization of payment as defined by the ISO 15118 were argued to be inadequate for secure billing operations [7]. Moreover, other attacks that have been studied include data sniffing [8], interruption of charging sessions [9], and shifting the financial burden of EV charging onto the victim's EV [10].

Although there are no instances of malware injection in V2G communication that have been reported, recent studies suggest that this may change in the future. The Idaho National Laboratory (INL) warned about the potential risk of EVs spreading malware to public charging stations [11]. Yet, the existing works, e.g., [5]–[10], have not thoroughly studied the feasibility of injecting malicious traffic over V2G front-end communications. Also, the existing studies do not present effective defense mechanisms against such injection attacks.

B. Objectives and Contributions

The objectives of this paper are twofold. First, to demonstrate the feasibility of injecting malicious traffic in the V2G front-end communications between the EVSE and EV. Second, to develop an effective intrusion detection system that can

detect injection attacks over V2G front-end communications. Toward these objectives, we have carried out the following:

- We developed a testbed based on MiniV2G [12] to emulate benign and malicious front-end V2G communications between the EVSE and EV following the ISO 15118 protocol. In the malicious scenario, we designed an attacker communication controller in the testbed based on the Open vSwitch, Parasite6, and V2Gdecoder such that the malicious actor (either at the EV or the EVSE) can inject malicious traffic into the victim. We investigate malicious injections that include run-time and DoS instances, remote code executions, and other malware instances. Our experimental results demonstrate that such injections can be successfully accomplished.
- We developed machine learning-based intrusion detection system that can efficiently detect injection attacks on front-end V2G communications. Specifically, we created a dataset of benign and malicious V2G traffic. The labeled dataset consists of three cyber features. We trained and tested a set of supervised machine learning models, namely, support vector machine (SVM), deep feedforward, recurrent, and convolutional neural networks. Our experimental results demonstrate that efficient detection of injection attacks can be attained with up to 95% detection rate and down to 7% false alarms.

The rest of the paper is organized as follows. Section II presents the system and attack model, V2G testbed, injection attack strategy, and experimental demonstration of injection attacks. Section III discusses benign and malicious dataset collection, training of machine learning-based intrusion detection models, and performance evaluation results. Finally, conclusions are given in Section IV.

II. Injection Attack Strategy

This section presents the system and attack model, V2G testbed, and our proposed strategy for injection attacks.

A. System and Attack Model

The system under consideration consists of EVs and a public electric vehicle charging station. The station consists of a set of EVSE, each transferring energy to a single EV at a time. The set of EVSE are connected through a local area network (LAN) to the station's charging management system, as shown in Fig. 1. Each EVSE has a physical outlet to charge an EV. To initiate and complete the charging transaction, the EV communication controller (EVCC) of the charging EV and the supply equipment communication controller (SECC) of the EVSE communicate using EXI message format following the ISO 151118 standard, as shown in Fig. 1. We consider herein wired communications between the SECC and EVCC over the charging outlet. This paper focuses on the front-end V2G communications between the SECC and EVCC based on the ISO 151118 standard and does not consider the back-end communications (the LAN connection between the SECC and the charging management system).

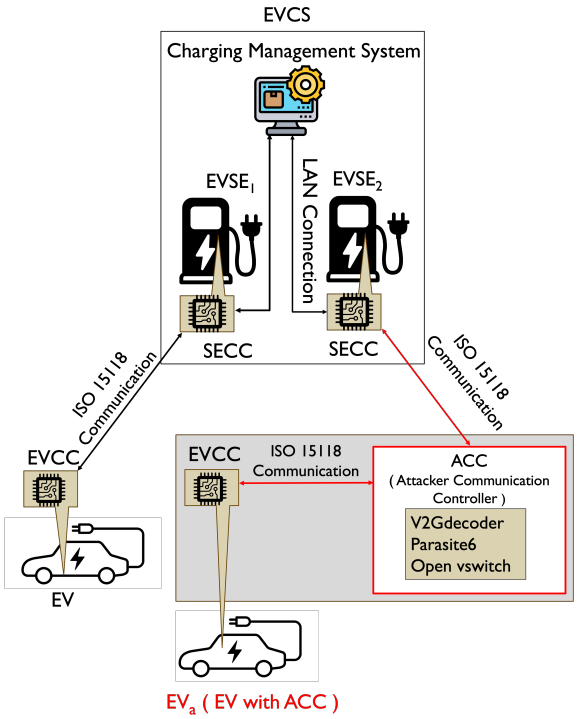

Fig. 1. System model showing the front-end V2G communication between the SECC and EVCC.

During normal operation, the SECC and EVCC exchange the sequence of request/response messages shown in Fig. 2 following the ISO 151118 standard [13]. At the beginning of normal V2G communication, UDP is used to exchange IP addresses and port numbers between the EVCC and the SECC through the SECC Discovery Protocol (SDP). First, EVCC sends the SDP Request message and receives a reply from the SECC with its IP address. Then, SECC and EVCC agree on either the Transport Control Protocol (TCP) or Transport Layer Security (TLS) protocol for further communication. Then, TCP/TLS sessions are established to send V2G messages between the SECC and EVCC. The communication session is divided into three phases, namely, setup, charging, and stopping. The EV and EVSE negotiate a number of parameters during the setup phase, including the protocol version, user identity, service details, payment options, and charging parameters. Energy transfer from the EVSE to the EV takes place during the charging phase. Energy transfer continues until a termination condition is satisfied or a service interruption is requested. Before disconnecting, the EV and EVSE run safety checks, which are followed by the session stop phase.

In the considered attack model, the attacker establishes an attacker communication controller (ACC) module in its EV, converting it to EV_a. This ACC intercepts the normal traffic between the SECC and the EVCC inside EV_a, modifies the EXI messages to inject malicious traffic, and redirects the injected traffic toward the victim EVSE. This way the attacker does not need to modify the firmware of the EVCC or gain

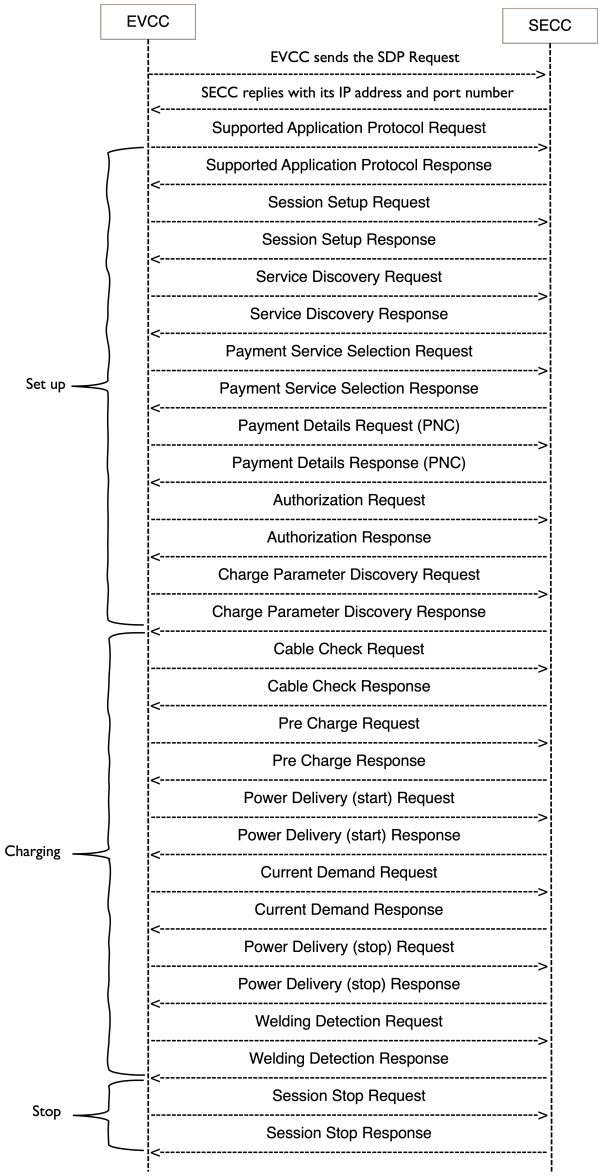

Fig. 2. Illustration of the front-end V2G communication requests and responses according to the ISO 15118 standard [13].

access to it. Instead, the attacker installs the ACC module inside its EV and converts it into EV_a. This is illustrated in Fig. 1. While the attack described herein considers an attacker that injects malicious traffic from EV_a to a victim EVSE, a similar strategy can be adopted by an attacker that owns an EVSE, establishes an ACC to convert the EVSE into $EVSE_a$ and injects malicious traffic to victim EVs. The next section discusses how to establish the ACC to enable injection attacks.

B. V2G Testbed with ACC Setup and Injection Strategy

To emulate normal V2G communications, the MiniV2G [12] emulator is used. It emulates EV charging based on MiniNet [14] and RiseV2G [15]. Specifically, MiniNet is an open-source communication network emulator and RiseV2G is a reference implementation of the ISO 15118 standard, which supports comprehensive AC and DC charging features with security measures. MiniV2G has classes for the EVSE and EV. The EVSE class provides the SECC functionalities, and the EV class provides the EVCC functionalities.

To launch traffic injection attacks, we upgraded the MiniV2G to introduce the ACC class, which includes the functionalities of Open vSwitch [16], Parasite6 [17], and V2Gdecoder [18], as shown in Fig. 1. In our implementation, the ACC wraps the EVCC functionalities to control V2G communication and inject malicious traffic. Specifically, Open vSwitch and Parasite6 enable traffic redirection, hence, setting ACC in the middle between SECC and EVCC, and V2Gdecoder facilitates the encoding and decoding of EXI payloads exchanged between EVCC and SECC.

Algorithm 1 summarizes the steps to launch the traffic injection attack and Fig. 3 illustrates the message timing diagram. EVCC sends an SDP request and waits for a reply from SECC. ACC sniffs the SECC reply to the SDP request and uses Open vSwitch and Parasite6 to establish itself in the middle between the SECC and EVCC by changing the MAC addresses, IP addresses, and port numbers. From now on, any message to be exchanged between the EVCC and the SESS will be redirected first to the ACC. Then, the ACC can use the V2Gdecoder to modify the contents of the exchanged EXI messages and inject malicious traffic into the SECC.

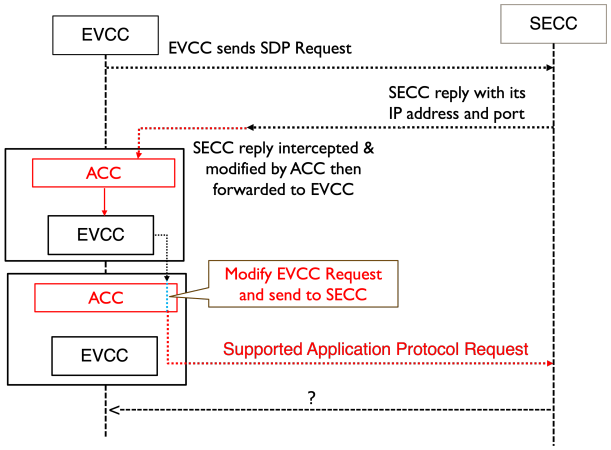

Fig. 3. Illustration of the attack during the V2G request message.

Possible Injections: In this paper, ACC modified the V2G messages to perform the following malicious injections:

- Run-time instances such as `${Runtime.getRuntime().exec("rm -rf /")}`.
- DoS instances like `${::-${::-${}}}`.
- Remote code execution such as `${jndi:ldap://attacker.com/a}`.
- Malware instances such as the Fork Bomb (rabbit virus, 60 bytes) and Tinba (Tiny Banker Trojan, 20 KB).

The aforementioned examples are meant to test the possibility of injecting malicious traffic between EVCC and EVSE.

Algorithm 1: Injection Attack Strategy for Front-End V2G Communications

EVCC: **begin**
  Send the SDP message through UDP to SECC;
end
ACC: **begin**
  Intercept the SDP request from EVCC;
  Generate a fake SECC reply with the ACC port number;
  Send the fake SECC reply to EVCC;
  Spoof the EVCC using Parasite6 and establish a connection with SECC;
end
EVCC: **begin**
  Receive the fake SECC reply from ACC;
  All EVCC traffic will be redirected to ACC by Open vSwitch;
end
ACC: **begin**
  Receive redirected EVCC traffic;
  Decode the EVCC message at ACC using V2Gdecoder;
  Modify the message to inject malware instances/malicious traffic as needed;
  Encode the modified message using the V2Gencoder;
  Forward the malicious message to SECC;
end
SECC: **begin**
  Receive and decode the malicious message from ACC;
end

Other malware instances can be injected if they maintain the maximum EXI message size by the ISO 15118 protocol (e.g., 64 KB for the ChargeParameterDiscoveryReq message).

C. Demonstration of Injection Attacks

Fig. 4 shows the normal V2G traffic between SECC and EVCC, captured by Wireshark, with the first row being the EXI of Supported Application Protocol Request and the second row being the reply. To demonstrate a successful injection attack, we first show the encoding of a malicious XML format to an EXI code at the ACC in Fig. 5. As shown in Fig. 5, the contents of the protocol namespace are modified to insert a JNDI lookup. If the logger at the victim SECC is vulnerable, then the JNDI lookup is called generating the canary token. Fig. 6 shows the injected traffic as captured by Wireshark. The first row in Fig. 6 shows the Supported Application Protocol Request being intercepted and modified to include the malicious EXI generated by ACC, which is sent to the SECC as depicted in the second row. Similarly, the SECC reply in the third row in Fig. 6 is intercepted and modified by ACC to continue the session. The decoding of the EXI message back to the XML format at the SECC is shown in Fig. 7. As aforementioned, this example is intended to demonstrate the possibility of injecting malicious traffic between the EVCC and SECC and other malware instances that conform to the EXI maximum allowed size in ISO 15118 can be injected.

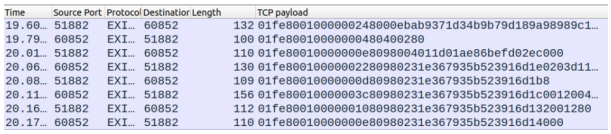

Fig. 4. Example of normal V2G traffic.

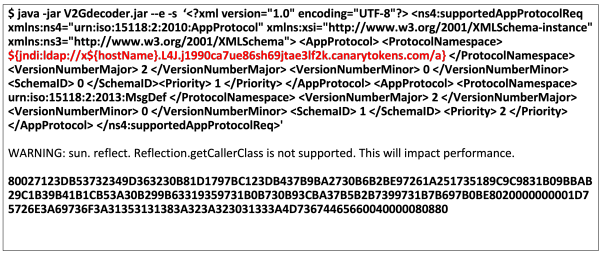

Fig. 5. Encoding of V2G message at the ACC.

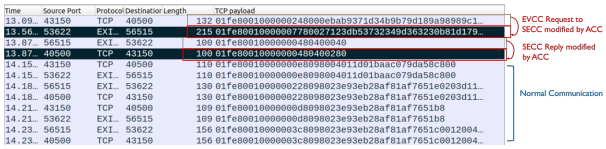

Fig. 6. Illustration of the attack during the V2G request message.

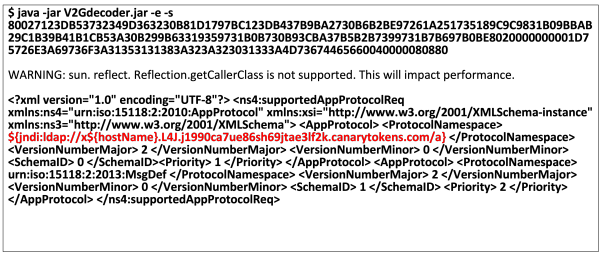

Fig. 7. Decoding of EXI to V2G message at the SECC.

III. Machine Learning-Based Detection Strategy

Section II demonstrated the feasibility of injecting malicious traffic into the V2G front-end communications. Adopting security techniques such as authentication and encryption (e.g., in TLS) is not helpful as the injected malicious traffic still is going to reach the victim. To deter such attacks, this section investigates the development of an effective strategy to detect malicious traffic injection attacks in V2G front-end communications. First, we discuss the collection of benign and malicious datasets. Then, we investigate four machine-learning models to develop the intrusion detection system, namely, SVM, deep feedforward neural network (FNN), recurrent neural network (RNN) based on long-short-term-memory (LSTM) cells, and convolutional neural network (CNN).

A. Dataset Generation

We used the testbed described in Section II.B to generate the required dataset. For normal traffic (benign data), MiniV2G was used to establish several charging transactions between EVs and the EVSE. For the malicious traffic, MiniV2G was used with the ACC class so that EV_a injects malicious traffic into the EVSE as described in Algorithm 1. The possible attacks described in Section II.B are considered while generating the malicious traffic. In each scenario, Wireshark was used to capture the relevant traffic. The captured packets were converted to comma-separated values (CSV) using pyshark [19]. The captured features include source and destination IP addresses, source and destination ports, round-trip time, payload length, and packet rate. The source and destination IP addresses were omitted as we are not considering any blacklisting of IP addresses. In addition, the port numbers were also omitted as they can be random, hence, adding no value to the detection strategy. As a result, our dataset consists of three cyber features, namely, round-trip time, payload length, and packet rate. The dataset samples were labeled such that $y = 0$ denotes benign traffic samples and $y = 1$ denotes malicious traffic samples (malicious traffic injections).

B. Machine Learning Models

We considered shallow (SVM) and deep (FNN, LSTM-RNN, and CNN) supervised machine learning models. The considered models cover both non-temporal (SVM and FNN) and temporal (LSTM-RNN and CNN) models that exploit correlation within the data to enhance the detection performance. The considered models are summarized next.

1) SVM: This model aims to find the optimal hyperplane that maximizes the margin between the two classes (benign and malicious). The data points near the hyperplane are the support vectors. This represents a shallow model.

2) FNN: This represents a deep model that consists of an input layer, a set of hidden layers, and an output classification layer, with interconnected artificial neurons. Forward propagation is used to process data, in which data travels from the input layer via the hidden layers to the output layer. Weights associated with neurons are iteratively adjusted during training to optimize the detection performance. FNNs are renowned for their capacity to approximate complex non-linear processes.

3) LSTM-RNN: This model processes sequential data, such as time series data, which is our case. This represents a deep model with a set of hidden layers and it has memory cells that can maintain information over extended sequences, making it appropriate for tasks requiring temporal modeling and context awareness. LSTM-RNNs are helpful in situations where the sequence of events and their dependencies are important.

4) CNN: This deep model exploits correlation in data by applying convolution layers. These layers apply filters to the input data, discover local patterns, and extract pertinent characteristics. In addition, pooling layers are used to minimize spatial dimensions and capture the most important characteristics. Due to their hierarchical and localized learning strategies, CNNs have excelled in classification tasks.

C. Hyper-parameter Optimization

We optimized the hyper-parameters of each model following a random grid search. For the SVM model, the optimal hyper-parameters were found to be the linear kernel, auto gamma, and regularization of $C = 1$. For the deep models, the optimal hyper-parameters Q were chosen from a search space that includes a number of layers $L = \{2, 3, 4\}$, a number of neurons/units $U = \{4, 8, 16, 32, 64\}$, a dropout rate $D = \{0, 0.2, 0.4, 0.5\}$, an optimizer $O = \{$SGD, Adam, Rmsprop$\}$, and an activation function $A = \{$ReLu, Sigmoid, Elu, Tanh$\}$. Table I summarizes the grid search results of the optimal hyper-parameters for the deep learning models.

TABLE I
SUMMARY OF OPTIMAL HYPER-PARAMETERS

Model	Q	Value
FNN	L	3
	U	64
	D	0.2
	O	Adam
	A	ReLu
RNN	L	3
	U	32
	D	0
	O	SGD
	A	ReLu
CNN	L	4
	U	32
	O	Rmsprop
	A	ReLu

D. Performance Evaluation Metrics

Three metrics have been considered to evaluate the performance of the intrusion detection models, namely, detection rate (DR), false alarm (FA) rate, and Accuracy. These are given as

$$\text{DR} = \frac{\text{TP}}{\text{TP} + \text{TN}}, \tag{1}$$

$$\text{FA} = \frac{\text{FP}}{\text{TN} + \text{FP}}, \tag{2}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{3}$$

where true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) represent the counts for different outcomes based on the predicted and actual labels. The positive label $y = 1$ denotes the malicious class and the negative label $y = 0$ represents the benign class. The DR is the ratio of correctly identified malicious samples. Furthermore, FA is the ratio of the number of benign samples that the model incorrectly identified as malicious. Finally, Accuracy denotes the percentage of correct predictions on the given data with respect to the actual label.

E. Results and Discussion

The dataset has $3,000$ samples, half of them represent benign traffic and the other half represent malicious injection traffic. The dataset was split into train, validation, and test sets with ratios $70\% : 10\% : 20\%$. The train and validation

sets were used to determine the model parameters and tune the hyper-parameters whose optimal values were found as in Table I. The detection results are summarized in Table II. A DR of 91.6% was obtained using SVM with a FA of 13.7% and an Accuracy of 92.3%. An improvement of 1.3% in DR, 3.6% in FA, and 0.6% in Accuracy was observed in FNN compared to SVM. The shallow SVM model and the deep supervised FNN model demonstrate a good detection performance, with the FNN model outperforming due to its deep architecture. Furthermore, the LSTM-RNN model improved the detection performance by more than 1% in DR and FA because it is effective in capturing and leveraging temporal dependencies, leading to better classification performance. Finally, CNN, being able to better capture correlation within the data, attained the highest DR of 95.1 %, lowest FA of 7.2%, and the highest Accuracy of 95.2%. In addition, Fig. 8 shows the relationship between FA and DR as described by the receiver operating characteristics (ROC) curves of all the developed models.

TABLE II
SUMMARY OF DETECTION RESULTS

Model	DR (%)	FA (%)	Accuracy (%)
SVM	91.6	13.7	92.3
FNN	93.3	10.1	93.7
LSTM	94.5	8.0	94.1
CNN	95.1	7.2	95.2

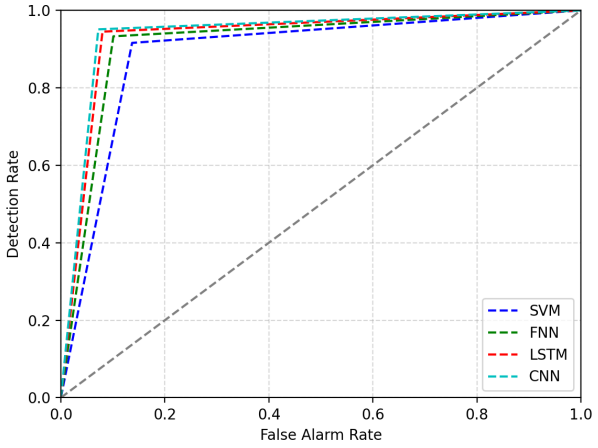

Fig. 8. ROC curve of the developed models.

IV. CONCLUSION

This paper investigated the potential risks associated with using public stations to charge EVs. A testbed was set up to emulate V2G front-end communications following the ISO 15118 standard. During the V2G communication, EVs attempted to inject malicious traffic to compromise the charging station. The same testbed can be used in reverse to attack EVs with compromised charging stations. Then, we examined a defense strategy that adopts machine learning models to detect such malicious injection attacks. The testbed was used to collect benign and malicious data, extract cyber features (round-trip time, payload length, and packet rate), and train the machine learning models. We compared the performance of shallow and deep learning models. Our results demonstrated that deep learning-based intrusion detection models can attain a detection rate up to 95% and a false alarm rate down to 7%.

REFERENCES

[1] U. DoE, "Electric vehicles registered in 2021," https://afdc.energy.gov/transatlas/#/, 2023, [Online: accessed 18-May-2023].

[2] S. Poudel, M. Abouyoussef, and M. Ismail, "EVCCS: Realistic simulation framework for electric vehicle commute and charge," in *2022 IEEE Vehicle Power and Propulsion Conference (VPPC)*, 2022, pp. 1–6.

[3] D. Singh, A. K. Biswal, D. Samanta, D. Singh, and H.-N. Lee , "Juice jacking: Security issues and improvements in USB technology," *Sustainability*, vol. 14, no. 2, 2022. [Online]. Available: https://www.mdpi.com/2071-1050/14/2/939

[4] "Shell recharge — Hubject and Greenlots to bring app free and card free charging to EV drivers," https://shellrecharge.com/en-us/solutions/news/hubject-and-greenlots-to-bring-app-free-and-card-free-charging-to-ev-drivers, (Accessed on 06/08/2023).

[5] S. Lee, Y. Park, H. Lim, and T. Shon, "Study on analysis of security vulnerabilities and countermeasures in ISO/IEC 15118 based electric vehicle charging technology," in *Proceedings of the 2014 International Conference on IT Convergence and Security (ICITCS)*, Beijing, China, October 2014.

[6] K. Bao, H. Valev, M. Wagner, and H. Schmeck, "A threat analysis of the vehicle-to-grid charging protocol ISO 15118," *Comput. Sci.-Res. Dev.*, vol. 33, pp. 3–12, 2018.

[7] C. Höfer, J. Petit, R. Schmidt, and F. Kargl, "POPCORN: privacy-preserving charging for emobility," in *Proceedings of the 2013 ACM Workshop on Security, Privacy & Dependability for Cyber Vehicles*, Berlin, Germany, November 2013, pp. 37–48.

[8] R. Baker and I. Martinovic, "Losing the car keys: Wireless phy-layer insecurity in ev charging," in *USENIX Security Symposium*, 2019.

[9] S. Köhler, R. Baker, M. Strohmeier, and I. Martinovic, "Brokenwire: Wireless disruption of ccs electric vehicle charging," 02 2022.

[10] M. Conti, D. Donadel, R. Poovendran, and F. Turrin, "Evexchange: A relay attack on electric vehicle charging system," in *Computer Security – ESORICS 2022*, V. Atluri, R. Di Pietro, C. D. Jensen, and W. Meng, Eds. Cham: Springer International Publishing, 2022, pp. 488–508.

[11] J. Johnson and T. Berg, "Review of electric vehicle charger cybersecurity vulnerabilities, potential impacts, and defenses," *Energies*, vol. 15, no. 11, 2022. [Online]. Available: https://www.mdpi.com/1996-1073/15/11/3931

[12] L. Attanasio, M. Conti, D. Donadel, and F. Turrin, "MiniV2G: an electric vehicle charging emulator," in *Proceedings of the 7th ACM on Cyber-Physical System Security Workshop*, ser. CPSS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 65–73. [Online]. Available: https://doi.org/10.1145/3457339.3457980

[13] M. Multin, "ISO 15118 as the enabler of vehicle-to-grid applications," *2018 International Conference of Electrical and Electronic Technologies for Automotive*, 2018.

[14] B. Lantz, B. Heller, and N. McKeown, "Mininet: An instant virtual network on your laptop," in *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 2010, pp. 191–206.

[15] "V2G clarity 2020 - Reference implementation supporting the evolution of the vehicle-2-grid communication interface ISO 15118," https://v2g-clarity.com/risev2g/, 2020, accessed on May 22, 2023.

[16] B. Pfaff, J. Pettit, B. Heller, P. Jain, R. Hundt, A. Hiltgen, C. Erickson, A. Bavier, D. Carver, J. Masters, and et al., "The design and implementation of open vswitch," in *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 2015, pp. 117–130.

[17] "THC. 2020. "THC-IPV6-ATTACK-TOOLKIT"." https://github.com/vanhauserthc/thc-ipv6, 2020, accessed on May 22, 2023.

[18] "Github - fluxius/v2gdecoder: V2gdecoder: encode and decode v2g messages on the fly," https://github.com/FlUxIuS/V2Gdecoder, (Accessed on 06/08/2023).

[19] K. Zorzos, M. Oikonomou, and P. Tsanakas, "Pyshark: A python wrapper for Wireshark's TShark," *Journal of Open Source Software*, vol. 3, no. 23, p. 524, 2018. [Online]. Available: https://doi.org/10.21105/joss.00524