Attack Design for Maximum Malware Spread Through EV Commute and Charge in Power-Transportation Systems

Sushil Poudel, Mahmoud Abouyoussef, J. Eileen Baugh, and Muhammad Ismail, *Senior Member, IEEE*

***Abstract*—The growing number of electric vehicles (EVs) on the roads led to a wide deployment of public EV charging stations (EVCSs). Recent reports revealed that both EVs and EVCSs are targets of cyber-attacks. In this context, a malware attack on vehicle-to-grid (V2G) communications increases the risk of malware spread among EVs and public EVCSs. However, the existing literature lacks practical studies on malware spread in power-transportation systems. Hence, this paper demonstrates malicious traffic injection and proposes strategies to identify target EVCSs that can maximize physical malware spread within power-transportation systems. We first show the feasibility of injecting malicious traffic into the front-end V2G communication. Next, we establish a model that reflects the logical connectivity among the EVCSs, based on a realistic framework for large-scale EV commute and charge simulation (EVCCS). The logical connectivity is then translated into a malware spread probability, which we use to design an optimal attack strategy that identifies the locations of target EVCSs that maximize the malware spread. We compare malware spread due to random, cluster-based, and optimal attack strategies in both urban (Nashville) and rural (Cookeville) U.S. cities. Our results reveal that optimal attack strategies can accelerate malware spread by $10-33\%$.**

***Index Terms*—Power-transportation systems, electric vehicle, electric vehicle charging station, malware attack, cybersecurity.**

Nomenclature

Abbreviations

ACC	Attacker communication controller
CAN	Controller area network
DoS	Denial-of-service
EV	Electric Vehicle
EVCSs	EV charging stations
EVSE	Electric vehicle supply equipment
EVCSMS	EV charging station management systems
EVCC	EV communication controller
EVCCS	EV commute and charge simulation
IDS	Intrusion detection systems
OSM	Open Street Map
SoC	State-of-charge
SQL	Structured Query Language
SECC	Supply equipment communication controller
V2G	Vehicle-to-grid

Sushil Poudel, J. Eileen Baugh, and M. Ismail are with the Department of Computer Science, Tennessee Technological University, Cookeville, TN, USA, email: {spoudel43, jebaugh42, mismail}@tntech.edu. M. Abouyoussef is with the Computer Science and Engineering Department, University of Central Arkansas, Conway, AR, USA, email: mabouyoussef@uca.edu

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

Parameters and Variables

$\mathcal{A}$	Targeted EVCS
$\mathcal{C}$	Set of C EVCSs of a city
C_i	EVCS i
D	Number of days
d	Day count
$e_{i,j}$	Edges connecting node i and node j in the graph
$f(\mathcal{A})$	Fitness function
G_c	Logical connectivity graph
G_e	Empirical connectivity graph
$\mathcal{I}$	Set of infected EVCSs
I	Number of infected EVCSs
K	Zones in a city (census tract)
m	Motif
$\mathcal{O}$	Set of centroids of zones
o	Centroid of each zone
$\mathcal{P}_v$	Charging pattern of EV number v
$\mathcal{P}$	Collective charging patterns for all EVs
R_{ij}	Number of times an EV charges at C_i then C_j
$\mathcal{S}$	Set of EVCSs
T_i	Count of EV charges at C_i
t_{pv}	Binary charging variable
U	Number of clusters
$\mathcal{V}$	Set of v EVs registered in a city
Z	City
z	Zone in a city
Λ	Average distance among EVCSs in the city
ρ_{ij}	Probability an EV charging at C_i then C_j

I. Introduction

Recent reports estimate that 125 million electric vehicles (EVs) will be on the roads by 2030 [2]. The growing trend of EV adoption is driven by financial and environmental advantages. Specifically, an EV costs one-fourth of the expenses of a gas-fueled vehicle while contributing to lowering carbon dioxide emissions [3], [4]. Overall, there are two methods to charge EVs. The first is dynamic wireless charging, where EVs charge on the move via induction coils, which has the limitations of weak power transfer and high power loss during induction [3]. The second is static charging, where *parked plug-in* EVs are charged from the power grid via a charge point. EVs can be charged either at home or at public EV charging stations (EVCSs). Reports have indicated that 35% of EV owners charge their EVs at home while the majority of EV owners (65%) rely on public EVCSs [5]. The charging pattern at the public EVCSs is intertwined with the behavior of EV owners, including their arrival and departure times and the EV's battery state-of-charge (SoC) levels [6].

EVs have coupled the power and transportation systems as EV charging via EVCSs is part of the power system while EV commute is part of the transportation system. Yet, EV charging at *public* EVCSs poses serious cybersecurity risks. In 2022, hackers attacked public EVCSs in Russia [7], England [8], and a Tesla car in Germany [9]. To initiate and complete EV charging, vehicle-to-grid (V2G) communications take place, e.g., via ISO 15118 [10], for session setup, service discovery, electrical parameters, payment details, authorization, cable check, and power delivery [11]. As a result, a malware-infected EV can infect EVCSs or vice versa. Hence, the propagation of malware can spread across other EVs and EVCSs. A similar cyber-attack strategy is reported against public charging of smartphones in 2011, namely, Juice Jacking [12]. Accordingly, the interaction between EVs and public EVCSs is a common vector for malware, e.g., ransomware, spread [13].

The vulnerability of EVCSs and EVs to cyber-attacks could potentially result in transferring them into botnets [14] disseminating malware that causes power outages, traffic congestion, etc. However, the existing literature lacks practical studies on physical malware spread in power-transportation systems. The existing studies focus on assessing threat levels and cyber-attack propagation via communication networks, rather than the physical spread of malware through EVs commuting and charging at public EVCSs. To defend the power-transportation system against malware attacks, this paper is the first to investigate the *physical spread of malware due to EV mobility and charging at public EVCSs*. The goal is to *identify target EVCSs that could serve as hubs for malware propagation, maximizing malware spread among EVs and EVCSs*. Identifying such stations would help system operators to defend the EVCSs against possible attacks and limit malware spread by isolating the potentially infected EVCS from EV charging, containing malware propagation in the system.

A. Related Works

EVCSs have Ethernet, serial, and USB ports for maintenance of firmware and software updates, which can be potential intrusion points [15]. Additionally, many EVCSs operate on Linux systems with vulnerable security credentials and communicate through the RS232 protocol [16], [17]. Weak access controls in EVCS operating systems allow administrative changes without requiring root user access [18]. EVCSs have vulnerabilities at the electric vehicle supply equipment (EVSE), EV connectors, Internet connections, and maintenance terminals, enabling malware injection and manipulating charging setting [19]. Idaho National Laboratory (INL) revealed security weaknesses in EVSE, such as outdated Linux kernels, vulnerable root processes, weakly hashed passwords, missing secure boot, unsigned firmware, exposed ports, direct processor control, and insecure coding practices [20]. Vulnerabilities in EVSE was identified including an exposed Structured Query Language (SQL) server that could lead to data exfiltration [21]. During the 2022 Russian-Ukraine conflict, EV chargers in Moscow were hacked. It was later revealed that a Russian EV charger provider had outsourced components to a Ukrainian company, giving the outsourced firm remote access and control over the charging functionality enabling remote changes to EVSE settings [22]. In this context, several studies in the literature assessed the security vulnerabilities in EVCSs [17], [23], [24]. Particularly, the security assessment of EV charging station management systems (EVCSMS) revealed vulnerabilities through reverse engineering and penetration testing, prompting vendors like Schneider Electric to acknowledge the need for improved security measures [25]. As a result, different types of intrusion detection systems (IDSs) have been developed to detect cyber-attacks in EVCS [26]–[28].

Furthermore, existing research has investigated cybersecurity threats related to in-vehicle communications based on the controller area network (CAN bus) [29]. Following this, several IDSs for in-vehicle network have been proposed [28], [30]. It is also important to highlight that malicious USB devices can be used to infect EVs [31], which can take place during mechanic repair or car rental [32]. Researchers also demonstrated a remote malware injection attack, named "TBONE" [33] targeting Tesla S, 3, X, and Y models, exploiting Internet connection manager vulnerabilities to load new Wi-Fi firmware, which controls in-vehicle systems, including steering and acceleration modes. Additionally, the boot security of Nvidia's system on chip, used in Tesla's autopilot and Mercedes-Benz's infotainment system, was bypassed with a voltage fault injection attack, enabling the execution of malicious software components [34]. These studies demonstrate the vulnerability of EVs and EVCSs to cyber-attacks.

The literature explored various EV charging-related cyber-attacks, including false data injection attacks for overcharging and denial of charging [35], man-in-the-middle attacks [36], payment fraud and battery damage [18], eavesdropping, tampering and forgery threats [37], and denial-of-service (DoS) [38]. The impacts of these cyber-attacks on the power system are reported in [17] and [39]. According to INL, there is a concern that EVs could potentially inject viruses into EVCSs leading to the further spread of malware among other EVs and EVCSs [40]. Additionally, cyber-attacks during EV charging such as charge manipulation attacks [41], delayed charging [42], and coordinated charging-discharging attack [43] are also studied in literature.While limited works have studied response strategies against malware attacks on EVCSs [44], [45], to the best of our knowledge, the physical malware spread among EVs and EVCSs remains unexplored.

There is a lack of practical studies in the existing literature on the physical spread of malware within power-transportation systems. Research has predominantly focused on assessing the spread of cyber-attacks via communication networks, rather than investigating how malware can physically spread through EVs commuting and charging at public EVCSs. Also, the existing literature does not present a dataset of EV commute and charge that can be used to study physical malware spread in public EVCSs. Some relevant studies [44], [45] rely on randomly generated hop distances and symmetric EV mobility matrices that do not accurately reflect real-world EV mobility patterns. Additionally, these studies focus on assessing threat levels and cyber-attack propagation via communication networks, rather than the physical spread of malware through EVs at EVCSs. They also do not address the design of attack strategies or identify the most impactful EVCSs.

B. Challenges

In an EV charging system, malware spread can occur on two levels [44]: (a) physical level due to the charging of an EV at different public EVCSs. In this case, an infected EVCS infects the charging EV, then the infected EV infects other EVCSs, which further spreads to other EVs, and so on; (b) cyber level where the malware propagates across the communication network connecting the EVCSs for energy management. This paper focuses on the physical spread of malware due to the mobility of infected EVs among public EVCSs. In this case, the mobility and charging patterns of EVs among the public EVCSs represent a key factor to the understanding of the physical malware spread. Specifically, the mobility and charging patterns of EVs provide information about the probability of EVs charging at EVCS X given that they charged in the previous time slot at EVCS Y, which translates into a logical connectivity among EVCSs that governs the physical malware spread. However, relevant EV datasets [46] are useful only in describing the spatial distribution of the charging load, which cannot be used to extract the probability of physical malware spread. Moreover, existing studies that simulate EV mobility among EVCSs (e.g., based on Markov Chain Monte Carlo) are either tied to specific geographical locations, which limits their applicability [47] or are not backed up with real data, which limits their practicality [44], [48]. Hence, there is a need for a method to generate a practical dataset that reflects the mobility and charging patterns of EVs among EVCSs. This dataset then can be used to find the probability of malware spread. Based on this study, system operators can identify target EVCSs that maximize the physical malware spread, thus, isolating such EVCSs when a malware attack is detected.

C. Contributions

To fill in the research gap, we carried out the following:

- We first employed MiniV2G [31] to illustrate malware injection attacks on the front-end V2G communication link, based on the ISO 15118 protocol, between an EVSE and an EV during the EV's charging process.
- Next, we studied the malware spread among EVCSs after the injection attacks. To do this, we proposed a framework for EV commute and charge simulation (EVCCS) and created a *practical* dataset that reflects EV mobility and charging patterns among public EVCSs. The EVCCS framework relies on publicly available datasets and statistics to mimic *real-life daily commute and charging events* in any given U.S. city, while considering: (a) the city population distribution among residential areas, workplaces, and entertainment regions, (b) the locations of public EVCSs and the number of registered EVs within the city, and (c) the average commute time, human daily motifs, and real-time traffic conditions.
- Based on the dataset from the EVCCS, we developed a probabilistic graph model to reflect the logical connectivity among the EVCSs due to EV mobility and charging events. The graph model describes the probability that a given EV charges at one EVCS in a given charging event and then charges at another EVCS in another charging event. We investigated the logical connectivity among EVCSs in two case studies representing examples of rural (Cookeville) and urban (Nashville) cities. Our results demonstrate that unlike Nashville, Cookeville presents a fully connected graph. However, Nashville presents a more dense but partially connected graph.
- Using the logical connectivity graphs and EVCCS, we compared three malware attack strategies on public EVCSs in terms of the speed of malware spread. The first attack strategy is a benchmark that targets public EVCSs at random. The second attack strategy clusters the EVCSs and then targets the centroid of each cluster to speed up the malware spread. The last attack strategy searches for the subset of EVCSs that maximizes the malware spread using a heuristic technique (genetic optimization). Our results demonstrate that the cluster-based and optimal attack strategies can speed up the malware spread by 20% and 32%, respectively, in an urban city (Nashville), and by 10% and 20% in a rural city (Cookeville). Further, the number of initially targeted EVCSs plays a vital role in speeding up malware spread. For example, by strategically targeting only 4 EVCSs out of 101 in Nashville, the speed of malware spread is increased by 33%.
- We proposed an empirical method to create an approximate logical connectivity graph based solely on the locations of EVCSs and their average separation in the city. This empirical graph presents a very simple way from an attacker perspective to identify target EVCSs without having access to any dataset about EV mobility and charging patterns among EVCSs in a city. Our results demonstrate that using the empirical connectivity graph, the attacker can speed up the malware spread by 20%.

The rest of this paper is organized as follows. To motivate our study, Section II demonstrates a malware injection attack on front-end V2G communications based on the ISO 15118 protocol. Then, Section III details the proposed EVCCS framework and derives the logical connectivity graph to infer the malware spread probability. Section IV describes the optimal and cluster-based malware attack strategies that aim to maximize the malware spread in the system and discusses the benchmark random attack and the empirical logical connectivity graph. Section V presents and discusses the simulation results. Finally, Section VI concludes this paper.

II. Injection Attack in V2G Communication

V2G communication enables EVs to exchange power and information with the EVSE at the EVCS. To demonstrate the feasibility of injecting malware in V2G communications, we employ the MiniV2G [31] simulator. MiniV2G is based on Mininet [49], an open-source communication network simulator, and reference implementation of ISO 15118 standard (RiseV2G) [50]. Our simulation environment comprises an EV and a public EVCS. The EVCS consists of a set of EVSE units that are interconnected via a local area network (LAN), all managed by an EVCS charging management system, as illustrated in Fig. 1. Each EVSE unit is equipped with a physical outlet to charge an EV. When the EV is connected to the EVSE to charge, V2G communications take place between the supply equipment communication controller (SECC) and the EV communication controller (EVCC), managed by the EVSE and EV, respectively. In our attack setup, the attacker deploys an attacker communication controller (ACC) module within their own EV, transforming it into EVa as illustrated in Fig. 1. ACC incorporates the capabilities of Open vSwitch [51], Parasite6 [52], and V2Gdecoder [53], wrapping the EVCC functionalities to regulate V2G communications and inject malicious traffic. Positioned between the SECC and EVCC, the ACC employs Open vSwitch and Parasite6 to redirect traffic. Additionally, V2Gdecoder handles EXI payload encoding and decoding between EVCC and SECC [53]. Specifically, any V2G communication between EVCC and SECC will first be redirected to the ACC module where V2Gdecoder can be used to encode the malware and inject it into the V2G traffic. In this work, we studied the injection of run-time instances such as `${Runtime.getRuntime().exec("rm -rf /")}`, DoS instances like `${::-${::-${}}}`, remote code execution such as `${jndi:ldap://attacker.com/a}`, and malware instances such as the Fork Bomb (rabbit virus, 60 bytes) and Tinba (Tiny Banker Trojan, 20 KB). For instance, the Fork Bomb malware would cause a DoS on EVCSs, disrupting the charging of EVs at all infected EVCSs. With wide EV adoption, this attack could lead to overvoltage in the power system and cripple the transportation system due to the inability of charging the EVs. It should be highlighted that other forms of malware can be considered given that they preserve the V2G XML message format, timing, and size. This includes ransomware similar in execution to the Mirai botnet, charge manipulation attacks [41], delayed charging attack [42], and coordinated charging-discharging attack [43].

To demonstrate the feasibility of the malware injection attacks, we show the following results. First, normal V2G communication traffic is illustrated in Fig. 2 and Fig. 3 shows an example of a successful malware injection attack. In the attack case, the ACC intercepts the regular communication between the SECC and the EVCC. Then, the ACC modifies the EXI messages to insert the malicious content and redirects the malicious traffic toward the targeted EVSE.

With this demonstration of a successful malware injection attack, the next section presents an EV commute and charge simulation framework that helps us investigate the malware spread pattern across the power-transportation system.

III. Large-Scale EV Commute and Charge Framework

This section discusses the process of creating a realistic database of a U.S. city and simulating the mobility and charging patterns of EVs within the city. The resulting dataset from this process will be used to extract the malware spread probability across the power-transportation system.

The process starts by identifying regions in the city for residential units, workplaces, shopping malls, restaurants, and entertainment. Then, we classified the amenities according to their land use and building type. In addition, the locations of public EVCSs within the city are identified. The aforementioned amenities are then associated with specific city zones. The number of registered EVs in the city are then distributed among the city zones. Using daily human motifs based on real-life statistics, the commute of EVs is simulated among the city zones and amenities. The SoC of the EV batteries are updated according to the commute distance, time, and real-time traffic conditions. Charging of EVs is carried out at the nearest public EVCS on a per-need basis. The details of this EVCCS framework are outlined below.

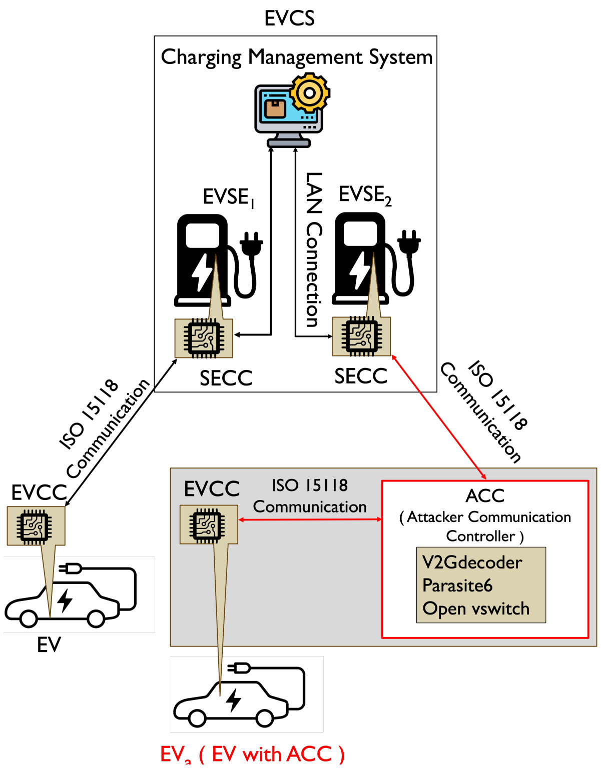
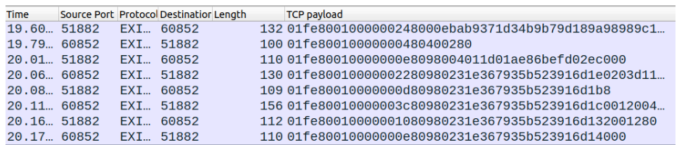
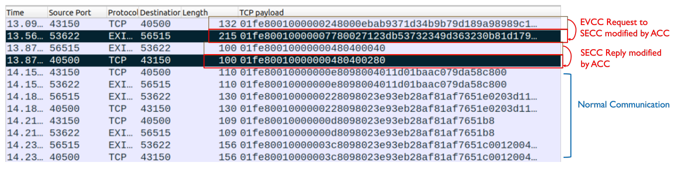


Fig. 1. V2G communication between SECC and EVCC.



```
Time      Source Port Protocol Destination Length  TCP payload
19.60…  51882     EXI…   60852          132 01fe80010000000248000ebab9371d34b9b79d189a98989c1…
19.79…  60852     EXI…   51882          100 01fe8001000000000480400280
20.01…  51882     EXI…   60852          110 01fe80010000000e8098004011d01ae86befd02ec000
20.06…  60852     EXI…   51882          130 01fe800100000002280980231e367935b523916d1e0203d11…
20.08…  51882     EXI…   60852          109 01fe80010000000d80980231e367935b523916d1b8
20.11…  60852     EXI…   51882          156 01fe80010000003c80980231e367935b523916d1c0012004…
20.16…  51882     EXI…   60852          112 01fe80010000001080980231e367935b523916d132001280
20.17…  60852     EXI…   51882          110 01fe80010000000e80980231e367935b523916d14000
```

Fig. 2. Example of normal V2G traffic.



```
Time      Source Port Protocol Destination Length  TCP payload
13.09…  43150     TCP    40500          132 01fe80010000000248000ebab9371d34b9b79d189a98989c1…
13.56…  53622     EXI…   56515          215 01fe8001000000077800271 23db53732349d363230b81d179…
13.87…  56515     EXI…   53622          100 01fe8001000000000480400040
13.87…  40500     TCP    43150          100 01fe8001000000000480400280
14.15…  43150     TCP    40500          110 01fe80010000000e8098004011d01baac079da58c800
14.15…  53622     EXI…   56515          110 01fe80010000000e8098004011d01baac079da58c800
14.18…  56515     EXI…   53622          130 01fe80010000000228098023e93eb28af81af7651e0203d11…
14.18…  40500     TCP    43150          130 01fe80010000000228098023e93eb28af81af7651e0203d11…
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…
```
EVCC Request to SECC modified by ACC
SECC Reply modified by ACC
Normal Communication

Fig. 3. An attack during a V2G request message.

A. Creating Geographical Database for the City

OpenStreetMap (OSM) [54] is used to identify residential units, workplaces, shopping malls, and entertainment within the city. It uses the city's map to create organized lists of these places. The OSM's Overpass application programming interface (API) is utilized to query the city's information,

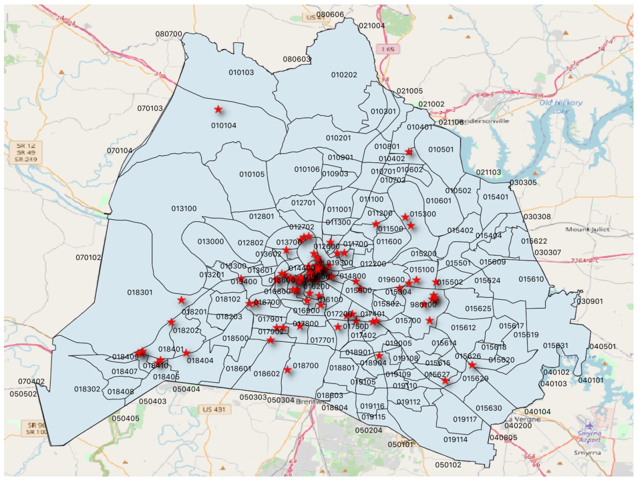


Fig. 4. Nashville's map featuring locations of public EVCSs.

passing the results output in a shapefile (.shp) format. OSM uses its own query language to create these queries. For instance, to locate all the restaurants within a designated area defined by latitude and longitude values in a boundary box (bbox) = (36.10745, −85.48559, 36.26983, −85.52532), the query is as follows:

(node [”restaurant”] bbox; way [”restaurant”] bbox;
relation [”restaurant”] bbox;); out body;

In this query, the terms “node,” “way,” and “relation” correspond to elements in OSM's conceptual data model, where “restaurant” is categorized as an amenity. Upon processing the query, the Overpass API generates a .shp file containing the geographical locations of all the restaurants within the designated area. Similarly, queries for different amenities such as residential units, shopping centers, hospitals, universities, schools, and offices are executed to obtain individual .shp file for each amenity. Additionally, the positions of public EVCSs within the city are collected in a .CSV file format from the U.S. Department of Energy's (DoE) Alternative Fuels Data Center [55]. Subsequently, this .CSV file is restructured into a .shp file format in the Quantum Geographic Information System (QGIS) [56], which is recognized as one of the most extensively employed applications for working with geographic data. QGIS combines the generated .shp files to craft a visual map of the city. Next, the city is divided into K zones, i.e., $Z = [z_1, z_2, z_3, \dots, z_K]$ based on census tract numbers in QGIS. Each zone is allocated a distinctive ID and census tract number. The centroid for each zone is defined as $O = [o_1, o_2, o_3, \dots, o_K]$. QGIS is then utilized to visualize and quantify the presence of various amenities within each distinct zone. Fig. 4 depicts the zones alongside the locations of public EVCSs within the city of Nashville, Tennessee.

B. Population Distribution and Number of Registered EVs

Data related to the city's overall population and the count of individuals working across different sectors is collected from the U.S. Census Bureau [57]. The city's population is then distributed between residential areas and workplaces, accounting for average commuting durations and the workforce population in each sector. Information about the typical time spent commuting to work within the city is sourced from BestPlaces [58]. For instance, Fig. 5 provides insight into the average work commute time in Cookeville.

Information about the number of registered EVs in a specific city can be obtained from Atlas EV Hub [59]. For instance, in 2022, there were a total of 116 registered EVs in Cookeville. Next, an object is created for each EV, encompassing distinct attributes such as house ID, driver ID, occupation, workplace, daily routines, battery SoC level, charging habits, and zone ID. These attributes of EV defines the person who drives it.

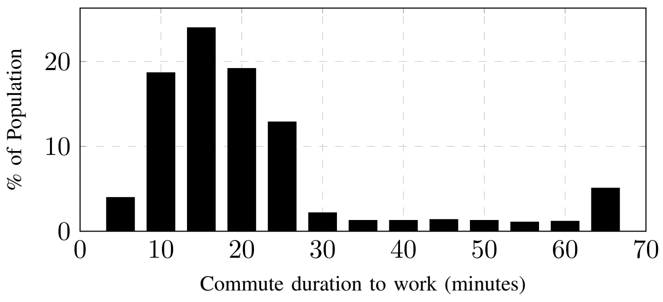


Fig. 5. Commute duration to work in Cookeville [58].

C. EV Commute and Charge

Modeling EVs as they move between different destinations is rooted in human daily patterns. Some of the common motifs derived from statistical data in [60], shown in Fig. 6, depict a sequence of activities. For example, the motif in Figure 6(c), HWLWSH, is defined as the activity of home (H), work (W), lunch (L), work (W), shopping (S), and home (H). The motifs in Figure 6(a)−(e) are for weekdays and the motif in Figure 6(f) is for weekends. When a motif is chosen for a specific day, the day's activities are simulated at 30-minute intervals, resulting in 48 motif for each EV per day. Moreover, accurate departure times for work are extracted from [61] and considered in the simulation framework to ensure a realistic representation. For instance, Table I provides a summary of departure times for commuting to work in Putnam County, Tennessee, a region that includes the city of Cookeville.

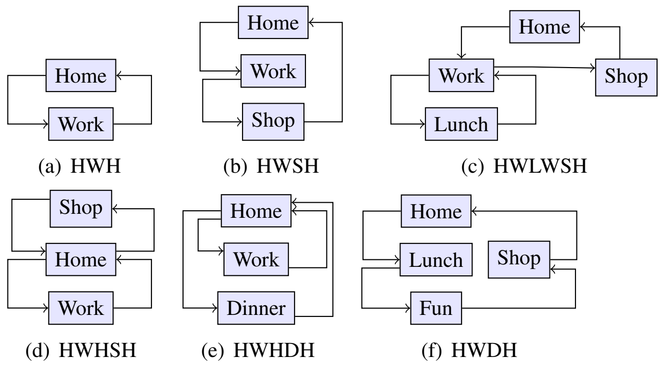


Fig. 6. Illustration of daily human motifs [60].

D. Generation of EV Objects and Simulation

Consider a set of V EVs defined by $\mathcal{V} = \{1, 2, \dots, V\}$, with V equals the number of registered EVs in the city, and a list of C EVCSs defined by $\mathcal{C} = \{1, 2, \dots, C\}$ with C equals the number of public EVCSs in the city. A total of V EV objects are instantiated with distinct attributes such

TABLE I. Commute time in Putnam County, TN [61].

Time of the day	Population departing	Time of the day	Population departing
12 AM - 5 AM	3.83%	8 AM - 8.30 AM	5.48%
5 AM - 5.30 AM	2.29%	8.30 AM - 9 AM	2.74%
5.30 AM - 6 AM	4.58%	9 AM - 10 AM	3.24%
6 AM - 6.30 AM	7.83%	10 AM - 11 AM	1.69%
6.30 AM - 7 AM	9.56%	11 AM - 12 PM	1.12%
7 AM - 7.30 AM	30.09%	12 PM - 4 PM	7.36%
7.30 AM - 8 AM	14.25%	4 PM - 12 AM	5.96%

as human ID, workplace, daily routines, EV's battery SoC, charging pattern, and zone ID. Each EV object's workplace is assigned based on factors that include the commute distance to work (e.g., based on Table I and Fig. 5 for Cookeville) and the number of employees in each sector. As an example in the City of Cookeville, Tennessee Tech University has approximately $10,000$ workers and students [62], while the population of Cookeville is around $35,000$ [63], and there were 116 registered EVs in Cookeville in 2022 [59]. Consequently, roughly $(10,000/35,000) \times 116 \approx 33$ EVs are assigned to EV owners affiliated with the Tennessee Tech University for work or education. The EVCCS framework creates commuting patterns for all city EVs based on daily motifs, chosen based on weekdays and weekends, and then structured into a 48-element sequence. Likewise, each EV's motif is also extended to 48 elements signifying activities in half-hour intervals. WAZE API [64] is used to determine in real-time the trip duration and commute distance between two locations while accounting for the traffic conditions. WAZE is a community-driven GPS-based navigation system that gives route details, trip time, traffic information, and map data. It enables us to capture real-time traffic conditions and rerouting.

When an EV's battery SoC hits a defined lower limit (e.g., 10%, around 20 minutes of driving), it heads to the closest EVCS. Similarly, if the SoC is insufficient for the next trip, the EV also heads to the nearest EVCS. On the contrary, if the battery SoC is adequate for the next destination, the EV travels there with the SoC decreased based on the commute's length and duration. Notably, the EVCCS models preferences of EVCS based on factors such as charger availability and number of EVSE within the EVCS.

Algorithm 1 provides a summary of the EVCCS framework. The following values were used in our evaluations: the total number of days 365, total number of motifs 48, the SoC threshold 10%, and the target charging level of the EV 100%.

E. Generation of Logical Connectivity Graph

Based on EVCCS, the charging pattern $\mathcal{P}_v$ of each individual EV is stored within its corresponding EV object v. The collective set of charging patterns for all EVs, denoted as $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_v\}$, is used to calculate the logical connectivity among public EVCSs. Let C_i and C_j denote two EVCSs. The interconnection signifies the likelihood that an EV charging at station C_i will choose station C_j for its next charging cycle. The following describes the development of this logical connectivity graph G_c. The total count of instances

Algorithm 1: EVCCS: Electric Vehicle Commute and Charge Simulation Framework

```
1  Choose a city and define its geographical boundaries;
2  Utilize a query with amenities over the city in the
     Overpass API of OpenStreetMap to obtain shapefiles;
3  Partition the city into $K$ zones, denoted as
     $[z_1, z_2, z_3, \ldots, z_K]$, and determine centroids
     $[o_1, o_2, o_3, \ldots, o_K]$ of these zones;
4  Identify the amenities within each zone and count their
     occurrences using QGIS;
5  Determine the actual driving distance $\Lambda_{o_i,o_j}$ and
     duration $t_{o_i,o_j}$ between centroids in
     $[o_1, o_2, o_3, \ldots, o_K]$ using the WAZE API;
6  Let $V$ represent the total number of registered EVs in
     the city, and set $v = 1$;
7  while $v \leq V$ do
8      Create an EV object ($v$) with a set of attributes;
9      Assign a job and a workplace to $v$, taking into
         consideration commute time and employee
         population in each sector as in [65] and [61];
10     Increment $v$ by 1;
11 end
12 Consider $C$ EVCSs in the city as in [55]; let $d$
     represent a day with $d = 1$ and $v = 1$;
13 while $d \leq 365$ do
14     while $v \leq V$ do
15         Assign each user 48 daily motifs based on their
             likelihood to occur;
16         Increment $v$ by 1;
17     end
18     Let $m$ denote a motif, starting with $m = 1$ and
         $v = 1$;
19     while $m \leq 48$ do
20         while $v \leq V$ do
21             if SoC of $v$ is $\leq 10\%$ or insufficient for the
                 next destination then
22                 Charge $v$ at the nearest station and
                     update its SoC to $100\%$;
23                 Update charging pattern of $v$ with the
                     corresponding station ID;
24             end
25             else
26                 $v$ travels to the intended destination,
                     decreasing SoC based on commute
                     distance and duration (obtained from
                     WAZE API);
27             end
28             Increment $v$ by 1;
29         end
30         Increment $m$ by 1 ;
31     end
32     Increment $d$ by 1;
33 end
```

where EVs charge at station C_i can be calculated as

$$T_i = \sum_{v=1}^{V} \sum_{p=1}^{|\mathcal{P}_v|} t_{pv}, \tag{1}$$

where,

$$t_{pv} = \begin{cases} 1, & \text{if } C_i \in \mathcal{P}_v, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The total number of times an EV charges at C_i in one cycle and subsequently at C_j in the next cycle is determined by

$$R_{ij} = \sum_{v=1}^{V} \sum_{p=1}^{|\mathcal{P}_v|} r_{pv}, \tag{3}$$

where

$$r_{pv} = \begin{cases} 1, & \text{if } \mathcal{P}_v(p) = C_j \text{ and } \mathcal{P}_v(p-1) = C_i, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Hence, the probability of an EV charging at C_i in one cycle while choosing C_j for the next cycle is calculated as

$$\rho_{ij} = R_{ij}/T_i. \tag{5}$$

The probabilities ρ_{ij} are calculated for all the EVCSs within the city to create a probabilistic charging graph G_c, which represents the logical connectivity graph. In this work, two case studies are considered for Cookeville and Nashville. The resulting connectivity graph for Cookeville is illustrated in Fig. 7. where nodes represent the ID of the EVCS, and edges denote the probability that an EV charged at EVCS X in one charging event will charge at EVCS Y in the next charging event. For instance, at EVCS 6000, 24% of the total EVs charged at EVCS 6000 will charge at EVCS 8000 in the next charging event, while 4% of EVs charged at EVCS 8000 will charge at EVCS 6000. Similarly, 32% of EVs charged at EVCS 8000 will charge at the same EVCS in the next charging event. There is a high transition probability from all public EVCSs to EVCSs 8000 and 7000, justified by Tennessee Tech University's proximity to public EVCS 8000 and a nearby grocery shop near EVCS 7000. Similarly, a connectivity graph for Nashville is generated; however, it is not shown due to its complexity. In Cookeville, there were 8 EVCSs and 116 EVs, whereas Nashville had approximately 3391 EVs and 250 EVCSs in 2022 [55].

The logical connectivity graph indicates the probability of malware spread from one station to another due to the mobility of EVs among the EVCSs. When a victim EV charges at an infected EVCS it gets infected. As the infected EV charges at another EVCS, as described by the probabilities indicated in the logical connectivity graph, the malware infects the EVCS, hence, spreading among all EVs and EVCSs.

IV. Malware Attack Strategies

This section identifies the set of EVCSs $\mathcal{C}$ that when targeted with the malware attack of Section II, the malware spread among the EVCSs requires the least number of days. An optimal attack strategy is proposed based on a black-box solver, namely, the genetic optimization algorithm. Furthermore, we consider three benchmark strategies, namely, random attacks, cluster-based attacks, and cluster-based attacks with an empirical connectivity graph.

A. Optimal Attack Strategy

The practical identification of the most effective EVCSs that would accelerate the malware spread is based on simulations (as given by EVCCS in Algorithm 1) and data-driven models (as given by the logical connectivity graph). This process cannot be described using explicit equations. Therefore, heuristic (black-box) optimization techniques can be used. In this work, we consider the genetic optimization as an example of such heuristic techniques. Algorithm 2 summarizes the attack strategy. The genetic algorithm defines its fitness function as the number of days required to infect 90% of the EVCSs in a city, which is determined by the EVCCS described in Algorithm 1. In Algorithm 2, the maximum number of generations is set to N. Initially, the C EVCSs are divided into S sets through random selection, ensuring that each element appears exactly once in one of the sets without repetition or omission. This step is pivotal as it leverages a hierarchical approach, potentially leading to a more balanced and diverse element selection, reducing the risk of local optima, and enhancing the overall quality of the final selection. Subsequently, populations of candidate solutions are initialized separately for all S sets, with chromosomes having a size of $|\mathcal{A}|$. The most suitable chromosome from each of the S sets is then identified by applying the genetic algorithm for N generations, using the fitness function $f(\mathcal{A})$ provided by the EVCCS. The final selected chromosomes from S sets are added to a temporary variable set, denoted as $\mathcal{T}$. Once again, populations of candidate solutions for set $\mathcal{T}$ are initialized, with chromosomes of size $|\mathcal{A}|$. The genetic algorithm is then applied to determine the fittest solution. Ultimately, the chromosome representing the $|\mathcal{A}|$ selected EVCSs that facilitate the fastest malware spread within the set of public EVCSs is obtained.

Algorithm 3 describes the calculation of the number of days to spread the malware, which represents the fitness function in Algorithm 2. In Algorithm 3, a while loop continues until the number of infected EVCSs (I) reaches or exceeds 90% of the total EVCSs ($\mathcal{C}$). Inside the while loop, $\mathcal{V}$ EVs are simulated as they move within the city using EVCCS. These EVs gradually deplete their charge levels as they travel. For each individual EV (v), the algorithm evaluates whether recharging is necessary, and if it occurs at a specific EVCS (C_i). Two key scenarios are considered: if an EV charges at C_i, the algorithm checks whether the EV is already infected ($v \in \mathcal{I}$) and whether C_i is not yet infected ($C_i \notin \mathcal{E}$). If these conditions are met, the EVCS is marked as infected by adding it to the set $\mathcal{I}$. Conversely, if the EV is not infected ($v \notin \mathcal{I}$) and C_i is already infected ($C_i \in \mathcal{E}$), the EV is marked as infected by adding it to set $\mathcal{I}$. With each iteration, a counter D increments by 1, tracking the days required for malware propagation.

B. Benchmark Strategies

This subsection presents three strategies that serve as benchmarks for comparison with the optimal attack strategy.

1) Random Attack Strategy: This represents the simplest attack strategy. Herein, the attacker randomly selects the EVCSs for malware injection attack. Let $\mathcal{A}$ represent the randomly attacked EVCSs from a pool of $\mathcal{C}$ EVCSs.

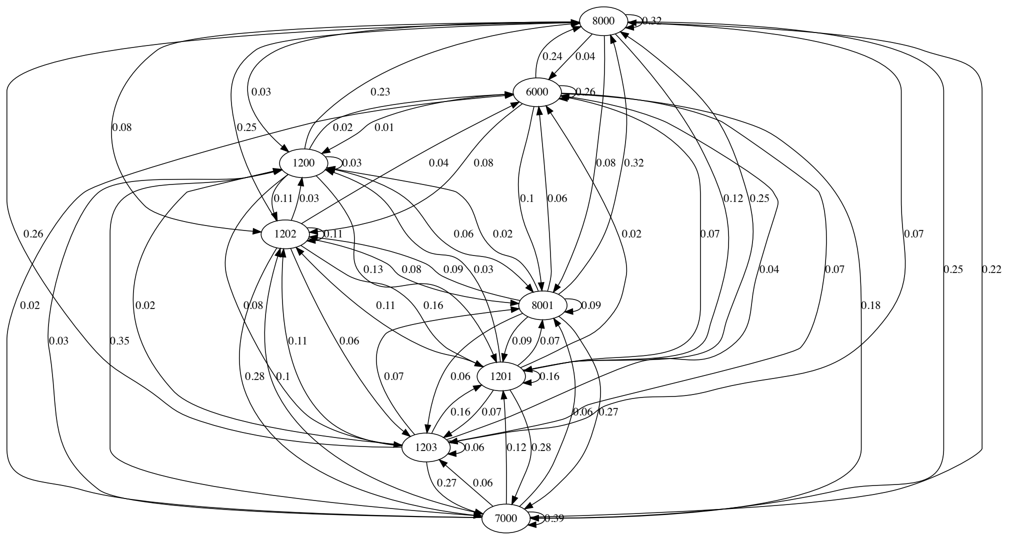


Fig. 7. Logical connectivity graph of Cookeville. The nodes represent the EVCSs in the city and the edges denote the probability of charging at EVCSs at consecutive charging events.

2) Cluster-based Attack Strategy: In a rural city like Cookeville, it is common to find residential, work, and shopping areas within the same zone. On the contrary, in an urban metropolis like Nashville, these activities are often distributed between different zones. In a large urban city, various distinct geographical zones exist each with its designated services, including workplaces, residential neighborhoods, and commercial districts. As a result, individuals residing in one zone are more likely to work in another zone and may choose to shop either within their own zone or in nearby areas. In this case, malware spread via randomly selected EVCSs can be time-consuming if the chosen EVCS do not have a significant impact. A more effective approach is to select EVCSs from different zones/clusters, which is done by applying clustering methods. One example is the Girvan-Newman algorithm [66], which divides the network graph into clusters based on the betweenness of its edges. By targeting nodes (EVCSs) with high-degree centrality from each cluster, attackers can accelerate malware spread among EVCSs.

Algorithm 4 summarizes the attack strategy based on four essential steps. First, the logical connectivity graph G_{c} is divided into clusters using the Girvan-Newman algorithm. Second, the degree centrality of each node within these clusters is calculated, quantifying its local connectivity. Third, the node with the highest degree centrality is selected as the central node within each cluster, indicating its prominence in mediating interactions within that cluster. Lastly, the algorithm outputs a list of nodes with the highest degree centrality. A subset of such central nodes, denoted by $\mathcal{A}$, are then targeted with the malware injection attack.

3) Cluster-based Attach Strategy with Empirical Connectivity Graph: To identify $\mathcal{A}$ in the clustering strategy of Section IV.B.2, the logical connectivity graph, G_{c} is required. This means that the attacker is running the EVCCS framework to construct the logical connectivity graph. Herein, we explore a simpler clustering-based strategy that does not rely on the logical connectivity graph. Instead, we construct an empirical connectivity graph, G_{e} to identify the relationship between the EVCSs. Unlike G_{c}, G_{e} is an undirected graph and has no weights on the edges. In the empirical graph, an edge is established between nodes when the spatial distance between two nodes is less than a specific threshold distance Λ. The threshold selection is based on the average distance between the EVCSs in the city. Whenever two EVCSs are separated by a distance less than this threshold, the two EVCSs are considered to be logically connected. When EVCSs are close to each other, drivers are more likely to choose between them based on convenience and availability. The closer the EVCSs, the higher the probability that a driver will charge at either one. The clustering-based algorithm described in Algorithm 4 is then applied on G_{e}.

Algorithm 5 describes the procedure to build the empirical connectivity graph. The algorithm takes inputs such as the collection of EVCSs within a city, the distances between them, and the predefined threshold distance Λ. Distances between EVCSs can be acquired using tools such as QGIS and WAZE, with WAZE providing travel distances between any pair of GPS points on the map. Subsequently, the threshold distance Λ can be computed by averaging the distances among the EVCSs. The process begins by initializing an edgeless graph, denoted as G_{e}, comprising $\mathcal{C}$ EVCSs. Then, we get the spatial distances, $\Lambda_{i,j}$, among all the EVCSs within $\mathcal{C}$ set using the WAZE API. Subsequently, we calculate the average distance among these EVCSs. This average distance serves as the designated threshold for establishing edge connections within the graph G_{e}. Hence, an edge $(e_{i,j})$ is created between pair of nodes in G_{e} if their spatial distance falls below the threshold value and if no connection between them has been established previously. Additionally, as a concluding step, if G_{e} represents a disconnected graph, we rectify its connectivity by introducing connections to the nearest nodes in terms of spatial distance. This step ensures that the graph is path-

Algorithm 2: Optimal attack strategy

```
Input: Logical connectivity graph $G_c$, Set of EVCSs
       in the city $\mathcal{C}$, EVCCS framework, maximum
       number of generation $N$
Output: Set of $\mathcal{A}$ targeted EVCSs
initialize: Fitness function $f(\mathcal{A})$ as the number of
       days to propagate the malware across 90%
       of EVCSs in the city, calculated from the
       EVCCS, $\mathcal{T} = [\cdot]$
function: FitEVCCS($N$): for $j \leftarrow 1$ to $N$ do
    Evaluate the fitness of each candidate solution by
    calling EVCCS to calculate the number of days it
    took to infect 90% of the EVCSs when targeting $\mathcal{A}$
    Select individuals for reproduction based on their
    fitness
    Perform crossover and mutation to create a new
    population
    Replace the old population with the new population
end
1 Create $S$ sets by randomly selecting vertices from the
  $G_c$, ensuring that each element from $G_c$ is included
  exactly once in one of the sets, with no repetitions or
  omissions.
2 Initialize a population of candidate solutions, each
  with a chromosome of size $|\mathcal{A}|$, for all the $S$ sets
3 for $i \leftarrow 1$ to $S$ do
4     FitEVCCS($N$)
5     Add selected vertices to set $T$
6 end
7 Initialize a population of candidate solutions, each
  with a chromosome of size $|\mathcal{A}|$, for the set $\mathcal{T}$
8 FitEVCCS($N$)
9 $\mathcal{A} \leftarrow$ a set of selected EVCSs to be attacked
```

connected, in other words, fully reachable.

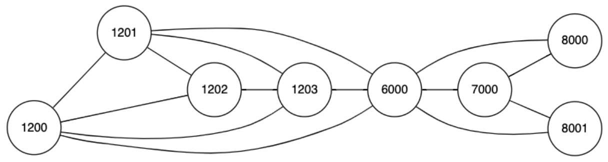


Fig. 8. Empirical charging graph of Cookeville

Fig. 8 shows an empirical connectivity graph for Cookeville, using a 2.5-mile threshold. In Fig. 8, EVCS 1201 connects with EVCS 1202 since they are within 2.5 miles, but it does not connect to EVCS 8000 as it exceeds the threshold.

V. Simulation Results

The EVCCS was developed in a Python 3.8.3 environment, leveraging NumPy, Matplotlib, NetworkX, and the genetic algorithm library. The implementation has been carried out on an MacOS with a 1.4 GHz processor and 8 GB RAM.

A. EVCS Radius

This metric measures the distance from each city's centroid to its respective EVCSs. In Cookeville, as shown in Fig. 9, all EVCSs are within a 3-miles radius from the city's centroid, indicating that they are likely to be utilized by the same group of EV owners. In contrast, as illustrated in Fig. 10, the majority of the EVCSs in Nashville are positioned approximately 25-miles away from the city's centroid, and they are widely distributed across the city with considerable distances between them, making it less likely for all EVCSs to be used by the same set of owners. The EVCSs in Nashville are often found to be far apart and distributed throughout the city. Furthermore, the EVCSs in Nashville tend to cluster within specific geographical areas. Consequently, it is highly probable that EV owners within the same territory utilize the same group of EVCSs, with minimal chances of an EV from one cluster using the EVCSs from another cluster. It is noteworthy that 101 EVCSs fall within a 30-mile radius from the centroid of the city, signifying a concentration of EVCSs in the downtown

Algorithm 3: Calculating the number of days for malware spread in $\geq$ 90% of EVCSs in the city

```
Input: EVCCS, set of EVCSs $\mathcal{C}$, set of targeted
       EVCSs $\mathcal{A}$, set of infected EVCSs including $\mathcal{A}$,
       set of all EVs $\mathcal{V}$, set of infected EVs $\mathcal{I}$, logical
       connectivity graph $G_c$
Output: $D =$ number of days to spread the malware
       across 90% of EVCSs in the city
Initialize: $D \leftarrow 0$, $\mathcal{I} \leftarrow \{\cdot\}$, $\forall v = \{1, 2, \ldots, V\}$
       randomly select initial EVCS from set $\mathcal{C}$ to
       charge
1  while $|\mathcal{E}| \leq 0.9 \times C$ do
2      $V$ EVs travel within the EVCCS, gradually
       depleting their charge levels as they move.
3      for $v \leftarrow 1$ to $V$ do
4          if $v$ requires recharging and charges at $C_i \in \mathcal{C}$
            then
5              if $v \in \mathcal{I}$ and $C_i \notin \mathcal{E}$ then
6                  $\mathcal{E} \leftarrow \mathcal{E} \cup \{C_i\}$
7              end
8              if $V \notin \mathcal{I}$ and $C_i \in \mathcal{E}$ then
9                  $\mathcal{I} \leftarrow \mathcal{I} \cup \{v\}$
10             end
11         end
12     end
13     $D = D + 1$
14 end
```

Algorithm 4: Cluster-based attack strategy

```
Input: Set of all EVCSs $\mathcal{C}$, logical connectivity graph
       $\mathcal{G}_c$, and number of clusters $U$
Output: Set of targeted EVCSs $\mathcal{A}$
1 Cluster $G_c$ into $U$ clusters using Girvan-Newman
  clustering algorithm [66]
2 Calculate the degree centrality of all the vertices of $G_c$
  after clustering,
3 Select the vertices from each cluster with the highest
  degree centrality and add to $\mathcal{A}$
```

Algorithm 5: Design of Empirical Connectivity Graph

```
Input: Set of all EVCSs $\mathcal{C}$, distances between EVCSs,
       and distance threshold $\Lambda$, i.e., average distance
       among EVCSs
Output: Empirical connectivity graph, $G_e$
Initialize: Edgeless Graph $G_e$ with $C$ EVCSs as
            nodes, $\omega \leftarrow 0$, $x_\omega \leftarrow \infty$
1  for $i \leftarrow 1$ to $C$ do
2      $\omega \leftarrow 0$
3      $x_\omega \leftarrow \infty$
4      for $j \leftarrow 1$ to $C$ do
5          if ($\Lambda_{\{i,j\}} \leq \Lambda$) and ($e_{i,j} \notin G_e$) then
6              connect edge $e_{i,j}$ in graph $G_e$
7          end
8          else
9              if $\Lambda_{i,j} \leq x_\omega$ then
10                 $\omega \leftarrow j$, $x_\omega \leftarrow \Lambda_{i,j}$
11             end
12         end
13     end
14     for $i \leftarrow 1$ to $C$ do
15         if $e_{i\omega} \notin G_e$ then
16             connect edge $e_{i,\omega}$ in graph $G_e$
17         end
18     end
19 end
```

area of Nashville.

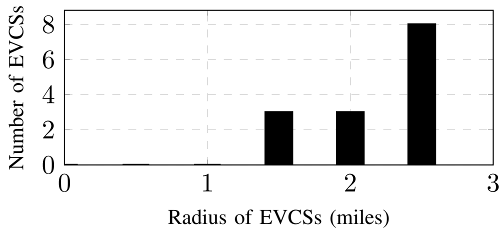


Fig. 9. Number of EVCSs from the centroid in Cookeville.

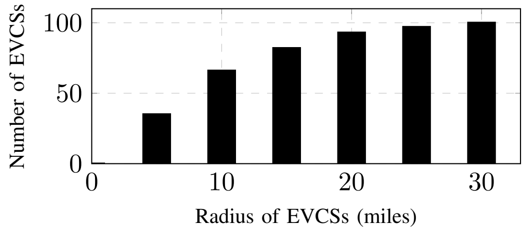


Fig. 10. Number of EVCSs from the centroid in Nashville.

B. Malware Spread

We conducted the following experiments in both rural and urban settings. Firstly, we examined the performance of various attack strategies, including random, cluster-based, and optimal methods, in terms of the time required for malware to spread across the EVCSs in Cookeville and Nashville. Our simulation results in Fig. 11 revealed that for Cookeville, the malware took > 12 days to spread using the random-based strategy, ≥ 12 days using the cluster-based strategies, and

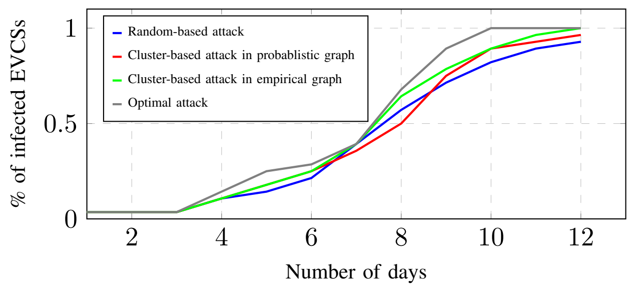


Fig. 11. Comparison of different strategies in Cookeville city.

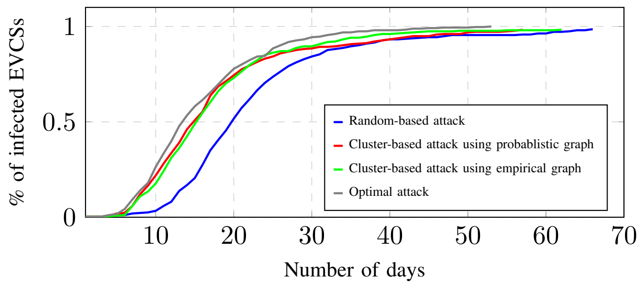


Fig. 12. Comparison of different strategies in Nashville city.

10 days using the optimal strategy *when attacking a single EVCS*. This indicates $> 16\%$ improvement in malware spread when employing the optimal strategy compared to the random one. Similarly, in the case of Nashville, as shown in Fig. 12, attacking a single EVCS resulted in the following to infect all the EVCSs: 69 days for the random attack strategy, 57 days for the cluster-based strategy, and 52 days for the optimal attack strategy. Here, the optimal attack showed a 9% improvement over the cluster-based method and a 32% improvement over the random-based approach. It is worth noting that the clustering strategies based on the probabilistic and the empirical connectivity graphs exhibit close performance with both outperforming the random attack strategy, and coming as second best after the optimal strategy. Specifically, compared with the random attack strategies, cluster-based attack strategies can speed up the malware spread by 10% in Cookeville and 20% in Nashville.

Next, we investigated the effect of the number of attacked clusters and the number of attacked EVCS per cluster in the cluster-based strategy in Cookeville and Nashville. Intuitively, increasing the number of clusters and the targeted EVCS per cluster accelerates malware spread as shown in Fig. 13 and 14. In both Cookeville and Nashville, targeting two EVCSs from two clusters speeds up malware spread by 10% and 17%, respectively, compared to targeting two EVCSs without clustering. Expanding clusters and targeting specific EVCSs accelerates malware spread, showing attackers with limited data can outperform random attacks.

Finally, we investigated the impact of the number of targeted EVCSs on malware spread in both Cookeville and Nashville, focusing on optimal attack strategies. For Cookeville, as shown in Fig. 15, as the number of initially targeted EVCSs increased from 1 to 4, the time required for malware spread was 10, 9, 9, and 8 days respectively. Hence, this increase in the number of targeted EVCSs yields an improvement of 20% reduction in the malware spread duration when targeting 4

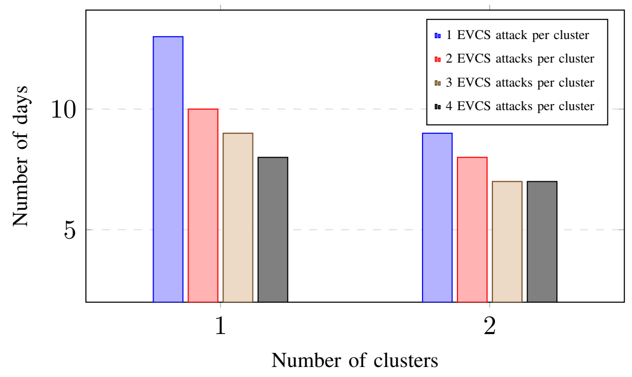


Fig. 13: Effect of number of clusters and targeted EVCSs in Cookeville city for cluster-based attack strategy.

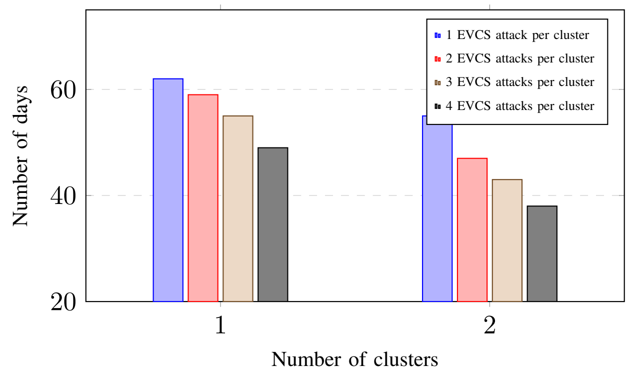


Fig. 14: Effect of number of clusters and targeted EVCSs in Nashville city for cluster-based attack strategy.

out of a total of 8 EVCSs in Cookeville (i.e., targeting 50% of the EVCSs in the city). On the other hand, in the case of Nashville, increasing the number of initially targeted EVCSs from 1 to 4 (out of 250 EVCSs) resulted in malware spread in 52, 43, 41, and 35 days respectively. This demonstrated a substantial improvement in the time to spread the malware, decreasing by 17%, 21%, and 33%, respectively, as the number of initial targets increased only from 1 to 4 (i.e., 1.6% of the EVCSs in Nashville) as illustrated in Fig. 16. These results underscore the significance of the number of attacked EVCSs in optimizing the speed of malware spread.

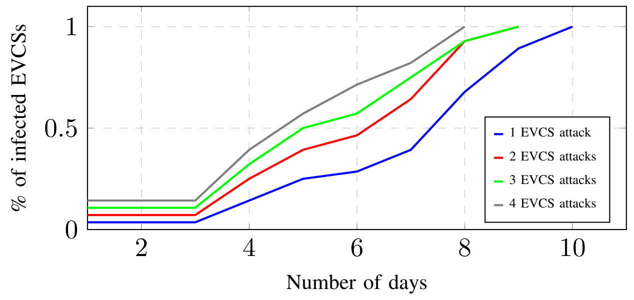


Fig. 15. Different number of targeted EVCSs in Cookeville.

VI. Conclusion

In this research, we studied the vulnerabilities in front-end V2G communication based on ISO 15118 and successfully executed malware injection attacks between EVs and EVCSs. Next, we studied the malware spread among EVCSs in the city. This was done by first establishing a comprehensive city dataset, facilitating the analysis of EV movement patterns and charging behaviors. Using this dataset, we constructed an EV commute and charging simulator (EVCCS) and developed probabilistic connectivity graphs, finding that in Cookeville the EVCSs exhibit full connectivity, while in the case of Nashville, the EVCSs are partially connected. Using the EVCCS and the probabilistic graphs, we studied the probability of malware spread among EVCSs and explored various attack strategies that aim at finding the locations and the number of target EVCSs who when attacked, the speed of malware spread can be maximized. Among these strategies, optimal attacks based on genetic algorithms emerge as the most effective strategy, selecting target EVCSs with high charge-sharing connectivity. Increasing the number of initially targeted stations significantly accelerates malware spread, particularly in Nashville (an example of an urban city), where a 17-day improvement and a 33% acceleration are achieved when targeting only 1.6% of the EVCSs in the city. The second best strategy is the clustering-based attacks that rely on empirical connectivity graphs as this strategy requires minimal information and exhibits a 32% improvement in malware spread compared with random attacks. These results and the developed tools (EVCCS) can play a significant role in future studies to protect the most effective EVCSs, hence, limiting malware spread among the stations in the city.

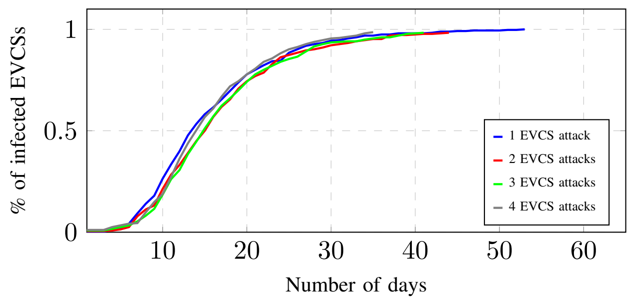


Fig. 16. Different number of targeted EVCSs in Nashville.

References

[1] 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.

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

[3] M. Abouyoussef and M. Ismail, “Blockchain-based privacy-preserving networking strategy for dynamic wireless charging of EVs,” *IEEE Transactions on Network and Service Management*, pp. 1–1, 2021.

[4] N. M. Manousakis et al., “Integration of renewable energy and electric vehicles in power systems: A review,” *Processes*, vol. 11, no. 5, p. 1544, May 2023. [Online]. Available: http://dx.doi.org/10.3390/pr11051544

[5] PWC, “Charging ahead! the need to upscale UK electric vehicle charging infrastructure,” https://www.pwc.co.uk/power-utilities/assets/electric-vehicle-charging-infrastructure.pdf, April 2018.

[6] M. Shafie-Khah et al., “An innovative two-level model for electric vehicle parking lots in distribution systems with renewable energy,” *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 1506–1520, 2018.

[7] 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, [Accessed on: May 30, 2022].

[8] BBC, “Isle of wight: Council’s electric vehicle chargers hacked to show porn site,” https://www.bbc.com/news/uk-england-hampshire-61006816, 2022, [Accessed on: May 30, 2022].

[9] M. McFarland, "Teen's Tesla hack shows how vulnerable third-party apps may make cars," https://www.cnn.com/2022/02/02/cars/tesla-teen-hack/index.html, 2022, [Accessed on: May 30, 2022].

[10] SS-EN ISO 15118-1: 2019, "Road vehicles—vehicle to grid communication interface—part 1: General information and use-case definition (iso 15118-1: 2019)," 2019.

[11] Hydro-Quebec, "Guidelines for the installation of electric vehicle charging stations at state-owned facilities," https://portal.ct.gov/-/media/DEEP/air/mobile/VW/EVSE-Guidelines_State_Facilities.pdf, 2021.

[12] I. Lütkebohle, "BWorld Robot Control Software," http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/, 2008, [Online].

[13] A. Sheikh, "The four most popular methods hackers use to spread ransomware," https://medium.com/@faizan81/the-four-most-popular-methods-hackers-use-to-spread-ransomware-5d4331260227, 2022.

[14] F. Wei and X. Lin, "Cyber-physical attack launched from evse botnet," *IEEE Transactions on Power Systems*, vol. 39, no. 2, 2024.

[15] S. Acharya et al., "Cybersecurity of smart electric vehicle charging: A power grid perspective," *IEEE Access*, vol. 8, pp. 214 434–214 453, 2020.

[16] R. Gottumukkala et al., "Cyber-physical system security of vehicle charging stations," *2019 IEEE Green Tech. Conf.*, pp. 1–5, 2019.

[17] Cyber Security Research and Development, "Cyber assessment report of level 2 AC powered electric vehicle supply equipment," 2018.

[18] Nadia Heninger et al., Ed., *28th USENIX Security Symposium, USENIX Security, Santa Clara, CA, USA.* USENIX Association, August 2019.

[19] S. Hamdare et al., "Cybersecurity risk analysis of electric vehicles charging stations," *Sensors*, vol. 23, no. 15, p. 6716, July 2023.

[20] K. Sarieddine et al., "Investigating the security of EV charging mobile applications as an attack surface," *ACM Trans. Cyber-physical Syst.*, vol. 7, no. 4, p. 1–28, October 2023.

[21] C. Hille et al., "EV charging: Mapping out the cyber security threats and solutions for grids and charging infrastructure," May 2018.

[22] Chris Jewers, "Russian electric vehicle chargers hacked," https://www.dailymail.co.uk/news/article-10565697/Russian-electric-vehicle-chargers-hacked-display-message-supporting-Ukraine.html, 2022, (Accessed on 07/06/2024).

[23] Y. Park et al., "Potential cybersecurity issues of fast charging stations with quantitative severity analysis," in *IEEE CyberPELS*, 2019, pp. 1–7.

[24] D. Sklyar et al., "Chargepoint home security research. 2018, kaspersky lab security services: Kaspersky," 2018.

[25] Cisomag, "Schneider electric patches 13 vulnerabilities affecting its EVlink charging stations," 2021, (Accessed on 10/04/2023). [Online]. Available: https://cisomag.com/schneider-electric-vulnerabilities-fixed/

[26] M. Girdhar et al., "Cyber-attack event analysis for EV charging stations," in *2023 IEEE Power Energy Society General Meeting*, 2023, pp. 1–5.

[27] S. B. Mitikiri et al., "Modelling of the electric vehicle charging infrastructure as cyber physical power systems: Review of components, standards, vulnerabilities, and attacks," 2023.

[28] M. ElKashlan et al., "Machine learning-based intrusion detection system for IoT electric vehicle charging stations (EVCSs)," *Electronics*, vol. 12, no. 4, p. 1044, February 2023.

[29] Roderick Currie, "Hacking the CAN bus: Basic manipulation of a modern automobile through CAN bus reverse engineering," in *Hacking the CAN Bus*, 2020.

[30] W. Ding et al., "Deepsecdrive: An explainable deep learning framework for real-time detection of cyberattack in in-vehicle networks," *Information Sciences*, vol. 658, p. 120057, February 2024.

[31] S. Checkoway et al., "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. USA: USENIX Association, 2011, p. 6.

[32] H. ElHusseini et al., "Blockchain, AI and smart grids: The three musketeers to a decentralized EV charging infrastructure," *IEEE Internet of Things Magazine*, vol. 3, no. 2, pp. 24–29, 2020 .

[33] "Tesla Car Hacked Remotely From Drone via Zero-Click Exploit," https://www.securityweek.com/tesla-car-hacked-remotely-drone-zero-click-exploit/, [Online].

[34] 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, 2023.

[35] 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, 2021.

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

[37] J. Xu et al., "The security of charging protocol between charging piles and electric vehicles," in *2019 IEEE Sustainable Power and Energy Conference (iSPEC)*, 2019, pp. 1315–1320.

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

[39] P. K. S. A. Vishnu et al., "Smart parking and charging management of electric vehicles in public parking space," in *Second International Conf. on Intelligent Computing and Control Systems*, 2018, pp. 1401–1406.

[40] J. Johnson et al., "Review of electric vehicle charger cybersecurity vulnerabilities, potential impacts, and defenses," *Energies*, vol. 15, no. 11, p. 3931, May 2022.

[41] 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. 1–1, 2024.

[42] 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, 2023.

[43] M. Ghafouri et al., "Coordinated charging and discharging of electric vehicles: A new class of switching attacks," *ACM Trans. Cyber-Phys. Syst.*, vol. 6, no. 3, Sep 2022.

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

[45] S. Mousavian et al., "Cyber attack protection for a resilient electric vehicle infrastructure," in *IEEE Globecom Workshops*, 2015, pp. 1–6.

[46] Y. Amara-Ouali et al., "A review of electric vehicle load open data and models," *Energies*, vol. 14, no. 8, p. 2233, 2021.

[47] A. Lojowska et al., "Stochastic modeling of power demand due to EVs using copula," *IEEE Transactions on Power Systems*, 2012.

[48] J. Munkhammar and M. Shepero, "Autonomous electric vehicle fleet charging in cities: Optimal utility estimates and monte carlo simulations," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 2017, pp. 1–6.

[49] B. Lantz et al., "Mininet: An instant virtual network on your laptop," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI).* USENIX Association, 2010, pp. 191–206.

[50] "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.

[51] B. Pfaff et al., "The design and implementation of open vswitch," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI).* USENIX Association, 2015, pp. 117–130.

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

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

[54] OpenStreetMap Team, "OpenStreetMap," https://www.openstreetmap.org/#map=4/38.01/-95.84.

[55] US DoE Vehicle Technologies Office, "Alternative fueling station locator," https://afdc.energy.gov/stations#/find/nearest.

[56] "Quantum Geographic Information System," https://qgis.org/en/site/.

[57] United States-Census Bureau, "City and Town Population Totals: 2010-2019," Available on: https://www.census.gov/data/tables/time-series/demo/popest/2010s-total-cities-and-towns.html.

[58] "Commuting in Cookeville, Tennessee," https://www.bestplaces.net/transportation/city/tennessee/cookeville/, 2020, [Online].

[59] " Cookeville EV information, howpublished = "https://www.atlasevhub.com/materials/state-ev-registration-data#data", note = "[online]"."

[60] J. Yin and G. Chi, "Characterizing people's daily activity patterns in the urban environment: A mobility network approach with geographic context-aware twitter data," *Annals of the American Association of Geographers*, vol. 111, no. 7, pp. 1967–1987, 2021.

[61] "What Time Does The Average American Leave for Work? - Overflow Data," https://overflowdata.com/demographic-data/what-time-does-the-average-american-leave-for-work/, 2016, [Online].

[62] "TNTech." [Online]. Available: https://www.tntech.edu/research/resources/institutional-information-archived.php

[63] "Cookeville information." [Online]. Available: https://putnamcountytn.gov/county-information

[64] Waze Team, "Waze API," 2023. [Online]. Available: https://developers.google.com/waze

[65] U.S. DoE, "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.

[66] L. Despalatovic, T. Vojkovic, and D. Vukicevcc, "Community structure in networks: Girvan-newman algorithm improvement," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2014, pp. 997–1002.