Received: 2 January 2020 | Revised: 10 May 2020 | Accepted: 8 June 2020

DOI: 10.1002/nem.2126

Clustering method in protocol reverse engineering for industrial protocols

Kyu-Seok Shim | **Young-Hoon Goo** | **Min-Seob Lee** | **Myung-Sup Kim**

Department of Computer and Information Science, Korea University, Sejong, South Korea

Correspondence
Myung-Sup Kim, Department of Computer and Information Science, Korea University, Sejong, South Korea.
Email: tmskim@korea.ac.kr

Funding information
Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government, Grant/Award Number: 2018-0-00539; Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Grant/Award Number: NRF-2018R1D1A1B07045742

Summary
Automation in all aspects of industrial activity is currently needed in today's industries. Networks, which are the most essential elements of automation, have been widely used in industrial sites to realize such needs. However, network security threats and malfunctions at industrial sites can cause considerable physical damage. Damage can be prevented, and threats can be detected through network traffic monitoring. However, industrial protocols use self-developed protocols to ensure rapid and efficient data transfer, and most self-developed protocols are private networking protocols. Efficient network traffic monitoring requires a detailed understanding of the structure of industrial protocols. Studies on existing protocol reverse engineering methods for commercial protocols have indicated that there are many limitations in applying these methods to industrial protocols. Therefore, in this paper, we propose a method of analyzing the structure of private protocols that can be employed as industrial protocols. This methodology consists of six modules: traffic collection, message extraction, message clustering by size, message clustering by similarity, field extraction, and session analysis. We collect traffic using the Schneider Modicon M580 and demonstrate the validity of the proposed methodology by comparing collected traffic with existing protocol reverse engineering methods.

1 | INTRODUCTION

An industrial control system (ICS) is a computer system that monitors and controls work processes based on the manufacturing, production, power generation, processing, smelting, infrastructure, and facilities operations in an industry. As existing industrial operations become impossible because of the scale of industrial sites, ICSs present promising solutions by automating industrial processes. Programmable logic controller (PLC) equipment have been recently employed for automatic control of industrial processes in an ICS. PLC is a controller that moves equipment according to user-generated logic circuits. In other words, PLC is a digitally controlled industrial power unit that performs control actions such as logic operations, sequencing, timers, counters, and arithmetic operations based on digital and analog input and output. PLC equipment is used to control the work processes in an ICS environment through an engineering workstation (EWS).

PLC operations involve a structure that uses a network to transmit user-controlled logic circuits from the EWS to the PLC. User operations involve PLC setup, security, and control. Self-developed protocols are employed to efficiently transmit structures set up in EWS to PLC. Most of the protocols developed are not disclosed to prevent security threats.[1–3] However, PLC and EWS require security threat analysis derived from traffic monitoring, ICS status, and current settings check. Therefore, gaining insight into the structure of the network protocols used by PLC and EWS for traffic monitoring is a crucial requirement.

Passive reasoning is traditionally executed to derive the structure of the protocol.[4–20] However, there remains many challenges because the types of protocols used by PLC equipment vary and the specifications of the protocols can vary depending on the environmental situation. Therefore, automated protocol reverse engineering methods are actively being studied. Many existing automated protocol reverse engineering technologies have been developed based on commercial protocols and are insufficient for analyzing protocols in ICS environments. Unlike commercial protocols, because peer-to-peer communication is usually involved between EWS and PLC, the EWS protocol sends and receives all commands and all packets through a single flow from a single connection, rather than generating several flows during the traffic collection process. For commercial protocols, a common field exists for each message, but for the EWS protocol, a detailed structure of the EWS protocol can be difficult to determine using conventional methods because different messages are transmitted for each packet except for the part that has been partially disclosed.

Therefore, methods are needed to accurately analyze the EWS protocol. Security accidents are on the rise as network technologies are becoming increasingly more prevalent in industrial sites.[21] Network traffic monitoring is necessary to prevent such accidents, and the structure of the protocol is essential. In this paper, we propose a system for deriving the structure of the Modbus/TCP protocol using a Schneider Modicon equipment, including representative PLC equipment. The system classifies protocol messages based on size and clusters the protocol messages using the mean-shift algorithm. These grouped messages are defined as one type. For each type, this system uses contiguous sequence pattern (CSP) algorithms to extract a common substring that defines the field.[22] The structure of the messages is analyzed after field definitions. Finally, the sequence and structure of the message types can be used to identify the types of messages used at industrial sites, the meaning of the fields, and the commands transmitted by the network traffic.

Following this introduction, this paper presents related research in Section 2. The proposed methods are described in Section 3, and validation of the proposed methods through experiments and results is presented in Section 4. Conclusions and future studies are included in Section 5.

2 | RELATED WORKS AND PROBLEM SCOPE

This paper focuses on two areas: industrial protocol and automatic protocol reverse engineering. There are many different types of industrial protocols that use independently developed protocols for each equipment. There are countless types of Modbus/TCP, PROFIBUS, CAN BUS, CANopen, DeviceNet, CC-Link, and so on. Automatic protocol reverse engineering refers to a system that automatically proceeds with work to derive the structure, specifications, and so on of a private protocol. Automatic protocol reverse engineering has been developed in a variety of ways.

2.1 | Industrial protocol

Currently, there are many industrial communication protocols used by industries. Industrial communication protocols include equipment such as human machine interfaces (HMI), PLCs, motors, and sensors that are connected to each other for expansion and efficiency. These industrial protocols are increasingly being used because they are simple and fast to connect, exhibit long connections, and can connect to several equipment. In addition, with growing automation in industries, the industrial protocol has become an essential element of industrial communication networks.

PROFIBUS is active in industrial automation systems, including factory automation and process automation.[23] PROFIBUS employs digital communication methods for data processing and auxiliary data transmission. It can achieve a maximum speed of 12 Mbps and supports up to 126 addresses. CAN BUS is a high-integrated serial bus system[24] designed for the auto industry. The scope of CAN BUS has been extended to field buses for industrial automation. CAN BUS ensures serial communications that provide physical link layers and data link layers at speeds of up to 1 Mbps. The CANopen and DeviceNet protocols are an umbrella concept of CAN BUS, ensuring interoperability with devices utilizing the same industrial network.[25,26] The CANopen network supports up to 127 nodes. DeviceNet, on the other hand, supports 64 nodes. Modbus is a simple, effective, and free serial bus[27] wherein up to 247 nodes can be connected within the same link. Modbus has an easy-to-deploy advantage. CC-Link is an open-structure industrial network protocol developed by Mitsubishi of Japan.[28] It has been widely used in Japan and other Asian regions.

Recently developed industrial sites require high cost efficiency. Consequently, industrial Ethernet communication protocols, which add Ethernet communication to industrial protocols, have been actively used. Ethernet protocols for industrial use have reduced latency and increased deterministic responses by utilizing modified media access control (MAC) layers. It provides flexibility in choosing a network topology and allows users to flexibly configure multiple nodes in a single system.

EtherCAT, developed by Beckhoff, realizes rapid packet processing and extends the connectivity of automated systems from PLC to I/O and sensors by accessing real-time Ethernet in automated applications.[29,30] Ethernet/IP, developed as the mainstay of Lockwell automation, is an application layer protocol, unlike EtherCAT, the MAC layer protocol.[31] PROFINET, developed by manufacturers of industrial equipment such as Siemens and GE, has three types and is suitable for use in the environment.[32] CC-Link IE is a protocol that introduces industrial Ethernet technology to CC-Link.[28] Finally, Modbus/TCP, which has been covered in this paper, is a new version that Modbus has recently developed.[33] Developed by Schneider Electric, this protocol implements Modbus message transmission using TCP/IP. Modbus/TCP can be easily implemented over standard Ethernet networks.

Most of these industrial protocols are private or partially public. Therefore, the only way of determining the specifications of industrial protocols is to manually analyze them. However, because there are many different types of protocols, it is impossible to manually analyze them whenever all protocols are used. Therefore, automated protocol reverse engineering systems are required for deriving the structure of industrial protocols.

2.2 | Automatic protocol reverse engineering

Most existing protocol reverse engineering methods are manually analyzed. However, because there are limitations in manually analyzing protocols as the types of protocols vary, several methods that automatically derive the structure of protocols have been studied. Automatic protocol reverse engineering technologies typically include Netzob, AutoReEngine, and FieldHunter.

Netzob is a semiautomatic methodology that automates some of the inference processes of the protocol structure using a methodology proposed by Bosert in 2011.[14] Netzob focuses on automating the inference process without involving the work of experts. Detailed vocabulary models and methodologies have been devised for this purpose. Netzob clusters similar types of messages using an unweighted pair group method with arithmetic mean (UPGMA). Clustering messages are defined as one Symbol. Symbol refers to a set of messages in the same format and role from a protocol point of view. Each symbol sorts a common string using the Needleman–Wunsch algorithm. The common strings are defined as static fields and alternative fields for the remainder of the messages. A field refers to a set of tokens that share a common meaning from a protocol perspective. A symbol consists of several fields, and each field can accept a unique value or multiple values.

AutoReEngine is a methodology proposed by Luo and Yu in 2013.[16] AutoReEngine receives network traffic for a single protocol as an input. AutoReEngine largely comprises four steps: “Data Pre-processing,” “Protocol Keyword Extraction,” “Message Format Extraction,” and “State Machine Inference.” In the data preprocessing step, input traffic is classified as a flow and packets in the flow are reassembled into messages. The protocol keyword extraction step largely proceeds in two steps. In the first frequency strings extraction step, a sequence of messages is entered and extracted from the field format candidate keyword using the Apriori algorithm. At this point, the length-1 items in the Apriori algorithm comprise 1 byte, the transaction comprises each message sequence, and the units of support include the session support rate (Rssr) and site-specific session set support rate (Rset). Rssr is the ratio of candidate sequences that contain candidate sequences for the entire flow, and Rset denotes the ratio of site-specific sessions that contain candidate sequences for the entire site-specific session. A site-specific session refers to a set of flows with the same server. In other words, Rsr and Rset are determined for candidates and for groups of items that occur gradually from length-1 to length-K, where the group of items that occur frequently are not extracted according to the default Apriori algorithm. In addition, the two items that satisfy the two sets of the threshold session support rate (Tssr) and the threshold site-specific session set (Tsets) are determined. A byte sequence, which includes a final collection of all frequently extracted items, is extracted and closed strings are determined for these byte sequences.

FieldHunter is a methodology proposed by Bermudez et al. in 2015 that receives network traffic for a single protocol as input.[19] FieldHunter first receives network flow as an input and divides the network flow into network messages. FieldHunter divides the units of network messages into PUSH flags for TCP and one packet for UDP. The syntax inference step first checks whether it is a text-based or binary-based protocol and differently performs message tokenization in the message tokenizing module. A key step in FieldHunter is semantics inference. Locate the field that corresponds to the type of meaning predefined in the semantics inference step in a heuristic way. There are six types of predefined meanings used by FieldHunter: message type, message length, host identifier, session identifier, transaction identifier, and accumulators. The primary means of determining whether the fields correspond to each type of meaning is to use the notion that different field types are completely different in vertical analysis, that is, there exists statistical characteristics for each field in different traces. For example, to deduce a field that corresponds to the host identifier, this system presents a field that always includes a corresponding unique value for each source IP address of different traces.

2.3 | Problem scope

As mentioned in this paper, protocol reverse engineering technologies developed for commercial protocols have been a subject of focus for a long time. However, there are many limitations to using this methodology for industrial protocols. First, Netzob and the proposed method follow a top-down approach, which involves setting the type of message and extracting fields based on the type. Therefore, Netzob is most similar to the proposed method. However, UPGMA, a core algorithm, is a method that recursively compares and determines the similarity between two messages, which is an algorithm with very high computational and time complexity. In fact, using Netzob, it was determined that when a large amount of data is entered, the system does not run, or it takes a long time to produce results.

AutoReEngine is a bottom-up method that extracts fields and derives the structure of messages from a combination of fields. Although different from the method proposed in this paper, the Apriori algorithm used to find the common substring in messages is similar. AutoReEngine searches for a common field in all messages. Therefore, there exists a passive part where users must specify the appropriate threshold value. If a user does not specify the appropriate threshold, the correct fields will not be extracted, nor would it be possible to derive the correct message structure without the correct fields.

The proposed method divides the clustering of messages into two parts to address these problems. The first part involves separating messages based on size. If industry protocols are of the same size, most messages are of the same type. However, the clustering process that measures similarity is necessary because different types of messages may be mixed despite of the same size. Therefore, the clustering algorithm mean-shift is applied only between messages separated by the next step. This method enables rapid clustering because of the small number of messages being clustered.

In addition, several types of messages are automatically inferenced using this methodology, even though semantic inference methods are used in FieldHunter. In FieldHunter, various algorithms are used for semantic inference and for finding matching fields. However, this method inferences the types of messages in the field extraction process. Message type can be inferred from clustering in the proposed method. Message length also separates messages based on size, which is the precluster stage, and so this method inferences the appropriate fields.

3 | THE PROPOSED METHOD

In this section, we present a description of the proposed automatic industrial protocol reverse engineering method. The automatic industrial protocol reverse engineering method consists of six stages. The structure of this system consists of the following steps: traffic collection, message extraction, message clustering based on size, message clustering based on similarity, field extraction, and session analysis. Each step is executed as shown in Figure 1.

Figure 2 illustrates the overall structure of the automatic industrial protocol reverse engineering system. First, the traffic collection phase collects traffic between the PLC equipment and the EWS program. Traffic associated with operations that performed the same function more than once is collected during this process. The nature of the connection

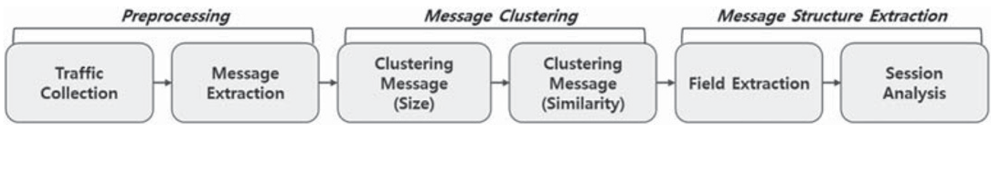


FIGURE 1 Sequence of protocol reverse engineering for industrial protocols

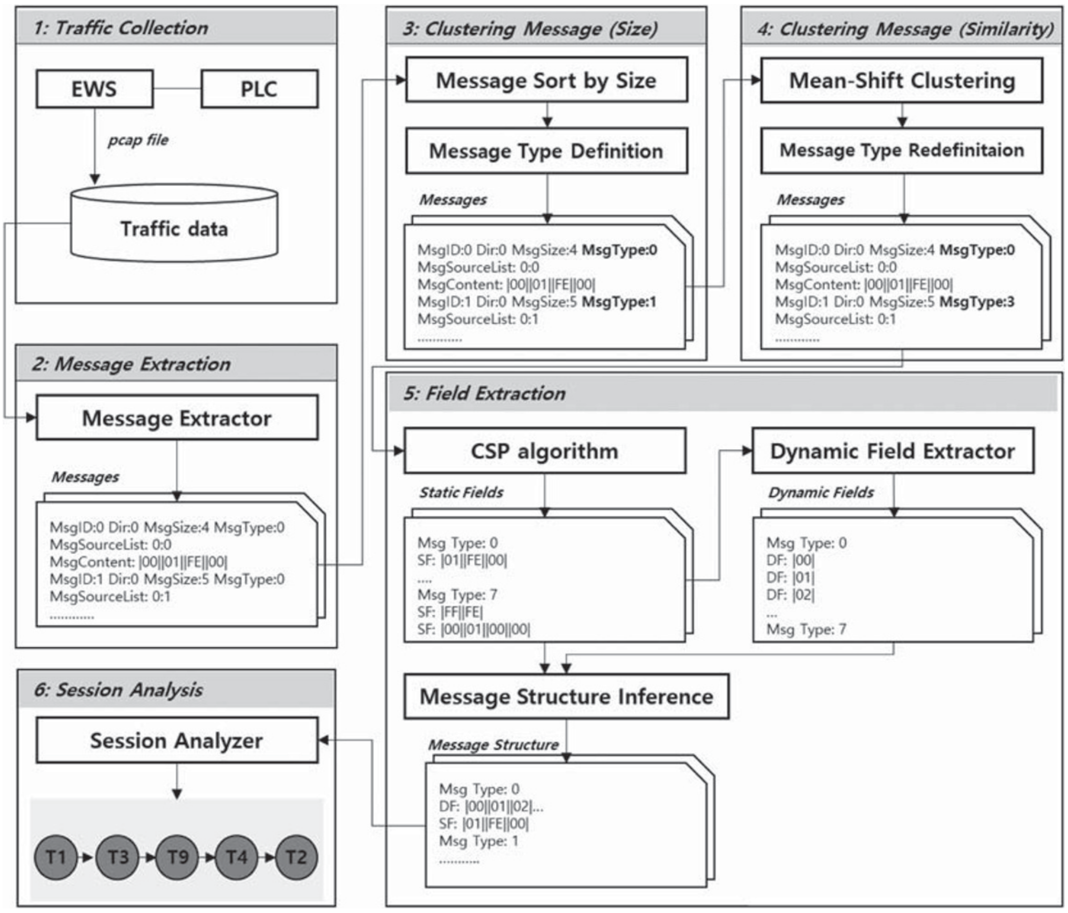


FIGURE 2 Overview of protocol reverse engineering for industrial protocol

between EWS and PLC generates one flow at the time of connection. Therefore, we execute at least two traffic collection processes to analyze the occurrence of one message in the next connection. The message extraction phase involves extracting messages from the traffic collected. This step defines a packet as a single message, separated by directions. Therefore, the outputs include request messages and response messages. The classification based on message size phase defines the size of the messages and specifies the message type based on size. Type definitions are numbered in increasing order of message size. The message clustering step recomposes similar messages by measuring their similarity. This step classifies messages of the same size as other types. The field extraction phase defines a field based on a common character string obtained from messages grouped into the same type. At this stage, commands, settings, and so on sent using network traffic can be inferred and abnormal traffic can be detected. In addition, this step ultimately extracts each type of message structure. In the last session analysis stage, the sequence of each type of message extracted is analyzed. Therefore, the order of the messages can be derived when performing the functions.

3.1 | Traffic collection

This step collects traffic between the EWS and PLC equipment. When collecting traffic, EWS specifies the function and executes that function to collect the traffic. At least two traffic sets must be collected by performing this process more than once. Industrial protocols transmit different messages from one flow. This is a different point from commercial protocols. For commercial protocols, the same message transmits multiple flows. Therefore, by analyzing the traffic set of commercial protocols, the message type can be obtained in one function; however, this is not possible for industrial protocols. Industrial protocols require more than one traffic set to extract both comparison from the same traffic sets and the comparison from other traffic sets.

3.2 | Message extraction

The collected traffic set is stored in a pcap-type. This step extracts a pcap-type file as a message format used in this method. When extracting in message form, divide by direction. Direction is a message sent from the EWS to the PLC and from the PLC to the EWS. In other words, direction is divided into request and response messages. This is because the message type in a request is different from the message type in a response. Information contained in the message has been presented in the following table.

Figure 3 shows the configuration of the message. Message information includes a unique ID to distinguish between different messages. Message information also includes the direction of the message as mentioned above. In addition, in the next step, messages are primarily grouped based on their size, and so the messages include message size information and type information to specify the type. In the current step, all types are zero values. The message source list includes the location information of the message. This includes the flow and packet information where the message was located because messages were extracted from flow information. This information will be used in the session analysis step. Finally, the message content includes the payload content that the message holds.

3.3 | Message clustering based on size

This step specifies the type by separating messages extracted from the message extraction step based on size. Figure 4 shows the process of clustering messages by size. The extracted messages are divided into request and response. Therefore, this step is performed twice. First, this step receives request messages. Messages entered are sorted based on size. Sorted messages are sequentially formatted starting with messages with the smallest size. For example, when there are six messages of sizes 5, 5, 5, 6, 7, and 8, messages of size 5 exhibit type 0 values, messages of size 6 exhibit type 1 values, messages of size 7 exhibit type 2 values, and messages of size 8 exhibit type 3 values.

There are two reasons for which this step is necessary. First, there are not many fields of variable lengths in the industrial protocol. In other words, messages of the same size are mostly of the same type. Second, clustering algorithms are algorithms with relatively high computational and time complexity. Therefore, the first classification is performed in the next step before the clustering algorithm is executed, reducing computational and time complexity.

3.4 | Message clustering based on similarity

This step measures the similarity between messages separated by size and performs more detailed clustering. In other words, similarity between messages of the same size is measured and classified. At this step, we used several algorithms to obtain clustering algorithms that best categorize messages. These algorithms include K-means, UPGMA, and mean-shift algorithms. Experiments employing each algorithm are presented in Section 4.

K-means is an algorithm that binds given data into k clusters and works in a way that minimizes the variance of each cluster and differences in distance.[34] This algorithm is a type of self-learning and acts as a label for nonlabeled input data. An advantage of this algorithm is that it runs fast and exhibits very good performance for certain types of data. However, a difficulty associated with this system is that the user must specify the value of k, and the user does not know how many data messages are classified. Therefore, the system uses the elbow method to employ K-means. The

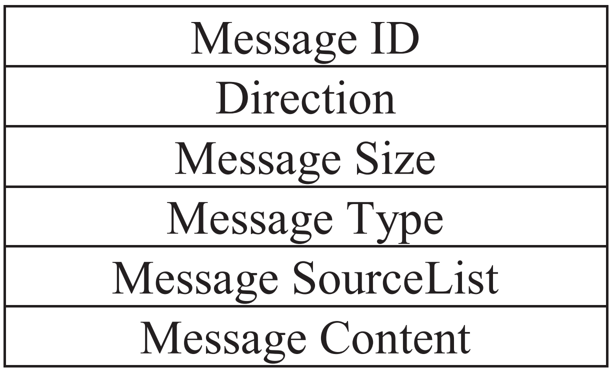
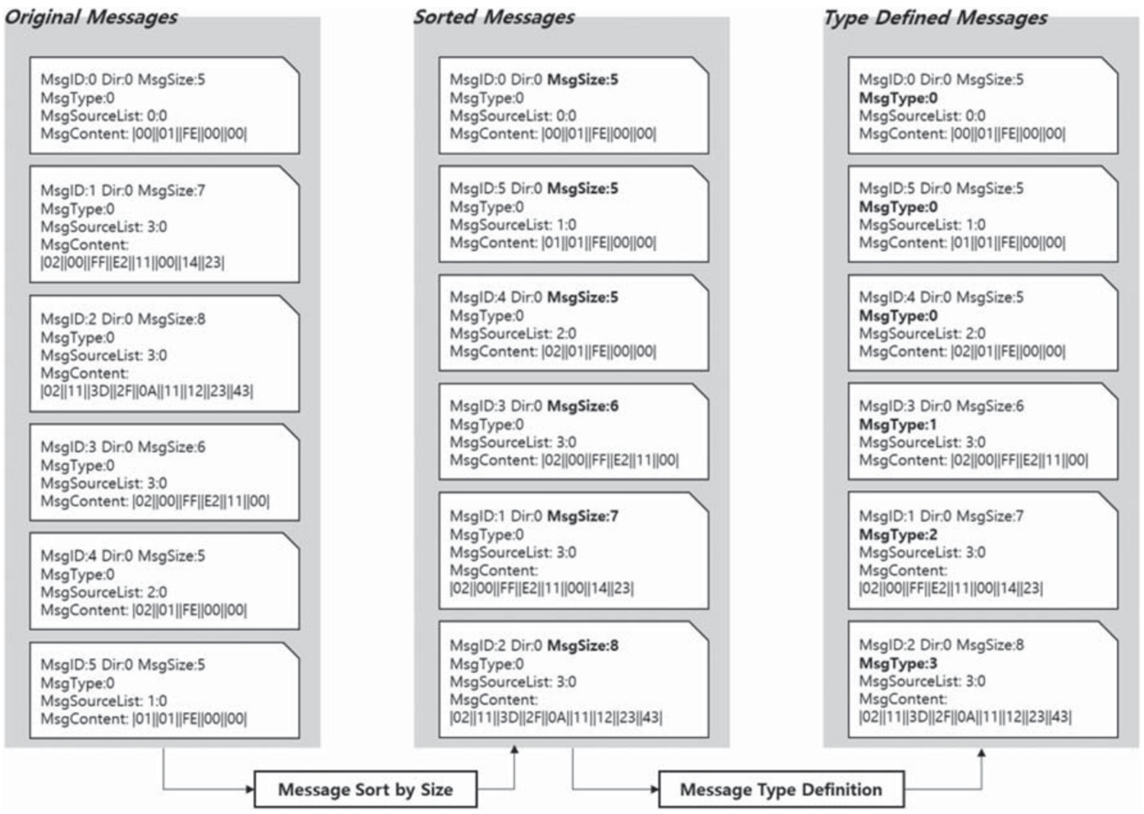


FIGURE 3 Message information



```
Original Messages
MsgID:0 Dir:0 MsgSize:5
MsgType:0
MsgSourceList: 0:0
MsgContent: |00||01||FE||00||00|

MsgID:1 Dir:0 MsgSize:7
MsgType:0
MsgSourceList: 3:0
MsgContent:
|02||00||FF||E2||11||00||14||23|

MsgID:2 Dir:0 MsgSize:8
MsgType:0
MsgSourceList: 3:0
MsgContent:
|02||11||3D||2F||0A||11||12||23||43|

MsgID:3 Dir:0 MsgSize:6
MsgType:0
MsgSourceList: 3:0
MsgContent: |02||00||FF||E2||11||00|

MsgID:4 Dir:0 MsgSize:5
MsgType:0
MsgSourceList: 2:0
MsgContent: |02||01||FE||00||00|

MsgID:5 Dir:0 MsgSize:5
MsgType:0
MsgSourceList: 1:0
MsgContent: |01||01||FE||00||00|
```

```
Sorted Messages
MsgID:0 Dir:0 MsgSize:5
MsgType:0
MsgSourceList: 0:0
MsgContent: |00||01||FE||00||00|

MsgID:5 Dir:0 MsgSize:5
MsgType:0
MsgSourceList: 1:0
MsgContent: |01||01||FE||00||00|

MsgID:4 Dir:0 MsgSize:5
MsgType:0
MsgSourceList: 2:0
MsgContent: |02||01||FE||00||00|

MsgID:3 Dir:0 MsgSize:6
MsgType:0
MsgSourceList: 3:0
MsgContent: |02||00||FF||E2||11||00|

MsgID:1 Dir:0 MsgSize:7
MsgType:0
MsgSourceList: 3:0
MsgContent:
|02||00||FF||E2||11||00||14||23|

MsgID:2 Dir:0 MsgSize:8
MsgType:0
MsgSourceList: 3:0
MsgContent:
|02||11||3D||2F||0A||11||12||23||43|
```

```
Type Defined Messages
MsgID:0 Dir:0 MsgSize:5
MsgType:0
MsgSourceList: 0:0
MsgContent: |00||01||FE||00||00|

MsgID:5 Dir:0 MsgSize:5
MsgType:0
MsgSourceList: 1:0
MsgContent: |01||01||FE||00||00|

MsgID:4 Dir:0 MsgSize:5
MsgType:0
MsgSourceList: 2:0
MsgContent: |02||01||FE||00||00|

MsgID:3 Dir:0 MsgSize:6
MsgType:1
MsgSourceList: 3:0
MsgContent: |02||00||FF||E2||11||00|

MsgID:1 Dir:0 MsgSize:7
MsgType:2
MsgSourceList: 3:0
MsgContent:
|02||00||FF||E2||11||00||14||23|

MsgID:2 Dir:0 MsgSize:8
MsgType:3
MsgSourceList: 3:0
MsgContent:
|02||11||3D||2F||0A||11||12||23||43|
```

FIGURE 4 Messages type clustering based on size

elbow method monitors the results by sequentially increasing the number of clusters and sets the number of previous clusters as *k* if the addition of one cluster does not produce better results than previously.

UPGMA is a hierarchical clustering method. This algorithm is one of the arithmetic equations used to compare similarities between data and data groups based on the qualitative results of the data.[35] UPGMA is a proven algorithm in the field of protocol reverse engineering used in Netzob. However, this algorithm is the weakest point of Netzob. Because the distances of each message are sequentially and recursively calculated, the computational complexity is significantly higher than other algorithms. Therefore, Netzob does not produce results when identifying the protocol structure of the data.

Mean-shift clustering is a centroid-based algorithm that focuses on finding the center point of each group that operates by updating the candidates of the center point to the average of the points in the sliding window.[36] The advantages of this algorithm include rapid and relatively accurate clustering. In addition, in contrast to K-means clustering, mean-shift is automatically detected in this algorithm, and so there is no need to select the number of clusters. However, a disadvantage includes the selection of a window size.

The proposed method uses the mean-shift algorithm. K-means is not available because it is not known that messages can be classified into several types. To address these disadvantages, we must use the elbow method. However, the elbow method is an algorithm that selects the maximum k. Therefore, messages of the same type are classified into different types. UPGMA exhibits a high level of computational complexity. The proposed method reduced the number of primary input data by grouping the data based on size. However, if there are many messages of the same type or a large volume of data is entered, the system load will prevent the results from being achievable. In addition, UPGMA requires a threshold such as K-means. It is not appropriate to apply UPGMA because it is not known in advance how many types of messages are classified, including the value of *K* that is classified for each type. Therefore, we use the mean-shift algorithm. Figure 5 shows the process of classifying messages by similarity through the mean-shift algorithm.

By performing this step, the types of messages can be determined. That is, this step determines the number of message types for the entered.

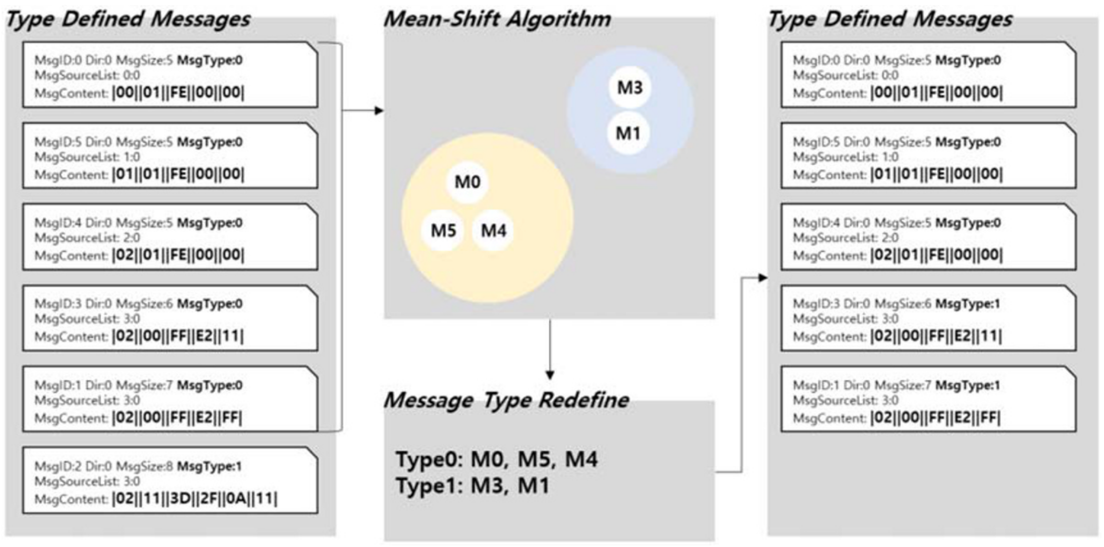

Type Defined Messages
MsgID:0 Dir:0 MsgSize:5 MsgType:0 MsgSourceList: 0:0 MsgContent: |00||01||FE||00||00|
MsgID:5 Dir:0 MsgSize:5 MsgType:0 MsgSourceList: 1:0 MsgContent: |01||01||FE||00||00|
MsgID:4 Dir:0 MsgSize:5 MsgType:0 MsgSourceList: 2:0 MsgContent: |02||01||FE||00||00|
MsgID:3 Dir:0 MsgSize:6 MsgType:0 MsgSourceList: 3:0 MsgContent: |02||00||FF||E2||11|
MsgID:1 Dir:0 MsgSize:7 MsgType:0 MsgSourceList: 3:0 MsgContent: |02||00||FF||E2||FF|
MsgID:2 Dir:0 MsgSize:8 MsgType:1 MsgSourceList: 3:0 MsgContent: |02||11||3D||2F||0A||11|
Mean-Shift Algorithm
M3
M1
M0
M5
M4
Message Type Redefine
Type0: M0, M5, M4
Type1: M3, M1
Type Defined Messages
MsgID:0 Dir:0 MsgSize:5 MsgType:0 MsgSourceList: 0:0 MsgContent: |00||01||FE||00||00|
MsgID:5 Dir:0 MsgSize:5 MsgType:0 MsgSourceList: 1:0 MsgContent: |01||01||FE||00||00|
MsgID:4 Dir:0 MsgSize:5 MsgType:0 MsgSourceList: 2:0 MsgContent: |02||01||FE||00||00|
MsgID:3 Dir:0 MsgSize:6 MsgType:1 MsgSourceList: 3:0 MsgContent: |02||00||FF||E2||11|
MsgID:1 Dir:0 MsgSize:7 MsgType:1 MsgSourceList: 3:0 MsgContent: |02||00||FF||E2||FF|

FIGURE 5 Clustering of message types based on similarity

3.5 | Field extraction

The field extraction step derives static fields and dynamic fields for messages. A static field refers to a series of common strings in the same type of messages. A dynamic field refers to the remainder of the except common strings in the same type of messages. Therefore, the message type consists of static fields and dynamic fields.

At this step, the static field is extracted using the CSP algorithm. The CSP algorithm extracts a common string based on the Apriori algorithm.[37] The static field extraction process that uses the CSP algorithm extracts the same type of messages into a set of sequences. The CSP algorithm generates content of length 1 from a set of sequences. Content of length 1 is divided into content that does not meet the minimum support through a minimum support examination and content that satisfies the minimum support requirements. Content that does not satisfy the minimum support requirement will be deleted, and content that satisfies the requirement will be created with a content length of 2. This process is repeated until the length can no longer be increased. Figure 6 presents an example of the processes involved in the extraction of static fields.

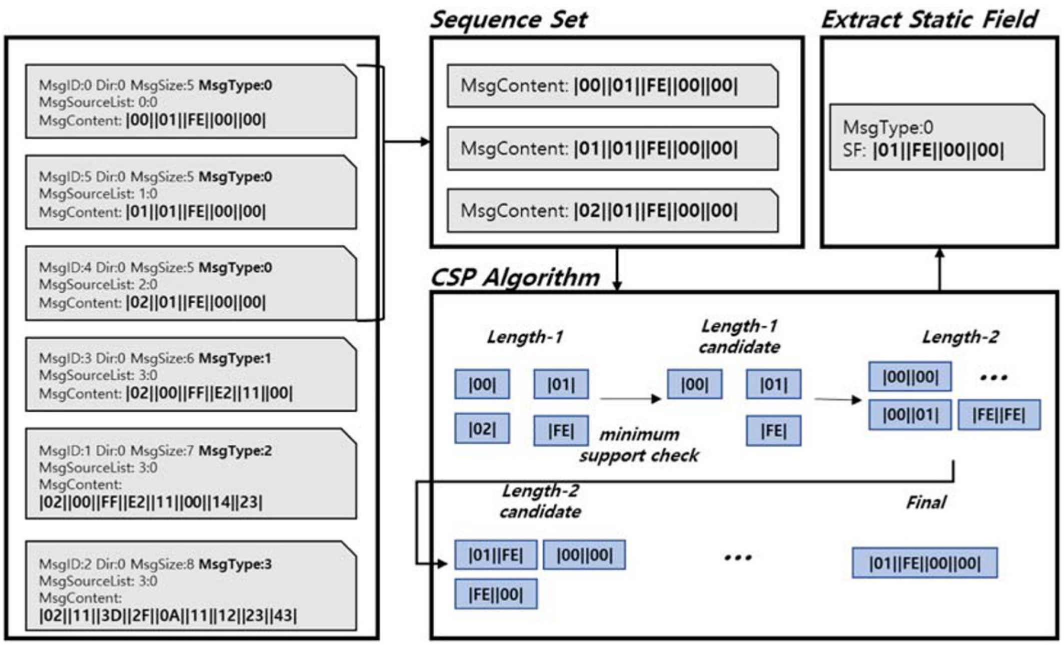

MsgID:0 Dir:0 MsgSize:5 MsgType:0 MsgSourceList: 0:0 MsgContent: |00||01||FE||00||00|
MsgID:5 Dir:0 MsgSize:5 MsgType:0 MsgSourceList: 1:0 MsgContent: |01||01||FE||00||00|
MsgID:4 Dir:0 MsgSize:5 MsgType:0 MsgSourceList: 2:0 MsgContent: |02||01||FE||00||00|
MsgID:3 Dir:0 MsgSize:6 MsgType:1 MsgSourceList: 3:0 MsgContent: |02||00||FF||E2||11||00|
MsgID:1 Dir:0 MsgSize:7 MsgType:2 MsgSourceList: 3:0 MsgContent: |02||00||FF||E2||11||00||14||23|
MsgID:2 Dir:0 MsgSize:8 MsgType:3 MsgSourceList: 3:0 MsgContent: |02||11||3D||2F||0A||11||12||23||43|
Sequence Set
MsgContent: |00||01||FE||00||00|
MsgContent: |01||01||FE||00||00|
MsgContent: |02||01||FE||00||00|
Extract Static Field
MsgType:0
SF: |01||FE||00||00|
CSP Algorithm
Length-1
|00| |01| |02| |FE|
minimum support check
Length-1 candidate
|00| |01| |FE|
Length-2
|00||00| |00||01| |FE||FE| •••
Length-2 candidate
|01||FE| |00||00| |FE||00| •••
Final
|01||FE||00||00|

FIGURE 6 The processes involved in extracting static fields

The CSP algorithm requires minimum support. Minimum support is a condition in which candidate content can be extended to the next length. In this study, the minimum support is always set to 100%; 100% refers to a common string that is extracted from all messages of the same type.

3.6 | Session analysis

The session analysis step presents the order of defined message types that occur when executing a function by aligning the sequence of deduced messages. Message information includes flow location and packet location information. Therefore, this information can be used to implement a sequence of message types within a session. The implemented results can analyze the structure of the protocol message in flow.

4 | EXPERIMENT AND RESULT

In this section, we evaluate the performance of the proposed method. The protocols used in this experiment are Modbus/TCP, Ethernet/IP, and FTP. The proposed method evaluates the message clustering capabilities of the industrial protocols Modbus/TCP and Ethernet/IP and evaluates objectivity by comparing the well-known protocol structures and outcomes by adding FTP protocols to experiments. We use the Schneider Electric Modicon M580 (PLC) and Unity Pro (EWS) for traffic collection. The Modicon M580 uses the industrial protocol, Modbus/TCP. Therefore, this section presents a description of the architecture of Modbus/TCP, refers to the traffic information collected for the experiment, and compares the results of applying different clustering algorithms to that traffic. The proposed method evaluates performance based on the analysis results of the Modbus/TCP protocol. Finally, the same traffic proves the validity of the proposed method using the comparative analysis and the results of Netzob and AutoReEngine.

4.1 | Modbus/TCP message structure

The Modbus/TCP is a typical industrial protocol with a partially public structure. The Modbus/TCP consists of six fields, as shown in Figure 7. Transaction ID indicates the sequence number of tasks associated with queries and responses. Protocol ID is fixed at 0x0000. Length indicates the distance between the length field and the end of that frame. Unit ID is a field that occupies a fixed number of 1 byte. Function code (FC) refers to the Modbus/TCP function code.

Function code is a command set code provided by the Modbus protocol. This service allows users to read or write values to slave memory (Coil, Register) using the function code. Although the value between function code 1 and 127 is used, the TCP Port supports the following values: 1, 2, 4, 5, 6, 15, and 16. For the following supported function codes, the data part is open to the public in the document. However, the function code is fixed at 90 when sending and receiving data using EWS. Therefore, the data part is private when communicating with EWS, and the structure of the data part with function code of 90 is extracted using the proposed method.

4.2 | Traffic collection information

The EWS can perform various functions. Typical functions include connecting with PLC, project transfer EWS to PLC, and project transfer PLC to EWS. The connection to PLC is the most basic function of connecting EWS and PLC. The function of project transfer EWS to PLC transmits project sets using EWS. The PLC, which receives the project, performs the task in the logic set using EWS. The function of project transfer PLC to EWS involves sending project

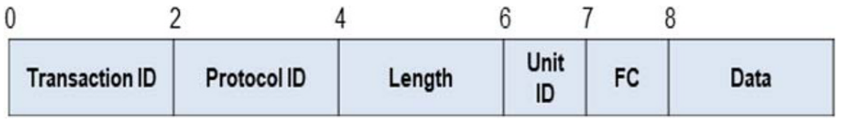


FIGURE 7 Structure of Modbus/TCP

information that is performed by existing PLC to EWS. We collect traffic and analyze protocol structures for the following three typical functions. Traffic collection information is shown in Table 1.

4.3 | Results of the message clustering algorithm

In the proposed system, we categorized messages based on type using clustering algorithms. The proposed algorithms are K-means, UPGMA, and mean-shift. This section applies each algorithm to protocols. The applied results were compared to prove the validity of applying the mean-shift algorithm. We manually classified the message types by analyzing the message types of the experimental data for accurate experimentation. The results of the classification are presented in Table 2.

Table 3 presents the results of applying the message clustering algorithm. When applying each algorithm, input data comprise messages classified by size. In other words, if there are five types of message sizes in the input data, clustering

TABLE 1 Information of evaluation traffic

Modbus/TCP	Connection	Transfer EWS to PLC	Transfer PLC to EWS
File	29	23	28
Flow	29	46	28
Packet	3506	7532	4523
Bytes	571 314	2 168 519	1 419 958
Messages	3506	7532	4523
Ethernet/IP	Connection	Transfer EWS to PLC	Transfer PLC to EWS
File	3	3	3
Flow	3	10	8
Packet	299	938	674
Bytes	30 314	89 010	88 935
Messages	299	938	674
FTP	-		
File	5		
Flow	8		
Packet	1139		
Bytes	101 524		
Messages	1139		

TABLE 2 Number of ground-truth message types by functions

	The ground truth message types
Modbus/TCP	
Connection	32 (19 + 13)
Transfer EWS to PLC	66 (22 + 44)
Transfer PLC to EWS	74 (63 + 11)
Ethernet/IP	
Connection	29 (16 + 13)
Transfer EWS to PLC	28 (15 + 13)
Transfer PLC to EWS	19 (11 + 8)
FTP	
-	47 (24 + 23)

TABLE 3 Result of message clustering by algorithms

	Size	K-means	UPGMA	Mean-shift
Modbus/TCP				
Connection	23 (17 + 6)	76 (27 + 49)	76 (27 + 49)	43 (27 + 16)
Transfer EWS to PLC	29 (18 + 11)	83 (36 + 47)	83 (36 + 47)	65 (27 + 38)
Transfer PLC to EWS	26 (20 + 6)	44 (14 + 30)	44 (14 + 30)	53 (43 + 10)
Ethernet/IP				
Connection	23 (13 + 10)	29 (16 + 13)	29 (16 + 13)	29 (16 + 13)
Transfer EWS to PLC	24 (13 + 11)	33 (17 + 16)	33 (17 + 16)	27 (14 + 13)
Transfer PLC to EWS	24 (12 + 12)	29 (14 + 15)	29 (14 + 15)	25 (13 + 12)
FTP				
-	56 (31 + 25)	97 (54 + 43)	97 (54 + 43)	63 (33 + 30)

algorithms are executed five times. We determined that the mean-shift algorithm is most similar to the results of manually classifying message types. K-means and UPGMA set the *K* value using the elbow value, and so the number of message types is the same.

4.4 | Result of Modbus/TCP structure analysis of the proposed system

In this section, we present a description of the results of the Modbus/TCP protocol analysis using the proposed method. We also present a comparison of the proposed method with existing protocol reverse engineering methods. Existing protocol reverse engineering methods include Netzob and AutoReEngine. Netzob and AutoReEngine are the most representative protocol reverse engineering methods used for commercial protocols. Netzob and AutoReEngine require initial setup values. Netzob must set a similarity percentage, and AutoReEngine must set a minimum support value because it is based on the Apriori algorithm. We applied several values to obtain the best initial settings, and we determined that the value was 50%.

Indicators for evaluating performance were evaluated in terms of conciseness and coverage. Conciseness evaluates the message type of input data to the manually generated ground-truth type and the message type extracted from each methodology. Conciseness is always based on the number of ground truth message types. If the message types extracted using each methodology are more than the number of ground truth message types, the result is a percentage of ground truth message types among the extracted message types. In the opposite case, the result is the percentage of the extracted message types among the ground truth message types.

Coverage evaluates the ability to cover all messages when a message type is extracted. The proposed method and Netzob first classifies the message type. Therefore, the proposed method and Netzob define the classification of only one message in the message type classification as uncoverable. This is because no field can be extracted if one message is classified into one type. AutoReEngine defines messages that cannot be covered by extracted fields.

$$\text{Conciseness} = \frac{\textit{The number of extracted message types}}{\textit{The number of ground truth message types}},$$
$$(\#\text{GT message types} > \#\text{extracted message types})$$

$$\text{Conciseness} = \frac{\textit{The number of ground truth message types}}{\textit{The number of extracted message types}},$$
$$(\#\text{GT message types} < \#\text{extracted message types})$$

$$\text{Coverage} = \frac{\textit{The number of covered messages}}{\textit{The number of messages}},$$

Table 4 represents the conciseness value of the Netzob, AutoReEngine, and the proposed method. The conciseness results indicate that the message types are classified most similar to the number of ground-truth message types. For

TABLE 4 Conciseness (proposed method vs. Netzob vs. AutoReEngine)

	Netzob	AutoReEngine	Proposed method
Modbus/TCP			
Connection	**36.36%** (32/88)	**62.75%** (32/51)	**74.41%** (32/43)
Transfer EWS to PLC	**27.27%** (66/242)	**57.89%** (66/114)	**98.48%** (65/66)
Transfer PLC to EWS	**33.64%** (74/220)	**19.27%** (74/384)	**71.62%** (53/74)
Ethernet/IP			
Connection	-	**40%** (29/72)	**90.63%** (29/32)
Transfer EWS to PLC	-	**28.7%** (28/99)	**96.43%** (27/28)
Transfer PLC to EWS	-	**56.06%** (19/34)	**76%** (19/25)
FTP			
	36.17% (17/47)	**97.92%** (47/48)	**74.60%** (47/63)

Modbus/TCP, it does not categorize in more detail than Netzob and AutoReEngine or subdivide the message type with. Netzob cannot analyze Ethernet/IP Protocol. Netzob uses a large amount of system resources because of the amount of data entered and the length of the data. Although Ethernet/IP protocol has less data input than Modbus/TCP protocol, it was not analyzed because Ethernet/IP payload contents are high. Although AutoReEngine was able to quickly analyze Ethernet/IP, the same message type was often divided into different message types. For the FTP protocol, AutoReEngine performed best. This is because the proposed method was developed for industrial protocols that do not have many variable lengths. However, the proposed method showed better performance than Netzob and confirmed that the common protocol FTP could also inference of the protocol structure.

Table 5 represents the coverage value of the Netzob, AutoReEngine, and the proposed method. The results of coverage show that the message types categorized by the proposed method contain the most messages. Netzob and AutoReEngine have high coverage, but given the result of conciseness, the other methods could see high coverage because the other methods deduced a protocol structure that was not detailed. The proposed method also showed the highest level of coverage for FTP, a commercial protocol.

Given only the results of coverage, it can be interpreted that the proposed method does not show values that differ from Netzob and AutoReEngine. However, when analyzed in combination with the result of conciseness, the proposed method is more similar to the correct message types than to other methods, and it can be confirmed that the content of the categorized message types contains the most messages.

As we can observe from the results of the experiment, our proposed method exhibits higher performance compared to existing methods. Netzob and AutoReEngine extract too many message types based on conciseness. These results can present several challenges in protocol structure analysis. In addition, the proposed method covered more messages

TABLE 5 Coverage (proposed method vs. Netzob vs. AutoReEngine)

	Netzob	AutoReEngine	Proposed method
Modbus/TCP			
Connection	**97.69%** (3425/3506)	**89.67%** (3144/3506)	**100%** (3506/3506)
Transfer EWS to PLC	**97.21%** (4397/4523)	**93.35%** (4222/4523)	**99.85%** (4516/4523)
Transfer PLC to EWS	**98.1%** (7389/7532)	**79.73%** (6005/7532)	**99.99%** (7531/7532)
Ethernet/IP			
Connection	-	**84.62%** (253/299)	**92.31%** (276/299)
Transfer EWS to PLC	-	**93.18%** (874/938)	**97.44%** (914/938)
Transfer PLC to EWS	-	**90.8%** (612/674)	**100%** (674/674)
FTP			
-	**82.53%** (940/1139)	**84.99%** (968/1139)	**88.94%** (1013/1139)

than Netzob and AutoReEngine. Therefore, the proposed method to make inferences the message types that include additional messages. And extracted message types are concise.

5 | CONCLUSION AND FUTURE WORK

Industrial protocols employed in industrial sites and infrastructure do not use standardized protocols but mostly independently developed protocols. As most of the independently developed protocols cannot be disclosed for reasons such as threats, it is very difficult to extract control device configuration information using the protocol's specifications or protocols. In this paper, a method is proposed to analyze the structure of the closed-door protocol for industrial use. This methodology can be used to enable efficient monitoring of network traffic for industrial protocols. Experiments have demonstrated that existing protocol reverse engineering methods exhibit several limitations in analyzing industrial protocols.

This methodology consists of six modules: traffic collection, message extraction, message clustering based on size, message clustering based on similarity, field extraction, and session analysis. The mean-shift algorithm was used for message clustering based on the similarity module. This methodology demonstrated that the structural analysis of the data part of the Modbus/TCP protocol, which was achievable. The proposed method contains the most messages from the industrial private protocol through the results of the experiment and classifies them into the simplest message types. This means that the most accurate message types can be classified, and information can be extracted about industrial process commands and so on, which are passed through traffic based on the categorized message types. This process enables monitoring of network traffic in industrial processes and can prevent network security threats.

As future work, we intend to apply more diverse industrial protocols to develop systems that can be used in industrial sites. In addition, we aim to develop a system that is applicable to commercial, private, and industrial protocols.

ACKNOWLEDGEMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2018R1D1A1B07045742) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government (MSIT) (No. 2018-0-00539, Development of Blockchain Transaction Monitoring and Analysis Technology).

ORCID

Kyu-Seok Shim https://orcid.org/0000-0002-3317-7000
Young-Hoon Goo https://orcid.org/0000-0002-3013-7011
Min-Seob Lee https://orcid.org/0000-0003-4854-9521
Myung-Sup Kim https://orcid.org/0000-0002-3809-2057

REFERENCES

1. Zetter K. Attack code for SCADA vulnerabilities released online. http://www.wired.com/threatlevel/2011/03/scada-vulnerabilities/, 2011.
2. Spenneberg R, Brüggemann M, Schwartke H. PLC-blaster: a worm living solely in the PLC. *Black Hat Asia*. 2016;16:1–16.
3. Langner R. Stuxnet: dissecting a cyberwarfare weapon. *IEEE Security & Privacy*. 2011;9(3):49-51.
4. Tridgell A. (2003). How Samba was written. [Online]. Available: http://samba.org/ftp/tridge/misc/french_cafe.txt
5. Pidgin. (2018). About Pidgin. [Online]. Available: http://www.pidgin.im/about
6. Caballero J, Song D. Automatic protocol reverse-engineering: message format extraction and field semantics inference. *Int J Comput Telecommun Netw*. 2013;57(2):451-474.
7. Liu M, Jia C, Liu L, Wang Z. Extracting sent message formats from executables using backward slicing. Proc. 4th Int. Conf. Emerg. Intell. Data Web Technol., X'ian, China, Sep. 2013, pp. 377–384.
8. Wang Y, Yun X, Shafiq MZ et al A semantics aware approach to automated reverse engineering unknown protocols. In: Proc. 20th IEEE Int. Conf. Netw. Protocols (ICNP), Oct. 2012, pp. 1–10.
9. Krueger T, Gascon H, Kramer N, Rieck K Learning stateful models for network honeypots. In: Proc. 5th ACM Workshop Secur. Artif. Intell., Raleigh, NC, USA, Oct. 2012, pp. 37–48.
10. Li H, Shuai B, Wang J, Tang C Protocol reverse engineering using LDA and association analysis. In: Proc. 11th Int. Conf. Comput. Intell.Secur. (CIS), Dec. 2015, pp. 312–316.
11. Beddoe MA. Network protocol analysis using bioinforomatics algorithms; 2004. [Online]. Available: http://www.4tphi.net/~awalters/PI/pi.pdf

12. Leita C, Mermoud K, Dacier M. ScriptGen: an automated script generation tool for Honeyd. In: Proc. 21st Annu. Comput. Secur. Appl. Conf., Tucson, AZ, USA, Dec. 2005, p. 2.
13. Cui W, Kannan J, Wang HJ. Discoverer: automatic protocol reverse engineering from network traces. In: Proc. 16th USENIX Secur. Symp., Boston, MA, USA, Aug. 2007, pp. 199–212.
14. Bossert G Exploiting semantic for the automatic reverse engineering of communication protocols. Ph.D. dissertation, Univ. Gif-sur-Yvette, Rennes, France, Dec. 2014.
15. Wang L, Jiang T. On the complexity of multiple sequence alignment. *J Comput Biol*. 1994;1(4):337-348.
16. Luo J-Z, Yu S-Z. Position-based automatic reverse engineering of network protocols. *J Netw Comput Appl*. May 2013;36(3):1070-1077.
17. Wang Y, Zhang N, Wu Y-M., Su B-B, Liao Y-J. Protocol formats reverse engineering based on association rules in wireless environment. In: Proc. 12th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun. Melbourne, VIC, Australia, Jul. 2013, pp. 134–141.
18. Ji R, Li H, Tang C Extracting keywords of UAVs wireless communication protocols based on association rules learning. In: Proc. 12th IEEE Int. Conf. Comput. Intell. Secur., Wuxi, China, Dec. 2016, pp. 310–313.
19. Bermudez I, Tongaonkar A, Iliofotou M, Mellia M, Munafo MM. Automatic protocol field inference for deeper protocol understanding. In: Proc. 14th IFIP Netw. Conf., Toulouse, France, May 2015, pp. 1–9.
20. Ladi G, Buttyan L, Holczer T Message format and field semantics inference for binary protocols using recorded network traffic. In: Proc. 26th Int. Conf. Softw., Telecommun. Comput. Netw., Split, Croatia, Sep. 2018
21. Stouffer K, Falco J, Scarfone K. Guide to industrial control systems (ICS) security. *NIST Special Publication*. 2011;800(82):16-16.
22. Shim K-S, Goo Y-H, Lee M-S, Hasanova H, Kim M-S Inference of network unknown protocol structure using CSP (contiguous sequence pattern) algorithm based on tree structure. Proc. of the NOMS 2018—IEEE/IFIP DISSECT workshop, Taipei, Taiwan, April. 23, 2018, pp. 1–4.
23. Davidson CC, Andel T, Yampolskiy M, McDonald JT, Glisson B, Thomas T (2018). On SCADA PLC and Fieldbus Cyber-Security. In: 13th International Conference on Cyber Warfare and Security pp. 140–149.
24. Van Herrewege A, Singelee D, Verbauwhede I. CANAuth—a simple, backward compatible broadcast authentication protocol for CAN bus. ECRYPT Workshop on Lightweight Cryptography. Vol. 2011. 2011.
25. Thompson S. Application of controller area network bus and CANopen protocol in Industrial Automation. Diss. Murdoch University, 2018.
26. Murvay P-S, Groza B. A brief look at the security of DeviceNet communication in industrial control systems. Proceedings of the Central European Cybersecurity Conference 2018. 2018.
27. Fovino IN, Carcano A, Masera M, Trombetta A. Design and implementation of a secure modbus protocol. In: *International Conference on Critical Infrastructure Protection*. Berlin, Heidelberg: Springer; 2009.
28. Suzuki K, Chino S, Sakurada H, Tarui I, Ban N, Charles P FDT technology for CC-link network. SICE Annual Conference 2011. IEEE, 2011.
29. Höfken H, Paffen B, Schuba M. ICS/SCADA security analysis of a Beckhoff CX5020 PLC. 2015 International Conference on Information Systems Security and Privacy (ICISSP). IEEE, 2015.
30. Langlois K, van der Hoeven T, Rodriguez Cianca D, et al. Ethercat tutorial: an introduction for real-time hardware communication on windows [tutorial]. *IEEE Robot Autom Mag*. 2018;25(1):22-122.
31. Faisal MA, Cardenas AA, Wool A. Profiling communications in industrial IP networks: Model complexity and anomaly detection. In: *Security and Privacy Trends in the Industrial Internet of Things*. Cham, Switzerland: Springer; 2019:139-160.
32. Feld J. PROFINET—scalable factory communication for all applications. IEEE International Workshop on Factory Communication Systems, 2004. Proceedings IEEE, 2004.
33. Goldenberg N, Wool A. Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *Int J Crit Infr Prot*. 2013;6(2):63-75.
34. Jain AK. Data clustering: 50 years beyond K-means. *Pattern Recognit Lett*. 2010;31(8):651-666.
35. Cumani S, Laface P. Exact memory–constrained UPGMA for large scale speaker clustering. *Pattern Recognit*. 2019;95:235-246.
36. Reddym CK, Bhanukiran B. A survey of partitional and hierarchical clustering algorithm. In: *Data Clustering*. Boca Raton, Florida, United States: Chapman and Hall/CRC; 2018:87-110.
37. Shim KS, Yoon SH, Lee SK, Kim SM, Jung WS, Kim MS. Automatic generation of snort content rule for network traffic analysis. *KICS*. 2015;40(04):666-677.

AUTHOR BIOGRAPHIES

Kyu-Seok Shim (kusuk007@korea.ac.kr) is a visiting professor in the Department of Computer and Information Science, Korea University, Korea. He received his B.S., M.S. and Ph.D. degrees in the Department of Computer Science and the Department of Computer and Information Science, Korea University, Korea, in 2014, 2016 and 2020, respectively. His research interests include Internet traffic classification, network management and protocol reverse engineering.

Young-Hoon Goo (gyh0808@korea.ac.kr) is a combined M.S.–Ph.D. degree student in the Department of Computer and Information Science, Korea University, Korea. He received his B.S. degree in the Department of Computer Science, Korea University, Korea, in 2016, respectively. His research interests include Internet traffic classification, network management and protocol reverse engineering.

Min-Seob Lee (chenlima2@korea.ac.kr, minseob.lee@ahnlab.com) is a junior developer in Ahnlab. He received B.S. and M.S. degrees in computer science from Korea University, Korea, in 2018 and 2020, respectively. He joined Ahnlab in 2020. His research interests include Internet traffic monitoring and analysis.

Myung-Sup Kim (tmskim@korea.ac.kr) is a professor in the Department of Computer and Information Science, Korea University, Korea. He received his B.S., M.S., and Ph.D. degrees in Computer Science and Engineering from POSTECH, Korea, in 1998, 2000, and 2004, respectively. From September 2004 to August 2006, he was a postdoctoral fellow in the Department of Electrical and Computer Engineering, University of Toronto, Canada. He joined Korea University in September 2006. His research interests include Internet traffic monitoring and analysis, service and network management, and Internet security.

How to cite this article: Shim K-S, Goo Y-H, Lee M-S, Kim M-S. Clustering method in protocol reverse engineering for industrial protocols. *Int J Network Mgmt*. 2020;e2126. https://doi.org/10.1002/nem.2126