

DataTransfer PDU-based Rollback Mechanism for Securing OCPP 1.6 Against Spoofing Attacks

Dong Uk Kim

Department of Computer Science and Engineering
Kyung Hee University, 446-701
 Republic of Korea
 g9896@khu.ac.kr

Keon Oh Kim

Department of Computer Science and Engineering
Kyung Hee University, 446-701
 Republic of Korea
 keonoh@khu.ac.kr

Sun-Moo Kang

Department of Computer Science and Engineering
Kyung Hee University, 446-701
 Republic of Korea
 extkang@khu.ac.kr

Choong Seon Hong

Department of Computer Science and Engineering
Kyung Hee University, 446-701
 Republic of Korea
 cshong@khu.ac.kr

Abstract—The Open Charge Point Protocol is a widely adopted communication standard that facilitates interoperability between Charge Points and Central Systems in the electric vehicle charging ecosystem. This paper explores the security implications of OCPP 1.6, with a particular focus on the risks posed by spoofed or unauthorized messages. We propose a rollback mechanism utilizing the DataTransfer PDU to mitigate these risks, allowing the Central System to trigger reversion of unauthorized changes at the CP level. The proposed solution ensures secure, efficient management of configurations and mitigates potential threats such as replay attacks, spoofed requests, and unauthorized modifications. This paper contributes to the growing body of knowledge on securing EV charging infrastructure by highlighting underexplored vulnerabilities in OCPP 1.6 and providing practical solutions to enhance the security and resilience of the protocol.

Index Terms—OCPP 1.6, Rollback Mechanism, Spoofing Attacks, EV Charging Security

I. INTRODUCTION

As the adoption of electric vehicles (EVs) accelerates globally, the need for secure and reliable charging infrastructure has become increasingly paramount [1]. A critical component of this infrastructure is the Open Charge Point Protocol (OCPP), which facilitates communication between Electric Vehicle Supply Equipment (EVSE) and Central Systems (CS) to manage charging sessions, monitor status, and perform remote operations [2]. OCPP version 1.6, one of the most widely used iterations of the protocol, introduces several key functionalities such as reservation systems, smart charging,

and offline operations, allowing for efficient management of charging stations. However, the growing reliance on OCPP 1.6 has also raised concerns about its security, particularly in the face of evolving cyber threats [3], [4].

Much research has been done to address vulnerabilities in areas such as websocket communication and JSON data processing, issues commonly associated with man-in-the-middle (MitM) [5], Distributed denial-of-service (D-DoS) [6], Packet Relay [7], [8] or Malware [9], [10] exploits. [3], [4].

In addition to managing reservations and configurations, the CS can issue various commands to the Charge Point (CP) such as CancelReservation, ChangeAvailability, ChangeConfiguration, and ClearCache. These commands are essential for the smooth operation of the charging infrastructure. However, unsolicited or spoofed confirmation messages sent in response to these commands can result in unauthorized changes, system misconfigurations, and disruption of operations. For instance, when a legitimate CancelReservation.req is sent, the CP processes the request and returns a confirmation (CancelReservation.conf). However, if an attacker sends a spoofed confirmation message for an invalid reservation ID, it could delete reservations without authorization. Similarly, commands like ChangeAvailability and ChangeConfiguration can be misused to alter the operational status or configuration settings of a CP, potentially rendering it unavailable or causing misconfigurations. [2], [11]

This paper addresses security gaps by analyzing vulnerabilities in OCPP 1.6, specifically those related to configuration changes and unintended messages. A new rollback mechanism is proposed to detect and reverse unauthorized operations when a central system receives a confirmation (conf) message without a corresponding request. This mechanism adds a layer of defense against spoofing attacks, ensuring the integrity of command execution and protecting the EV charging infrastructure from malicious tampering.

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the Convergence security core talent training business support program(IITP-2023(2023)- RS-2023-00266615), (No.RS-2024-00398993) supervised by the IITP(Institute for Information and Communications Technology Planning and Evaluation), and supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2023-00207816). *Dr. CS Hong is the corresponding author.

II. OVERVIEW OF OCPP 1.6

The OCPP is a widely adopted open communication protocol designed to facilitate seamless interaction between CPs and CS, playing a critical role in the electric vehicle charging ecosystem. OCPP enables interoperability between charging stations from different manufacturers and management systems, making it a cornerstone of the rapidly growing EV infrastructure.

OCPP 1.6 introduces several key features and improvements to meet the evolving demands of the EV market. One of the major enhancements is the introduction of Smart Charging capabilities, which allow for more efficient energy management in charging EVs. Smart Charging supports functions such as load balancing and dynamic energy allocation, which are essential for optimizing grid resources and ensuring a stable and efficient charging experience.

Additionally, OCPP 1.6 supports communication via JSON over WebSockets, which provides a modern, lightweight alternative for transmitting data between CPs and the Central System. This method allows for faster and more efficient communication, improving overall system performance and reliability.

OCPP 1.6 also enhances diagnostic capabilities, introducing features like enhanced Charge Point Statuses and the TriggerMessage function. The TriggerMessage allows the Central System to dynamically request specific information from the Charge Point, offering better visibility into the CP's operational state and facilitating more effective monitoring and troubleshooting.

A critical feature in OCPP 1.6 is the DataTransfer Protocol Data Unit(PDU), which allows for the exchange of vendor-specific or custom data that is not covered by standard protocol messages. This flexibility is especially useful for implementing custom functionalities, such as firmware updates, operational adjustments, or triggering a rollback mechanism when unauthorized or unintended changes occur in a CP's configuration. By utilizing the DataTransfer PDU, the Central System can initiate these custom commands while adhering to the OCPP framework, ensuring secure and flexible interactions between the CPs and CS.

The DataTransfer PDU [2] offers a powerful tool for ensuring advanced and secure operations within the OCPP ecosystem. The messages of these data transfers are defined in the following table:

TABLE I
DATATRANSFER.REQ

Data Field	Field Type	Card
Vendor ID	CiString255Type	1..1
Message ID	CiString50Type	1..1
Data	Text (e.g., "Our Proposed Message")	0..1

The following describes each data field:

- **Vendor ID:** This field identifies the vendor of the system that is sending the message. It ensures that the CP knows

which specific vendor implementation is initiating the `DataTransfer.req`.

- **Message ID:** The `Message ID` field provides an identifier for the specific custom message being sent. This field helps the CP recognize the type of operation or instruction being carried out.
- **Data:** This field contains any additional information or payload associated with the custom message.

III. THREAT MODEL

As the EV charging ecosystem expands, securing its infrastructure, especially communication protocols such as OCPP 1.6, has become critical. One emerging risk involves unsolicited or spoofed confirmation (conf) messages sent to charge points, potentially leading to unauthorized or malicious configuration changes. This section outlines a comprehensive threat model for a rollback mechanism designed to address such risks by detecting and reversing unauthorized actions.

A. Attacker Capability

In this paper, we assume that the attacker has gained sufficient knowledge of the system through reconnaissance and analysis to create and inject a spoofed message. As a result, the attacker gains the ability to intercept and analyze communications between the CS and the CP. Using this information, the attacker can impersonate the Central System by spoofing its credentials and mimicking its behavior. The attacker then crafts and sends unauthorized req messages (such as `ChangeConfiguration.req`, `ClearCache.req`, or `SetChargingProfile.req`) to the CP, tricking it into executing malicious operations.

B. Threat Scope

The scope of this work focuses specifically on preventing and mitigating damage caused by spoofed messages, where an attacker impersonates the CS and sends unauthorized requests to the CP. Our approach aims to detect these fraudulent messages and prevent the CP from executing unintended actions such as changing configurations or clearing caches. It is important to note that this study does not address the reconnaissance and analysis activities that an attacker may undertake prior to the attack. We assume that the attacker has already acquired the necessary information to craft spoofed messages, and we focus solely on detecting and mitigating those malicious messages once they are sent. Additionally, we do not cover the secondary effects or broader consequences of the attack, such as potential service disruptions, grid instability, or other impacts that may result from the successful execution of spoofed messages. The primary objective is to address the immediate threat posed by the spoofed messages themselves.

IV. PROPOSED ROLLBACK MECHANISM

The proposed rollback mechanism leverages the DataTransfer PDU in OCPP 1.6 to detect and revert unauthorized changes made to a CP by spoofed or unsolicited requests. The CS uses this mechanism to ensure that configuration

changes, such as ChangeConfiguration.req or ClearCache.req, are only executed if they originate from a valid request. If a suspicious or unsolicited conf message is detected, the rollback mechanism can trigger a corrective action through a DataTransfer request, instructing the CP to revert to its previous safe state.

The core objective is to maintain the integrity of the CP by verifying that only authorized configuration changes are applied and to undo any changes resulting from fake or replayed messages [12], [13].

A. Mechanism Implementation

The following figure is a message diagram that presents an overview of our proposed rollback mechanism:

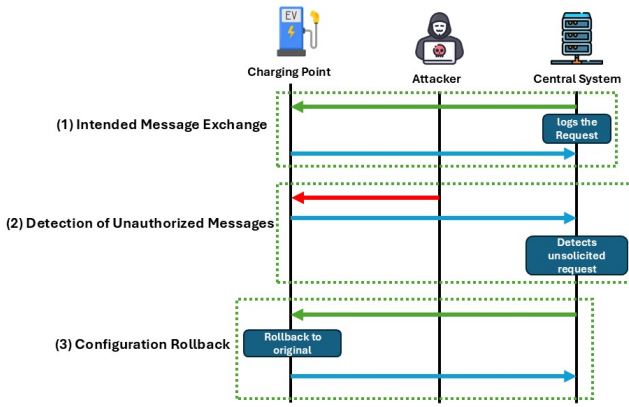


Fig. 1. Message Diagram of Proposed Mechanism

1) *Message Tracking:* The CS maintains a request log for every outgoing req message, including details such as the message ID, type (e.g., ChangeConfiguration.req), timestamp, and expected response type. This log is essential for matching incoming conf messages to corresponding requests.

2) *Verification and Detection:* Upon receiving a conf message, the CS checks the request log:

- If matching request exists: The CS verifies that the conf message is legitimate and processes it accordingly.
- If no matching request exists: The conf message is treated as unsolicited or potentially spoofed. In this case, the rollback mechanism is triggered.

3) *Triggering the Rollback:* When an unsolicited or suspicious conf message is detected, the CS sends a DataTransfer.req message to the CP, instructing it to roll back the unauthorized changes. This DataTransfer message contains details about the original configuration, instructing the CP to revert to its previous state before the unauthorized changes occurred.

The rollback message, implemented within the data field of the data transfer message, is organized as follow Table: The

TABLE II
ROLLBACK MESSAGE STRUCTURE WITHIN DATATRANSFER.REQ

Data Field	Field Type	Card
RollbackType	CiString50Type	1..1
Timestamp	DateTimeType	0..1

following describes each data field:

- **RollbackType:** This field specifies the type of control message that needs to be rolled back. Examples include CancelReservation, ChangeConfiguration, SetChargingProfile, or ClearCache. This ensures that the CP identifies the specific operation to be reverted.
- **Timestamp:** This field indicates when the original control message was applied. It helps the CP target the exact instance of the change, ensuring that only the specific action made at that time is rolled back, especially useful when similar changes have occurred at different times.

The CP receives this DataTransfer request and executes the rollback action, restoring its configuration to the safe state. The behavior of the proposed rollback mechanism is summarized below in Algorithms 1 and 2, which detail the process for both CS and CP:

Algorithm 1 CS Side - Message Verification and Rollback Algorithm

```

1: Initialize CS request log  $L_{CS}$  for outgoing req messages
2: while WebSocket session is online do
3:   Wait for outgoing req message  $m_{req}$  to be sent to CP
4:   Log the request in  $L_{CS}$ :  $\{messageID, type, timestamp\}$ 
5:   Wait for incoming conf message  $m_{conf}$  from CP
6:   Extract  $messageID$  from  $m_{conf}$ 
7:   if  $messageID$  exists in  $L_{CS}$  then
8:     Verify  $m_{conf}$  and process it
9:   else
10:    Treat  $m_{conf}$  as unsolicited or spoofed
11:    Trigger rollback mechanism to revert unauthorized changes
12:   end if
13: end while

```

V. EXPERIMENT

A. Experiment Setup

The primary goal of this experiment is to evaluate the performance and effectiveness of the proposed rollback mechanism in mitigating unauthorized or spoofed control messages within an OCPP 1.6 implementation. For this purpose, we utilized the mobilityhouse OCPP 1.6 implementation in Python [14], available on GitHub, to simulate both the CP and the CS components. The experiment was conducted in a controlled environment, where the CP was implemented on a Raspberry Pi running a Linux-based system, and the CS was implemented

Algorithm 2 CP Side - Message Tracking and Rollback Detection Algorithm

```

1: Initialize CP message log  $L_{CP}$  to store incoming requests
2: while WebSocket session is online do
3:   Wait for incoming req message  $m_{req}$  from CS
4:   Extract  $messageID$ ,  $type$ ,  $timestamp$  from  $m_{req}$ 
5:   Log the request in  $L_{CP}$ :  $\{messageID, type, timestamp\}$ 
6:   Process the req message according to its type (e.g., ChangeConfiguration)
7:   Send conf message back to CS with corresponding  $messageID$ 
8: end while

```

on a Windows-based server. The actual hardware configuration of the lab is shown in Figure 2. The architecture of the

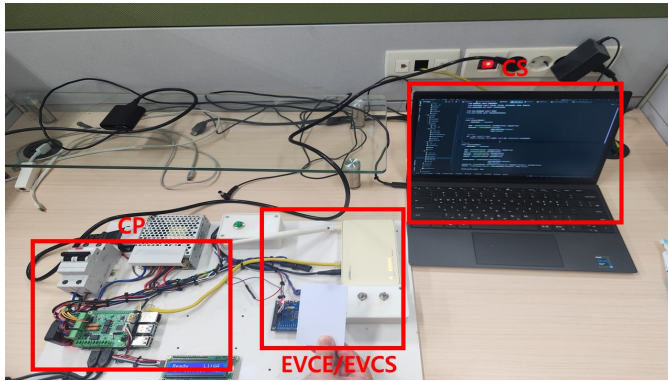


Fig. 2. Experiment Setup

test environment consisted of a Central System capable of managing multiple Charge Points. In this experiment, we focused on simulating the behavior of one CP interacting with the CS. The CP was configured to process requests such as ChangeConfiguration, SetChargingProfile, and ClearCache, while the CS was designed to manage these operations and, more importantly, detect unauthorized changes and trigger a rollback via DataTransfer.req.

B. Attack Scenario

To evaluate the proposed rollback mechanism, we designed a scenario in which a CP receives both a legitimate control message and a spoofed control message. In this modified attack scenario, we assume that the attacker has gained sufficient knowledge of the system through reconnaissance and analysis to create and inject a spoofed message. First, the CS sends a legitimate request to the CP to delete the reservation with reservation ID 1500. This is handled correctly and the reservation is removed as expected. Next, the attacker injects a spoofed message to delete reservation ID 2500 (Victim). However, with the proposed rollback mechanism, the system detects the unauthorized action and automatically recreates the reservation with ID 2500, reversing the attacker's action.

C. Results and Observations

As shown in Fig.3, the behavior of the system during the experiment is clearly illustrated across the CP, CS, and attacker components. The logs on the CP side show that a legitimate cancellation request for reservation ID 1500 is successfully processed, and the CP returns an 'Accepted' status. Shortly thereafter, the attacker sends a spoofed cancel reservation message for reservation ID 2500, which is also processed by the CP and marked as "Accepted", highlighting the vulnerability to spoofing attacks. The CS identifies the CP's unsolicited confirmation of the deletion of reservation ID 2500 as an unintended action. In response, the CS triggers the rollback mechanism by sending a DataTransfer.req message instructing the CP to reverse the action. The CP receives the rollback request and successfully restores the deleted reservation ID 2500, as shown in the Rollback:CancelReservation log. The interaction log shown in Fig.3 shows the process at the command prompt where the spoofed message is detected and reverted through the rollback mechanism.

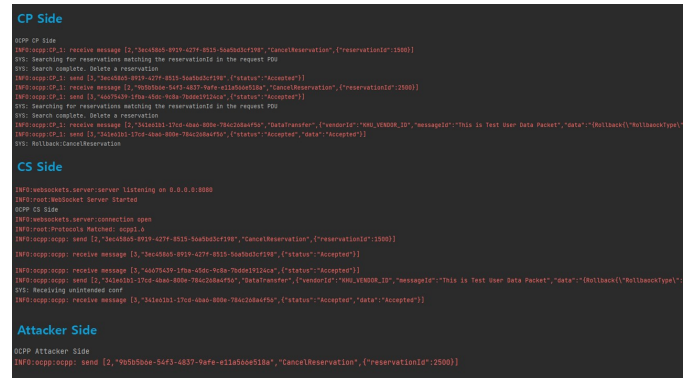


Fig. 3. Experiment Results

VI. LIMITATION AND FUTURE WORK

The introduction of OCPP 1.6 has brought significant advancements to the EV charging ecosystem, including Smart Charging, JSON-based communication, and the DataTransfer PDU for custom data exchange. However, these enhancements come with new security challenges, particularly the risk of unauthorized or spoofed messages that could lead to misconfigurations at CP. This paper has proposed a rollback mechanism utilizing the DataTransfer.req to address such vulnerabilities, allowing the CS to efficiently revert unauthorized configuration changes at the CP level.

Despite its potential to enhance security, the proposed mechanism comes with certain limitations. Rollback delay, caused by message processing time, can result in temporary misconfigurations, affecting CP operations until the rollback is executed. Moreover, for large-scale deployments, maintaining request logs and handling multiple rollback requests efficiently could pose scalability challenges. Additionally, the mechanism's effectiveness relies on secure communication channels between the CS and CP. If the communication is compromised,

rollback requests themselves could be spoofed, undermining the security of the system.

Looking ahead, several improvements can be made to strengthen the proposed solution. Automated detection of attack patterns through machine learning or anomaly detection can enhance the mechanism's responsiveness and accuracy, reducing the time between detecting unauthorized actions and initiating rollbacks. Additionally, implementing a more sophisticated logging mechanism that uses cryptographic checksums can ensure the integrity of the request logs and further prevent spoofing or tampering with rollback commands.

VII. CONCLUSION

In conclusion, while the rollback mechanism provides an effective solution to certain security vulnerabilities in OCPP 1.6, ongoing improvements and vigilance are necessary to adapt to evolving threats in the EV charging infrastructure. By addressing these limitations and incorporating future enhancements, the security and reliability of OCPP 1.6 can be further strengthened, ensuring the safe operation of EV charging networks.

REFERENCES

- [1] S. Hemavathi and A. Shinisha, "A study on trends and developments in electric vehicle charging technologies," *Journal of energy storage*, vol. 52, p. 105013, 2022.
- [2] Open Charge Alliance, "Open Charge Point Protocol, Protocol Spec. Version 1.6," <https://www.openchargealliance.org/downloads/>, 2019, accessed: 2024-09-30.
- [3] Z. Garofalaki, D. Kosmanos, S. Moschogiannis, D. Kallergis, and C. Douligeris, "Electric vehicle charging: A survey on the security issues and challenges of the open charge point protocol (ocpp)," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 3, pp. 1504–1533, 2022.
- [4] S. Acharya, Y. Dvorkin, H. Pandžić, and R. Karri, "Cybersecurity of smart electric vehicle charging: A power grid perspective," *IEEE access*, vol. 8, pp. 214 434–214 453, 2020.
- [5] L. Buschlinger, M. Springer, and M. Zhdanova, "Plug-and-patch: Secure value added services for electric vehicle charging," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, 2019, pp. 1–10.
- [6] G. S. Morrison, "Threats and mitigation of ddos cyberattacks against the us power grid via ev charging," Master's thesis, Wright State University, 2018.
- [7] Z. El-Rewini, K. Sadatsharan, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan, "Cybersecurity challenges in vehicular communications," *Vehicular Communications*, vol. 23, p. 100214, 2020.
- [8] A. Irshad, M. Usman, S. A. Chaudhry, H. Naqvi, and M. Shafiq, "A provably secure and efficient authenticated key agreement scheme for energy internet-based vehicle-to-grid technology framework," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 4425–4435, 2020.
- [9] A. Gharaibeh, M. A. Salahuddin, S. J. Hussini, A. Khreishah, I. Khalil, M. Guizani, and A. Al-Fuqaha, "Smart cities: A survey on data management, security, and enabling technologies," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2456–2501, 2017.
- [10] A. Gopstein, C. Nguyen, C. O'Fallon, N. Hastings, D. Wollman *et al.*, *NIST framework and roadmap for smart grid interoperability standards, release 4.0*. Department of Commerce. National Institute of Standards and Technology . . . , 2021.
- [11] C. Zhao and X. You, "Research and implementation of ocpp 1.6 protocol," in *2017 2nd International Conference on Machinery, Electronics and Control Simulation (MECS 2017)*. Atlantis Press, 2016, pp. 283–291.
- [12] 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.
- [13] A. G. Morosan and F. Pop, "Ocpp security-neural network for detecting malicious traffic," in *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, 2017, pp. 190–195.
- [14] T. M. House, "Ocpp: Open charge point protocol implementation in python," <https://github.com/mobilityhouse/ocpp>.