Enhance the ICS Network Security Using the Whitelist-based Network Monitoring Through Protocol Analysis

Kyu-Seok Shim, Ilkwon Sohn, Eunjoo Lee, Woojin Seok and Wonhyuk Lee*

National Institute of Supercomputing and Networking Advanced KREONET Center, Korea Institute of Science and Technology Information, Daejon, Korea
E-mail: kusuk007@kisti.re.kr; d2estiny@kisti.re.kr; saranha@kisti.re.kr; wjseok@kisti.re.kr; livezone@kisti.re.kr
**Corresponding Author*

Received 09 September 2020; Accepted 31 October 2020;
Publication 15 February 2021

Abstract

In our present technological age, most manual and semi-automated tasks are being automated for efficient productivity or convenience. In particular, industrial sites are rapidly being automated to increase productivity and improve work efficiency. However, while networks are increasingly deployed as an integral part of the automation of industrial processes, there are also many resultant dangers such as security threats, malfunctions, and interruption of industrial processes. In particular, while the security of business networks is reinforced and their information is not easily accessible, intruders are now targeting industrial networks whose security is relatively poor, wherein attacks could directly lead to physical damage. Therefore, numerous studies have been conducted to counter security threats through network traffic monitoring, and to minimize physical loss through the detection of malfunctions. In the case of industrial processes, such as in nuclear facilities and petroleum facilities, thorough monitoring is required as security issues

Journal of Web Engineering, Vol. 20_1, 1–32.
doi: 10.13052/jwe1540-9589.2011

can lead to significant danger to humans and damage to property. Most network traffic in industrial facilities uses proprietary protocols for efficient data transmission, and these protocols are kept confidential because of intellectual property and security reasons. Protocol reverse engineering is a preparatory step to monitor network traffic and achieve more accurate traffic analysis. The field extraction method proposed in this study is a method for identifying the structure of proprietary protocols used in industrial sites. From the extracted fields, the structure of commands and protocols used in the industrial environment can be derived. To evaluate the feasibility of the proposed concept, an experiment was conducted using the Modbus/TCP protocol and Ethernet/IP protocol used in actual industrial sites, and an additional experiment was conducted to examine the results of the analysis of conventional protocols using the file transfer protocol.

Keywords: Network security, protocol analysis, traffic monitoring, ICS network, clutsering algorithm, Apriori algorithm.

1 Introduction

In recent years, there have been many studies focusing on the automation of manual or semi-automated tasks in various fields. These generally lead to higher production efficiency, lower defect rates, and/or cost reduction due to the increased speed of operation. The most popular field for these studies has been industrial automation. The most critical element when implementing automation in an industrial environment is the communication network. An industrial control system (ICS) typically issues commands to devices connected via a network, and each device functions according to the commands it receives. It therefore follows that the network is an essential element in industrial automation.

In an ICS environment, a programmable logic controller (PLC) is the component that controls each device by issuing the commands. The ICS configures a logic circuit for each device using a PLC and communicates this information via the network. Most network protocols used by PLCs are developed by the ICS vendors and are designed to achieve effective data transmission. Most vendors do not disclose network protocol specifications due to intellectual property and security reasons [1–3]. Therefore, an ICS environment manager cannot monitor industrial processes or extract information related to them by analyzing the network traffic. Currently, the only network monitoring solutions available are those provided by the ICS

vendors, but these are usually very costly, and can even create security threats if they are not updated frequently.

Although an ICS/Supervisory Control and Data Acquisition (SCADA) system can be created as an isolated internal network environment, it is often still possible for an attacker to gain access through various paths. Since an engineering workstation (EWS) can typically be accessed from a normal office computer, a PLC project file can be infected from the connection point where the EWS is installed, and it may then be vulnerable to an attack. Other attack paths include connection points for remote maintenance and diagnosis, log devices, remote access modems, wireless systems, and USB ports. Therefore, to minimize the probability of such attacks, it is necessary to detect malicious behavior by monitoring the ICS/SCADA network and protecting it by implementing whitelist-based network management. An ICS/SCADA administrator must continually monitor the ICS processes to prevent security threats and to verify the operational status of the ICS and all its industrial control devices. A user-generated logic circuit is configured on an EWS and transmitted to the ICS using a proprietary protocol. Therefore, it is necessary to understand the structure of the ICS/SCADA network protocol in order to monitor the settings and commands sent to and from the ICS, and thereby attempt to prevent security breaches [4].

The ICS network environment should also consider scalability. Nodes can be added at any time in the industrial field. The manager should inspect the added equipment for hazards and determine if it can be performed normally. That is, the manager must be understand of the information on all equipment in the management area. However, the current administrator cannot know the normal range of transmitted data through the network because there is no information on the network protocol. In this paper, we propose the protocol analysis based on whitelist network traffic in the industrial management area. Through this proposed method, the administrator can grasp the normal range of network traffic within the industrial management area and detect abnormal traffic and threats using this information.

Existing techniques for analyzing the structure of confidential protocols generally employ passive analysis methods [5–21]. These passive analysis methods collect the data traffic of confidential protocols and identify the location and structure of the commands within each message. However, in modern network environments, the protocol used by each ICS is different, the message structure may be changed according to the network environment, and the protocol structure may be continuously updated for efficient data transmission. Therefore, due to the limitations of passive analysis methods, a

number of studies have been conducted to derive methods that can automatically analyze confidential protocols. These automatic analysis methods can be classified into two main types. The first is a top-down type that extracts messages from the protocol, classifies similar messages, and deduces fields from the messages in the same group. Netzob is one of the most widely used top-down methodologies. The second is a bottom-up type that extracts messages from the protocol, extracts common fields from the messages, and deduces the message type using the extracted fields. A popular example of the bottom-up methodology is AutoReEngine.

In contrast to commercial network protocols, an ICS network protocol usually exchanges all the information in a single flow instead of multiple flows. In the case of a commercial protocol, there is a common field for each message and a new flow is created whenever new information is exchanged. Therefore, a commercial network protocol can group common messages through multiple flows and it is easy to identify the sequence of messages of the same type. However, analyzing an ICS protocol requires locating common fields in a single flow. Therefore, in this study we propose a method to determine a common field, which is the most important factor in deducing the structure of an industrial protocol. The proposed method collects more than two message flows, performs pre-processing that extracts messages from the collected data, and then clusters messages of the same type using the k-means algorithm. Common fields are then extracted from the clustered messages using the Apriori algorithm [22] and messages of the same type are merged using these extracted fields.

This paper also covers related research in Section 2. In Section 3, the proposed method is described, and its validity is proven through experimentation. The results of the experiments are presented in Section 4, and the final section presents conclusions and suggestions for future studies.

2 Related Work

This study concentrates on two areas: The first is ICS/SCADA security issue, and the second is automatic protocol reverse engineering. As industrial fields became automated, there have been continuous security issues and the number of issues has been constantly increasing. This study examines such security issues. Also, protocol automatic reverse engineering refers to a system that automatically processes tasks to derive the structure and specifications of a private protocol.

2.1 ICS/SCADA Security Issue

As industrial systems become more automated, security concerns have grown and the number of issues has been escalating. In the past, ICS/SCADA environments have usually been operated in a completely closed network. More recently, however, due to the combination of the Internet with sensors or actuators (IoT – Internet of Things), data-based operation has become feasible, leading to improved efficiency but also increased security threats. In other words, though connections to the outside world have been increased to enhance operational convenience and work efficiency, these connections are also being exploited as external intrusion routes.

There have been various cases of ICS security breaches. In 2010, some functions of the centrifuge at a nuclear power plant in Iran were stopped due to a PLC infection. The infection was caused by hackers who had acquired administrator credentials using Stuxnet, which is a malicious computer worm, in order to exploit the vulnerability of the system. Since that incident, attacks targeting energy industries such as the oil industry, nuclear facilities, nuclear power, and water resources, have been steadily increasing, including the Saudi Aramco hacking in 2012, the Ukrainian power grid shutdown in 2015, and ransomware attacks on one of the Michigan municipal utilities in the United States in 2016. These incidents are linked by the tremendous human danger and financial damage as a result of ICS security being compromised. Therefore, ensuring the security of ICS/SCADA networks is essential to prevent the threat of attack, rather than having to deal with the consequences. To achieve this, the ICS/SCADA network environment should be under continuous monitoring and must provide immediate controllability. One way to provide such security is by using a whitelist-based network control system.

Whitelist-based network control allows the passage of valid network traffic and blocks all other traffic based on its detection and identification. Therefore, when using this method it is important to be able to classify and detect valid traffic. To identify valid traffic in an industrial network, an administrator would need information regarding the industrial network protocol structure. However, the network protocol depends on the ICS equipment being used, which invariably uses a protocol developed directly by the ICS vendor. These protocols are designed to transmit data effectively to the ICS equipment in an encrypted or binary form to ensure security. In addition, users do not have access to the protocol structure because it is generally not disclosed by the vendor. Therefore, users must be able

to deduce information about the ICS protocol structure in order to implement whitelist-based network monitoring. This paper proposes a method of deducing the ICS network protocol structure.

2.2 Automatic Protocol Reverse Engineering

Automatic protocol reverse engineering refers to a system that automatically analyses tasks to derive the structure and specifications of a confidential network protocol. Existing protocol reverse engineering methods deduce a protocol structure through a manual analysis process. However, the limitation to using passive methods is that these protocol structures are all diversified and periodically updated. Therefore, there have been studies conducted to develop an automatic method to deduce a protocol structure. Some of the most widely used automatic protocol reverse engineering methods include Netzob and AutoReEngine.

Netzob uses a semi-automatic methodology that automates some of the inference processes of protocol architecture, as proposed by Bosert et al. in 2011 [14]. The purpose of Netzob is to infer a protocol structure automatically. To achieve this, Netzob clusters similar types of messages through the unweighted pair group method with arithmetic (UPGMA). It receives a threshold value input by a user to determine the degree of similarity. The clustered messages are defined as symbols for each group. In other words, if the messages are clustered into three groups, a total of three symbols are created. A symbol denotes a set of messages having the same format and function in terms of the protocol. Each symbol extracts common strings using the Needleman-Wunsch algorithm. The common strings are defined as a static field and the remaining messages are defined as a dynamic field. A field represents a set of tokens that share a common meaning in terms of protocol. A symbol consists of static and dynamic fields, and each field can be unique or have multiple values.

AutoReEngine receives network traffic for a single protocol as input to the methodology proposed by Luo et al. in 2013 [16]. AutoReEngine consists of four main steps: data pre-processing, protocol keyword extraction, message format extraction, and state machine inference. In the data pre-processing step, the input traffic is assembled in a flow form, and the packets in a flow are reassembled into messages. The frequent protocol keyword extraction step is executed in two sub-steps. The first step is to enter a sequence of messages and extract a field-format candidate keyword by using the Apriori algorithm. The Apriori algorithm extracts a keyword that satisfies the minimum support

of the maximum length, by deleting all candidate sets except for those satisfying the minimum support among the candidates with length 1 or more. AutoReEngine defines length 1 (L1) as 1 byte and extracts strings. The units of support include session support rate (Rssr) and site-specific session set support rate (Rset). Rssr is the ratio of candidate sequences that contain candidate sequences to the total flows, and Rset represents the ratio of site-specific sessions that contain candidate sequences to the total site-specific sessions. A site-specific session refers to a set of flows with the same server. A byte sequence, which is a nested group of the final collection of all extracted items, is extracted using the support values and sequences, and a common string is defined for this byte sequence.

2.3 Problem Scope

Netzob and AutoReEngine are not specifically designed for industrial protocols. Therefore, they both have limitations in analyzing the confidential protocol structure of an industrial protocol.

Netzob is a top-down method for protocol reverse engineering. Similar to Netzob, the proposed method is also a top-down model as it extracts fields after classifying messages by their format. However, UPGMA, the core algorithm of Netzob, determines similarity by recursively comparing two messages and therefore, has a high computational and time complexity. In fact, Netzob may sometimes be unable to derive results or may take a long time, depending on the amount, length, and similarity of the input data. In other words, it is heavily data-dependent.

AutoReEngine employs a bottom-up model that extracts and combines fields to derive a message structure. It is a different model from the proposed method but similar in its use of the Apriori algorithm to find a common field in a message type. AutoReEngine searches for a common field from all messages. Therefore, a user must set an appropriate threshold whenever the system is executed. This kind of user intervention can be seriously detrimental to the system results. If the threshold is not appropriate, correct fields cannot be extracted, and consequently a correct message structure cannot be inferred.

The proposed method divides message clustering into two parts to overcome the limitations described earlier. The first part is to classify messages by size. Due to the characteristics of the ICS protocol, there are not many fields with a variable length; thus, a message of the same size is most likely to be a message of the same message type. However, there may be

other formats of messages even if their size is the same; hence, the second part needed is a clustering process to measure similarity. In addition, the k-means algorithm, which is a clustering algorithm, uses messages of the same length; thus, it does not require a process of lowering accuracy such as padding, and it does not perform the clustering algorithm for all the messages, which would require a high computational complexity. Therefore, it has the advantage of lowering system-wide computational complexity. In addition, user intervention is not required, because the support value of the Apriori algorithm can be set to 100% when extracting fields from grouped messages.

3 Proposed Method

In this section, a field extraction method is proposed to infer an industry protocol structure. The preliminary step before extracting fields consists of message extraction, message classification by size, clustering based on message type, and merging based on message type. A message classified by type goes through a field extraction step for each type. After field extraction, the meaning of each field is determined.

3.1 Overview

Figure 1 is overview of the proposed method. The structure of the proposed method consists of pre-processing and main processing as follows. The purpose of pre-processing is to extract messages and classify them according to type. The pre-processing task consists of two steps: message extraction and message classification by message size. In the main stage, a static field that is commonly found in the messages classified by type is searched for and extracted. The static field extraction in the main stage can be a command or setting included in an industrial protocol. And the extracted static fields are used to merge the message types. The message type, the field containing a message, and the meaning of a field used in the industrial protocol can be inferred through the pre-processing and main processing as follows.

3.2 Pre-processing of Protocol Analysis

The pre-processing stage of message field extraction consists of message extraction, message classification by size, and message type clustering.

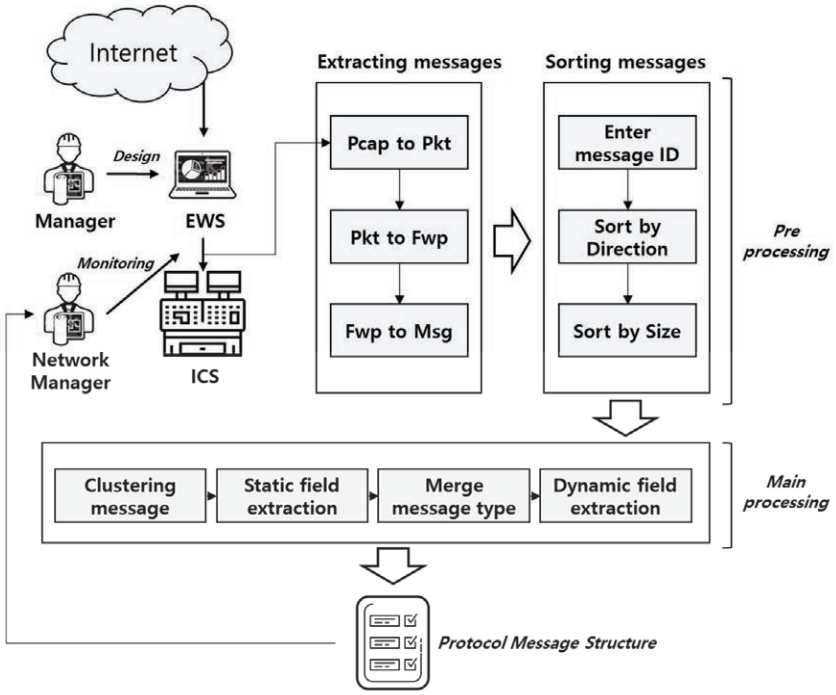


Figure 1 Overview of the proposed method.

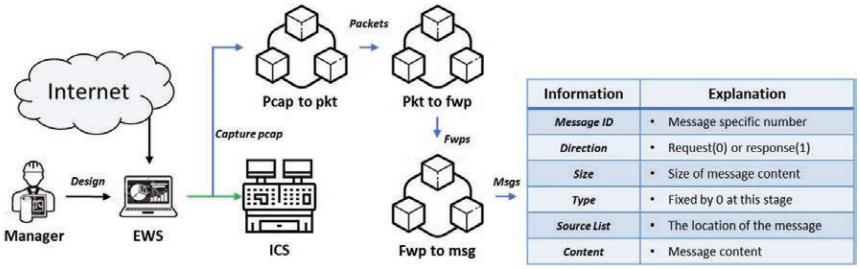


Figure 2 The process of message extraction.

3.2.1 Message extraction

The message extraction step is the step to extract the message from the collected protocol traffic for analysis. At this stage, extracting a message refers to defining a packet as a single message and distinguishing its direction. The extracted message contains six pieces of information, as listed in Figure 2.

Message ID refers to an ID unique to a message. In essence, the message ID cannot be the same as that of another message. The Message ID is information that distinguishes a message from other messages. Direction is information that distinguishes between messages sent from PLC to EWS and messages sent from EWS to PLC. Message size is the size of the content that the actual message contains. This information is necessary because the first step after extracting the message is to cluster the message based on size. Message type is information for storing the type of paper each time you cluster. Message source list stores flow and packet information on where messages are located. Message content stores content containing messages.

3.2.2 Sorting messages

The extracted messages are the first message type to be determined by size in the classification phase based on message size. This process first aligns the extracted messages by size. Aligned messages are formatted from small messages. There are two reasons for classifying by size. First, most industrial protocol messages do not have variable lengths. In essence, messages of the same size are mostly of the same type. Second, clustering algorithms are computational time complexity algorithms. Otherwise, all messages would immediately trigger the clustering algorithms. In such a case, not only is it incorrectly classified but it also causes system overload in the form of excessive CPU memory usage. Therefore, implementing the primary classification before the clustering algorithm is performed can solve these problems.

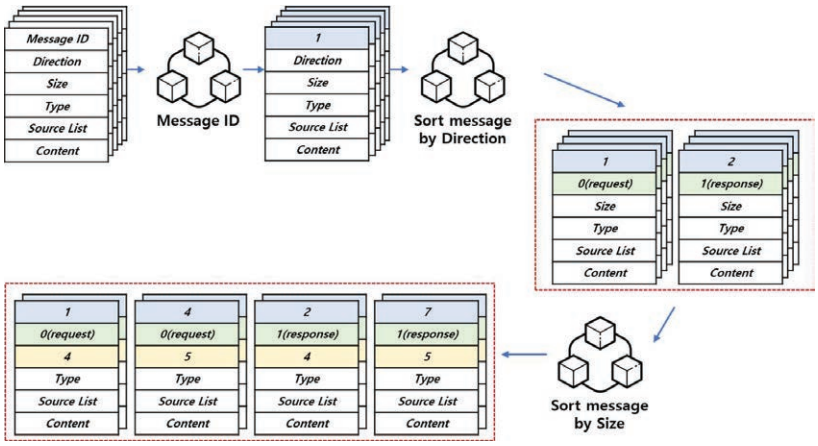


Figure 3 The process of sorting messages.

3.2.3 Clustering messages based on similarity

Messages classified by size are mostly of the same type but may include different types. Therefore, it is necessary to refine the type of message by determining the similarity of the messages. The algorithm used in this step to determine the similarity of messages is mean shift. The mean shift algorithm is a centroid-based algorithm whose goal is to find the center point of each group and it operates by updating the candidate for the center point with the average of the points in a sliding window. The mean shift algorithm clusters messages quickly and accurately [36].

At this stage, several conditions are required for selecting clustering algorithms for classifying types. Because the first is to reclassify messages classified by size, it is not known how many groups of sizes are clustered based on similarity measurement. Thus, algorithms that require user input, such as the K-means algorithm that should be classified into N groups, are not available. The elbow method is typically used to address these shortcomings. The Elbow method is an algorithm that selects maximum k. Second, the clustering algorithm should be selected as quickly and accurately as possible because each message classified by size needs to be clustered. The clustering algorithm used by Netzob is UPGMA. UPGMA is an algorithm that compares each message peer to peer in order to group multiple messages [34, 35]. Consequently, the time and calculation required are high. In the case of the Mean-Shift algorithm, it is not necessary to provide k as in the k-means algorithm before clustering; however, a user must configure the region of interest (ROI). Therefore, user intervention is required, as in other algorithms. In the methodology proposed in this study, the k-means algorithm with relatively low time complexity is selected, and k is determined through the elbow method. At this time, k is a high value; hence, it can be classified in detail, and handled owing to post-processing by a message type merging step.

3.3 Message Field Extraction

The step of extracting the message field involves the steps of extracting common character columns by message type, extracting fields, and merging message types using the extracted fields. It also deduces the meaning of the extracted fields.

3.3.1 Static field extraction based on message type

Field extraction steps extract the static and dynamic fields of messages. Static field refers to a series of strings found in common in messages of the same

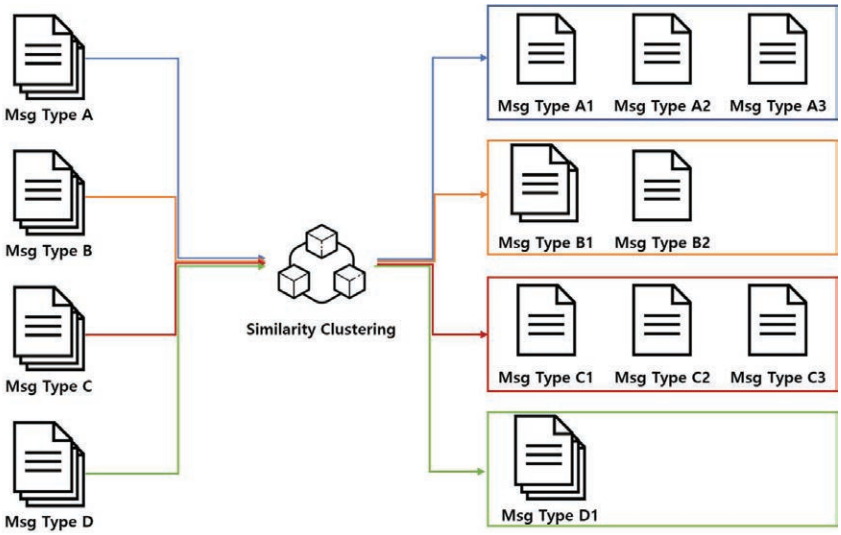


Figure 4 Clustering messages based on similarity.

type. A dynamic field is any part of a message of the same type except for a series of strings found in common. Therefore, the message type consists of static fields and dynamic fields.

At this stage, the static field is extracted using a CSP algorithm [37]. CSP algorithm is an algorithm that extracts common strings based on the Apriori algorithm. The static field extraction process using the common spatial pattern (CSP) algorithm extracts messages of the same type as a set of sequences, and generates content of length 1 from a set of sequences. The contents of length 1 are divided into content that is not satisfied by minimum support through minimum support check and content that is satisfied. Content that does not meet the minimum support level will be deleted and content that is satisfied will be created as content with a length of 2. This process is repeated until the length no longer increases. Figure 5 outlines an example of the process of extracting static fields.

The CSP algorithm requires minimum support. Minimum support is a condition under which candidate content can be extended to the next length. In this study, the minimum support level is always set to 100%. 100% means a string that is extracted in common from all messages of the same type, i.e., it is the static field that extracts all the strings common to the same type of message. The dynamic field is the rest of the message in the same type of message except for a series of strings found in common. Each message may

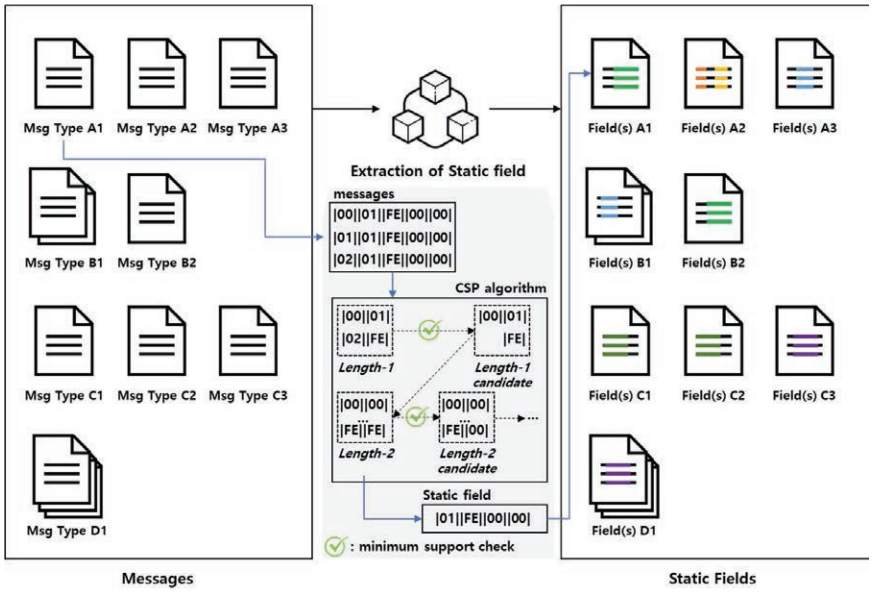


Figure 5 Extraction of static field.

have the same string, or another string. In essence, a variable string is defined by the dynamic field.

3.3.2 Message type merge based on static fields

Messages that have completed the classification phase based on message type are mostly categorized by type. Categorized messages are used to extract message fields. However, with the presence of fields with variable length, messages of the same type but of different sizes were not considered. Thus, the merger phase based on this type of message has different sizes but combines the same type of message into one type.

Messages of the same type but of different sizes cannot be used to determine the similarity of messages due to their different sizes. Therefore, to find the same message types with different sizes, we use the fields that were extracted from the previous step. We match the content and order of the fields extracted from each type to find the types of messages containing variable lengths.

This phase requires several conditions to merge message types, as outlined in Figure 6. First, the message types being merged should have the same number of extracted fields. The most basic condition should be: the

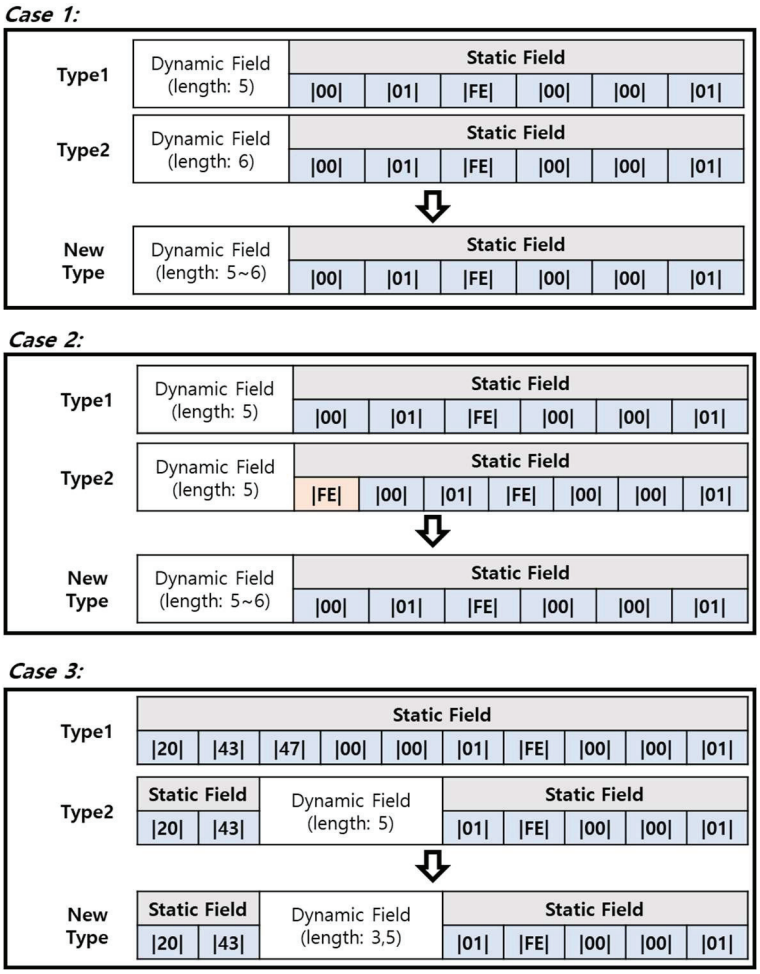


Figure 6 Merge of message types.

same number of extracted fields if they are of the same type. Second, the fields extracted from the message types being merged should be included. If the preceding two conditions are met, the two message types are merged into one message type. However, the nature of the message may also be a coverage relationship, although the number of extracted fields varies. For example, if one message is contained in a single message type, the content of the message is defined as a static field. When two or more messages from different message types are included, such that two or more static fields are

extracted and extracted static fields are included in one message, the two types should be merged.

4 Experiment and Result

In this section, the performance of the proposed method is evaluated. Two protocols were selected to evaluate the performance of this method. One protocol is Modbus/TCP and the other is FTP which is not an industrial protocol. The reason for selecting FTP as a target protocol is because objective evaluation can be conducted, due to the fact that the protocol specifications have already been disclosed. The structure of each protocol is explained and the traffic information collected for the experiment is described. The structure of each protocol is the type of protocol message and the commands included in the messages. In addition, the clustering results of each protocol are compared with the ground-truth results. The results of processing each protocol type using the existing protocol reverse engineering methods, Netzob and AutoReEngine, are compared with the result of the proposed method, in order to evaluate its performance.

4.1 The Structure of Ground Truth Message

4.1.1 The structure of Modbus/TCP message

Modbus/TCP is a typical industrial protocol that has a partially disclosed structure. The Modbus/TCP message consists of six fields, as shown in Figure 7. A transaction ID represents the sequence number of an operation related to query and response and a protocol ID has a fixed value of 0x0000. The length the message length after the length field and the unit ID field has a fixed value of 1 byte. Finally, the function code (FC) has a different value for each function of the Modbus/TCP protocol, and the data field contains the content of the message.

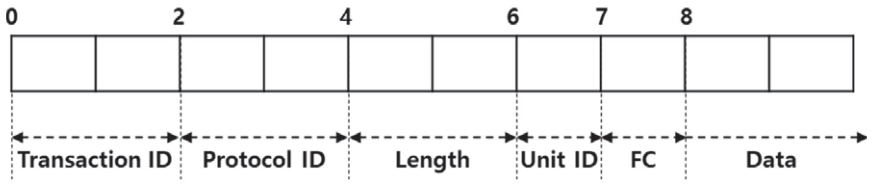


Figure 7 The structure of modbus/TCP.

The function code is a command set code provided by the Modbus/TCP protocol. The Modbus/TCP protocol allows users to read or write the values of slave memory (coil registers) by using the function codes. The traffic between the EWS and ICS equipment, which is the traffic collected in this experiment, always has a function code of 90. The goal of this method is to infer the data structure by extracting the data field from messages with a function code of 90.

4.1.2 The structure of file transfer protocol (FTP)

The FTP protocol transmits files between a server and a client based on TCP/IP. In this experiment, the structure of FTP, whose specifications are known, is inferred to replace the analysis results of an industrial protocol that does not have objective protocol specifications. In the FTP protocol, a request message mainly consists of command and content, and a response message consists of a number indicating the status or error of a command and its description.

A request message includes access commands, file management commands, file transfer commands, data format commands, port definition commands, and other commands. The access commands and file management commands are listed in Table 1. A response message displays a number indicating the status or error of a command in 3 bytes. The first byte indicates the command state, the second byte indicates the cause, and the final byte provides additional information. In addition, the text section contains mandatory parameters or additional descriptions. The first byte can be a number from 1 to 5: 1 indicates a server sending another response after a task starts before receiving another command; 2 indicates that a task is completed and another command is received; 3 indicates a command is received but additional information is required; 4 indicates that a command cannot be performed due to a temporary error, but the same command can be used later; and 5 indicates that a command cannot be executed and no reattempt is allowed. The second byte can be a number from 0 to 5. 0, representing grammar: 1 represents information, 2 represents connection, 3 represents authentication and account, 4 represents unspecified, and 5 represents file system.

Based on this information, the performance of the proposed method is evaluated by determining whether each command and its command status code are properly classified at the clustering step, and whether each command is properly extracted at the field extraction.

Table 1 Types of FTP request messages

Cmd Type	Cmd	Argument	Description
Access	USER	User ID	User Information
	PASS	User Password	Password
	ACCT	Account to be charged	Account Information
	REIN		Reinitialize
	QUIT		Logout of the system
	ABOR		Abort the previous command
File Management	CWD	Directory name	Change to another directory
	CDUP		Change to parent directory
	DELE	File name	Delete a file
	LIST	Directory name	List subdirectories of files
	NLIST	Directory name	List subdirectories of files without attributes
	MKD	Directory name	Create a new directory
	PWD		Display name of current directory
	RWD	Directory name	Delete a directory
	RNFR	File name (old)	Identify a file to be renamed
	RNTO	File name (new)	Rename the file
	SMNT	File system name	Mount a file system

Table 2 Information on collected traffic

	Modbus/TCP			FTP
Function	Connection	Transfer from EWS to ICS	Transfer from ICS to EWS	–
File	29	23	28	3
Flow	29	46	28	6
Packet	4,458	5,367	8,850	259
Bytes	571,314	1,419,958	2,168,519	21,022
Messages	3,506	4,523	7,532	88

4.2 The Information on Collected Traffic

To collect Modbus/TCP protocols, which are industrial protocols, various functions must be performed in EWS after establishing a connection between

EWS and ICS. Typical functions include connection with ICS, project transmission to ICS, and project reception from ICS. The connection function with ICS is the most basic function involved in connecting between EWS and ICS. The project transfer function to ICS is a function that sends projects set by EWS so that they can be performed in ICS. The function of receiving a project from the ICS is to receive project information that is being carried out by the existing ICS to the EWS.

We collect traffic for the following three typical functions and analyze the protocol structure. The traffic information collected is presented in Table 2.

4.3 The Result of Clustering Message Types

This section analyzes the clustering results for each protocol. First, define the correct message type for each protocol. Because there is no published information on the message types in the Modbus/TCP protocol, the message types are classified directly, and the classified results are compared. FTP is classified using command language and command code. Because the proposed method is clustering by separating the request message from the response message, even when classifying the message type directly, it classifies the direction and also classifies the type. Table 3 presents the number of ground truth message types for each protocol.

The proposed method is to receive a message and divide the direction. The messages are grouped by size, and because industrial protocols have few length variable fields, primary clustering is possible. Table 4 presents the results of clustering based on size.

The proposed method is for the same size in message types classified by size, but the message types are classified using the similarity clustering

Table 3 Number of ground truth message types

	Modbus/TCP			FTP
Function	Connection	Transfer from EWS to PLC	Transfer from PLC to EWS	–
Request	18	24	73	16
Response	8	62	11	11

Table 4 Result of clustering based on size

	Modbus/TCP			FTP
Function	Connection	Transfer from EWS to PLC	Transfer from PLC to EWS	–
Request	17	18	20	15
Response	6	10	6	24

Table 5 The result of clustering based on similarity

	Modbus/TCP			FTP
Function	Connection	Transfer from EWS to PLC	Transfer from PLC to EWS	–
Request	27	27	78	32
Response	16	68	12	22

Table 6 Message Type 5 in Modbus/TCP

Message Type	Content
5	O |FE| |07| |00| |00| |00| |00| |00| |00|
	P |FE| |07| |00| |00| |00| |00| |00| |00|
	Q |FE| |07| |00| |00| |00| |00| |00| |00|
	R |FE| |07| |00| |00| |00| |00| |00| |00|
	S |FE| |07| |00| |00| |00| |00| |00| |00|
	V |FE| |07| |00| |00| |00| |00| |00| |00|
	W |FE| |07| |00| |00| |00| |00| |00| |00|
	X |FE| |07| |00| |00| |00| |00| |00| |00|
	Y |FE| |07| |00| |00| |00| |00| |00| |00|
	Z |FE| |07| |00| |00| |00| |00| |00| |00|
	[|FE| |07| |00| |00| |00| |00| |00| |00|
	\ |FE| |07| |00| |00| |00| |00| |00| |00|

algorithm and the mean shift algorithm to classify messages of different types. Table 5 presents the result of message type classification using mean shift. Messages classified based on type using the mean shift algorithm are entered in the field extraction step.

4.4 The Extraction of Static Fields and Dynamic Fields

Messages classified by type are entered in the field extraction step, which finds common character columns in the messages entered for each type. Table 6 presents the messages included in Message Type 5 among Modbus/TCP protocol messages. The field extraction step extracts "|FE| |07| |00| |00| |00| |00| |00| |00|" into a fixed field with a common substring, and extracts "O, P, Q, R, S V, W, X, Y, Z, [, \" with dynamic field. Because dynamic fields occur sequentially, it is possible to simultaneously perform semantic input.

Table 7 Message types in FTP

Message Type	Content
4	CWD /home|0D||0A|
5	CWD |EB||AC||B8||EC||84||9C||0D||0A|
	CWD |EA||B3||B5||EA||B0||7A||0D||0A|
9	CWD /dev/cdrom|0D||0A|
13	CWD /home/kusuk007/|0D||0A|
14	CWD /home/kusuk007/|EB||AC||B8||EC||84||9C||0D||0A|
15	CWD /home/kusuk007/|EB||8B||A4||EC||9A||B4||EB||A1||93||0D||0A|

Table 8 Result of extraction of static fields in FTP messages

Message Type	Content
4	CWD /home|0D||0A|
5	CWD |EA||B3||B5||EA||B0||7A| |0D||0A|
	|EB||AC||B8||EC||84||9C|
9	CWD /dev/cdrom|0D||0A|
13	CWD /home/kusuk007/|0D||0A|
14	CWD /home/kusuk007/|EB||AC||B8||EC||84||9C||0D||0A|
15	CWD /home/kusuk007/|EB||8B||A4||EC||9A||B4||EB||A1||93||0D||0A|

For FTP protocols, the following types of messages are often categorized because many variable fields do not have a definite length. If one message type is defined as only one message, then that message is defined as a static field. Table 7 shows that only the Message Type 5 of the following FTP messages is extracted as "CWD." "|0D||0A|" is extracted as static fields, while "|EB||AC||B8||EC||84||9C|", and "|EA||B3||B5||EA||B0||7A|" are defined as dynamic fields.

4.5 The Result of Merging Based on Message Type Using Static Fields

This step combines messages of the same type into one type, although of different sizes. Because it is difficult to determine the similarity of messages of different sizes, this step merges message types using extracted fields. The type defined in this step is the final message type. Thus, Table 8 Shows that

Table 9 Result of merging of message types in FTP

Msg Type		Content	
n	CWD	/home	|0D||0A|
		|EA||B3||B5||EA||B0||7A|	
		|EB||AC||B8||EC||84||9C|	
		/dev/cdrom	
		/home/kusuk007/|	
		/home/kusuk007/|EB||AC||B8||EC||84||9C|	
		/home/kusuk007/|EB||8B||A4||EC||9A||B4||EB||A1||93|	

Table 10 Number of FTP message types after merging

	Modbus/TCP			FTP
Function	Connection	Transfer from EWS to PLC	Transfer from PLC to EWS	–
Request	18	25	75	19
Response	8	61	10	12

messages classified into six types can be merged into one message type. Table 9 shows that message types can be merged into one type because "CWD " and "|0D||0A|" have been extracted from the static field of Message Type 5, and "CWD " and "|0D||0A|" are included within the static field of Message Type 4, 9, 13, 14, and 15.

Table 10 shows the results after merging the message types. Most message types are simpler than just clustering algorithms. In particular, the FTP message types where variable length fields are present has been greatly reduced.

4.6 The Result of Quantitative Evaluation

In this section, the results of the experiment are evaluated quantitatively. The results of the proposed method compare the results with the existing Netzob and AutoReEngine methodologies. First, the proposed method evaluates the clustering results for coverage and conciseness. The expressions of each evaluation index are as follows:

The Equation (1) is coverage. Coverage evaluates the ability to cover all messages when a message type is extracted. The proposed method and Netzob first classify the message type. Therefore, the proposed method and

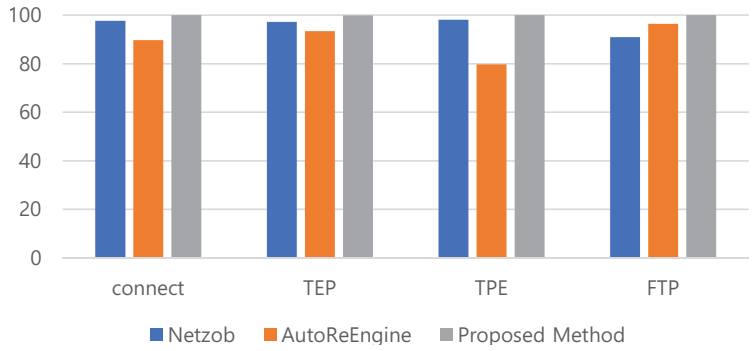


Figure 8 The result of coverage.

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. The Equations (2) and (3) is conciseness. Conciseness evaluates the brevity of the message type classification. The highest evaluation of brevity is the same number of ground truth message types and the same number of categorized message types. Thus, 100% conciseness is when there is the same number of ground truth message types and the same number of categorized message types. Because the results of the experiment may be less or more than the number of ground truth message types, conciseness is evaluated in two cases.

$$\text{Coverage} = \frac{\text{Number of covered messages}}{\text{Total number of messages}} \tag{1}$$

$$\text{Conciseness} = \frac{\text{Number of classified message types}}{\text{Number of ground truth message types}} \tag{2}$$

$$(\#\text{GT message types} \geq \#\text{classified message types})$$

$$\text{Conciseness} = \frac{\text{Number of ground truth message types}}{\text{Number of classified message types}} \tag{3}$$

$$(\#\text{GT message types} < \#\text{classfied message types})$$

The coverage results show that most methodologies have high levels, but the proposed method scores far higher on conciseness. In the cases of coverage, the proposed method extracted the message types that can cover all messages, while the existing methods resulted in not covering all messages. Uncovered messages will be excluded from management traffic, and excluded

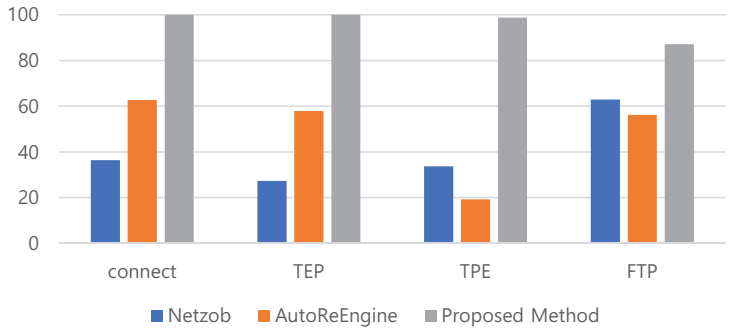


Figure 9 The result of conciseness.

messages will cause security threats or malfunctions. Also, the conciseness scored the highest value for the proposed method. This is the result that the proposed method is best classified by message types.

In the case of Netzob and AutoReEngine, the value of conciseness is low because most of the message types are classified in detail. If the message type is classified in detail, it is difficult to expect an accurate result from field extraction because in many cases, only the same messages are classified into the same type. Although the proposed method is classified in detail in the clustering process, since there is a process of merging message types using fields later, it is possible to extract a relatively accurate message type. In particular, on conciseness, the proposed method records higher values than other methodologies in Modbus/TCP protocols. This proves that the proposed method is optimized for industrial protocols with low variable length and multiple message types.

The second evaluation is the field extraction assessment. The field extraction evaluation evaluates precision and recall. For field evaluation, precision evaluates the accuracy of field extraction, while recall evaluates the degree of extraction of the field to be extracted. The formulas for precision and recall used for field evaluation of field extraction are as follows:

$$\text{Precision} = \frac{\text{Number of ground truth static fields in extracted static fields}}{\text{Number of extracted static fields}} \tag{4}$$

$$\text{Recall} = \frac{\text{Number of extracted static fields in ground truth static fields}}{\text{Number of ground truth static fields}} \tag{5}$$

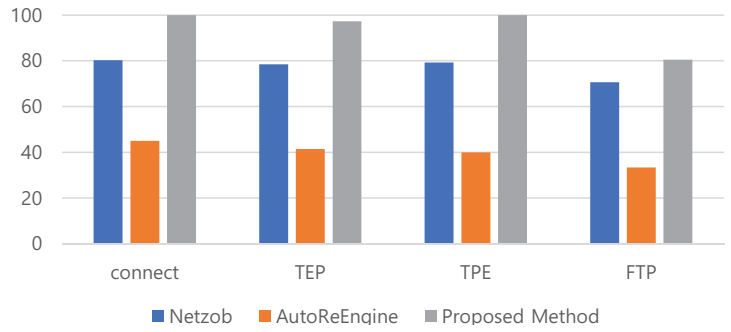


Figure 10 The result of precision.

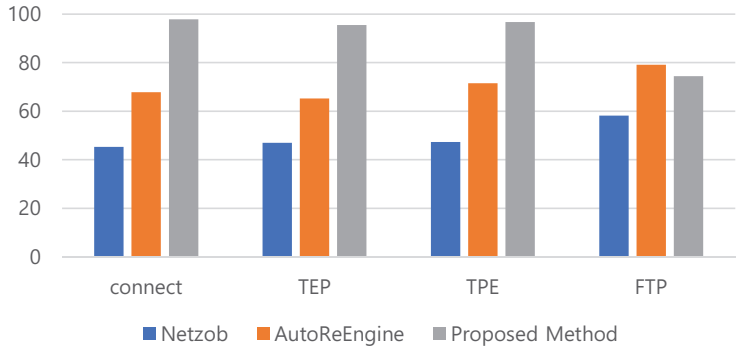


Figure 11 The result of recall.

Precision refers to the ratio of extracted static fields to the fields included in the ground truth static fields i.e., the proportion of extracted static fields that were properly extracted. Recall refers to the ratio of static fields extracted from the correct field. Netzob extracted the fewest fields. Thus, Netzob has a higher precision than AutoReEngine. However, Netzob has lower recall results than AutoReEngine. AutoReEngine extracted quite a number of fields, unlike Netzob. AutoReEngine has low precision and high recall because it extracted fields from the data portion.

The proposed method is relatively accurate in dividing the types of messages and extracting fields from the divided messages, thus its precision and recall are higher than those for AutoReEngine and Netzob. However, the recall value for FTP targets is lower than that for AutoReEngine. The proposed method is a methodology for industrial protocols and thus there are fields that cannot be extracted for analyzing commercial protocols. However, the extracted fields have been correctly extracted.

5 Conclusion

Networks are becoming increasingly interconnected in many industrial fields as a natural result of the implementation of automation; consequently, network security threats experienced in these industrial fields are also increasing. Industrial site security threats have the potential to cause more severe human danger and property damage than in other fields and therefore, network monitoring of industrial sites is becoming critically important. Industrial protocols used in industry and infrastructure sectors do not use standardized protocols, but mostly independently developed protocols. Because the independently developed protocol is largely undisclosed, it is very difficult to extract information on the specification of the protocol or the setup of the control device using the protocol. In this study, a method for analyzing the structure of industrial private protocols is proposed. This methodology can be used to enable efficient network traffic monitoring of industrial protocols. Experiments demonstrate that traditional protocol reverse engineering methods have many limitations in analyzing industrial protocols. We used Schneider Modicon M580 to prove the possibility of industrial protocol analysis using the proposed method and used the FTP commercial protocol where there is a correct answer for objective evaluation.

This methodology consists of seven modules: traffic collection, message extraction, message clustering based on size, message clustering by simulation, field extraction and session analysis. In the message clustering based on similarity module, the mean shift algorithm was used. Furthermore, static fields were extracted from the field extraction module using the CSP algorithm. In addition, message type inference was confirmed using static fields extracted to combine messages of different sizes but of the same type into one type. Through this methodology, we have demonstrated that structure analysis of the data part of the Modbus/TCP protocol, an industrial protocol, is possible.

With future research, we will improve the system performance for identifying protocol structures in real time. Currently, the collected traffic is received through input, but the protocol structure will be continuously updated with future traffic in real time.

Acknowledgements

This research was supported by Korea Institute of Science and Technology Information (KISTI).

References

[1] K. Zetter. 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 16, 2016.
[3] Langner, Ralph. "Stuxnet: Dissecting a cyberwarfare weapon." IEEE Security & Privacy 9.3 pp. 49–51. 2011.
[4] Stouffer, Keith, Joe Falco, and Karen Scarfone. "Guide to industrial control systems (ICS) security." NIST special publication 800.82: pp. 16–16. 2011
[5] Pidgin. (2018). About Pidgin. [Online]. Available: http://www.pidgin.im/about
[6] J. Caballero and D. Song, "Automatic protocol reverse-engineering: Message format extraction and field semantics inference," Int. J. Comput. Telecommun. Netw., vol. 57, no. 2, pp. 451–474, Feb. 2013.
[7] M. Liu, C. Jia, L. Liu, and Z. Wang, "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] Y. Wang 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] T. Krueger, H. Gascon, N. Kramer, and K. Rieck, "Learning stateful models for network honeypots," in Proc. 5th ACM Workshop Secur. Artif.Intell., Raleigh, NC, USA, Oct. 2012, pp. 37–48.
[10] H. Li, B. Shuai, J. Wang, and C. Tang, "Protocol reverse engineering using LDA and association analysis," in Proc. 11th Int. Conf. Comput. Intell.Secur. (CIS), Dec. 2015, pp. 312–316.
[11] M. A. Beddoe. (2004). Network Protocol Analysis Using Bioinforomatics Algorithms. [Online]. Available: http://www.4tphi.net/~awalters/PI/pi.pdf
[12] C. Leita, K. Mermoud, and M. Dacier, "ScriptGen: An automated script generation tool for Honeyd," in Proc. 21st Annu. Comput. Secur. Appl.Conf., Tucson, AZ, USA, Dec. 2005, p. 2.
[13] W. Cui, J. Kannan, and H. J. Wang, "Discoverer: Automatic protocol reverse engineering from network traces," in Proc. 16th USENIX Secur.Symp., Boston, MA, USA, Aug. 2007, pp. 199–212.

[14] G. Bossert, "Exploiting semantic for the automatic reverse engineering of communication protocols," Ph.D. dissertation, Univ. Gif-sur-Yvette, Rennes, France, Dec. 2014.

[15] L. Wang and T. Jiang, "On the complexity of multiple sequence alignment," J. Comput. Biol., vol. 1, no. 4, pp. 337–348, 1994.

[16] J.-Z. Luo and S.-Z. Yu, "Position-based automatic reverse engineering of network protocols," J. Netw. Comput. Appl., vol. 36, no. 3, pp. 1070–1077, May 2013.

[17] Y. Wang, N. Zhang, Y.-M. Wu, B.-B. Su, and Y.-J. Liao, "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] R. Ji, H. Li, and C. Tang, "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] I. Bermudez, A. Tongaonkar, M. Iliofotou, M. Mellia, and M. M. Munafo, "Automatic protocol field inference for deeper protocol understanding,"in Proc. 14th IFIP Netw. Conf., Toulouse, France, May 2015, pp. 1–9.

[20] G. Ladi, L. Buttyan, and T. Holczer, "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] A. Tridgell. (Aug. 2003). How Samba Was Written. [Online]. Available: http://samba.org/ftp/tridge/misc/french_cafe.txt

[22] Kyu-Seok Shim, Young-Hoon Goo, Min-Seob Lee, Huru Hasanova and Myung-Sup Kim, "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] Tovar, Eduardo, and Francisco Vasques. "Real-time fieldbus communications using Profibus networks." IEEE transactions on Industrial Electronics 46.6 (1999): 1241–1251.

[24] Van Herrewege, Anthony, Dave Singelee, and Ingrid Verbauwhede. "CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus." ECRYPT Workshop on Lightweight Cryptography. Vol. 2011. 2011.

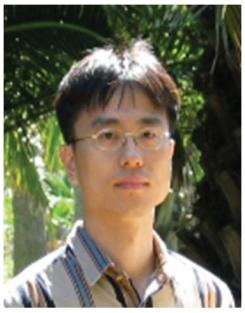
[25] Cena, Gianluca, and Adriano Valenzano. “A protocol for automatic node discovery in CANopen networks.” IEEE Transactions on Industrial Electronics 50.3 (2003): 419–430.
[26] Lian, Feng-Li, James R. Moyne, and Dawn M. Tilbury. “Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet.” IEEE control systems magazine 21.1 (2001): 66-83.
[27] Fovino, Igor Nai, et al. “Design and implementation of a secure modbus protocol.” International conference on critical infrastructure protection. Springer, Berlin, Heidelberg, 2009.
[28] Noguchi, Satoshi, et al. “FDT technology for CC-link network.” SICE Annual Conference 2011. IEEE, 2011.
[29] Bonney, Gregor, et al. “ICS/SCADA security analysis of a Beckhoff CX5020 PLC.” 2015 International Conference on Information Systems Security and Privacy (ICISSP). IEEE, 2015.
[30] Seno, Lucia, and Claudio Zunino. “A simulation approach to a Real-Time Ethernet protocol: EtherCAT.” 2008 IEEE International Conference on Emerging Technologies and Factory Automation. IEEE, 2008.
[31] Brooks, Paul. “Ethernet/IP-industrial protocol.” ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No. 01TH8597). Vol. 2. IEEE, 2001.
[32] Feld, Joachim. “PROFINET-scalable factory communication for all applications.” IEEE International Workshop on Factory Communication Systems, 2004. Proceedings IEEE, 2004.
[33] Goldenberg, Niv, and Avishai Wool. “Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems.” International Journal of Critical Infrastructure Protection 6.2 (2013): 63–75.
[34] Jain, Anil K. “Data clustering: 50 years beyond K-means.” Pattern recognition letters 31.8 (2010): 651–666.
[35] Gronau, Ilan, and Shlomo Moran. “Optimal implementations of UPGMA and other common clustering algorithms.” Information Processing Letters 104.6 (2007): 205–210.
[36] Cheng, Yizong. “Mean shift, mode seeking, and clustering.” IEEE transactions on pattern analysis and machine intelligence 17.8 (1995): 790–799.
[37] K. S. Shim, S. H. Yoon, S. K. Lee, S. M. Kim, W. S. Jung, M. S. Kim, “Automatic Generation of Snort Content Rule for Network Traffic Analysis,” KICS, Vol. 40, No. 04, pp. 666–677, April, 2015.

Biographies

Kyu-Seok Shim is a postdoctoral researcher in Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea. He received his B.S., M.S., and Ph.D. degree in 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, protocol reverse engineering and quantum key distribution.

IlKwon Sohn is a senior researcher in Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea. He received his B.S., and Unified M.S. & Ph.D. degree in the School of Electrical Engineering, Korea University, Korea, in 2011, and 2018, respectively. His research interests include quantum error correction, quantum key distribution, and quantum computation.

Eunjoo Lee is a postdoctoral researcher in Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea. She received B.S. degree in Physics from Hanyang University, Korea and Ph.D degree in Physics from Korea Advanced Institute of Science and Technology (KAIST). She was a former postdoctoral researcher of quantum optics group in Korea Research Institute of Standards and Science (KRISS). Her interests include fiber optics, single photon generation in telecom band, quantum optics experiment and quantum communication with single photons and continuous variables.



Woojin Seok is a principal researcher in Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea. He received his B.S. in the School of Computer Engineering, Kyungpuk University, M.S. in the school of Computer Science in UNC Chapel Hill, and Ph.D. degree in the School of Computer Engineering, Chungnam University, in 1998, 2003, and 2008, respectively. His research interests include TCP protocol, QKD network, and wireless network such as LoRa and private 5G.

Wonhyuk Lee is a senior researcher in Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea. He received his B.S., and M.S. & Ph.D. degree in the School of Electrical, Electronical and Computer Engineering, Sungkyunkwan University, Korea, in 2001, 2003 and 2010, respectively. His research interests include quantum Network Management, Network Performance Enhancement, and QKD network.