The Open Charge Point Protocol (OCPP) Version 1.6 Cyber Range A Training and Testing Platform

David Elmo II
Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all

Part of the Computer Engineering Commons, and the Computer Sciences Commons

Repository Citation
https://corescholar.libraries.wright.edu/etd_all/2788

This Thesis is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.
THE OPEN CHARGE POINT PROTOCOL (OCPP) VERSION 1.6
CYBER RANGE
A TRAINING AND TESTING PLATFORM

A Thesis submitted in partial fulfillment of the
requirements for the degree of
Master of Science in Cyber Security

by

DAVID ELMO II
B.S. The University of Toledo, 2006

2023
Wright State University

Junjie Zhang, Ph.D.
Thesis Director

Thomas Wischgoll
Interim Chair, Computer Science and Engineering

Committee on Final Examination:

Krishnaprasad Thriunarayan Prasad, Ph.D.

Bin Wang, Ph.D.

Junjie Zhang, Ph.D.

Shu Schiller, Ph.D.
Interim Dean, College of
Graduate Programs & Honors Studies
ABSTRACT

Elmo II, David. MSCS, Department of Computer Science and Engineering, Wright State University, 2023. The Open Charge Point Protocol (OCPP) Version 1.6 Cyber Range A Training And Testing Platform

The widespread expansion of Electric Vehicles (EV) throughout the world creates a requirement for charging stations. While Cybersecurity research is rapidly expanding in the field of Electric Vehicle Infrastructure, efforts are impacted by the availability of testing platforms. This paper presents a solution called the “Open Charge Point Protocol (OCPP) Cyber Range.” Its purpose is to conduct Cybersecurity research against vulnerabilities in the OCPP v1.6 protocol. The OCPP Cyber Range can be used to enable current or future research and to train operators and system managers of Electric Charge Vehicle Supply Equipment (EVSE). This paper demonstrates this solution using three attack types, Denial of Service, Machine-in-the-Middle, and Log4shell.
# Contents

**Introduction** .................................................................................................................................................. 1  
1.1 Motivation .................................................................................................................................................. 3  
1.2 Research Statement .................................................................................................................................. 5  
1.3 Thesis Outline .......................................................................................................................................... 5  

## 2 Related Work

2.1 NREL CEEP ................................................................................................................................................ 7  
2.2 Issues and Challenges of OCPP .............................................................................................................. 9  
2.3 Vulnerabilities in Commercial Charging Stations .................................................................................. 10  

## 3 Background

3.1 Building the System ................................................................................................................................. 20  
3.1.1 The Server .......................................................................................................................................... 21  
3.1.2 The Client .......................................................................................................................................... 21  
3.1.3 The Attack Machine .......................................................................................................................... 22  
3.2 Challenges ............................................................................................................................................. 23  

## 4 Attack Methodology

4.1 Denial of Service ....................................................................................................................................... 24  
4.1.1 Configuration One – AWS .................................................................................................................. 26  
4.1.2 Configuration Two – Pure VM .......................................................................................................... 27  
4.2 Machine-In-The-Middle .......................................................................................................................... 30  
4.3 Log4Shell .............................................................................................................................................. 36  

## 5 Discussion

......................................................................................................................................................................... 43  

## 6 Conclusion

......................................................................................................................................................................... 45  

References ......................................................................................................................................................... 46
List of Figures

Figure 2.3-1: Overview of charging infrastructure for EVs ...................................................... 12
Figure 2.3-2: Modes of Charging at Different Levels ................................................................. 13
Figure 2.3-3: EV Communications Ecosystem ............................................................................ 17
Figure 4.1-1: Client Boot Sequence ............................................................................................. 27
Figure 4.1-2 Server Resource Utilization. ................................................................................... 28
Figure 4.1-3 Shows a snapshot of the “tcpdump” capture on the client machine. ..................... 28
Figure 4.1-4 DoS Attack Command and Configuration .............................................................. 29
Figure 4.1-5 Attack System Utilization. ....................................................................................... 29
Figure: 4.2-1 Client Traffic to Server. ......................................................................................... 31
Figure: 4.2-2: MITIM Traffic Between Client/Server. ................................................................. 33
Figure 4.2-3 Server Traffic From MITM ..................................................................................... 33
Figure: 4.2-4 Curl Request ......................................................................................................... 34
Figure 4.2-5 "update_firmware" Command From Server. ......................................................... 34
Figure 4.2-6: MITM Traffic From Server ..................................................................................... 35
Figure 4.2-7: Client Traffic from MITM ....................................................................................... 35
Figure 4.3-1: Log4Shell Information Flow ................................................................................... 38
Figure 4.3-2: Phase 1 Traffic Flow ............................................................................................... 40
Figure 4.3-3: Contents of the “malicious.conf” file ................................................................. 40
Figure 4.3-4: Phase 2 Traffic Flow ............................................................................................... 41
Figure 4.3-5: Phase 3 Traffic Flow ............................................................................................... 42
List of Tables

Table 3-1: Types of charging infrastructure ................................................................. 15
Table 3-2: Client to Server Command Examples .......................................................... 22
Table 4-1 Cyber Range IP Table ..................................................................................... 34
Acknowledgement

First and foremost, I would like to thank God for leading me to this path and providing me the courage and dedication to stay the course through all the challenges. I would like to thank my wife Ashley and my children for their support and dedication through this experience. I may have been doing the studying, but they are the ones who made the real sacrifice and kept me on track with their love, support, and encouragement. To my parents, thank you for your love and support to our family through everything. My deepest thanks to Jay Johnson and George Fragkos at Sandia National Laboratories for their guidance and collaboration on building this project. It would not have been possible without them. I would like to thank my thesis advisor Dr. Zhang for his support and guidance through my research. I would also like to thank my mentor and friend COL Carlos Pena, US Army, for helping me to get where I am over the last five years. Lastly, I would like to thank the Professors in the Cybersecurity Program and Faculty of Wright State University for their teaching and guidance throughout the Cybersecurity program.
Introduction

Electric Vehicles (EV) are rapidly replacing fossil-fuel powered vehicles both in the commercial and private sectors. The global expansion of EVs creates a requirement for a massive increase in charging stations. In the US, there is currently no industry standard for the communications protocol between the client charging station and the server management systems. Companies are at liberty to utilize their own proprietary communication protocols. This creates vulnerabilities to EV operators, supporting infrastructure, and puts sensitive information at risk. Recent passed legislation demonstrates the increased political will to reduce carbon emissions and allocate large sums of money to achieve this expansion. “Electric transportation got a jolt of support from the 2021 Infrastructure Investment and Jobs Act — which funds $7.5 billion in EV charging infrastructure. Most recently, the Inflation Reduction Act provided tax credits for both new and used electric passenger vehicles as well as for commercial vehicles, and California announced it will ban the sale of new internal combustion engine-powered vehicles by 2035.” [1] With millions of EVs forecasted to be on the road in the coming years, charging their batteries needs to be managed to prevent overloading the electricity grid and minimize costly upgrades that are ultimately paid for by consumers. [2] The number of charge points in the US is poised to grow from about 4 million today to an estimated 35 million in 2030. [1] The vast increase in these charging stations and the impact they have on commercial and consumer transportation creates a potential threat to the overall US way of life if not properly studied and vulnerabilities addressed.

As the proliferation of charging stations has grown throughout the world so has their capability and sophistication. Current charging station clients communicate with customers through a wide variety of methods such as RFID cards, mobile apps, and traditional credit cards
to manage transactions. This level of interconnection is a replication of the same information
security issues that are present in today’s computing environment. Research to support protecting
this information and the risks associated with this level of system interaction must be studied to
protect users and reduce the risk of interruption in critical supply chains across the globe.

Failure to protect this information could lead to severe consequences in the consumer,
commercial and strategic categories. This Thesis will focus on and demonstrate three types of
attacks that individually or combined, can have impacts at the aforementioned categories. These
attacks result in the attacker’s ability to manipulate, monitor, or deny information transfer
between the charging client and charging management server.

At the consumer level, PII and financial information transmitted between the client and
server, is vulnerable to interception if not properly encrypted. The transaction information
transmitted across the communications link leaves consumers vulnerable to the theft of that PII
and/or their financial information by way of “machine-in-the-middle” (MITM) attacks. This
information becomes susceptible to sale on the black-market which leaves consumers vulnerable
to identity theft and fraudulent credit card charges.

In the commercial sector, vulnerabilities may lead to attackers conducting price changes
on charging units by manipulating the configuration and pricing information in the individual
units. MITM and Log4shell attacks allow an attacker to gain root access to a client or server.
These attacks could result in financial loss to commercial entities due to modified prices and
energy theft by the attacker. Additionally, attackers can conduct denial of service attacks on
charging clients. This type of attack has both short- and long-term implications. In the short term,
loss of sales affects immediate profits. However, impacts on the company’s reputation can have
much greater consequences in the long-term future of a company.
At the strategic level, the ability to achieve root access to multiple charging systems throughout a region can have significant impacts to the power grid, federal emergency response, and commercial transit of goods and services. The ability for attackers to control charging stations can impact the power-grid by changing charging electric loads during high-peak times away from “smart-grid” configurations, resulting in over and underpowering grid areas. [3] The white house is attempting to push the entire federal vehicle fleet to 100% zero-carbon emission vehicles by 2027. [4] Without readily accessible charging stations federal emergency response effort may be hindered resulting catastrophic outcomes. Additionally, without availability of charging stations within acceptable proximity of each other, delays of commercial electric based transportation are inevitable. The inability of operators to charge their vehicles within a platform’s limitation will cause drivers to have to stop more frequently than fossil-fuel powered transportation which delays deliveries.

1.1 Motivation

The lack of EV client-server communication standardization makes it difficult to ensure proper security measures are being implemented, and security vulnerabilities patched, to reduce the possible attack vectors in the client to server communications channel. Some countries are working to implement a national protocol standard which includes methods to protect sensitive information such as PII, financial information, or information about the EV using the charging station. There is also a rise in the number of companies globally adopting the Open Charge Point Protocol (OCPP) standard for this communication.

OCPP is a global open-source communication protocol between charging stations and the back-end systems that manage the individual stations. More specifically it controls
communications between the charging point (CP) and the charging station management system (CSMS). [5] The protocol is supported by the Open Charge Alliance (OCA) with more than 220 member-companies active in the electric mobility area. [5], [6] The OCA’s mission is to “foster global development, adoption, and compliance of communication protocols in the EV charging infrastructure and related standards through collaboration, education, testing, and certification.” [6] OCPP is a widely known and readily available protocol used to communicate between the EVSE client and the Charging Station Management System (CSMS), the server. OCPP is quickly becoming a go to protocol for EV equipment across multiple vendor platforms due to it being free for commercial vendors to use. [6] One research community recommendation is that “OCPP systems should be standardized and certified by the Open Charge Alliance (OCA). OCA certification would ensure uniformity and cross-vendor compatibility.” [3], [7] Many companies are beginning to implement this protocol in their system including several in the US. SemaConnect, a US based company, charging stations and the SemaConnect Network are OCPP 1.6 certified. “OCPP is currently being used in 148 countries across all six continents and is supported by more than 65,000 installed and operating charging stations.” [3], [6]

The vast number of charging stations require central control servers to control software, updates, current flow, financial transactions, etc... “Charging providers and users alike seek to optimize their use of the growing network of fast chargers through a variety of highly interconnected and internet-enabled tools. EVSE must communicate with cloud services, EVs and their battery management systems, and much more.” [5] This is done through a variety of protocols such as Bluetooth, Wi-Fi, WAN, and Cellular.

This increase in sophistication comes with potential security vulnerabilities. Between 2018 and 2021 cyberattacks against electric vehicles increase 225%. [8] “Smart charging and
vehicle-to-grid (V2G) charging technologies connect an EV to the electricity grid. This is done using a charging device which includes a data connection and ensures the exchange of information and control commands between various entities in the EV ecosystem including CP, CP operators, and grid network operators, among others.” [2] Any connected infrastructure is a potential target for cyber-security attacks, with motives including information theft, cyber-warfare, or organized crime. [9]

1.2 Research Statement

The OCPP Cyber Range presented in this Thesis, is a platform that can be expanded and used to test cybersecurity tactics, techniques, and procedures to develop methods to protect the charging ecosystem consisting of the client and CSMS, utilizing OCPP version 1.6. It demonstrates the need to secure and encrypt the communication between the EVSE and CSMS by experimentally exploiting the OCPP client and server’s communications to deny or modify its intended actions. This work focuses on version 1.6 because of its wide level of implementation in the field and its stability. The exchange of information between the OCPP server and client is the primary target for these experiments. For this purpose, the main attacks this research will focus on will be denial of service (DoS), machine-in-the-middle and Log4Shell.

1.3 Thesis Outline

The remaining sections of the thesis are organized as follows; The Second Chapter is related work, that has been or is being conducted in this field. The Third Chapter will focus on the background information covering the EVSE, Electric Vehicle Management Server (EVSM), OCPP, how the OCPP cyber range is built and challenges to the build. The Fourth chapter will cover attack methodology and analysis of the resulting data. The Fifth Chapter will cover a brief
discussion of future use and research with the system. The Sixth and final chapter is the conclusion.
2 Related Work

The growth of Electric Vehicles has prompted a new level of security research across the community. While still a growing area of research, there are several related works whose focus are closely related to this Thesis. First, is the Identification and Testing of Electric Vehicle Fast Charger Cybersecurity Mitigations by the National Renewable Energy Laboratory (NREL). Second, an article published by research in the Institute of Electrical and Electronics Engineers (IEEE) entitled “Electric Vehicle Charging: A survey on the security issues and challenges of the Open Charge Point Protocol.” Third, is research conducted by Sandia National Laboratories to test known/published attacks against EV charging equipment.

2.1 NREL CEEP

Testing on commercial EVSE systems creates a challenging situation for many cybersecurity researchers. The lack of access to commercial grade equipment and open-source software creates a gap in the ability for researchers to test theories, identify vulnerabilities and generate mitigations to potential threats. By restricting the pool of researchers, companies open themselves to vulnerabilities remaining unknown except to attackers for extended periods of time. The use of cyber emulation using virtualization has proven to be invaluable “for research, development, testing, validation, and modeling. “[10]

“In 2019, Idaho National Laboratory (INL), Oak Ridge National Laboratory, and the National Renewable Energy Laboratory (NREL) were awarded a scope of work called Consequence-Driven Cybersecurity for High-Power Charging Infrastructure, and the laboratories have jointly worked to identify, evaluate, and mitigate potential cyber-related consequences associated with high-power fast chargers.”[11] The focus of their research was high-
consequence events (HCE) which constitute situations that lead to significant outcomes, such as the loss of system control, equipment damage, and/or financial loss. [11] 50 Concepts were developed as potential focus areas. Those areas were then categorized into six different groups. The areas they determined to need the most focus were the following five outcomes:

1. “Grid Impacts (power flow dynamics, generator, or substation operation change)
2. Safety (fire, electrocution)
3. Loss of service (exceeding circuit limits or forcing mis-operation)
4. Hardware damage (blown fuses, overloaded components, bricked communications)
5. Data theft/altercation (credit card data, personal information)” [11]

To achieve their goals, the creation of a simulation platform was required that combines cyber-physical systems with a simulation and monitoring system with the ability to monitor necessary parameters. The goal of this build was to “combine currently available systems and network virtualization, power simulation, real-time data streaming, and power hardware-in-the-loop technologies to build a comprehensive, real-time visualization tool. This tool is a desirable capability for many organizations, especially those related to research and education.” [11] The solution created by the NREL is called the Cyber-Energy Emulation Platform (CEEP). [30] One of the key objectives to the NREL’s project was to build a simulation platform that would allow researchers to view a cyber event in a simulated environment from multiple perspectives. [10] The overall goal of this system was to “develop a 3D platform that would, through a single web interface, provide users with a vantage from which to observe cybersecurity experiments unfolding, quickly highlight anomalies and attacks, and interact with the system for in-depth analysis.” [10] CEEP provides the following core features; “Cyber emulation layer, Power co-
simulation framework, Data collection and storage, Data parsing and transport, Real-time visualization, and Binding and control scripts.” [10]

The “most recent progress provides results of test scenarios that compare a baseline implementation under attack on both unprotected systems and systems with potential mitigations implemented.” [11] “OCPP is used by charge network operators to enable and monitor charging operations across all chargers at a site or across a broad region. [11]. With its potential for accelerating cybersecurity research as well as education and training, the development team sees the CEEP as an emulation tool for pushing the frontiers of our current understanding of complex and interconnected systems at scale. Currently, CEEP is in active development at NREL and is in a closed alpha development stage. This platform has been awarded U.S. Provisional Patent Application No. 62/913232.” [11]

2.2 Issues and Challenges of OCPP

An article published in IEEE entitled “Electric Vehicle Charging: A survey on the security Issues and Challenges of the Open Charge Point Protocol” [12] conducts a survey that focuses on security issues in the OCPP protocol posed by various scholars. The authors analysis goes on to “identify open security issues for OCPP and propose future research directions for the security enhancement of the protocol.” [12]

This article builds on a study which revealed severe vulnerabilities in the charging stations of 16 different vendors that could be exploited. [13] Another such article goes on to mention that local power supplies are vulnerable to the lack of security on EVs and EV charger data. [14] The key contributions of this article are “An analysis of the security issues and threats, and of the related countermeasures proposed by other scholars. [12] They go on to provide an explanation of
existing security threats proposed in the previous article and associate the threat to the specific subsystem of the EV infrastructure. Finally, they identify open issues and offer proposals of future areas of research for OCPP 2.0. [12]

2.3 Vulnerabilities in Commercial Charging Stations

Sandia National Laboratories has been conducting research to identify vulnerabilities in commercial charging stations. During the 9th Embedded Security in Cars (escar) USA Conference in Ypsilanti, Michigan, Jay Johnson, of Sandia National Laboratories, presented a talk on vulnerabilities in the EVSE components and protocols. The focus of their research was high and low power level 2 chargers, back-end cloud networks, OCPP 1.6 and ISO 15118-2 PKI requirements. [15] During their research they “surveyed known/published vulnerabilities by testing them against current industry commercial grade equipment. The result of this information was the presentation of vulnerability information to industry partners who owned the tested equipment. [9] Their research helped to patch many of the vulnerabilities found, before they were publicly disclosed.

One of the attacks selected by the team included “Brokenwire”, a novel attack developed by the University of Oxford. This attack targeted the charging connector’s communications link. It demonstrated that a high-power Combine Charging System (CCS) session could be terminated through a transmitted RF attack utilizing a commercial off-the-shelf software defined radio and an amplifier. [15]

Sandia also explored Java Log4j/Log4Shell Privilege escalation. Their research showed the possibility of launching a Log4j/Log4Shell attack successfully against a simulated EVSE to achieve privilege escalation. They did this by combining the Log4j vulnerability with a vehicle
to grid (v2g) injector and a flaw in the HomePlugPHY key collection to achieve their desired privilege escalation. [15] Their research also found that vehicles may be vulnerable to this attack. [15]

“Their research concluded by making several recommendations. “First, the need for the cybersecurity community to continue to identify EV charger vulnerabilities. Second, federal and state governments should seek policies to improve the security of EVSE systems. Finally, they recommend the need for a comprehensive national cybersecurity approach must include information sharing programs and incident response strategies.” [15]
3 Background

There are a variety of topologies and component configurations that are common in commercial grade Electric Vehicle Supply Equipment (EVSE). Each vendor modifies components to fit their various needs. However, certain basic components are in most commercial systems. These include but are not limited to, “external EV connectors, an authentication terminal (e.g., the front console), and a maintenance terminal(s) that may be internal to the EVSE housing. The EVSE also often has a cellular or other internet connection for the EVSE operator or service provider to capture data on charging sessions, push new firmware, and collect prognostics and user data using Open Charge Point Protocol (OCPP), IEEE 2030.5, or proprietary protocols.” [15]

There are three types of EV charges currently in circulation. Level I, Level II, & Level III. (Figure 3-1) The sophistication of the charging station varies based on the level, the vendor, and the intended use. As sophistication increases so do the vulnerabilities and attack vectors within the system. Level I (L1) chargers come with the vehicle at the time of purchase and run on 120v AC power. These devices can be plugged into any US outlet without specific installation. L1 chargers, have a power output of 2.4kW and charge at a rate of approximately 40mi over an eight-hour period or 5mi per hour. [16] L1 chargers are the
slowest of the charging methods. At maximum output of 16 amps and 1.9kW, L1 chargers can provide a full charge in-between 8-16hrs depending on the size of the battery. [16]

“Level II (L2) chargers are most common in public and residential locations. These chargers use a single phase 240v power supply with a maximum output of 40 amps for residential and a three-phase maximum current flow of 400v 80amps for commercial units. The maximum charging power at L2 stations is around 12kW, equating to about 100mi of charging every eight hours. L2 chargers can provide a quick charge for people traveling longer distances through the ability to conduct fast charging. Over-voltage and over-current protection are built into L2 systems.” [16] L2 are the most common method of charging for the every-day user. With both L1 and L2 charges the AC/DC converter on board the EV is utilized as the method of direct access to the DC battery on board the vehicle as depicted in Figure 2.3-2.

Level III (L3) chargers are the fastest method of charging and the most expensive. Currently, L3 chargers are only available in the commercial sector and can be found, most commonly, in areas such as rest stops, shopping centers, and entertainment districts due to their rapid ability to recharge. [16] In both L1 and L2 chargers, electricity is delivered to the AC/DC power converter on board the vehicle. “DC chargers deliver power directly to the battery and bypass the on-board converter completely (Figure 3-2). L3 chargers do not comply with industry standards and are not universally compatible. The charge time for a battery
from 0% to 80% is 15 to 20 minutes. Charge specs for L3 charges range from 200v to 600v with power outputs ranging from 36 to 240kW.” [16]

The vast number of charging stations require central control servers to control software updates, current flow, financial transactions, etc... “Charging providers and users alike seek to optimize their use of the growing network of fast chargers through a variety of highly interconnected and internet-enabled tools. Most “EVSE will communicate with cloud services, EVs and their battery management systems, and much more.” [15] This is done through a variety of protocols such as Bluetooth, Wi-Fi, WAN, and Cellular.

There are three basic methods of managing charging infrastructure, Non-Networked, Closed Network and Open Network charging Infrastructure. Table 3-1 provides a brief description of each. Non-Networked is, as it sounds, not controlled beyond the plug and charge connection. Closed and Open network charging stations are connected to a controlling infrastructure through a networked connection. “New managed charging technologies such as smart chargers and vehicle to grid (V2G) technologies are being introduced. Electric vehicles, plugged using these new charging technologies, can help balance supply and demand across a green electricity system and reduce costs for the consumer.” [2], [17], [18]
These systems communicate via protocols between the client and the management system. While many of these protocols are proprietary and vendor dependent there is currently a global push for standardization across platforms. Arguably the most notable is OCPP version 1.6 and 1.6j variant. OCPP stands out as the de facto protocol, used in 148 countries across all 6 continents and is supported by more than 65,000 installed and operating charging stations. [12], [19] OCPP is managed by the Open Charge Alliance (OCA) who’s mission is to “foster global development, adoption, and compliance of communication protocols in the EV charging infrastructure and related standards through collaboration, education, testing, and certification.” [20]

OCPP is the result of an idea proposed by the “Dutch foundation ElaadNL for an open protocol to support the communication between the charging points and the back-end systems.” [15], [20] The protocol started in 2009 and was originally dubbed the “Open Charge Point Protocol Forum.” [19] Their objective was to increase interoperability by developing a method for charging stations to communicate that was open source and therefore would allow charging companies to build networks that were vendor agnostic. The hope was to streamline proliferation

<table>
<thead>
<tr>
<th>Non-Networked</th>
<th>Closed Networked</th>
<th>Open Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charging Stations Work As Stand-Alone Units With No Network Connectivity</td>
<td>EV Chargers Based On Proprietary Protocols To Communicate With Charging Networks.</td>
<td>Allows Owners Of Charging Stations To Flexibly Switch Between Charging Networks And Equipment Vendors Without Any Change In Infrastructure.</td>
</tr>
<tr>
<td>Uses The Standard Electricity Rates Offered By The Grid Supplier.</td>
<td>Typically Needs Entire System Replacement To Switch To Other Networks Or Charging Equipment Due To Vendor Lock-In.</td>
<td>Offers Enhanced Security And Integration With Energy Management Systems And V2G Applications</td>
</tr>
<tr>
<td>Example: Residential Chargers</td>
<td>Example: Tesla Chargers</td>
<td>Example: OCPP Compliant EV Charger And Software</td>
</tr>
</tbody>
</table>

Table 3-1: Types of charging infrastructure. [43]

These systems communicate via protocols between the client and the management system. While many of these protocols are proprietary and vendor dependent there is currently a global push for standardization across platforms. Arguably the most notable is OCPP version 1.6 and 1.6j variant. OCPP stands out as the de facto protocol, used in 148 countries across all 6 continents and is supported by more than 65,000 installed and operating charging stations. [12], [19] OCPP is managed by the Open Charge Alliance (OCA) who’s mission is to “foster global development, adoption, and compliance of communication protocols in the EV charging infrastructure and related standards through collaboration, education, testing, and certification.” [20]

OCPP is the result of an idea proposed by the “Dutch foundation ElaadNL for an open protocol to support the communication between the charging points and the back-end systems.” [15], [20] The protocol started in 2009 and was originally dubbed the “Open Charge Point Protocol Forum.” [19] Their objective was to increase interoperability by developing a method for charging stations to communicate that was open source and therefore would allow charging companies to build networks that were vendor agnostic. The hope was to streamline proliferation
of the EVSE, increasing availability to consumers, which reduces the burden of resupplying electric vehicles and allows for their utilization to become more widespread.

Prior to OCPP, “charging networks (which typically control pricing, access, and session limits) were closed and did not allow for site hosts to change networks, should they want different network features or pricing. Instead, they had to completely replace the hardware (the charging station) to get a different network.” [21], [22] This method of control was slow and costly to vendors which posed a risk to the overall goal of moving away from fossil-fuel engines.

This thesis will focus solely on the implementation of OCPP v1.6. While there are newer versions of OCPP, v1.6 is still widely implemented in the field throughout the EU and servers as a pertinent training platform for people training in the field of EVSE management system protection. Due to its long-term implementation in the field, it is the most thoroughly tested for bugs and therefore is currently the most stable version of OCPP used in commercial environments.

OCPP is a demand-response protocol which mainly provides the messages for the communication between the Charging Station (CS) and the Charging Station Management System (CSMS), although in practice it is not exclusively used for that communication. [12] "The “OCPP offers a uniform solution for the method of communication between charge stations and any central system. With this protocol, it is possible to connect any central system with any charge station, regardless of the vendor.” [5]

As previously mentioned, topographies and configurations may vary based on vendor discretion however basic setup is generally consistent amongst EVSE networks. Figure 3-3 depicts a general EVSE layout. The four main components depicted are the connection from the
vehicle the charging client, authentication, and interaction terminal of the client, EVSE 
operators/vendor’s cloud, and maintenance terminal. [15] Each of the components in Figure 3-3 
may be vulnerable to attack in some way.

In level 1 and 2 chargers the connection from the vehicle to the charger (Figure 3-3 block 1) 
is an industry standard J1772 connector. This is standard across North America except for Tesla. 
Tesla has their own connection but includes the required adapter to utilize the J1772 connector. 
[23] However, level 3 chargers do not require an industry standard connection. Currently there 
are three different types of connectors, Combined Charging System (CCS), CHAdeMO, and 
Tesla. Each of these connectors is met with a different level of capability and thus are open to 
potential attacks. For example, the previously mentioned “Brokenwire” attack specifically 
targeted a leak in the data channel within the cable exposing the data link between the vehicle 
and charger to corruption from an outside RF source. [15]

The authentication terminal is used for authenticating users, completing financial 
transactions required for charging, and other such tasks. It communicates using mobile apps or

Figure 2.3-3: EV Communications Ecosystem. [9]

on-screen transactions. On-screen transaction validation is generally conducted using a credit
card, RFID tag or near field communication using mobile device, such as apple pay. This connection (Figure 3-3 block 2) can also serve as an attack vector for malicious actors. Both commercial L2/L3 and residential L1 chargers have been known to have vulnerabilities. Many times, this is caused by 3rd party plugins used for billing, navigation, [15] or system management. “Kaspersky Lab revealed security flaws in the ChargePoint Home smartphone application for EV charging. This flaw would enable a remote attacker to intrude into the charger and tamper with EV charging via the WiFi connection to the charging device.” [3], [7] ”Security flaws were also identified in EV-link chargers produced by Schneider Electric. This flaw would allow a remote attacker to bypass hard-coded authentication credentials, inject malware, and disable the charger.” [3], [24]

This Thesis will focus exclusively on the attack vector available in the communication between the client and server (Figure 3-3 block 3). This communications link is where the server communicates with and controls the client. This link is responsible for conducting price changes, validating charging sessions, initiating firmware updates, and monitoring the status of individual charging clients. While many vendors in the US utilize their own communications protocol, OCPP is gaining ground in becoming the go to protocol for this line of communication. OCPP has two different communications protocol options, JavaScript Object Notation (JSON) and Simple Object Access Protocol (SOAP).

JSON “is a light-weight data-interchanging format. It is built on two structures, a collection of name value pairs and an ordered list of values.” [25] “One of the main benefits of JSON comes in the format which is text only. This enables JSON to easily be sent between two computers and can be used by any programming language.” [26] In OCPP these JSON files are transported over websockets established on top of a TCP connection. The websocket connection
starts as an “HTTP GET” request with an upgrade. The resulting websocket connection enables duplex communication over a TCP connection. [27]

“SOAP is a lightweight XML-based protocol for exchanging information in a decentralized, distributed environment. By combining SOAP-based requests and responses with a transport protocol, such as HTTP, the internet becomes a medium for applications to publish database-backed Web services.” [28] One of the main reasons this project focused on JSON over SOAP, is that SOAP not available in the newer versions of OCPP.

OCPP is currently on Version 2.0.1, but, as previously stated, version 1.6 is still widely used in the field.” [15] This makes it a viable training platform for server and control system operators. “Unfortunately, OCPP did not include PKI encryption until Version 1.8 [40] [12], therefore many EVSE systems rely on running v1.6 which is unencrypted and requires the use of virtual private networks (VPNs), isolated cellular networks, or other protections to avoid reconnaissance and hacking attempts.” [15] However, the ability of the attacker to gain access to the communications channel opens this link to a wide variety of attack vectors. This makes them vulnerable to machine-in-the-middle attacks and replay attacks. Additionally, “OCPP v1.6 does not require client authentication to begin communication with the server, this creates a vulnerability to a denial-of-service attack.” [19]

“Generally, EVSE are firewalled from the internet, however multiple devices have been found using the “Shodan” tool and other targeted searches. [29] Not only do these internet connections create the potential for the EVSE to be exploited from the internet, but there is also a risk that EVSE vendors or operator systems could be compromised by using the EVSE as an entry vector into their networks. This would result in an attacker potentially controlling large fleets of EVSE devices which could impact power grid operations, transportation systems, or
other critical infrastructure. The ability to pivot between vehicle, EVSE, and cloud interconnected domains was the focus of previous attack tree research [15], [30]

3.1 Building the System

The system is built on a basic websocket server using OCPP v1.6 protocol files from Mobility house hosted in a github.com public repository. [31] Both the client and server are operating system (OS) agnostic which allows them to be hosted on any platform that has python installed. For this research, virtual machines were utilized to build and test all aspects of the system. The system was created using a Linux Ubuntu distribution hosted by Virtualbox on a Windows 10 system. The connection was bridged to allow for discovery by the home network router. The attack machine is a Kali Linux VM also hosted by the Virtualbox system.

There are four main parts to the OCPP system divided into two main subsections, the OCPP Client and OCPP Server. Both the client and the server have two parts to their communications process, the websocket, enabling the communication between the OCPP client and OCPP server, and an asyncio http server used to handle asynchronous communication between an external system operation and the OCPP entities [32], [33] ie: client and server requests between servers and communication between the operator and the OCPP client or server. The aiohttp server enables this by creating a single object that can be used for multiple individual requests and by default can make connections with up to 100 different servers at a time. [32] Each system (client and server) is hosted in a separate VM. Additionally, a logging extension is added to the client for logging purposes which uses Apache Log4J. [34] In the interest of testing the log4shell attack on the system, log4j version 2.14 was used in the client’s logging framework.
3.1.1 The Server

When the OCPP server program starts it creates a websocket server which listens on port 9000. When the client attempts to connect, the server receives the incoming request, and checks the client is attempting to connect with the correct subprotocols, in this case OCPP v1.6. If these checks pass, the server issues a “charge_point_id,” to keep track of which websocket is assigned to which charge point. The server creates a .config file to store the “websocket_connection_id” for each connection.

The OCPP server then creates an aiohttp server and saves the port assigned to it in the “charge_point_id .config” file for each connection. This aiohttp server and port are the interface to issue commands to the OCPP server. Requests are submitted using the “curl” command in the terminal window in the form of an http request. Commands are sent from the operator to the http server which then sends them to the OCPP server for processing. This method allows for the users of both the EVSE and the CSMS to interact with the system in real-time. “The OCPP server communicates with the OCPP client via the websocket, throughout the connection for authorizing, initiating, and stopping the charging sessions with the EV. The OCPP server also provides the operator an ability to configure and control the fast charger functions.” [11]

3.1.2 The Client

At the start of the client program the system checks to ensure that it has two arguments, argument one is the “name of CP”, argument two is the “id_tag of CP”. If both arguments are present, the client sends a websocket request to the server’s IP address at port 9000, along with the arguments passed at startup, and the “id of the subprotocol” used in this case OCPP v1.6. While the CP establishes a connection with the OCPP Server, the client creates an http server for
communication with the OCPP Client. The system can be configured to assign a new port per connection. However, at this time it is configured to connect the client on port 8082.

The boot notification method is the first communication sent over the newly established websocket connection. This exchange communicates, in the form of messages, and provides details about the CP. This includes information about the CPs configuration and vendor. The CP receives necessary administrative information from the OCPP server to include acceptance or denial of the connection request. [19] Next, an Authorization request and the current meter values of the client are sent to the server. All of this takes place at the start of every connection. Once the initial communication and administrative exchanges are complete, the OCPP client sends a heartbeat to the OCPP server at intervals of 10 seconds during the entire connected period.

Additional commands from the client and server are communicated using curl commands entered through the terminal. Commands entered by the user are received through the aiohttp server and then transmitted to the OCPP Server/Client. From there they are transmitted over the websocket connection maintained between the client and server and executed accordingly. Outputs of these commands are displayed in the console window of the client or server. Table 3-2 depicts sample commands used by the client, sent to the server.

<table>
<thead>
<tr>
<th>Type</th>
<th>Command</th>
<th>Argument 1</th>
<th>Argument 2</th>
<th>Argument 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>curl <a href="http://127.0.0.1:8082/ask_for_update">http://127.0.0.1:8082/ask_for_update</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PUT</td>
<td>curl -X <a href="http://127.0.0.1:8082/security_event">http://127.0.0.1:8082/security_event</a></td>
<td>security_type</td>
<td>tech_info</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>curl <a href="http://127.0.0.1:8082/meter_values">http://127.0.0.1:8082/meter_values</a></td>
<td>id_tag</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3-2: Client to Server Command Examples

3.1.3 The Attack Machine

The Kali-Linux attack machine utilizes several open source and custom tools to perform tests against the OCPP client and server, depending on the attack. This research conducts three
different attacks against the OCPP system. The denial-of-service attack was attempted against two different configurations of the attack using “hping3” and “Metasploit.” The machine-in-the-middle attack utilizes a tool called “mitmproxy” [35] and “Wireshark.” [36] The log4shell attack is conducted utilizing “mitmproxy” toolset, an FTP server, LDAP server, and python HTTP server.

3.2 Challenges

Several challenges were encountered while trying to conduct this research. Most commercial systems configurations are proprietary property therefore are not available for testing. Several companies and research labs were contacted with requests for help with existing infrastructure for testing. None of them were able to offer the use of current systems. Available open-source systems had limited functionality and/or did not comply with OCPP v1.6. This required the building of a unique simulator that could replicate the environment required to explore the vulnerabilities of OCPP v1.6 utilizing json files.

The infrastructure to host the system was carefully thought out. The original build hosted the system in a cloud environment for ease of access and availability of resources. Amazon Web Service (AWS) was utilized to host the first configuration. The second configuration used virtual machines (VMs) hosted on a laptop.

The scope of embedded systems in EVSE created another issue. Without the electrical systems required to generate specific information, such as meter values, the information would need to be hard coded. While this is a great temporary solution, adding the embedded systems would generate a more realistic environment and is a potential option for future enhancements.
4 Attack Methodology

Three types of attacks are explored throughout this thesis. The first is the “Denial-of-Service attack (DoS).” The DoS is conducted in two different configurations. This was done to test the effectiveness of the attack in different environments. The remaining two attacks are conducted in the same VM environment with some specific additions between the second and third attack. The second attack is the “Machine-in-the-Middle” attack. The final attack is the Log4shell attack.

4.1 Denial of Service

The DoS attack is caused by the depletion of resources on the target machine or network, causing the system to be degraded or unavailable to legitimate users. “A denial-of-service condition is accomplished by flooding the targeted host or network with traffic until the target cannot respond or crashes, preventing access for legitimate users. DoS attacks can cost an organization both time and money while their resources and services are inaccessible.” [37] OCPP v1.6 is vulnerable to the DoS attack due to a lack of authentication required to connect to the machine. This means that any attempt to contact the server on the designated port is processed by the server whether the traffic is legitimate or not.

Three types of denial of service (DoS) attacks were explored during the course of this research; ICPM Flood Attacks, SYN Flood and Distributed Denial of Service (DDoS). [37] During a “ICMP Flood” the attacker sends Internet Control Message Protocol packets (ping requests) to the target computer usually using a spoofed IP address. [37] This usually done
through use of a piece of malware that sends the packets in an infinite loop draining resources from the target machine.

A “SYN Flood” attack is very similar to a “ICMP Flood Attack” however instead of ICMP packets, SYN requests are used to flood the network. This is effective due to a vulnerability created in the way the TCP three-way-handshake is accomplished. When a SYN request is sent to a server the server saves a record of the SYN request from the client and then sends a SYN-ACK packet to the client. At this point the connection is considered “half-open” while it waits for the ACK response from the client. [38] During this waiting period the connection continues to utilize finite resources. Additional SYN packets received require more resources, until all useable resources are expended. The outcome produces a degradation or complete denial of system access by a legitimate user.

The DDoS attack is very similar to the previous two attacks but on a larger scale. Unlike the previous two attacks, the DDoS attack utilizes more than one computer for the attack. Using botnets, attackers can generate large amounts of traffic against a target machine. [39]

The DoS attack was attempted in two different configurations. The first configuration is using an Amazon Web Service (AWS) host utilizing a Linux-Ubuntu distribution for the OCPP client, server, and attack machine. The second configuration is accomplished utilizing only VMs running through a type one hypervisor on the host machine. Several tools were utilized to conduct this test including “Metasploit,” “Netstat,” and “tcpdump.” All tools used were open-source programs available to the public and utilized without modifications.

The threat model of the DoS attack is very minimal in OCPP v1.6. For this attack, it is assumed that the attacker has access to the network communication channel the server and client
are communicating over. Once inside the network, no other authentication is required for connection to the client or the server.

The focus of this attack is to trigger an observable spike in network and/or CPU utilization. It is expected that an attack from a single machine will not cause enough resource depletion to crash the system. Therefore, observations in the forementioned data points are assumed to be multiplied in a DDoS situation and ultimately cause the desired outcome of a successful Denial-of-Service attack.

4.1.1 Configuration One – AWS

This configuration consists of three AWS machines. The cloud-based AWS EC2 instances, client and server, consisted of a single core 64-bit processor using 1GB of RAM and 8GB of storage running ubuntu though terminal-based interaction only. The attack Platform consisted of a Kali-Linux box with 4GB of Ram and 30GB of storage on a x86 64-bit machine with a single core and sustained clock speed of 2.5ghz

Initial attacks were targeted toward the server on port 9000 which was the designated port for server listening. After those attacks showed no obvious signs of degradation, attacks were created against the machine without specifying a specific port. Still no noticeable degradation of traffic was observable from the server. The presence of flood traffic was confirmed using “tcpdump” [40] on both the server and on the attack platform through monitoring of the eth0 interface. Multiple instances, iterations, and configurations of the attack, failed to produce the desired outcome. No noticeable degradation of performance was apparent in the server or the network.
After further research it was determined that AWS is equipped with built-in safeguards against DoS/DDoS attacks called AWS Shield, along with other cybersecurity countermeasures that were not anticipated. AWS Shield provides managed protection against DDoS. [41] While not significant to the testing results, it shows strong support for the use of cloud-based hosting of the OCPP Client and OCPP Server [2] due to the built in countermeasures. This leads to the recommendation that companies who lack their own robust cybersecurity infrastructure should consider the use of cloud hosted CSMS systems to capitalize on the organic protection mechanisms that are inherent in services like AWS.

4.1.2 Configuration Two – Pure VM

The second configuration uses a VM based client, server, and attack machine, all hosted on the same laptop. Both the client and server are Ubuntu distributions of Linux with the attack machine running Kali Linux. The client and the server are equipped with two CPU cores each. The client is equipped with 4GB of RAM. The server is equipped with 2 GB of RAM. The

![Client Boot Sequence](image)

*Figure 4.1-1: Client Boot Sequence.*
Storage capacity of the client and server is 30GB each, dynamically allocated. They are running on a virtual network card.

The attack setup is orchestrated toward the server. At the start of the attack the client and server are connected and communicating. Figure 4.1-1 shows the traffic of messages during the connection process between the client and the server. Continued steady state shows the traffic of the heartbeat message exchange at an interval of 10 seconds between transmissions.

Figure 4.1-2 depicts system utilization to include the CPU and network traffic flow during normal traffic. Due to the sterile environment of the server, only one client attached, all resources utilization prior to the attack is low. Over the course of 30 seconds, it can be observed that CPU utilization averages approximately 15-20% utilization per CPU. Network traffic registers approximately 200 byte/s.

Figure 4.1-3 Shows a snapshot of the “tcpdump” capture on the client machine.
Figure 4.1-3 depicts a “tcpdump” capture of the OCPP Server machine’s network traffic. The program monitors port 9000 on the server machine for traffic. It can be observed that the server is receiving one packet approximately every 10 seconds depicted in the time interval between packets. This packet represents the heartbeat packet between the client and server. The first attack scenario utilized an ICMP flood attack. The attack configuration was set to a packet size of 300 bytes sending 100 packets. The network traffic showed a slight increase during this attack. However, CPU utilization did not show any significant up-tick in usage.

![Figure 4.1-4 DoS Attack Command and Configuration](image)

The second attack scenario is configured to conduct a SYN Flood attack using “hping3” [42] against port 9000 on the server. “hping3 is a network tool able to send custom ICMP/UDP/TCP packets and to display target replies like ping does with ICMP replies. “hping3” can be used to test firewall rules, perform (spoofed) port scanning, test network performance using different protocols, and more” [43]

Figure 4.1-4 shows the command and configuration of the attack from the Kali Linux machine against the OCPP server. Figure 4.1-5 depicts a significant spike in both the system resource utilization and network traffic.

![Figure 4.1-5 Attack System Utilization](image)
During normal communications between the server and one client, the server utilized less than 5% of its CPU capacity per CPU and processed the receipt of 100-200 bytes of network traffic per second. During the attack, CPU one was at a steady state of 85% utilization with CPU 2 fluctuating between 30-40% utilization. Even in this controlled environment, one virtual machine created a significant degradation of service in the OCPP server. Doubling the attack by implementing a second attack machine could reasonably be expected to deplete the remainder of the server’s available resources and crash the machine or render it unavailable to legitimate users.

The lack of required authentication to connect to the server causes a situation where every attempted connection is handled by the server. This allows for the attack to flood the server with half open illegitimate connections which quickly diminish the ability of the server to process the requests. It should be noted that at termination of the attack the server returns to regular resource utilization indications.

4.2 Machine-In-The-Middle

The machine-in-the-middle (MITM) attack “is a type of eavesdropping attack, where attackers interrupt an existing conversation or data transfer.” [44] It is caused by an interception of traffic by a third-party during transmission between a transmitter and receiver, and then retransmission of that information to the legitimate receiver. During that interception, information can be monitored, recorded, or modified before broadcast from the MITM attacker to the original intended receiver. Figure 4.2-1 represents normal flow vs MITM flow of traffic between two parties and the presences of a MITM attacker.
The intent of the attacker is to remain in this advantageous position without the knowledge of the legitimate parties. For this scenario, the purpose of the attack is to wait for the client or server to initiate a “firmware update” request at which time the attacker will detect and modify the TCP packet associated with the “firmware update” file “location” and “filename” prior to relaying it to the client machine. The modified packet will cause the client to seek a “firmware update” from the attacker’s File Transfer Protocol (FTP) server instead of the CSMS’ FTP server, causing the client to download a malicious file from the attacker.

This attack comes with several assumptions for the purposes of this research. First, it is assumed that the attackers have access to the subnet that communications are taking place on. Second, it is assumed that prior to this attack, the malicious party conducted a successful Address Resolution Protocol (ARP) cache poisoning attack against the router. It is also assumed that the attackers are familiar with the OCPP v1.6 JSON files and commands, which are readily available on the OCA website.

ARP is a protocol that enables network communications to reach a specific device on a network. This is done by associating the IP address to the Media Control Access (MAC) address of the device’s network card. “This association allows the router to relay traffic from one device to another using the IP address. The poisoning attack takes place when the attacker uses a spoofing tool to send out forged ARP responses which are received by the router.” [45] This
causes the router to update its table to associate the victim’s IP address with the attacker’s MAC address. [46] When traffic is sent to the victim’s IP address, it is routed to the attacker’s machine due to the MAC address in the routing table associating the attacker with the victim’s IP address. This is an over-simplified explanation of this attack for informational purposes. ARP Spoofing can be successfully accomplished using the “Ettercap” tool. “Ettercap is a comprehensive suite for man in the middle attacks. It features sniffing of live connections, content filtering on the fly and many other interesting tricks. It supports active and passive dissection of many protocols and includes many features for network and host analysis.” [47]

To simulate the ARP cache poisoning attack, the IP address and port number of the attack machine are hard coded in the client’s source code. With the ARP spoofing complete, the MITM begins at the start of the connection between the client and the server. The client sends its credentialing information and awaits confirmation from the server of an authorized connection. (Figure: 4.2-1) This request is sent to the attacker by the client, who believes it is sending the request to the OCPP server. The MITM machine receives the request and simply sends it forward to the OCPP server (Figure: 4.2-2). The OCPP server, believing it is receiving a request from the OCPP client, processes the request and sends a confirmed connection response back to the MITM machine who forwards it to the OCPP client. Once complete, the server and the client begin to pass heartbeat confirmation messages every 10 seconds and await further instructions. Figure 4.2-3 depicts the traffic being received by the client from the MITM and sent by the server.
Notice all communication in Figure 4.2-1 and Figure 4.2-3 look legitimate from the view of the console. Of note, is the difference in the IP address which is not the standard server IP. This is due to the hard coding of the attack machine IP to bypass the need for an ARP spoofing attack prior to the experiment. With the completion of a successful ARP spoofing attack there would be no visible indication of the attack taking place.

At this point the MITM is in a passive mode and only monitoring and relaying traffic it receives. During its monitoring the MITM scans all traffic searching for key words in the websocket TCP packets, that will cause it to advance its attack to the next steps. Using “mitmdump” the attacker creates an “packet-injection” file to tell the program what words to search for and what to do when those words are detected. When the MITM

Figure 4.2-2: MITIM Traffic Between Client/Server.

Figure 4.2-3 Server Traffic From MITM.
detects a packet with the words “firmware update” sent from the server to the client, it will modify the “location” and “filename” prior to sending it back to the client. Table 4-1 shows the IP address for the client, MITM, and server.

<table>
<thead>
<tr>
<th>System</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCPP Client</td>
<td>192.168.50.220</td>
</tr>
<tr>
<td>OCPP Server</td>
<td>192.168.50.151</td>
</tr>
<tr>
<td>MITM Machine</td>
<td>192.168.50.63</td>
</tr>
</tbody>
</table>

*Table 4-1 Cyber Range IP Table*

To start the MITM attack the client initiates a request to the server for a “firmware update.” This is done utilizing a “curl” request through the terminal window, which is transmitted to the OCPP client who sends an http request through the aiohttp server on the OCPP client. (Figure: 4.2-4) The OCCP server’s aiohttp server receives the request and sends it to the OCPP server. The OCCP server sends the “initiate firmware update” message to the OCPP client via websocket.

The received HTTP request from the client can be seen in the server terminal window (Figure 4.2-5). The highlighted portion shows the request and who it was received from. The server processes this request and returns a packet instructing the client to “initiate a firmware update,” this tells the client to download the update configuration file at IP address “192.168.50.151” where it will retrieve a file called “firmware_update.conf.”
The packet with these instructions is intercepted enroute by the MITM machine (Figure 4.2-6). The MITM scans the packet and finds the instructions to “update_Firmware” with a location and filename. The MITM modifies the IP address of the FTP server and the name of the file, and then sends the modification to the client. The modified packet is received by the client who will attempt to connect to the OCPP server at the malicious FTP server’s IP address (Figure 4.2-7).

The lack of authentication or certificate verification between the client and server communications in OCPP v1.6 makes this type of attack possible. OCPP v1.6 relies on security from the underlying method of communication to provide security for server/client communications. [44] This flaw in the protocol must be corrected in the OCPP protocol. Many companies have chosen mitigation measures with VPN or cellular connection from the client-proxy to the central server however this does not protect against insider threats with access to the communications.
link. “It should be noted that newer versions of OCPP 1.6j and 2.0.1 provide certificate verification of certain requests.” [48]

The second piece to this vulnerability is in how the client processes the “initiate firmware update” instruction. When the instruction is received the client receives two pieces of information from the server. One is the IP address and filename of the update, which the MITM attack machine changed to the IP address of the attack machine and the filename to “malicious.conf” and the other, which is a response payload (data structure) that has the accurate information from the server for the update. The client never looks at the response payload. It just sends it back to the server to confirm receipt of the information. The vulnerability comes when the client does not compare the contents of the two packets. The OCPP protocol, being a request/response protocol, only takes the response packet and sends it back to the OCPP server to confirm receipt of the information. Upgrading current systems utilizing OCPP1.6 would be prudent to prevent this attack.

4.3 Log4Shell

The final attack explored in this Thesis is the log4shell attack, also known as CVE 2021-44228, which utilizes the log4j exploit discovered in 2021. [49] This exploit allows the attacker to gain root access to the victim machine using a combination of the MITM and log4shell vulnerabilities. OCPP Is vulnerable due to a lack of information verification prior to initiating actions to retrieve remote files. This attack introduces several components to the OCPP Cyber Range that have not previously been discussed including Log4j, Java Naming and Directory Interface (JNDI), and the Lightweight Directory Access Protocol (LDAP).
Most systems maintain a way to log system events including connection error messages, software updates, system events, and more. One common and highly used method to do this is the Log4j framework. Log4j is an open-source Java-based logging framework that collects and manages information about system activity. [50] It is used in many computer systems and servers to maintain system activity information such as network connection errors, connection events, and system update events. Due to being readily available and free, software developers integrate it into many different architectures during their build. “In 2013 a feature was added to the Log4j library called Java Naming and Directory Interface (JNDI). The purpose of this feature was to help with data storage and retrieval.” [50]

“JNDI is an application programming interface (API) that provides the naming and directory functionality to applications written using Java.” [51] The purpose of JNDI is to lookup information such as log messages. Additionally, JNDI is capable of remote lookups on external servers such as an LDAP server. [52] “The architecture of JNDI consists of an API and service provider interface (SPI). The SPI enables a variety of naming and directory services to be plugged in transparently, thereby allowing the Java application using the JNDI API to access their services. JNDI is included in the Java SE Platform. To use JNDI, the system needs to have one or more service providers. The Java Development Kit (JDK) includes service providers for LDAP and the Domain Name Service (DNS) protocol,” to name a few. [51]

LDAP is an open, vendor-neutral application protocol for accessing and maintaining data such as usernames, passwords, printer connections, and other static data within directories. [53] The LDAP server’s primary role is to enable users to find data about organizations, persons, etc... It allows information for multiple databases to be stored in one central location. The JNDI API is used to access this server and retrieve the required information from the LDAP server.
LDAP works with most vendor directory services such as Active Directory (AD). [54] It should be noted that LDAP servers require authentication prior to accessing the database. [54] For the purpose of this attack the LDAP server in the OCPP Cyber Range will be hosted on the attacker machine, however it would be present in the legitimate environment under normal circumstances also.

The FTP Server, previously mentioned in the MITM attack, facilitates the ability of the client to download the required files from the CSMS. In normal cases the client would retrieve the firmware update file from this FTP server. In this scenario, the attacker is also hosting an FTP server, on which the malicious files are stored. It should be noted that all servers on the attack machine are malicious versions of servers found on the legitimate management server and required for the functionality of the charging management command and control.

In 2021 The National Institute of Standards and Technology (NIST) who manages the Common Vulnerabilities and Exposure (CVE) database, released CVE 2021-44228 (Log4Shell)
which alerted administrators to a critical vulnerability in the use of Log4j. [49] The vulnerability was found in the JNDI features of Log4j. The alert received a “Base Score” of 10/10 which is reserved for only the most critical vulnerabilities. NIST warned that “JNDI features used in configuration, log messages, and parameters do not protect against attacker-controlled LDAP and other JNDI related endpoints. CVE 2021-44228 states, “if an attacker can control the log messages or log message parameters, they can execute arbitrary code on the victim machine, loaded from LDAP servers, when message lookup substitution is enabled.” [49] In a normal situation information pulled from the LDAP server is printed in the Log4j log. However, due to a flaw in the JDNI feature of Log4j, the information is accessed and executed instead of just printed to the log.

This attack will be described in three phases. Figure 4.3-1 shows the overall traffic flow throughout the attack. The first phase is almost identical to the previous MITIM attack. Figure 4.3-2 depicts steps 1-5 of the Log4Shell attack starting with a request for “firmware update”. Instead of being initiated by a “curl” request through the aiohttp server, it is initiated by the OCPP client Log4j extension. Phase one starts with the request to update firmware sent from the Log4j extension of the OCPP client to the OCPP client aiohttp server. Phase one ends when the Log4j extension receives the command to “initiate firmware update” from the OCPP client, along with the location and filename of the new firmware file from the OCPP server. To start, Log4j sends an http request to the OCPP client’s aiohttp server on port 8082 (step 1). The aiohttp server on the OCPP client sends an http request to the aiohttp server on the OCPP Server for a “firmware Update.” (step 2) The aiohttp server in the OCPP Server forwards this request to the OCPP server (step 3) who sends the “initiate firmware update” command to the OCPP client via a websocket connection (step 4). This command is accompanied by a location
(192.168.50.151:21) which the OCPP client will use to retrieve the firmware and the filename (firmware_update.conf) from the OCPP server’s FTP server.

The websocket connection between the OCPP client and the OCPP server is where the MITM attack machine monitors communication. It is important to note that the “mitmdump,” in the current configuration, is only capable of monitoring websocket connections. At this point the MITM is scanning each transmission of the websocket connection for the message “Firmware Update.” When it finds this message, it modifies the transmission “location” and “filename” to those of the attack machine’s FTP server and filename (192.168.50.63, “malicious.conf.”) It then relays the modified transmission to the OCPP client. This completes phase one of the attack.

Phase two begins with the request for the “malicious.conf” file from the FTP server on the attack machine (step 5). It ends when Log4j receives the exploit.class from the attack

![Figure 4.3-2: Phase 1 Traffic Flow][56]

![Figure 4.3-3: Contents of the “malicious.conf” file][56]
machine (step 10). Figure 4.3-4 depicts phase two traffic flow. The OCPP Log4j extension sends a file request for the “malicious.conf” file, received in phase one, to the FTP server located on the attack machine (step 5). The FTP server returns the requested “malicious.conf” file to the Log4j extension of the OCPP client (step 6). The Log4j extension processes the malicious.conf file which tells it to contact the LDAP server and retrieve the “Exploit.class” file. However, due to the flaw the Log4j library, instead of printing it to the log it processes it as a command and attempts to execute it. Figure 4.3-3 shows a screen shot of the “malicious.conf” file. The “jndi:ldap” at the beginning of the line tells the system to use the JNDI feature of Log4j when contacting the LDAP server. At the end of the line “/Exploit” is printed immediately following the port number. This line tells the Log4j to contact the LDAP server and request the “Exploit.class” from it. (step7)

Figure 4.3-4: Phase 2 Traffic Flow [56]

The LDAP server, being only able to receive requests, sends the “Exploit.class” to the python http server on the attack machine (step 8). The http server sends the “Exploit.class back
to the aiohttp server on the OCPP client machine (step 9). The aiohttp server on the OCPP client sends the http response to the Log4j with “Exploit.class” which ends phase two of this attack.

Phase three of this attack starts when the Log4j processes the “Exploit.class” and ends with the attacker obtaining root access to the client machine. Figure 4.3-5 depicts the traffic flow (steps 10-12) in phase three of the attack. When the system attempts to log the class, the flaw in the Log4j causes it to execute the “Exploit.class” as a command (step 10) which initiates a NetCat utility, as root, on port 4444 on the OCPP client machine (step 11). “NetCat is a utility tool that uses TCP and UDP connections to read and write to a network. NetCat can be used to create a backdoor that an attacker can return to at any time.” [55] At this point the attacker opens a NetCat session on the attack machine and requests to contact the OCPP client on port 4444. The OCPP client responds with an acceptance of the request and the resulting session gives root access to the attack machine on the OCPP client system, ending phase three and completing the attack.
5 Discussion

The main intent of this project was to provide a training platform for operators and cybersecurity researchers to learn and test future vulnerabilities to prevent cybersecurity attacks on EV infrastructure. This platform can achieve both goals. It is modular enough to be modified to the needs of the researcher and lightweight enough to be deployed on almost any infrastructure. With slight configuration, significant feature additions are available to researchers.

The OCPP Cyber Range brings with it countless possibilities for testing OCPP v1.6 and OCPP v1.6j. While some of these attacks have established mitigation techniques, the ability to conduct them provides a baseline for further research into vulnerabilities in the protocol. The selected attacks pave a way for future research to explore other aspects of follow-on attacks. Denial of service attacks are still prevalent in every aspect of networked infrastructure. The machine-in-the-middle attacks enable countless attack vectors for malicious actors. The Log4shell’s ability to produce a root shell opens countless possibilities for future attack vectors against EV clients and servers. This system is not meant to be the answer to all attack vectors but rather provide a readily available platform for future researchers to start their research. This baseline can drastically reduce their learning and preparation curve prior to deep diving attacks.

This system is still in its infancy. Sandia National Laboratories will utilize this system going forward in multiple other aspects of research such as the plug-in-play certificate-based authentication and billing methods. This allows users to plug their vehicle in without having to authenticate at the control console or pay for power using credit cards or RFID cards. This system will provide the foundation for testing of future cybersecurity research in this area which
will enable researchers to discover and stay ahead of malicious actors who are searching for attack vectors in these new developments.
6 Conclusion

This paper presented a system called the OCPP Cyber Range which was designed for training EV management system operators and provided a testing range for cybersecurity research. Three attacks were presented to validate the proof of concept, denial of service, machine-in-the-middle, and Log4shell. All these attacks have proved to cause significant degradation, denial, and disruption of IT infrastructure.

EVs are becoming more widespread and intertwined in day to day logistical and transportation requirements across the world. It is paramount that research is done to identify vulnerabilities early and training is provided to the operators and maintainers of these systems. This system offers a steppingstone that is available to conduct future research in the field of EVSE infrastructure.
References

[1] “US electric vehicle charging market growth: PwC.”


