Forensic Science International: Digital Investigation 45 (2023) 301565

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS 2023 USA - Proceedings of the Twenty Third Annual DFRWS Conference

PREE: Heuristic builder for reverse engineering of network protocols in industrial control systems

Syed Ali Qasim*, Wooyeon Jo, Irfan Ahmed

Virginia Commonwealth University, Richmond, VA, 23284, USA

ARTICLE INFO

Article history:

Keywords:
Control logic
Industrial control systems
Forensics
SCADA

ABSTRACT

Industrial control systems (ICS) play a critical role in the operation of our vital infrastructures. They consist of field sites and a control center, with programmable logic controllers (PLCs) used at field sites to control physical processes directly. These systems communicate with the control center using proprietary protocols for remote monitoring, control, and configuration. The ability to reverse engineer these protocols can improve digital forensics techniques for investigating ICS attacks. The existing methods for reversing ICS protocols are manual forensics, binary analysis, probabilistic methods, or predefined network traffic analysis tools. ICS protocols, designed to operate in industrial environments, exhibit overlapping functionality, like uploading/downloading control logic to a PLC, which results in shared standard fields, such as function code and PLC memory address. Our hypothesis is that knowledge of one ICS protocol can aid in reverse engineering other proprietary ICS protocols. The paper introduces a heuristic builder, PREE, which enables control engineers with ICS protocol knowledge to create heuristics for identifying fields in other ICS protocols. We test our hypothesis by creating seven heuristic variants using the rolling window, vertical window, and frequency table techniques. We evaluate our heuristics on six ICS protocols, i.e., Modbus TCP, UMAS, ENIP, Omron FINS, CLICK, and PCCC. The evaluation involves five PLCs from four vendors: Modicon M221, Allen Bradley 1400 and 1100, Omron CP1L, and AutomationDirect CLICK Koyo. Results show that PREE can effectively identify common fields in multiple protocols, such as function code, message type, message length, PLC memory address, data size, and session/transaction IDs. PREE outperforms existing reverse engineering tools like NetPlier, Netzob, and Discoverer in terms of accuracy, conciseness, completeness, and consistency. We also demonstrate PREE's applications in a vulnerability study on CLICK Koyo PLC and present SNORT rules for investigating various attacks on it.

1. Introduction

A Programmable Logic Controller (PLC) is a critical component of Industrial Control Systems (ICS) (Ahmed et al., 2012, 2016). These devices are placed at field sites to control physical processes and send their current state to the control center using proprietary protocols. However, their critical nature makes them a target for attackers over networks to disrupt physical processes (Ayub et al., 2021a, 2021b, 2023; Zubair et al., 2022a; Qasim et al., 2022). Investigating such attacks is challenging due to the need for an appropriate forensic method to analyze the proprietary protocols used in PLC communication (Rais et al., 2021, 2022; Awad et al., 2023).

Protocol knowledge is valuable for security applications such as fuzzing (Luo et al. Sun; Luo et al., 2020; Niedermaier et al., 2017), intrusion detection (Yang et al., 2019; Li et al., 2020; Yoo et al., 2019a), malware injection (Ayub et al., 2023; Yoo et al., 2019b; Kalle et al., 2019; Senthivel et al., 2018; Zubair et al., 2022b), vulnerability discover (Ayub et al., 2021b; Qasim et al., 2022) and forensics (Ahmed et al., 2017; Senthivel et al., 2017; Qasim et al., 2019, 2020). Since protocols are proprietary, network protocol reverse engineering is typically used to uncover the format and semantics of protocol messages. Existing methods include tedious manual analysis, complex binary analysis (Narayan et al. Clancy; Lin et al., 2008; Chang et al., 2018), or pre-installed capabilities for network traffic analysis (Ládi et al., 2018; Kim et al., 2021; Wang et al., 2020; Shim et al. Kim; Wu et al., 2019).

In manual network traffic forensics, the user can compare messages within or across sessions to identify protocol field

* Corresponding author.
E-mail addresses: qasimsa@vcu.edu (S.A. Qasim), iahmed3@vcu.edu (I. Ahmed).

https://doi.org/10.1016/j.fsidi.2023.301565

positions and guess their meanings. Although this is a common approach in security, it faces challenges such as large data volumes, changing control logic message fields, unreadable binary messages, context-specific fields affecting meaning, extracting client-server sessions from network dumps (Narayan et al. Clancy), etc. With Industry 4.0, manual semantic forensics is no longer feasible due to the growing connectivity of heterogeneous PLC networks from different vendors, making it difficult for experts to learn all protocols.

These challenges have driven the forensic community to develop automated protocol reverse engineering tools in two directions: Binary (taint) Analysis and Network Trace Analysis. In Binary Analysis, the reverse engineering tool inputs a message to an available executable file of the program or protocol and monitors control flow, called instructions, and memory usage to learn the protocol format and field semantics. In Network Trace Analysis, network traffic between communicating entities is captured and protocol fields and boundaries are identified through machine learning and data analytics techniques like clustering and differential analysis (Wu et al., 2019).

ICS protocols support communication between PLCs and enable remote monitoring, control, and configuration by a control center. They inherently overlap and share many standard fields such as function code and PLC memory address. ICS protocols have a consistent usage pattern due to their repetitive control logic operations on PLCs. A pattern recognition tool called Ratcliff/Obershelp was used for fuzzing in a study (Niedermaier et al., 2017), demonstrating that ICS protocols' consistency makes them suitable for primitive protocol reverse engineering. Therefore, it is hypothesized that knowledge of one ICS protocol can aid in identifying standard fields in others.

This paper proposes a heuristic builder, the Protocol Reverse Engineering Engine (PREE), to allow control engineers to use their ICS protocol knowledge to create heuristics for protocol message fields. PREE applies these heuristics to network traffic from an unknown ICS protocol to automatically discover the locations and semantics of similar fields in the protocol messages. It analyzes network dumps at message and session levels and provides data analysis functions to assist heuristic building, such as analyzing message sections and comparing messages within and across sessions.

We evaluated PREE on six ICS protocols (Modbus TCP, M221, ENIP, OmronFINS, CLICK, and PCCC) using five PLCs from four ICS vendors (Modicon M221, Allen Bradley 1400 and 1100, Omron CP1L, and AutomationDirect CLICK Koyo). We used three different techniques (rolling window, vertical window, and frequency table) and created seven heuristics to discover similar fields in multiple protocols. The heuristics effectively identified eight protocol fields, including function code, message type, transmission length, PLC memory address and data size, and session ID.

Our contributions are summarized as follows.

- We present PREE, a heuristic builder for control engineers to use their domain knowledge to reverse engineer ICS protocols.
- We develop eight heuristic algorithms to find eight distinct fields in ICS protocols using three techniques
- We evaluate PREE on six real-world ICS protocols in five PLCs and demonstrate its effectiveness in finding similar fields in different protocols.
- We compare PREE with the existing binary protocol reverse engineering tools like NetPlier, Netzob, and Discoverer.
- We conduct a vulnerability study on CLICK Koyo PLC and develop SNORT rules to investigate and discover several attacks on CLICK PLC to show the application of PREEknowledge.

The remaining paper is organized as follows: Section 2 discusses the background and related work. Section 3 presents the PREE architecture and the heuristic algorithms. Section 4 and 5 presents the PREE implementation and evaluation. Section 6 compares PREE with existing reverse engineering tools. Section 7 shows offensive and defensive applications of PREE. Section 8 concludes the paper and presents future work.

2. Background and related work

There are many tools available for reverse engineering protocols to discover protocol message format or the state machine. Most of these tools fall under two techniques; the first is the program analysis technique where protocol binaries are analyzed to reverse engineer the protocol. The second is network trace analysis where different network dumps are analyzed to extract protocol details. Our focus is on tools developed using network trace analysis and is close to PREE.

Ladi et al. (Ládi et al., 2018) presented a four-phase approach to reverse engineer binary protocols. They captured network traffic, constructed and optimized an acyclic graph of the messages exchanged, and assigned pointers at the first byte of each packet to monitor processing. The algorithm starts with a root node and adds nodes for different fields as it moves the pointers of all packets. They developed some heuristics to identify constant bytes, length fields, counters, enumerated types, and highly variable bytes. The approach was evaluated on Modbus and MQTT protocols.

Kim et al. (2021) proposed a 4-step method for reverse engineering the Modbus/TCP protocol and creating an intrusion detection system. They used 9 tuples to group similar messages, then applied multiple sequence alignment to categorize bytes into constant, categorical, and variable categories. Next, they identified header/payload boundaries through local sequence alignment and inferred payload fields by categorizing bytes and analyzing their behavior. The result was a successful reverse engineering of the Modbus/TCP protocol and an intrusion detection system.

Wang et al. (2020) proposed an approach to find feature words in unknown protocols using V-grams and XGBoost. Binary messages were converted to hexadecimal data, grouped by length, and aligned using PMSA. V-grams were generated and feature words were extracted and ranked using XGBoost. They evaluated their approach based on the S7 protocol.

Shim et al. (Shim et al. Kim) proposed a six-stage model for identifying message formats in ICS protocols. They captured communication between PLC and engineering software, then grouped messages based on size and refined groups with K-Means, UPGMA, and mean shift clustering. Then they used a contiguous sequence pattern (CSP) algorithm to extract static/dynamic fields and generated message formats. The approach was evaluated on Modbus/TCP, ENIP, and FTP protocols.

Wu et al. (2019) presented an HMM-based approach for identifying ICS message formats. They tokenized application layer data into two categories: text (printable bytes) and binary (non-printable) using ASCII encoding. Consecutive printable bytes form one text token and non-printable bytes as a binary token. They then grouped messages with similar token patterns into clusters and inferred different message formats using an HMM-based sequence

alignment algorithm. Their approach was evaluated on Modbus/TCP and IEC 61850 protocols.

3. Overview of PREE architecture

PREE helps users develop and implement heuristics by using network dumps. Fig. 1 illustrates the bottom-up overview of PREE. It has a three-layer model: a data pre-processing layer, where network dumps are organized into data structures; a data analytics layer, offering analytics functions for the session and message-level analysis; and a heuristic builder where the user can develop and execute their heuristics for protocol reverse engineering. PREE works similarly to MySQL. To reverse engineer a protocol, the user provides network dumps with targeted protocol and metadata (e.g. PLC and engineering workstation IPs and ports) to PREE. After processing the dump, the user can use PREE's analytics functions to write heuristics, similar to MySQL queries.

3.1. Data pre-processing

PREE starts with data pre-processing. This involves extracting client-server sessions, making request-response message pairs, and grouping messages.

Session Extraction: In a network dump, multiple client-server sessions may exist. To analyze them, we must first separate these sessions. PLCs in ICS environments have fixed ports, such as Allen–Bradley MicroLogix 1100 and 1400 using port 44818 and Modicon M221 using port 502. However, the client-side port, used by the engineering software, is often machine-dependent and changes. PREE identifies and separates different sessions by using a four-tuple: source IP, source port, destination IP, and destination port.

Message Pairing: After separating messages from different sessions, the next step is to pair request and response messages and arrange them in order of exchange. This pairing and maintaining the sequence helps identify common fields in request and response messages and fields that show a consistent change along the session.

Message Grouping: Grouping similar messages together is important in ICS protocols that have more than one message format. This helps the user develop heuristics for different groups and discover different message formats in network dumps. Grouping can be based on message payload length or total size.

3.2. Data analytics

PREE's data analytics layer offers useful functions for analyzing network dumps and discovering protocol fields. Table 1 lists the available functions, split into two categories: message-level and session-level analysis.

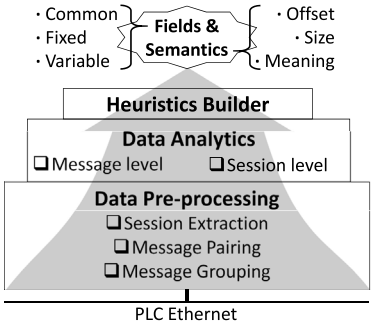

Fig. 1. Protocol Reverse Engineering Engine (PREE) model.

Message-Level Analysis: During our research, we found that certain protocol fields, such as the "Length field" and "Checksum field", can be identified using the information within the message. The values in the Length and Checksum fields represent the actual length and checksum of a specific section of the message. To aid users in discovering these protocol fields, PREE offers several message-level functions, such as "h_move" and "window_gen" that can be used to identify correlations between different sections of a message.

Session-Level Analysis: The second category focuses on protocol fields that change or show a pattern throughout a session, e.g., the "Transaction ID" present in many ICS protocols increases with every new message in the session. PREE provides the user with several functionalities, such as comparing bytes at the same index in different messages, finding all the values seen at a fixed location in all the messages, etc to perform the session-level analysis.

3.3. Heuristic building

We observe that the common fields in ICS protocols can be divided into three categories based on their behavior during communication. Fig. 5 illustrates the classification of different fields in the Modicon M221 message into one of these categories.

Configuration Fields: The fields depend on the ICS environment and can be configured by using engineering software. Their values typically remain constant throughout the communication session. An example of a configuration field is "PLC ID".

Fixed Fields: The second category consists of fields with constant values across all messages and sessions. Though it's challenging to gather semantic information from these fields through differential analysis, their patterns can aid in identifying proprietary protocols. Thus, we label them "Protocol Identifiers".

Variable Fields: The third category includes fields with values in different messages across sessions. For instance, the length, checksum, function code, etc. Change per message, while the session ID changes between sessions.

Finding Configuration Fields: No heuristics are required as the values of "Configuration Fields" are known to the user. They can be located in messages using the "find_msg" function of PREE, which takes the target sequence of bytes (configuration field value) and returns its location or index in all messages of a session if found.

Finding Fixed Fields: Like "Configuration Fields", no heuristics are required to locate "Fixed Fields". Users can use the "find_feq" function to generate a frequency table showing the frequency of values at each index in all messages of the session. Fixed fields can be found where the frequency is 100%, meaning the value stays the same.

3.4. Heuristics for variable fields

Finding the location and meaning of "Variable fields" is difficult because the variance depends on the field's nature. For example, the "Transaction ID" in a message increases over time, the "Length" and "CRC" fields depend on the payload, and the "Session ID" is initiated by the PLC or engineering software. To handle these variations, we used three techniques and created eight heuristics in total.

Rolling window: In the Rolling Window heuristic, PREE employs a sliding window of varying sizes (1,2, …,n bytes) over the message and applies the user-defined function (which could be designed to find the length, checksum, etc) to all substrings of the messages. If the output of the function matches the value within the window, the location is labeled as a potential field. To minimize false positives, only potential fields that consistently appear across similar messages are selected. Fig. 2 shows the implementation of

Table 1
Summary of PREE data analytics functionalities.

Function	Description	Type
sim_msg	Find similarity between two messages	Message-Level
find_msg	Search the given sequence of bytes in messages	Message-Level
diff_msg	Find difference between tow messages	Message-Level
h_move	Give all possible substrings and their indices in a message	Message-Level
window_gen	Generates substrings inside a window given message, window size and increment	Message-level
longestSubstringFinder	Find the longest common subsequence of two messages	Session-Level
v_move	Gives array of substring inside the given window for all messages	Session-Level
find_feq	Makes frequency table containing frequency of each byte at each index in the pcap file	Session-Level
freq_match	Find Messages that have bytes with frequency > given threshold	Session-Level
freq_change	Find indices in messages with frequency change lower than given threshold	Session-Level

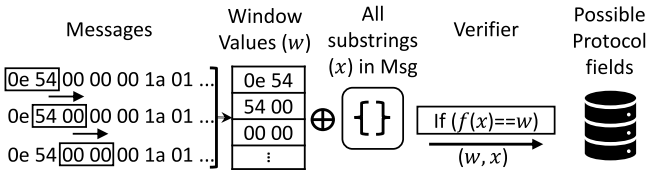

Fig. 2. Rolling window approach to find message-level fields.

this technique in PREE. Using this technique we developed and executed two heuristics to find the length and checksum fields.

- **Length field:** If the user provides the function f(x) that calculates the length of the payload, the value inside any window that matches the output of f(x), the current location of the window can be marked as a length field.
- **Checksum field:** Similarly, if the user has developed a potential checksum function, he can use the rolling window technique to identify the location of checksum field.

Vertical Window: The value of some fields changes in a fixed pattern throughout a session. This can be detected using the vertical window approach (Fig. 3). Using a user-defined function "f(x)", PREE moves a window of varying sizes overall messages in a session. For each consecutive message pair, i. e y and y+1, it checks if f(y) = y+1. If this condition is true for all message pairs, the current window location can be labeled a potential protocol field based on f(x). Using this technique we developed two heuristics to find the Transaction ID and PLC Memory Address fields.

- **Transaction ID:** Transaction ID in a protocol increases constantly with each new message. If a user defines f(x) to add a fixed number to x, the sliding window can represent a potential "Transaction ID".
- **PLC Memory Address:** Uploading/downloading control logic involves sending a series of messages with PLC memory address and data size to be read/written i. e in consecutive messages, the address changes by the size of data written/read. To identify the "PLC Memory Address", the vertical window can use f(x) to add the current memory address and data size.

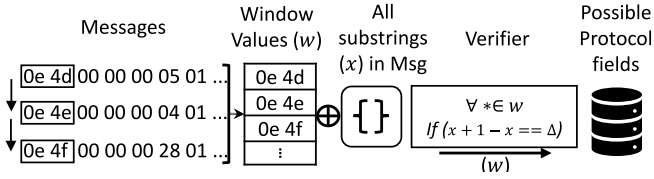

Fig. 3. Vertical window approach to find session-level fields.

Frequency table: The frequency table is useful in identifying variable fields in messages that don't have a specific pattern and depend on software or PLC, such as "Function Codes" and "Session ID". The frequency table feature of PREE can be used to locate these fields in message headers by creating a table of all messages and storing the frequency and values of each byte at each index in a session. This enables the development of various heuristics to find protocol fields.

- **Session ID:** The Session ID is established in the initial messages between PLC and software and stays constant. To find it, the frequency table can be queried for indices with limited changes and these bytes can be searched in the initial messages. If found, these indices may indicate the "Session ID" in the protocol.
- **Function Code:** The function code is a field with a limited set of codes used by software to send requests to the PLC. If the request is accepted, the PLC replies with a success code. If not, a failure code is sent. In a session with no failures, the function code can be found by querying the frequency table for indices with limited variance in request messages and constant values in response messages. These indices may indicate the location of the "function code" in the ICS protocol.
- **Message Type ID:** The message type ID is a field that identifies the message as a request or response. It has unique values in request messages and different unique values in response messages. To find this field, separate frequency tables for request and response messages can be created, then compare bytes with 100% frequency in each table.

4. Implementation

We developed PREE using Python and Scapy (Biondi, 2022). It is designed as a simple and portable Python library, consisting of four main modules: data processing, data analytics, storage, and query builder. To use PREE, network dumps in the form of pcap files and metadata such as client and server machine IP addresses and port numbers are required.

Data Processing: PREE receives pcap files and metadata (IP & port no.), extracts sessions from the network dump, and identifies request and response messages. It stores them as an ordered dictionary, where requests are keys and responses are values. Multiple pcap files can be processed, creating dictionaries for each. Messages with varying lengths are grouped by length for analysis.

Data Analytics: The data analytics module provides the user with different functionalities to analyze the network dumps and find various relations, patterns, and trends across different

messages. The details of functions provided by the data analytics module are in Table 1.

Heuristics Builder: The Heuristics Builder allows for interaction between the user and the PREE. The user can download the PREE and use it either within their programs or through a Python shell. With the help of various functions, the user can process pcap files and make queries to gather information necessary for creating and implementing heuristics.

Storage: The storage module offers various functions to store intermediate results and final protocol message formats, as one protocol field may lead to others.

5. Evaluation

5.1. Data collection

For evaluating PREE and our heuristic algorithms, we analyzed six widely-used ICS protocols such as Modbus, EtherNet/IP, etc (Luo et al., 2019; Yusheng et al., 2017). We generated network dumps by connecting to different PLCs using their engineering software and capturing network communication using Wireshark. To ensure diversity and comprehensive coverage of message formats, we performed various actions such as transferring control logic between PLC and engineering software, changing PLC mode, etc.

5.2. Evaluation metrics

Several studies have manually reverse-engineered some of the protocols evaluated in this paper, identifying the location and meaning of certain protocol fields for use in offensive or defensive applications. For the purposes of this paper, we consider the established location and meaning information for each protocol's fields as the **ground truth**. With the ground truth defined, we evaluated PREE using three metrics as shown in Fig. 4.

Coverage evaluates the percentage of messages covered by PREE as protocol fields and is calculated as the ratio of bytes labeled by PREE to the total bytes in the message.

Conciseness measures the stability of how PREE identified the protocol fields compared to the ground truth and is calculated as the ratio of fields extracted by PREE to the total number of fields in the ground truth.

Perfection evaluates the quality of how we perfectly extracted out of the existing ground truth fields. It is the same as having true-positive as a numerator but divided by the total number of ground truth fields.

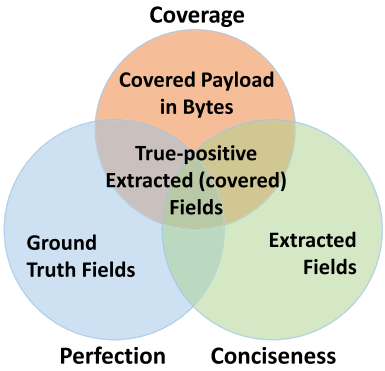

Fig. 4. Three metrics used for evaluating PREE.

$$Conciseness = \frac{of\ extracted\ ground\ truth\ fields}{of\ extracted\ fields}$$

$$Coverage = \frac{of\ labeled\ bytes}{of\ extracted\ bytes}$$

$$Perfection = \frac{of\ extracted\ ground\ truth\ fields}{of\ total\ ground\ truth\ fields}$$

5.3. Evaluation methodology

Table 2 shows that some fields are common across ICS protocols. After collecting network dumps from multiple protocols, we applied various heuristic algorithms created with PREE to each protocol, to determine the location and meaning of these fields. To assess conciseness, perfection, and coverage, We compared our results with ground truth from previous ICS protocol reverse engineering studies.

5.4. Modbus

In our experiments, we applied different heuristic algorithms developed with PREE to identify fields listed in Table 2 in the Modbus protocol. Using the rolling window heuristic, we found the "Length field" is 2 bytes located at bytes 5-6th, representing the length of the region from byte 6 to the end of the message (payload). The "Transaction ID" was identified using the vertical window heuristic and was found to be two bytes, represented by the first two bytes of the message, and incrementing by one with each new message. Finally, using a frequency table with 100% frequency (bytes that had the same value in all the messages), we identified the Protocol Identifiers. As shown in Fig. 5, PREE was able to achieve 100% coverage and identified 4 fields in the Modbus message. Table 3 compares the location and semantics of fields identified by PREE and the ground truth. PREE was able to achieve 100% conciseness and perfection. Our results align with the results of previous manual reverse engineering studies on Modbus protocol (ground truth) (Qasim et al., 2020; Kalle et al., 2019).

5.5. UMAS

We used the frequency table heuristic to identify the first byte as a "Protocol Identifier" in the Modbus payload. We found a 2-byte "Length field" at bytes 7–8 that represents the remaining message length. We identified the M221 function code as the 3rd byte, which changed in request messages and was constant in response messages. The "PLC Memory Address" was found at bytes 4–5. Fig. 6 shows that PREE was able to achieve 100% coverage in request and 98% coverage in response messages. We found 5 fields in the UMAS message. Furthermore, as shown in Table 4, the fields and semantics identified by PREE also matched with the manual forensic studies (Yoo et al., 2019b; Qasim et al., 2019; Kalle et al., 2019) and achieved 100% conciseness and 80% perfection.

5.6. ENIP

ENIP is widely used by Allen–Bradley PLCs, such as the MicroLogix 1400 and MicroLogix 1100, for communication with RSLogix engineering software. For evaluation, we captured the communication between a MicroLogix 1400 PLC and RSLogix during different engineering operations. Using the rolling window technique we found three 2-byte "length fields" at offsets 3–4, 35–36, and

Table 2
Common fields in different ICS protocols.

Semantic	Modbus	Modbus M221	ENIP	PCCC	CLICK	Omron FINS	Field Type
PLC ID	✓						Configuration
Transaction/Message ID	✓		✓	✓	✓	✓	Variable
Session ID			✓			✓	Variable
Message Type ID			✓	✓	✓	✓	Variable
Message Length	✓	✓	✓	✓	✓	✓	Variable
Function Code		✓	✓	✓	✓	✓	Variable
PLC Memory Data Size		✓	✓	✓	✓	✓	Variable
PLC Memory Address		✓	✓	✓	✓	✓	Variable
Protocol Identifiers	✓	✓	✓	✓	✓	✓	Fixed

Table 3
Comparison of PREE and ground truth in Modbus.

Field	PREE Location	Ground Truth Location	PREE Semantic	Ground Truth Semantic	#PREE types	# Ground Truth types
1	1–2	1–2	Transaction ID	Transaction ID	1	1
2	5–6	5–6	Length	Length	1	1
3	3–4	3–4	Protocol ID	Protocol ID	1	1
4	7	7	Protocol ID	Protocol ID	1	1

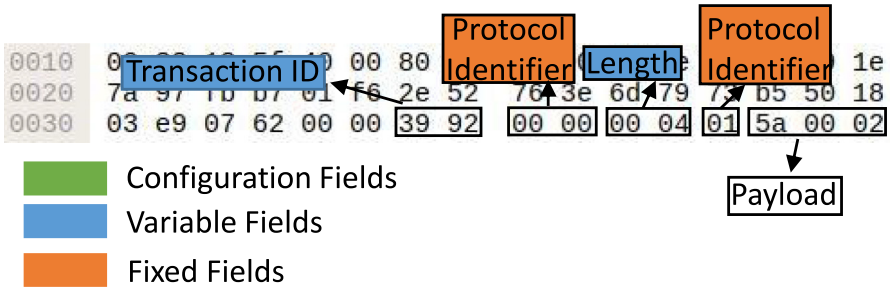

Fig. 5. Fields identified in the Modbus message.

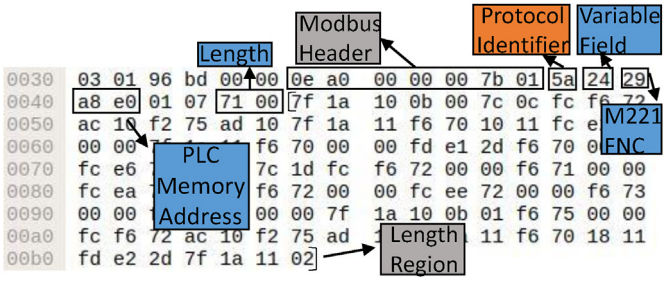

Fig. 6. Fields identified in UMAS request messages.

54–55, indicating multiple data layers. A 100% frequency threshold in the frequency table identified multiple protocol IDs, and a 90% frequency revealed a 4-byte session ID at bytes 5–8. The vertical window approach showed a constant 2-byte "Transaction ID" at bytes 13–14, incrementing by 2 in each message. A 6-byte session field was also found but varied in different sessions. Furthermore, frequency table heuristics identified an additional field: "Message Type ID" at bytes 29–30 with values "0500" in requests and "0004" in responses. Finally, we found the location of IP address of the PLC at 37-50th byte by directly searching it in the message as Field 12 in Table 5. As shown in Fig. 7 we could identify 14 fields in the ENIP message and achieve a 98% coverage. Table 5 shows that not only

Table 4
Comparison of PREE and ground truth in UMAS.

Field	PREE Location	Ground Truth Location	PREE Semantic	Ground Truth Semantic	#PREE types	# Ground Truth types
1	1	1	Protocol ID	Protocol ID	1	1
2	3	3	Function Code	Function Code	1	1
3	4–5	4–5	PLC Memory Address	PLC Memory Address	1	1
4	8–9	8–9	Length	PLC Memory Data Size	1	1

Table 5
Comparison of PREE and ground truth in ENIP.

Field	PREE Location	Ground Truth Location	PREE Semantic	Ground Truth Semantic	#PREE types	# Ground Truth types
1	1–2	1	Protocol ID	Protocol ID	1	1
2	3–4	3	Length	*NA*	1	1
3	5–8	4–5	Session ID	*NA*	1	1
4	9–12	8–9	Protocol ID	PLC Memory Data Size	1	1
5	13–14	13–14	Transaction ID	Transaction ID	1	1
6	15–20	15–20	Session Field	Session Field	1	1
7	21–28	21–28	Protocol ID	Protocol ID	1	1
8	29–30	29–30	Message Type	Message Type	2	2
9	31–32	31–32	Protocol ID	Protocol ID	1	1
10	34	34	Protocol ID	Protocol ID	1	1
11	35–36	35–36	Length	Length	1	1
12	37–50	*NA*	PLC IP	*NA*	1	1
13	52–53	52–53	Protocol ID	Protocol ID	1	1
14	54–55	54–55	Length	Length	1	1

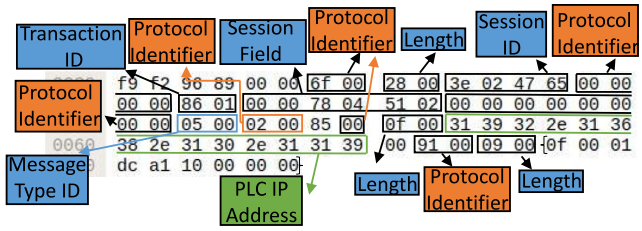

Fig. 7. Fields identified in ENIP request Message.

the location and semantics discovered by PREE matched with the existing manual reverse engineering efforts (Yoo et al., 2019b; Qasim et al., 2019; Kalle et al., 2019), it was also able to identify an extra field PLC IP Address (configuration field) that was not discovered (hence marked NA, not applicable) in the previous works. PREE achieved 100% conciseness and perfection.

5.7. PCCC

PCCC is a proprietary protocol used by many Allen–Bradley PLCs. For MicroLogix 1400 and 1100, PCCC messages are embedded in ENIP payloads. After analyzing the ENIP protocol, we applied heuristic algorithms to find PCCC protocol fields. Using the frequency table technique, we found the "Message Type ID" at first and a protocol identifier at the 2nd byte. The "Message Type ID" remained "0f" for all request messages and "4f" for all response messages. We identified the "Transaction ID" at the 3-4th byte in the PCCC message, which increments by 4 with each new message. Using rolling window heuristics, we identified the "Function Code" at the 5th byte and the "PLC Memory Data Size" at the 6th byte. As shown in Fig. 8, we were able to identify 5 fields in the PCCC

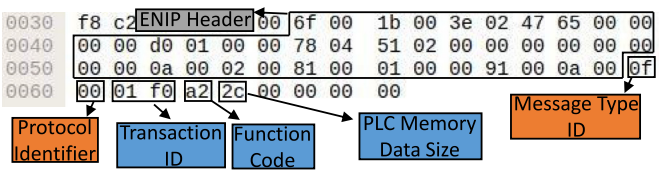

Fig. 8. Fields Identified in PCCC request Message.

protocol and achieve 60% coverage. Table 6 shows that the location and semantics of different fields match with previous works (Senthivel et al., 2017, 2018) done on PCCC. Furthermore, we identified a one-byte *Protocol Id* at 2nd byte that was not labeled in the previous work (*NA*). PREE achieved 100% conciseness and 62.5% perfection for PCCC protocol as shown in Table 9.

5.8. CLICK

AutomationDirect has developed its own application layer proprietary protocol for communication between CLICK PLC and its engineering software. Using the frequency table technique with 100% frequency, we were able to identify "Protocol Identifiers". We also found a variable field at the 12th byte that was the same in all the request-response messages except one. We also detected a one-byte "length" field at the 9th byte and the "PLC Memory Data Size" field using the rolling window technique. The "Transaction ID", a two-byte field, was found at the 5-6th byte using the vertical window heuristic. Additionally, the Memory Address heuristic helped us identify the "PLC Memory Address" field's location (16-19th byte) and the read and write function codes at the 13th and 14th bytes. As shown in Fig. 9, we identified 8 fields in the CLICK protocol, achieving 85% coverage. Table 7, shows that the semantics of fields and their location identified by PREE matched the ground truth (Ayub et al., 2021b).

5.9. OMRON FINS protocol

The Omron FINS protocol is a proprietary protocol used by OMRON PLCs for communication with engineering software. We used the OMRON CP1L PLC and CX-Programmer in our experiments. Using the frequency table technique with a 100% threshold, we identified the "Protocol Identifier" and "Message Type ID" fields in OMRON FINS network dumps. The two-byte "length field" was found at 7-8th bytes using the rolling window technique. It indicates the number of bytes after it in the message. The one-byte "Transaction ID" was discovered at the 26th byte with the vertical window heuristic. It increases by one in new command messages and stays constant in command-response messages. As shown in Fig. 10 we identified 12 fields and achieved 23% coverage. Table 8 show the details of the location and semantics of different fields identified using PREE. As we did not find any ground truth for the

Table 6
Comparison of PREE and ground truth in PCCC.

Field	PREE Location	Ground Truth Location	PREE Semantic	Ground Truth Semantic	#PREE types	# Ground Truth types
1	1	1	Message ID	Message ID	2	2
2	2	*NA*	Protocol ID	*NA*	1	*NA*
3	3–4	3–4	Transaction ID	Transaction ID	1	1
4	5	5	Function code	Function code	1	1
5	6	6	Length	PLC Memory Data Size	1	1

Table 7
Comparison of PREE and ground truth in CLICK.

Field	PREE Location	Ground Truth Location	PREE Semantic	Ground Truth Semantic	#PREE types	# Ground Truth types
1	1–4	1–4	Protocol ID	Protocol ID	1	1
2	5–6	5–6	Transaction ID	Transaction ID	1	1
3	9	9	Length	Length	1	1
4	10–11	10–11	Protocol ID	Protocol ID	1	1
5	15	15	PLC Memory Data Size	PLC Memory Data Size	1	1
6	16–19	16–19	PLC Memory Address	PLC Memory Address	1	1
7	20	20	Length	PLC Memory Data Size	1	1

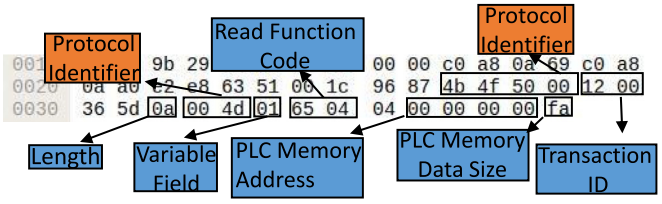

Fig. 9. Fields identified in CLICK PLC request message.

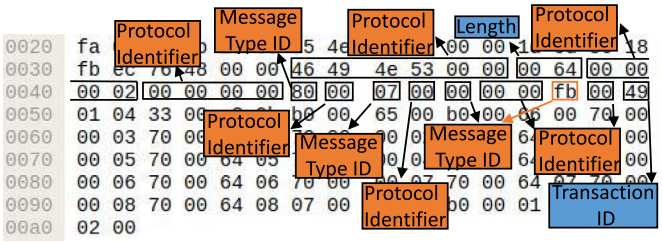

Fig. 10. Fields identified in OMRON FINS command message.

OMRON FINS protocol, the ground truth location and semantic columns show NA. Similarly, we were not able to compute the conciseness and perfection as shown in Table 9.

6. Comparison with existing tools

6.1. Comparison metrics

In this section, we will compare the proposed PREEwith other protocol reverse engineering tools. There are five evaluation metrics: V-measure, homogeneity, completeness, conciseness, and perfection (Rosenberg and HirschbergV-measure, 2007). The score of the V-measure v can be computed from homogeneity h and completeness c as:

$$v = 2 \cdot \frac{h \cdot c}{h + c} \tag{1}$$

$$h = \frac{\sum_{i}^{k} \textit{of elements in cluster}_i \textit{ from single class}}{\sum_{i}^{k} \textit{of elements in cluster}}$$

$$c = \frac{\sum_{i}^{c} \textit{of elements in class}_i \textit{ assigned to single class}}{\sum_{i}^{c} \textit{of elements in class}}$$

where k represents the number of clusters, and c represents the number of classes (keywords).

Evaluation metrics such as ***h*** and ***c*** for probabilistic approaches indicate how well the clustering results are classified into classes. Since PREE is not based on probabilistic methods, it is considered that clustering has been successfully performed if the semantics of the key fields are correctly identified. If the ground truth keyword, in this case, function code (service code) is inferred, it has h and c score of 100%. For instance, assuming a network using only two function codes, h decreases if the network is classified as one without being able to distinguish between the two, and c decreases if more than two are excessively classified.

Table 8
Comparison of PREE and ground truth in OMRON FINS.

Field	PREE Location	Ground Truth Location	PREE Semantic	Ground Truth Semantic	#PREE types	# Ground Truth types
1	1–6	*NA*	Protocol ID	*NA*	1	1
2	7–8	*NA*	Length	*NA*	1	1
3	9–16	*NA*	Length	*NA*	1	1
4	17	*NA*	Message Type ID	*NA*	1	1
5	18	*NA*	Protocol ID	*NA*	1	1
6	19	*NA*	Message Type ID	*NA*	1	1
7	20	*NA*	Protocol ID	*NA*	1	1
8	21	*NA*	Message Type ID	*NA*	1	1
9	22–23	*NA*	Protocol ID	*NA*	1	1
10	24	*NA*	Message Type ID	*NA*	1	1
11	25	*NA*	Protocol ID	*NA*	1	1
12	26	*NA*	Transaction ID	*NA*	1	1

Table 9
Summary of fields identified by PREE.

Results	Modbus TCP	Modbus M221	CLICK	ENIP	PCCC	Omron FINS
Ground Truth Fields	4	5	6	13	8	15
PREE Identified	4	4	6	14	5	13
Conciseness	100%	100%	100%	100%	100%	–
Perfection	100%	80%	100%	100%	62.5%	–

6.2. Existing/comparison tools

NetPlier proposed by (Ye et al., 2021a) is the most recent state-of-the-art automatic protocol reverse engineering study and provides protocol reverse engineering with probabilistic approaches based on Netzob (Bossert et al., 2014) and MAFFT (Katoh et al., 2002). The main probabilistic approaches based on NetPlier can be evaluated with homogeneity and completeness scores, which indicate how well clustering contains one class (keyword) (Rosenberg and HirschbergV-measure, 2007). They also performed a comparative analysis with the other protocol reverse engineering tools Netzob and Discoverer (Cui et al., 2007).

6.3. Experiment methodology

For fairness, we compared the results of PREE and other tools with a Modbus dataset known to have been used by NetPlier (Ye et al., 2021b) as shown in Table 10. According to (Ren et al., 2018), NetPlier collected this dataset through a network security monitoring tool called Bro (Paxson, 1999)and used this dataset for anomaly detection. Even though NetPlier extracted and evaluated 1000 messages from the dataset, the exact purification method was not disclosed. We evaluate all of the Modbus packets in the dataset.

Table 10
Comparison of PREE with existing tools in Modbus.

	PREE	NetPlier	Netzob	Discoverer
Homogeneity	100%	100%	73%	100%
Completeness	100%	100%	70%	55%
Conciseness	100%	70%	59%	4%
Perfection	100%	5%	8%	5%

6.4. Comparison results

As shown in Fig. 11, the proprietary protocols UMAS (embedded in Modbus) and PCCC (embedded in ENIP) were compared using PREE and NetPlier. To compare the two, additional development was done for NetPlier. A total of 64,525 Modbus (UMAS) packets were used for the evaluation and 16,601 for ENIP (PCCC). PREE extracted function codes flawlessly in all four protocols. However, due to the embedded proprietary protocols in Modbus and ENIP, packet lengths became more variable, leading to a drop in NetPlier's

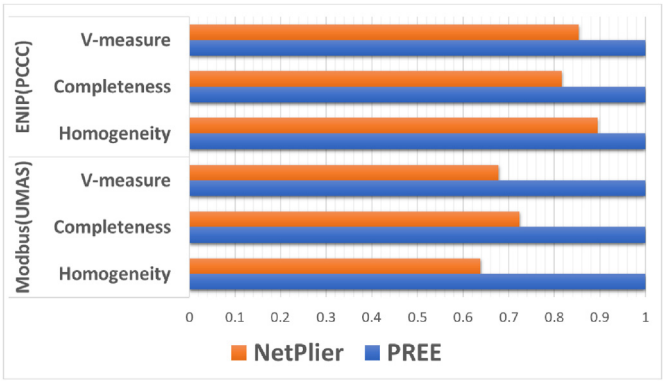

Fig. 11. Comparison of PREE with NetPlier on proprietary protocols (UMAS embedded in Modbus and PCCC embedded in ENIP).

performance of over 30%. Nonetheless, NetPlier showed better performance in ENIP (PCCC) than in Modbus (UMAS), and the completeness and homogeneity varied depending on the protocol's propensity.

NetPlier infers keyword candidates to cluster classes accurately. Function codes are typically used as the key field to indicate the class, as is the case in most other situations. However, in instances where a proprietary protocol is embedded in a known protocol (e.g., UMAS in Modbus), there are two kinds of function code fields, one in the known protocol and one in the proprietary protocol. As a result, even if NetPlier correctly infers the function code in the known protocol (Modbus) as a keyword, maintaining a homogeneity of 1.0, the completeness remains low because the proprietary function code is the true classification.

NetPlier, Netzob, and Discoverer were unable to infer proprietary function fields such as Modbus (UMAS) and ENIP (PCCC). This trend occurred due to two main reasons. Firstly, the probabilistic approach is not appropriate for analyzing proprietary protocols with dual structures. Secondly, entropy-based alignment performed by MAFFT does not handle proprietary fields located within the payload effectively.

The performance of proprietary protocols, such as Modbus (UMAS) and ENIP (PCCC), was inversely proportional to the number of internal function types. Although performance tended to increase on a large scale as the number of packets increased, it never exceeded a certain value. This is due to the proprietary function field not being recognized as a possible keyword candidate based on entropy. It's worth noting that poor metrics were not solely caused by proprietary protocols. The example packets' usage patterns differed somewhat from the actual behavior of PLCs.

7. PREE applications for vulnerability study and forensic analysis of different attacks

7.1. PREE application 1: vulnerability study on CLICK PLC

Network-based attacks on PLCs often require knowledge of proprietary protocols. For example, reverse engineering was used to exploit PLC authentication in Adeen et al. (Ayub et al., 2021b). Kalle et al. (2019) leveraged knowledge of the Modbus protocol for a control logic injection attack on Schneider Electric's Modicon M221. Syed et al. (Qasim et al., 2022) demonstrated a new type of attack that disrupts the physical process by targeting the control engine (responsible for executing control logic). They identified messages for starting/stopping the control engine of a PLC by analyzing network communication between the PLC and software, then modified fields and replayed the messages to stop the engine. We extended their work by conducting a control engine attack on the CLICK PLC using protocol knowledge obtained through PREE.

Adversary Model: We assume the adversary is inside the ICS network and can communicate with the target PLC, sniff its communication with engineering software, initiate connections, and send malicious messages.

Experimental Setup: We used AutomationDirect's CLICK Koyo PLC model C0-10DD2E-D with firmware version 2.60. The engineering software was running on Windows 10 in a virtual machine, and the attacker scripts ran on an Ubuntu 18.04 virtual machine, both in the same network.

Attack Implementation: We started by changing the mode of a PLC using the engineering software, captured the network traffic, and analyzed the differences to identify the messages responsible for switching the PLC from start to stop. Once identified (Fig. 12), we created a python script that modifies the message using the protocol knowledge from PREE and sends it to the target PLC to change its operation mode.

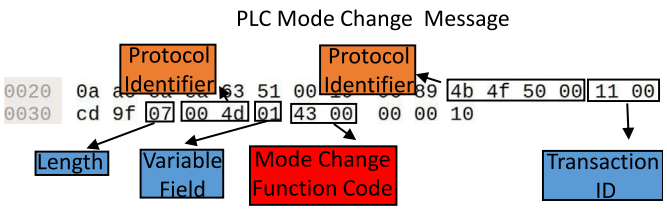

Fig. 12. Message of a control engine attack on CLICK PLC, showing the function code used to change the PLC mode.

7.2. PREE application II: forensic analysis of different attacks on CLICK PLC using SNORT

The communication between a PLC and the attacker, if captured during an attack, can serve as a forensic artifact. A forensic investigator with knowledge of the PLC protocol can use SNORT to analyze network dumps (pcap files) for evidence of an attack. SNORT is a widely used IPS/IDS tool, and it is used in projects such as (Quickdraw-snort and https, 2022). It analyzes messages using user-defined rules and generates alerts for matching packets.

Snort Rule for Detecting Control Engine Attack: To investigate if control logic engine attacks happened on a CLICK PLC, we used PREE to analyze attack messages and find fields in the CLICK protocol. Fig. 13 shows the SNORT rules we developed. While analyzing a network dump a SNORT rule raises an alert if it finds a message containing the signature of a PLC mode change i. e“4 b 4f 50 00” at offset 0 and “07 00 4 d 01 43 00” starting from offset 8. Our evaluation shows that this rule effectively detects control engine attacks on a CLICK PLC.

Snort Rule for Control Logic Injection Attack: To detect control logic injection attacks on the CLICK PLC, where an attacker tries to download malicious control logic to the PLC, we need to identify the request message used for writing data and its signature. We captured network communication while downloading a benign control logic on the PLC and identified write request messages (those with the largest size and larger than the corresponding response message). Using PREE, we extracted protocol identifiers, length, and function code and developed a SNORT rule as shown in Fig. 13. The SNORT rule raises an alert when it detects a message with the signature “4 b 4f 50 00” at offset 0 and “0a 00 4 d 01 65 05” starting from offset 8. In this way, the forensic investigator can analyze the network dump to find evidence of a control logic injection attack on the targeted PLC.

Snort Rule for Control Logic Theft Attack: To detect control

Snort Rule Template for Detecting Control Engine Attack

```
alert udp any any -> PLCIP 25425 (content:"|4b 4f 50 00|";offset:0;
 depth:4; content:"|07 00 4d 01 43 00|"; offset:8;
 depth:6; msg:“PLC Mode change attempted“)
```

Snort Rule Template for Detecting Control Logic Injection Attack

```
alert udp any any -> PLCIP 25425 (content:"|4b 4f 50 00|";offset:0;
 depth:4; content:"|0a 00 4d 01 65 05|"; offset:8;
 depth:6; msg:“Control Logic write attempt“)
```

Snort Rule Template for Detecting Control Logic Theft Attack

```
alert udp any any -> PLCIP 25425 (content:"|4b 4f 50 00|";offset:0;
 depth:4; content:"|0a 00 4d 01 65 04|"; offset:8;
 depth:6; msg:“Control Logic Read attempt“)
```

Fig. 13. Snort rules to detect different attacks on CLICK PLC.

logic theft attacks, where an attacker tries to read the logic on a PLC, we need to identify the unique signature of read request messages. We captured the communication between the PLC and engineering software during a control logic upload and found the read request messages by looking for smaller requests compared to other requests and smaller than their corresponding response. Then we used PREE to discover different fields in these messages and developed the snort rule that raises an alert whenever it detects a message in the network dump with the following signature; bytes "4 b 4f 50 00" at offset 0, and "0a 00 4 d 01 65 04" starting from offset 8. In this way, a forensic investigator can discover if a control logic theft attack happened on the targeted PLC.

8. Conclusion

In industrial control systems, proprietary protocols are commonly used to establish communication between PLCs and their engineering software. The knowledge of the location and meaning of various fields in these protocols can enhance the effectiveness of existing security solutions, support the development of new tools, and aid the forensic community in investigating network-based attacks on PLCs. Consequently, we introduce a novel tool for reversing proprietary ICS protocols: the protocol reverse engineering engine PREE.

Our observations revealed that many ICS protocols have similar characteristics and share common fields due to operational requirements. This led us to propose the hypothesis that knowledge of one ICS protocol can aid in reverse engineering other proprietary protocols. In this paper, we present PREE, a tool that helps users develop heuristics for identifying fields in proprietary ICS protocols. To test our hypothesis, we employed three techniques to develop seven heuristics, which were applied to six different protocols (Modbus, UMAS, ENIP, PCCC, CLICK, OMRON FINS). Our evaluation results indicate that PREE is able to identify several common fields in these protocols, such as "Length", "Transaction ID", "Message Type ID", etc. Furthermore, we showed the practical application of protocol knowledge to investigate 3 different network-based attacks on CLICK PLC.

Disclaimer

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

Acknowledgment

This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 17STCIN00001-05-00.

References

Ahmed, I., Obermeier, S., Naedele, M., R III, G.G., 2012. Scada systems: challenges for forensic investigators. Computer 45, 44–51.
Ahmed, I., Roussev, V., Johnson, W., Senthivel, S., Sudhakaran, S., 2016. A SCADA system testbed for cybersecurity and forensic research and pedagogy. In: Proceedings of the 2nd Annual Industrial Control System Security Workshop. ICSS).
Ahmed, I., Obermeier, S., Sudhakaran, S., Roussev, V., 2017. Programmable logic controller forensics. IEEE Security Privacy 15 (6), 18–24.
Awad, R.A., Rais, M.H., Rogers, M., Ahmed, I., Paquit, V., 2023. Towards generic memory forensic framework for programmable logic controllers. selected papers of the Tenth Annual DFRWS EU Conference Forensic Sci. Int.: Digit. Invest. 44, 301513. https://doi.org/10.1016/j.fsidi.2023.301513. https://www.sciencedirect.com/science/article/pii/S2666281723000148. URL.
Ayub, A., Johnson, J., Ahmed, I., et al., 2021. Attacking the Iec-61131 Logic Engine in Programmable Logic Controllers in Industrial Control Systems. Tech. Rep.. Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States).
Ayub, A., Yoo, H., Ahmed, I., 2021. Empirical study of plc authentication protocols in industrial control systems. In: 2021 IEEE Security and Privacy Workshops (SPW). IEEE, pp. 383–397.
Ayub, A., Zubair, N., Yoo, H., Jo, W., Ahmed, I., 2023. Gadgets of gadgets in industrial control systems: return oriented programming attacks on plcs. In: 2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). IEEE.
Biondi, P., 2022. The Scapy Community. https://scapy.net/.
Bossert, G., Guihéry, F., Hiet, G., 2014. Towards automated protocol reverse engineering using semantic information. In: Proceedings of the 9th ACM Symposium on Information. computer and communications security, pp. 51–62.
Chang, Y., Choi, S., Yun, J.-H., Kim, S., 2018. One step more: automatic ics protocol field analysis. In: D'Agostino, G., Scala, A. (Eds.), Critical Information Infrastructures Security. Springer International Publishing, Cham, pp. 241–252.
Cui, W., Kannan, J., Wang, H.J., 2007. Discoverer: automatic protocol reverse engineering from network traces. In: USENIX Security Symposium, pp. 1–14.
S. Kalle, N. Ameen, H. Yoo, I. Ahmed, Clik on plcs! attacking control logic with decompilation and virtual plc, Proceedings 2019 Workshop on Binary Analysis Research.
Katoh, K., Misawa, K., Kuma, K.-i., Miyata, T., 2002. Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. Nucleic Acids Res. 30 (14), 3059–3066.
Kim, H., Kim, S., Jo, W., Kim, K.-H., Shon, T., 2021. Unknown payload anomaly detection based on format and field semantics inference in cyber-physical infrastructure systems. IEEE Access 9, 75542–75552.
Ládi, G., Buttyán, L., Holczer, T., 2018. Message format and field semantics inference for binary protocols using recorded network traffic. In: 2018 26th International Conference on Software. Telecommunications and Computer Networks (SoftCOM), pp. 1–6.
Li, H., Wang, B., Xie, X., 2020. An improved content-based outlier detection method for ics intrusion detection. EURASIP J. Wirel. Commun. Netw. (1).
Lin, Z., Jiang, X., Xu, D., Zhang, X., 2008. Automatic protocol format reverse engineering through context-aware monitored execution. NDSS 8, 1–15.
Luo, Z., Zuo, F., Jiang, Y., Gao, J., Jiao, X., Sun, J., 2019. Polar: function code aware fuzz testing of ics protocol. ACM Trans. Embed. Comput. Syst. 18, 1–22. https://doi.org/10.1145/3358227.
Luo, Z., Zuo, F., Shen, Y., Jiao, X., Chang, W., Jiang, Y., 2020. Ics protocol fuzzing: coverage guided packet crack and generation. In: 2020 57th ACM/IEEE Design Automation Conference. DAC), pp. 1–6.
Z. Luo, F. Zuo, Y. Jiang, J. Gao, X. Jiao, J. Sun, Polar: function code aware fuzz testing of ics protocol, ACM Trans. Embed. Comput. Syst. 18 (5s).
J. Narayan, S. K. Shukla, T. C. Clancy, A survey of automatic protocol reverse engineering tools, ACM Comput. Surv. 48 (3).
Niedermaier, M., Fischer, F., von Bodisco, A., 2017. Propfuzz — an it-security fuzzing framework for proprietary ics protocols. In: 2017 International Conference on Applied Electronics. AE), pp. 1–4.
Paxson, V., 1999. Bro: a system for detecting network intruders in real-time. Comput. Network. 31 (23–24), 2435–2463.
Qasim, S.A., Lopez, J., Ahmed, I., 2019. Automated reconstruction of control logic for programmable logic controller forensics. In: Lin, Z., Papamanthou, C., Polychronakis, M. (Eds.), Information Security. Springer International Publishing, Cham, pp. 402–422.
Qasim, S.A., Smith, J.M., Ahmed, I., 2020. Control logic forensics framework using built-in decompiler of engineering software in industrial control systems. Forensic Sci. Int.: Digit. Invest. 33, 301013.
Qasim, S.A., Ayub, A., Johnson, J., Ahmed, I., 2022. Attacking the iec 61131 logic engine in programmable logic controllers. In: Staggs, J., Shenoi, S. (Eds.), Critical Infrastructure Protection XV. Springer International Publishing, Cham, pp. 73–95.
Quickdraw-snort. https://github.com/digitalbond/Quickdraw-Snort, 2022.
Rais, M.H., Awad, R.A., Lopez Jr., J., Ahmed, I., 2021. Jtag-based plc memory acquisition framework for industrial control systems. Forensic Sci. Int.: Digit. Invest. 37, 301196.
Rais, M.H., Awad, R.A., Lopez Jr., J., Ahmed, I., 2022. Memory forensic analysis of a programmable logic controller in industrial control systems. Forensic Sci. Int.: Digit. Invest. 40, 301339.
Ren, W., Yardley, T., Nahrstedt, K., 2018. Edmand: edge-based multi-level anomaly detection for scada networks. In: 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm). IEEE, pp. 1–7.
Rosenberg, A., Hirschberg, J., V-measure, 2007. A conditional entropy-based external cluster evaluation measure. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. EMNLP-CoNLL), pp. 410–420.
Senthivel, S., Ahmed, I., Roussev, V., 2017. Scada network forensics of the pccc protocol. Digit. Invest. 22, S57–S65.
Senthivel, S., Dhungana, S., Yoo, H., Ahmed, I., Roussev, V., 2018. Denial of engineering operations attacks in industrial control systems. In: Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, pp. 319–329.
K. Shim, Y.-H. Goo, M.-S. Lee, M.-S. Kim, Clustering method in protocol reverse engineering for industrial protocols, Int. J. Netw. Manag. 30.
Wang, R., Shi, Y., Ding, J., 2020. Reverse engineering of industrial control protocol by xgboost with v-gram. In: 2020 IEEE 6th International Conference on Computer

and Communications (ICCC). IEEE, pp. 172–176.
Wu, Z., Shu, M., Shi, J., Cao, Z., Xu, F., Li, Z., Xiong, G., Yiu, S.M., 2019. How to reverse engineer ics protocols using pair-hmm. In: Satapathy, S.C., Joshi, A. (Eds.), Information and Communication Technology for Intelligent Systems. Springer Singapore, Singapore, pp. 115–125.
Yang, H., Cheng, L., Chuah, M.C., 2019. Deep-learning-based network intrusion detection for scada systems. In: 2019 IEEE Conference on Communications and Network Security (CNS), pp. 1–7.
Ye, Y., Zhang, Z., Wang, F., Zhang, X., Xu, D., 2021. Netplier: Probabilistic Network Protocol Reverse Engineering from Message Traces. NDSS, pp. 1–18.
Ye, Y., Zhang, Z., Wang, F., Zhang, X., Xu, D., 2021. Netplier tool data. https://github.com/netplier-tool/NetPlier/tree/master/data.
Yoo, H., Kalle, S., Smith, J., Ahmed, I., 2019. Overshadow plc to detect remote control-logic injection attacks. In: Perdisci, R., Maurice, C., Giacinto, G., Almgren, M. (Eds.), Detection of Intrusions and Malware, and Vulnerability Assessment. Springer International Publishing, Cham, pp. 109–132.
Yoo, H., Ahmed, I., 2019. Control logic injection attacks on industrial control systems. In: Dhillon, G., Karlsson, F., Hedström, K., Zúquete, A. (Eds.), ICT Systems Security and Privacy Protection. Springer International Publishing, Cham, pp. 33–48.
Yusheng, W., Kefeng, F., Yingxu, L., Zenghui, L., Ruikang, Z., Xiangzhen, Y., Lin, L., 2017. Intrusion detection of industrial control system based on modbus tcp protocol. In: 2017 IEEE 13th International Symposium on Autonomous Decentralized System. ISADS), pp. 156–162. https://doi.org/10.1109/ISADS.2017.29.
Zubair, N., Ayub, A., Yoo, H., Ahmed, I., 2022. Control logic obfuscation attack in industrial control systems. In: 2022 IEEE International Conference on Cyber Security and Resilience (CSR). IEEE, pp. 227–232.
Zubair, N., Ayub, A., Yoo, H., Ahmed, I., 2022. Pem: remote forensic acquisition of plc memory in industrial control systems. Forensic Sci. Int.: Digit. Invest. 40, 301336.