Reverse Engineering Industrial Protocols Driven By Control Fields

Zhen Qin*, Zeyu Yang*, Yangyang Geng†, Xin Che*, Tianyi Wang*, Hengye Zhu*, Peng Cheng*, and Jiming Chen*
*The College of Control Science and Engineering, Zhejiang University, Hangzhou, China
†Information Engineering University, China

IEEE INFOCOM 2024 - IEEE Conference on Computer Communications | 979-8-3503-8350-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/INFOCOM52122.2024.10621405

***Abstract*—Industrial protocols are widely used in Industrial Control Systems (ICSs) to network physical devices, thus playing a crucial role in securing ICSs. However, most commercial industrial protocols are proprietary and owned by their vendors, which impedes the implementation of protections against cyber threats. In this paper, we design REInPro to Reverse Engineer Industrial Protocols. REInPro is inspired by the fact that the structure of industrial protocols can be determined by a particular field referred to control field. By applying a probabilistic model of network traffic behavior, REInPro automatically identifies the control field and groups the associated network traffic into clusters. REInPro then infers critical semantics of industrial protocols by differentiating the features of corresponding protocol fields. We have experimentally implemented and evaluated REInPro using 8 different industrial protocols across 6 Programmable Logic Controllers (PLCs) belonging to 5 original equipment manufacturers. The experimental results show REInPro to reverse-engineer the formats and semantics of industrial protocols with an average correctness/perfection of 0.70/0.58 and 0.96/0.39.**

I. Introduction

Industrial protocols are widely used in Industrial Control Systems (ICS) to network physical devices according to the *specifications* (e.g., format, semantics). A common practice in defining industrial protocols' specifications is to use a *control field* to specify the structure of communication messages. For example, the monitoring/programming systems in Fig. 1 read and write the memory of the Programmable Logic Controllers (PLCs) following a uniform message structure specified by the control field `0x01`. However, most protocol specifications of commercial controllers are proprietary and owned by vendors [1], highly impeding the implementation of protection approaches, such as intrusion detection [2]–[4], devices testing [5]–[11], honeypot implementations [12], etc.

To bridge the gap between security application requirements and unknown protocol specifications, researchers and cybersecurity companies have put much effort into reverse-engineering industrial protocols based on the program of protocol implementation and communication traffic, respectively. However, program-based methods require access to program binaries [13]–[15], which are difficult to obtain from most PLCs. For example, Siemens provides the encrypted firmware to update their PLCs [16]. In contrast, the traffic-based protocol analyzing methods are more feasible because the required network traffic is much easier to obtain. One common approach in analyzing the communication traffic is to align the message sequence [17]–[24]. Then based on the aligned sequence, the communication traffic will be further clustered according to the defined similarity scores. By identifying the commonality within each message cluster, the protocol formats are finally abstracted. Another approach to reverse-engineer protocols from network traffic relies on frequent itemsets [25]–[27]. This approach is based on the observation that a subsequence is more likely to represent a key field if it appears continuously and with high frequency. Taking high-frequency sets as candidate fields, semantic information (e.g., the length field of the protocol) is inferred by applying corresponding statistical methods. However, the above methods still face challenges when applied to industrial protocols:

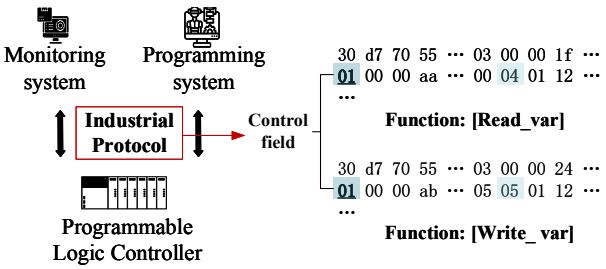


Fig. 1. An exemplary communication between control devices and monitoring/programming systems using industrial protocols.

- **The different communication tasks may have a similar structure for industrial protocol.** The similarities of messages in different communication tasks will lead to incorrect message clustering, which voids the sequence alignment based protocol reverse engineering. As shown in Fig. 1, the messages of reading and writing PLC memory — specified by the fields of `0x04`/`0x05` — have similar structures.
- **The communication tasks of the industrial protocol are decided by multiple keywords.** Industrial protocols contain multiple keywords that work together to define communication tasks and ensure reliable communication. However, the diverse statistical properties of these keywords (e.g., entropy) make them indistinguishable when mining the frequent itemsets and further decrease the performance of protocol reverse engineering.

This work was supported in part by the National Natural Science Foundation of China under Grant 62293510/62293511, 62303411, and 61833015, the Zhejiang Provincial Natural Science Foundation under Grant LZ22F030010, and the Fundamental Research Funds for the Central Universities under Grant 226-2023-00111.

979-8-3503-8350-8/24/$31.00 ©2024 IEEE 2408

In this work, we propose REInPro, which automatically Reverse-Engineers Industrial Protocols from network traffic. REInPro is built on the fact that a specific field of industrial protocols — i.e., control field — determines the structure of communication messages. Inspired by NetPlier [28], which leverages a probabilistic model to characterize the behavior of protocol keywords, REInPro identifies the control field based on its unique behaviors: (i) messages with the same control field typically have similar structures, including the arrangement of message fields; (ii) the control field generally appears at the front of the message sequence. Atop on the identified control field, REInPro groups the associated network traffic into clusters to further infer critical field semantics. Specifically, REInPro leverages the unique characteristics of each kind of field: ❶ inferring sequence numbers by Hamming distance; ❷ inferring address fields according to the IP address (or port number); ❸ extracting length fields by size-based classification; ❹ inferring function codes using entropy. Ultimately, REInPro acquires the specification of an unknown protocol by merging adjacent fields with the same semantics. We have evaluated REInPro using 8 different industrial protocols across 6 PLCs that belong to 5 original equipment manufacturers. The experimental results show REInPro to infer control fields with 100% accuracy, and further the formats of industrial protocols with an average correctness/perfection of 0.70/0.58, outperforming Netzob (0.51/0.31), FieldHunter (0.41/0.17) and NetPlier (0.61/0.45).

In summary, this paper makes the following contributions:

- REInPro, a method to Reverse Engineer Industrial Protocols has been proposed. REInPro is grounded on a set of *control fields*, which determine the protocol structure and enable the accurate clustering of communication messages.
- REInPro identifies the semantics of critical fields of industrial protocols and generates protocol specifications, facilitating the implementation of ICS protection strategies.
- A set of communication messages specifying various communication functions of different PLCs has been collected, which is available at https://github.com/hi-zer/Industrial-protocols-dataset.

II. Background

A. Industrial Protocol Primer

Industrial protocols define the rule of communication between the control devices (e.g., PLC) and other entities (e.g., programming software) in ICSs. By leveraging the predetermined rule, devices can easily understand and interpret the commands they receive, allowing for efficient and effective communication between devices. For example, the S7Comm protocol is a proprietary protocol applied between Siemens PLCs and their programming software. A part of the specification for the S7Comm protocol is explained in Fig. 2. The fields (i.e., the blocks displayed in the figure) are the basic units that compose a protocol. They contain specific semantic information, such as:

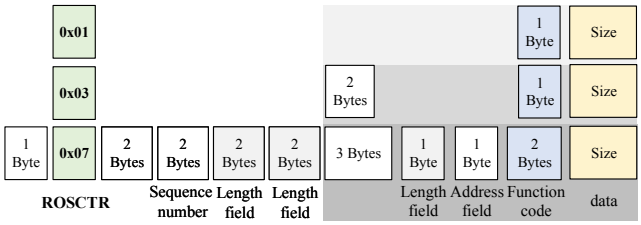


Fig. 2. General specification of S7Comm protocol.

- **Sequence Number.** The sequence number is used for flow control to keep track of the order in which messages are sent. It is a numerical value that is incremented or remains unchanged with each new message, helping the sender and receiver ensure message freshness, prevent replay attacks, and maintain proper message sequencing.
- **Address field.** The address field denotes the destination or source of messages and specifies the devices intended to send or receive messages. It plays a crucial role in ensuring that messages are correctly routed and delivered to the intended recipients in a communication system.
- **Length field.** The length field defines the length of the whole message or a portion (i.e., the parameter or payload section) of the message. The length field is used to format and segment messages.
- **Function code.** Function code is a field that specifies the type of function or operation to be performed by the recipient device. The value of function code defines the specific functionalities of the message, such as reading, writing, uploading, downloading, etc.

These fields are crucial in securing/exploiting industrial protocols as they provide knowledge of the message structure and purpose of the message exchanged between devices.

B. Control Field Identification

Most commercial industrial protocols, such as S7Comm by Siemens and PCCC by Rockwell, are designed to perform different functions using different message types, which are decided by a special field — i.e., the *control field*. Taking the ROSCTR field of S7Comm in Fig. 2 as an example, the structure of messages is decided by the values of the control field. Specifically, the ROSCTR value of {`0x01`, `0x03`, `0x07`} specifies the message sequence for {JOB, ACK, Userdata}, respectively, and allocates the corresponding function code at the {17th, 19th, 22nd-23rd} byte(s) of the message. More detailed examples of control field and corresponding message format are summarized in Table I, where $\{m^{c0}, m^{c1}\}$, $\{m^{s0}, m^{s1}\}$, and $\{m^{c2}, m^{s2}, m^{c3}, m^{s3}\}$ are the clusters determined by the value of the control field.

NetPlier [28], a protocol reverse engineering tool, applies a probabilistic model to extract clustering keywords. By deploying a multiple sequence alignment algorithm (MSA), NetPlier first generates keyword candidates and their corresponding probabilities. NetPlier then exploits four constraints on message observations to identify the keyword further: (i) *Similarity*, postulating messages with the same keyword exhibit

TABLE I
THE MESSAGES AND FIELDS OF S7COMM PROTOCOL

ID	[0,1]	[2,3]	[4]	[5-7]	[8]	[9,10]	[11,12]	[13,14]	[15,16]	[17]	[18]	[19]	[20]	[21]	[22,23]	Data	Type
m^{c_0}	0300	0031	02	f08032	**01**$^\bullet$	0000	0e00	0020	0000	$1a^*$						...	JOB
m^{c_1}	0300	0023	02	f08032	**01**$^\bullet$	0000	0100	0012	0000	$1b^*$						...	JOB
m^{s_0}	0300	0014	02	f08032	**03**$^\bullet$	0000	0e00	0001	0000	00	00	$1a^*$				...	ACK
m^{s_1}	0300	00f7	02	f08032	**03**$^\bullet$	0000	0100	0002	00e2	00	00	$1b^*$				...	ACK
m^{c_2}	0300	001d	02	f08032	**07**$^\bullet$	0000	2300	0008	0004	00	01	12	04	11	4701^*	...	Userdata
m^{s_2}	0300	002b	02	f08032	**07**$^\bullet$	0000	2300	000c	000e	00	01	12	08	12	8701^*	...	Userdata
m^{c_3}	0300	0021	02	f08032	**07**$^\bullet$	0000	5203	0008	0008	00	01	12	04	11	4401^*	...	Userdata
m^{s_3}	0300	017d	02	f08032	**07**$^\bullet$	0000	5203	000c	0160	00	01	12	08	12	8401^*	...	Userdata

The fields with "$\bullet$" and "$*$" represent the control field and corresponding function code, respectively.

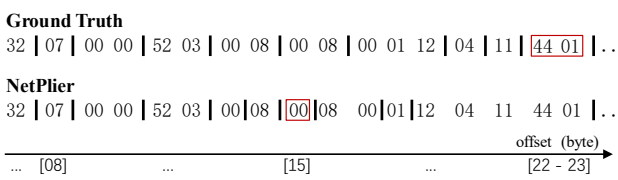


Fig. 3. Comparison of inferred format for S7Comm protocol.

more similarity; (ii) *Remote Coupling*, expecting messages from one side (client/server) to have corresponding responses on the other (server/client); (iii) *Dimension*, limiting the cluster quantity generated by the keyword while maintaining sufficient messages in each cluster; (iv) *Structure Coherence*, demanding consistent gap numbers in messages with identical keywords, as produced by the MSA algorithm.

However, it should be noted that the effectiveness of NetPlier would be decreased when applied to industrial protocols. Fig. 3 presents the results of NetPlier's reverse engineering, where the 15th byte of the message (i.e., `0x00`) is recognized as a keyword. However, according to the ground truth of the S7Comm protocol (refer to the message of m^{c_3} in Table I), the correct keywords are the bytes with offset 22 to 23. Besides, the MSA algorithm in NetPlier makes it inefficient when reverse-engineering industrial protocols. Given the fact that industrial protocols have much longer messages (over hundreds of bytes) compared to the normal network protocols, an excessive amount of unnecessary filling gaps will be introduced. This fact, in turn, greatly increases the complexity of the MSA process.

C. Basic Idea of REInPro

NetPlier's probabilistic model is a potent strategy for keyword identification. However, it tends to generate numerous unnecessary gaps and is affected by the varying field values in industrial protocols, making it incapable of identifying the control fields that determine the structure of ICS protocols. Our method, REInPro, identifies the control field based on its unique behaviors. Firstly, based on the analysis of the industrial protocol mentioned above, it was observed that messages sharing the same control field exhibit a similar arrangement of fields in terms of format, regardless of variations in field values. This observation inspires us to improve the similarity quantification by introducing the lateral distance distribution of individual messages as a structural abstraction. REInPro considers that some fields in the same message type may

TABLE II
COMMON INDUSTRIAL PROTOCOLS

Protocol	Number of Messages client	server	total	Covered/Total Function code
Modbus	144	145	289	10/21
Dnp3	302	354	636	22/25
S7Comm	915	956	1871	27/Unknown
S7Comm_Plus	507	1262	1769	8/Unknown
PCCC	700	698	1398	17/Unknown
Omron_Fins	800	798	1598	18/Unknown
Delta_P	666	670	1336	18/Unknown
Hollysys_P	326	364	690	10/Unknown

carry identical semantic information but different values. For instance, in many industrial protocols, the session ID field is randomly generated for tracking communication sessions. While session ID values may differ among messages of the same type, they should be considered structurally similar due to the shared semantic information they carry. Calculating value-based similarity alone does not accomplish this. The randomness of the session ID can result in a significant lateral distance between the same fields in different messages, which can be applied for structure abstraction. By considering both value and structure similarity, REInPro offers a more comprehensive and precise measure of similarity between message sequences for industrial protocols, even when certain fields have differing values but similar structural attributes.

Furthermore, given that the control field generally appears toward the front of a message sequence, we have incorporated an offset constraint accordingly. By employing the probabilistic model of network traffic behavior, REInPro automatically identifies the control field and groups the corresponding network traffic into clusters. After the messages have been correctly clustered, their structures become fixed, priming them for subsequent inference of format and semantics.

III. DESIGN OF REINPRO

As depicted in Fig. 4, taking the network traffic of various communication functions as input, REInPro identifies the control field, infers the field semantics, and generates the protocol specification. In the following, we detail the design of REInPro in conjunction with the S7Comm protocol.

A. Traffic Collection

Sufficient traffic is the prerequisite for reverse engineering the corresponding communication protocol. While some

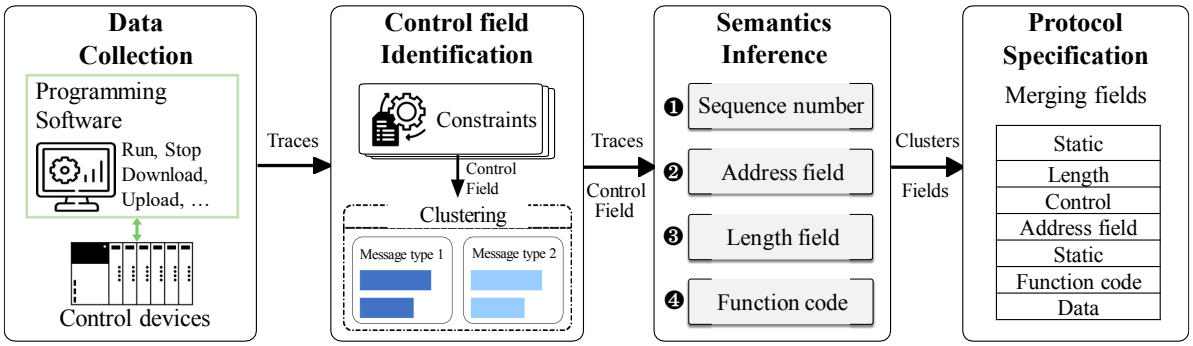


Fig. 4. An overview of REInPro.

publicly available communication datasets exist for research purposes, the generally accepted datasets for reverse engineering industrial protocols are not yet accessible. However, capturing sufficient traffic from the real-world control system is impractical because communication thereof barely changes during regular operation.

In this paper, we built a dataset by manually clicking through all the functions the PLC programming software provided, covering a wide range of function codes. The communication traffic of 8 widely used industrial protocols has been collected, as summarized in Table II. For instance, the traces of Dnp3 contain 636 messages, covering 22 (out of 25) function codes. When collecting traces in a complex industrial environment, there may be interference traces that do not conform to the target protocol. The software Wireshark [29] is used to filter irrelevant traces, such as TCP handshakes.

B. Control Field Identification

The message structure is determined by the value of the control field for industrial protocols. Thus for further analysis, we cluster messages by the result of control field identification. Taking the probabilistic framework of NetPlier [28] as input, REInPro identifies the protocol's control field based on its unique behaviors.

1) Message Similarity: To eliminate the influence of payload fields in calculating similarity, we trim all message sequences to the shortest length, computed as the minimum length among all message sequences. NetPlier computes the similarity score between two message sequences with Eq. (1):

$$s = \frac{\text{Number of identical bytes}}{\text{Sum of total bytes of the two messages}}, \quad (1)$$

where identical bytes refer to those bytes at the same position whose values are the same.

To optimize the similarity quantification, we introduce the *structure similarity* using the Hamming distance distribution of communication messages. Note that different protocol fields have different behavior in Hamming distance. For example, the protocol field of session IDs exhibits a large Hamming distance due to its discontinuous states. Similarly, discontinuous states among different protocol fields will also induce a significantly large Hamming distance. Thus, a transition in Hamming distance — from large to small or vice versa, suggests a potential switch between different types of fields. Hamming distance measures the number of different bits between two bytes [30]. Given the message sequence M, the Hamming distance between two adjacent bytes M_x^k and M_x^{k+1} is calculated as:

$$\mathrm{D}(M_x^k) = \frac{1}{8}\sum_{i=0}^{7}[M_x^k[i] \oplus M_x^{k+1}[i]], \quad (2)$$

where $M_x^k[i]$ is the i-th bit of the k-th byte in message M_x.

To incorporate the structure of messages into the similarity consideration, we adjust the initial value-matched score as:

$$\mathrm{s} = \begin{cases} 1.0, & V(M_x^k) = V(M_y^k)\ \&\ D(M_x^k) = D(M_y^k), \\ 0.5, & V(M_x^k) = V(M_y^k)\ or\ D(M_x^k) = D(M_y^k), \\ 0, & \text{otherwise}, \end{cases} \quad (3)$$

where V represents the value of the byte, and M_x^k and M_y^k are the k-th bytes in the x-th and y-th messages, respectively. Using this similarity matrix, we can compute the similarity between all pairs of message sequences.

2) Offset Constraint: As it has been observed that the control field generally appears at a position towards the front of the message sequence, an offset constraint is introduced to assign a higher score to fields with a more forward position. The metric for this constraint is described below.

$$p_o = 1 - \frac{f.\text{offset}}{l_{\min}}, \quad (4)$$

where f.offset is the offset of the candidate control field, and $l_{\min}$ is the minimum length of the message sequence. The field that satisfies these constraints is considered the most probable control field, based on which messages are grouped into clusters. If no control field exists, messages are forwarded directly to semantics inference without clustering.

C. Semantics Inference

Once messages have been effectively clustered, REInPro applies heuristic techniques to each cluster for semantics inference, aiming to infer five types of field semantics: the control field (Section III-B), the sequence number, the address field, the length field, and the function code.

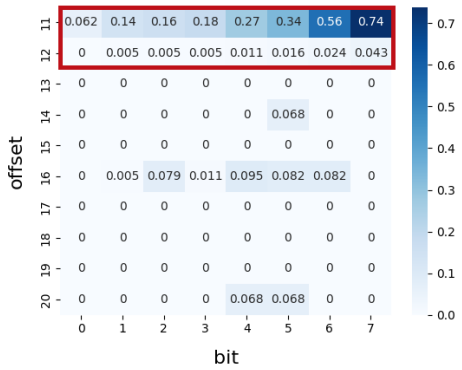


Fig. 5. The average Hamming distance distribution for the S7Comm communication messages.

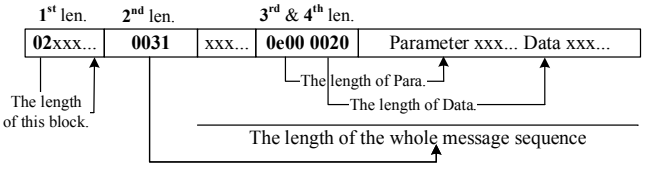


Fig. 6. The length field description of S7Comm protocol.

1) Sequence Number: The sequence number conveys the temporal aspects of a message sequence. To estimate the average rate of change for each bit in a series of messages, we transform the raw messages into bit-level data and track the changes using the Hamming distance. A counter is kept for each bit, which is incremented whenever a bit changes. The average Hamming distance for i-th bit, denoted $\bar{D}_i$, is calculated using Eq. (5):

$$\bar{D}_i = \frac{1}{|M|} \sum_{j=0}^{|M|-1} [M_j[i] \oplus M_{j+1}[i]], 0 \leq i \leq n, \qquad (5)$$

where $M_j[i]$ represents the i-th bit of the j-th message in the sequence, n is the number of bits in each message, and $|M|$ is the total number of messages. If the *i*-th bit is zero (one) in M_j and the *i*-th bit is one (zero) in M_{j+1}, the counter for the *i*-th bit increments by one. Finally, the Hamming distance of each bit is calculated by dividing the corresponding bit counter by the total number of message sequences. Each bit's final estimated Hamming distance is distributed between 0 and 1. The closer to 0, the less the bit changes, while the closer to 1, the more frequently the bit changes.

If the average Hamming distance in a byte increases from one end to the other, this byte will be considered a sequence number. If two adjacent bytes both have monotonic Hamming distance, REInPro will combine them together as the sequence number. For example, Fig. 5 depicts the average Hamming distance of the S7Comm communication messages, where the bytes with offset 11 to 12 are inferred as a sequence number.

2) Address field: The address field can be obtained by correlating the fields with the message destination. First, the original messages are split into dynamic and static fields, and the number of distinct values for each dynamic field is counted. If the number of distinct values equals the number

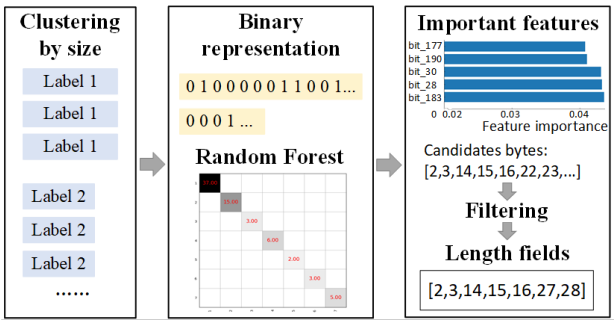


Fig. 7. Inferring the length field of S7Comm protocol.

of interacting devices, the field is considered a candidate for the address field. Then, messages are clustered by IP address or port number. If the candidate field becomes static, it is regarded as an address field.

3) Length field: The length field, indicating the size of associated variable-length data, can vary in number and location within a message, as demonstrated in Fig. 6. Encoding size for these fields can differ among protocols and even within the same protocol, like the 1^{st} length field using one-byte encoding, in contrast to two bytes for the remaining length fields. Furthermore, length fields may not always directly precede the data they reference, such as the 3^{rd} and 4^{th} length fields denoting the sizes of the parameter and data separately. Given these variations in encoding units and potential disjunctions between length fields and their related data, identifying multiple length fields via linear correlation with message length or through comparison with subsequent field sizes becomes a complex task.

To recognize multiple length fields, we utilize a classifier based on random forest. The idea is that multiple length fields in industrial protocols of the same type are typically located at the same position, making it possible to extract them by identifying important classification features that denote information about physical length. Specifically, to extract the length fields from protocol messages, a clustering approach is utilized where messages are grouped based on their physical length, and a corresponding length label is assigned to each cluster. However, clusters may suffer from class imbalance due to varying frequencies of collected functions or the presence of heartbeat packets, and REInPro addresses by oversampling the minority samples. Pre-processing is necessary to represent the data for applying machine-learning techniques. Since the length fields belong to the header data, the message sequences can be truncated to a minimum length l_{min}, eliminating the need for alignment. To create a fine-grained approach, the byte-level raw packet is converted to bit-level data, containing only two values (0 and 1). This conversion eliminates the need for normalization and creates a pipeline to the classifier. The data, once suitably formatted, can be processed through various classifiers to infer the length field. Based on the proposed method, the entire length field inference procedure can be summarized and depicted in Fig. 7.

Step 1 : Cluster messages by their physical length and

assign them with corresponding length labels.

$Step\ 2$: Oversample the minority samples to mitigate the issue of class imbalance.

$Step\ 3$: Truncate the message sequences to l_{min} and convert raw byte-level data to bit-level. Create a training set $\mathcal{M}_{train} = \{(\mathcal{M}_1, l_1), \ldots, (\mathcal{M}_i, l_i), \ldots, (\mathcal{M}_n, l_n)\}$ with 80% of the dataset. The testing set is built accordingly.

$Step\ 4$: If the classification accuracy on the testing set exceeds 90%, extract the significant features with a weight over 0.02. Convert these bit-level features back to byte-level to produce the length field candidates.

$Step\ 5$: Apply a filtering rule to infer the candidates for length fields, which asserts that the value of the field should be less than or equal to the sum of the subsequent bytes.

Note that the type of classifier does not impact the results of length field extraction. Empirical experiments have shown that classifiers such as Decision Tree, Random Forest, and Extra Tree can lead to identical results in inferring the length field. REInPro is implemented using a Random Forest classifier with n_estimators=1000 and max_depth=None, facilitating comprehensive and unrestricted tree growth, respectively. Given the inferred candidates of length fields, REInPro then deploys a linear regression to validate them. If only one length field candidate is identified, REInPro verifies whether its value highly correlates with the message length. However, if the protocol length field has several candidates, REInPro verifies, in turn, whether the sum of values of their subset has a linear correlation with the message length. REInPro only keeps those candidates that satisfy the correlation.

4) Function code: The function codes in industrial protocols define the operations performed by PLCs. The existing methods in inferring function codes are primarily built on the value distribution and offsets of the candidate fields [24], [27], [31], [32], which however, have the following limitations. Firstly, these methods are highly sensitive to the threshold. This threshold can be challenging to determine in real datasets, especially when dealing with diverse protocols, and may need to be re-evaluated for each new protocol. Secondly, an industrial protocol may contain more than one function code. In the S7Comm protocol, for example, the function group field and function code work together to define the function of a communication message. The methods may produce multiple false positives due to the fields with similar entropy and value range in complex protocols.

To address these limitations, we propose a novel method based on the correlation between the function and length of a message. Note that developing the message formats closely correlates to specific functions, so different message sequences with distinct functions exhibit varying field types and arrangements. Take the S7Comm protocol as an example, the messages defined by the function code of start/stop (`0x28`/`0x29`) are associated with zero-byte payload. Whereas the messages defined by the function code of read/write (`0x04`/`0x05`) carry a much longer payload, including the addresses to read/write and the concomitant values. The above facts induce reliable correlations between message length and function, motivating us to pay attention to the results of length field inference. The length field inference algorithm outputs the important bytes that are significantly correlated to message length. Besides the length fields, the function codes also influence the message lengths, even if their values are not directly linear to the message length. Hence, the fields that are excluded by the length field inference are further regarded as candidates of function codes. Then we decide if a candidate field w_k is a function code by the normalized entropy $H(w_k)$ in Eq. (6):

$$H(w_k) = -\frac{1}{\log(|w_k|)} \sum p(v) \log_2[p(v)], \tag{6}$$

where v is the value of candidate field w_k, and $p(v)$ is the corresponding probability of v. By taking the filtered fields from the length field inference, we reduce the number of fields to be considered for inferring function code. It is worth noting that during the experiments, for five out of the eight protocols, only the function codes were filtered out from the length fields, which resulted in high accuracy for function code inference.

D. Protocol Specification Generation

After the semantics inference process, REInPro derives the specification of the targeted protocol based on the reverse-engineered results. In particular, the adjacent fields with the same semantics, including the static field, are combined as one protocol field. If the structures defined by control fields possess the same format and semantics, REInPro will further merge these similar structures. In the end, REInPro outputs the reverse-engineered specifications, including the format and semantics for industrial protocols.

IV. Evaluation

We have evaluated REInPro with 8 industrial protocols: Modbus, Dnp3, S7Comm, S7Comm_Plus, PCCC, Omron_Fins, Delta_P, Hollysys_P. The specifications for {Modbus, DNP3} protocols are publicly available [33], [34], while the specifications for {S7Comm, S7Comm_Plus, PCCC, and Omron_Fins} protocols are obtained from Wireshark [29]. The specifications for {Delta, Hollysys} protocols are manually reverse-engineered. The ground truth of the above protocol specification is detailed in [35].

A. Experiments Setup and Evaluation Metric

REInPro is implemented on a laptop with Intel Core i5 1135G7 2.4GHz CPU and 4GB memory. In comparison, the classic reverse-engineering schemes of Netzob [17], NetPlier [28], FieldHunter [25], and IPART [27] are also implemented.

We evaluate the inferred specifications against the ground truth to assess the correctness and perfection of the inferred fields and semantics. Correctness in field inference is calculated as the ratio of correctly segmented fields to all inferred fields, while perfection is the ratio of accurately inferred fields to all fields defined in the protocol specification. Similarly, in semantics inference, correctness is defined as the ratio of bytes with correct semantics to all bytes of the inferred fields, while perfection is the ratio of bytes with accurate

TABLE III
EVALUATING MESSAGE FORMAT INFERENCE WITH DIFFERENT REVERSE-ENGINEERING SCHEMES.

Protocol	Netzob			FieldHunter			NetPlier			REInPro		
	Corr.	Perf.	F1_Score	Corr.	Perf.	F1_Score	Corr.	Perf.	F1_Score	Corr.	Perf.	F1_Score
Modbus	0.60	0.33	0.43	0.50	0.33	0.40	0.67	0.50	0.57	0.67	0.50	0.57
Dnp3	0.60	0.44	0.51	0.33	0.11	0.17	0.73	0.67	0.70	0.73	0.67	0.70
S7Comm	0.50	0.18	0.26	0.39	0.15	0.22	0.46	0.29	0.36	**0.74**	**0.62**	**0.67**
S7Comm_Plus	0.25	0.06	0.10	0.40	0.14	0.18	–	–	–	**0.48**	**0.41**	**0.44**
PCCC	0.37	0.23	0.28	0.33	0.08	0.13	0.42	0.17	0.24	**0.51**	**0.27**	**0.35**
Omron_Fins	0.67	0.44	0.53	0.60	0.19	0.29	0.67	0.44	0.53	**0.91**	**0.57**	**0.70**
Delta_P	0.60	0.35	0.44	0.50	0.24	0.32	0.60	0.35	0.44	**0.75**	**0.71**	**0.73**
Hollysys_P	0.46	0.46	0.46	0.25	0.15	0.18	0.71	0.71	0.71	**0.80**	**0.86**	**0.83**
Average Accuracy	0.51	0.31	0.38	0.41	0.17	0.24	0.61	0.45	0.52	**0.70**	**0.58**	**0.63**

The notation of "–" denotes that the reverse engineering is time out and infers nothing.

TABLE IV
EVALUATING FIELD SEMANTICS INFERENCE WITH DIFFERENT REVERSE-ENGINEERING SCHEMES.

Protocol	Netzob			FieldHunter			IPART			REInPro		
	Corr.	Perf.	F1_Score	Corr.	Perf.	F1_Score	Corr.	Perf.	F1_Score	Corr.	Perf.	F1_Score
Modbus	1.00	0.13	0.22	0.50	0.38	0.43	0.75	0.38	0.50	1.00	0.50	0.67
Dnp3	1.00	0.08	0.14	1.00	0.08	0.15	1.00	0.23	0.38	1.00	0.54	0.70
S7Comm	1.00	0.09	0.16	0.45	0.20	0.28	0.56	0.20	0.29	1.00	**0.61**	**0.76**
S7Comm_Plus	**1.00**	0.06	0.11	0.80	0.12	0.21	0.10	0.10	0.10	0.86	**0.17**	**0.29**
PCCC	**1.00**	0.02	0.03	0.43	0.05	0.09	0.22	0.03	0.05	0.94	**0.12**	**0.22**
Omron_Fins	1.00	0.07	0.13	1.00	0.21	0.35	1.00	0.11	0.19	1.00	**0.32**	**0.49**
Delta_P	**1.00**	0.18	0.31	0.40	0.18	0.25	0.63	0.27	0.38	**1.00**	**0.46**	**0.63**
Hollysys_P	**1.00**	0.06	0.11	0.50	0.18	0.26	0.50	0.12	0.19	0.88	**0.41**	**0.56**
Average Accuracy	**1.00**	0.08	0.15	0.64	0.18	0.25	0.60	0.18	0.26	0.96	**0.39**	**0.54**

semantics to all bytes detailed in the protocol specification. In addition, we calculate the F1_score to balance the correctness and perfection of the above calculations:

$$\text{F1\_Score} = 2 * \frac{\text{correctness} * \text{perfection}}{\text{correctness} + \text{perfection}} \tag{7}$$

B. Results of Reverse Engineering

As Table III and Table IV show, REInPro reverse engineers the formats and semantics of 8 industrial protocols with an average correctness/ of 0.70/0.58 and 0.96/0.39, respectively. Note that the inaccuracies of REInPro mainly happen on the fields with static and encrypted values. Specifically, the static fields among different message sequences exhibit uninformative behaviors, and the encrypted messages will also obfuscate the network behaviors using chaotic encoding. For example, the S7Comm_Plus protocol contains multiple encrypted fields, which result in a high chaotic data streams and further reduce REInPro's efficacy. Moreover, in the PCCC protocol, numerous fields remain static, providing no valuable information for inference.

Performance on Format Inference. We compare REInPro with three state-of-the-art tools: Netzob [17], FieldHunter [25], and NetPlier [28], as shown in Table III. REInPro outperforms the other tools in most industrial protocols. FieldHunter, which uses n-grams to generate fields, has inferior performance due to its tendency to generate longer fields, making it less effective for industrial control protocols with finer encoding granularity. Sequence alignment's exponential complexity leads to high computational and time demands as the number of traces increases. This results in NetPlier's timeout for S7comm_Plus. The format inference results for S7Comm_Plus and PCCC protocols are low across all techniques due to the protocol complexity and encrypted data. Our evaluation reveals that REInPro significantly improves accuracy, particularly in complex protocols such as S7Comm and PCCC. In summary, REInPro achieves an average correctness of 0.70 and perfection of 0.58, outperforming Netzob (0.51, 0.31), FieldHunter (0.41, 0.17), and NetPlier (0.61, 0.45).

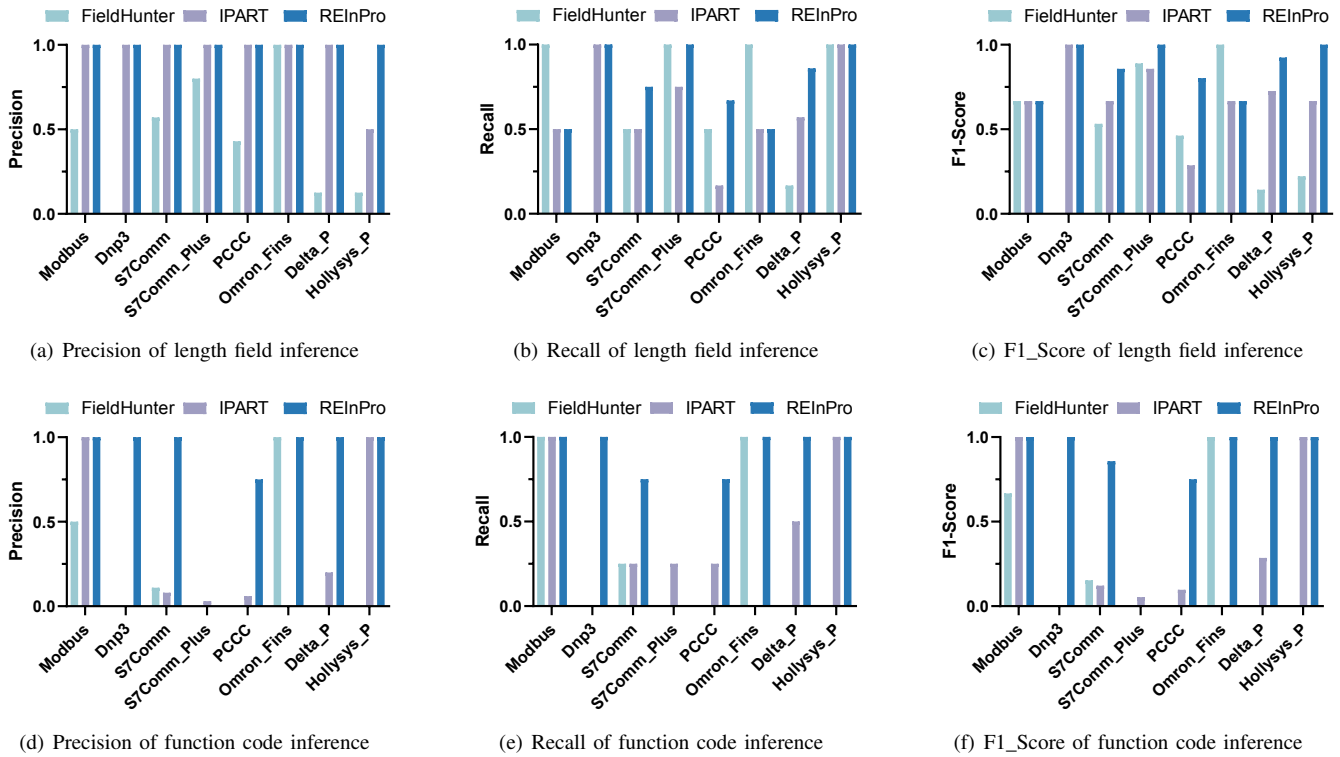
Performance on Semantics Inference. Netzob primarily focuses on inferring the semantics of length fields for industrial protocols, while NetPlier emphasizes keyword identification. In addition to Netzob and FieldHunter, we further compare our approach with the semantics inference method of IPART [27], a protocol reverse engineering tool designed for industrial protocols.

Table IV presents the semantics inference results of the four tools, revealing that REInPro acquires richer semantics for all eight protocols. Netzob achieves 100% correctness, as it only infers the length field representing the whole message's length. It is important to note that the semantics inference perfection is limited by the traffic-based approach's inability to capture the true semantics of static or encrypted fields. As a result, inferred static fields are not counted as correctly identified fields, leading to lower perfection. This limitation is especially apparent in protocols such as PCCC and S7comm_Plus, which contain multiple static and encrypted fields.

We have compared REInPro with the classic methods FieldHunter and IPART in inferring the semantics of function codes

TABLE V
THE ACCURACY OF MESSAGE FORMAT AND FIELD SEMANTICS INFERENCE WITH/WITHOUT USING CONTROL FIELD.

Protocol	Format inference						Semantics inference					
	Without control field			With control field			Without control field			With control field		
	Corr.	Perf.	F1_Score	Corr.	Perf.	F1_Score	Corr.	Perf.	F1_Score	Corr.	Perf.	F1_Score
Modbus	0.67	0.50	0.57	0.67	0.50	0.57	1.00	0.50	0.67	1.00	0.50	0.67
Dnp3	0.73	0.63	0.68	0.73	0.67	0.70	1.00	0.54	0.70	1.00	0.54	0.70
S7Comm	0.64	0.36	0.46	**0.74**	**0.62**	**0.67**	1.00	0.28	0.44	1.00	**0.61**	**0.76**
S7Comm_Plus	0.38	0.07	0.12	**0.48**	**0.41**	**0.44**	0.50	0.09	0.15	**0.86**	**0.17**	**0.29**
PCCC	0.43	0.17	0.24	**0.51**	**0.27**	**0.35**	0.50	0.05	0.09	**0.94**	**0.12**	**0.22**
Omron_Fins	0.91	0.57	0.70	0.91	0.57	0.70	1.00	0.32	0.49	1.00	0.32	0.49
Delta_P	0.64	0.59	0.61	**0.75**	**0.71**	**0.73**	0.60	0.27	0.38	**1.00**	**0.46**	**0.63**
Hollysys_P	0.64	0.54	0.59	**0.80**	**0.86**	**0.83**	0.50	0.41	0.45	**0.88**	0.41	**0.56**
Average Accuracy	0.64	0.42	0.51	**0.70**	**0.58**	**0.63**	0.76	0.31	0.42	**0.96**	**0.39**	**0.54**



(a) Precision of length field inference (b) Recall of length field inference (c) F1_Score of length field inference

(d) Precision of function code inference (e) Recall of function code inference (f) F1_Score of function code inference

Fig. 8. The inference quality of length field and function code.

and length fields. As shown in Fig. 8, REInPro outperforms FieldHunter and IPART in length field inference, except for Omron_Fins, where FieldHunter achieves better performance. This is because Omron_Fins encodes length fields in four bytes, which matches FieldHunter's n-gram generation. For function code inference, REInPro shows significant improvement in all seven protocols. However, all methods exhibit poor results in inferring the S7Comm_Plus protocol due to its complex format and encrypted fields. On average, REInPro achieves a perfection of 0.39 for semantics inference, outperforms Netzob (0.08), FieldHunter (0.18), and IPART (0.18).

C. Effectiveness of Control Field in REInPro

Out of the 8 evaluated protocols, 6 of them contain control fields, all of which are identified with 100% accuracy by REInPro. We compare the same reverse engineering strategy applied to message sequences with and without using control field identification. As presented in Table V, both format and semantics inference exhibit improved performance after clustering. For the three protocols — Modbus, Dnp3, and Omron_Fins — the results are identical before and after clustering. This can be explained by the fact that Modbus and Omron_Fins do not contain a control field, while the protocol format header for Dnp3 remains the same irrespective of the control field's value. Consequently, REInPro does not enhance the performance of these three protocols.

D. Application of REInPro: Fuzzing

To demonstrate the practical application of our reverse-engineering results, we integrate our approach with Boofuzz,

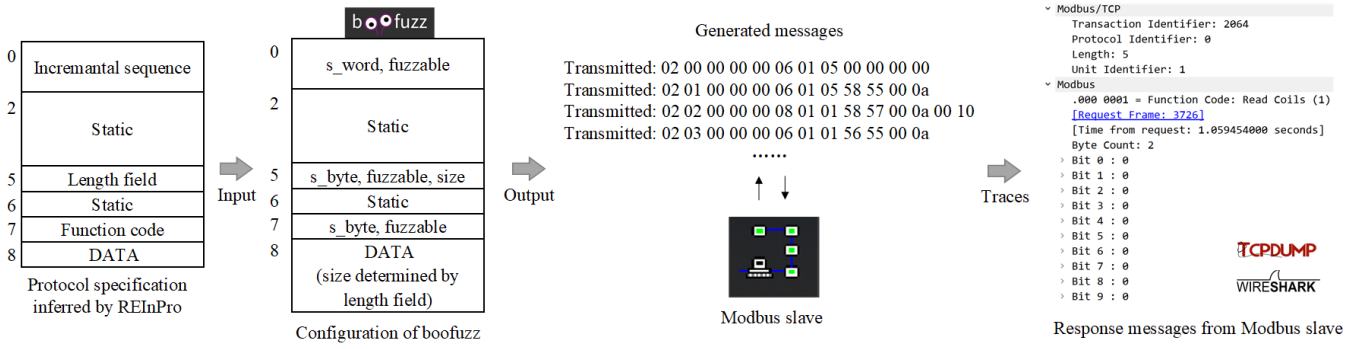


Fig. 9. Fuzzing the protocol of Modbus based on REInPro.

an open-source black-box fuzzer [10]. Take the Modbus protocol (Fig. 9) as an example, we have corroborated the effectiveness of REInPro's reverse-engineering results in fuzzing protocol. Specifically, the mutated messages are generated according to the inferred specifications, including the field size (e.g., n bytes), field semantics (e.g., length field, function code) and their interdependence. The format and semantics inferred by REInPro help Boofuzz to test validity of each field. For example, during PLCs' interaction, if a message does not comply with the protocol's length certification, the PLC may not respond correctly. REInPro's ability to infer multiple length fields enables the imposition of length constraints on fuzzing, increasing the message acceptance rate of fuzzing tests. By analyzing interaction traces between Boofuzz and the Modbus slave, it shows that all sent messages were identified as Modbus protocol by Wireshark, with some response messages corresponding to the correct command request. Our results indicate that REInPro's reverse engineering outcomes can be leveraged to generate messages that communicate effectively with PLC slaves.

V. Discussion

Static Fields and Encrypted Messages. The unchanging of static fields renders them uninformative in reverse engineering protocols, as they do not contribute to the variability and communication patterns. On the other hand, the chaotic bit encoding in encrypted messages causes the bit variation to behave randomly and similarly, making it uninformative as well. However, the encrypted messages can be identified and located by measuring their entropy [36].

Finite State Machines Inference. Finite state machines are defined by their states, an initial state, and the conditions or inputs prompting state transitions, usually driven by the content and type of message sequences. Successful inference of finite state machines requires accurate abstraction of message sequences. REInPro proves effective in this regard as it can efficiently abstract messages by accurately inferring control fields and function codes, which are key determinants of functionality.

VI. Related Work

Reverse-engineering the communication protocol plays a vital role in protecting control systems [37], [38]. Several automated methods have emerged to identify protocol formats and infer their underlying semantics. The Protocol Informatics (PI) [39] is the first to deploy sequence alignment algorithms for protocol reverse engineering, forming the basis for subsequent techniques. Discoverer [19] improves the robustness of sequence alignment by segregating messages into binary and text tokens. Netzob [17] infers the message formats by integrating the alignment algorithm with *a prior* knowledge of field semantics. By applying the multiple sequence alignment, NetPlier [28] identifies the message formats based on the protocol keywords captured using a probabilistic model. A novel dual-track framework that co-optimizes the keyword extraction and message clustering is proposed in [40]. The advanced machine learning methods are also applied to identify the protocol format [41]–[44]. In parallel with the alignment-based methods, Zhang et al. leverage the occurrence frequency of bytes to infer the protocol format [26]. Along with the inference of protocol format, methods that reverse-engineer the corresponding semantics have also been proposed. FieldHunter [25] and IPART [27] allocate semantics to the identified fields by utilizing the specific data features and a series of correlation metrics. BinaryInferno [45] deploys a set of semantics identifiers to cross-check the identified field semantics. Note that most above methods necessitate inputting a similar type of message. REInPro reverse-engineers the industrial protocols by utilizing the control fields to automatically determine the message type.

VII. Conclusion

In this work, we presented REInPro, a method to reverse-engineer industrial protocols based on control fields. REInPro identifies the control field and clusters the network traffic using a probabilistic model. The evaluation using 8 industrial protocols from 6 different PLCs has corroborated REInPro to identify the format of industrial protocols with an average correctness/perfection of 0.70/0.58, outperforming Netzob (0.51/0.31), FieldHunter (0.41/0.17), and NetPlier (0.61/0.45).

REFERENCES

[1] Y. Huang, H. Shu, F. Kang, and Y. Guang, "Protocol reverse-engineering methods and tools: a survey," *Comput. Commun.*, vol. 182, pp. 238–254, 2022.

[2] H. Pearce, S. Pinisetty, P. S. Roop *et al.*, "Smart I/O Modules for Mitigating Cyber-Physical Attacks on Industrial Control Systems," *IEEE Trans. Industr. Inform.*, vol. 16, no. 7, pp. 4659–4669, 2019.

[3] Z. Yang, L. He, H. Yu, C. Zhao, P. Cheng, and J. Chen, "Detecting PLC intrusions using control invariants," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9934–9947, 2022.

[4] H. Pu, L. He, C. Zhao, D. K. Yau, P. Cheng, and J. Chen, "Detecting replay attacks against industrial robots via power fingerprinting," in *Proc. 18th ACM Conf. Embed. Netw. Sens. Syst.*, 2020.

[5] D. Tychalas, H. Benkraouda, and M. Maniatakos, "ICSFuzz: Manipulating I/Os and repurposing binary code to enable instrumented fuzzing in ICS control applications," in *USENIX Secur. Symp. (USENIX Security)*, 2021.

[6] Y. Sun, S. Lv, J. You, Y. Sun, X. Chen, Y. Zheng, and L. Sun, "IPSpex: Enabling Efficient Fuzzing via Specification Extraction on ICS Protocol," in *Proc. Appl. Cryptogr. Netw. Secur.*, 2022.

[7] J. Antunes, N. Neves, M. Correia, P. Verissimo, and R. Neves, "Vulnerability discovery with attack injection," *IEEE Trans. Softw.*, vol. 36, no. 3, pp. 357–370, 2010.

[8] Z. Yang, L. He, H. Yu, C. Zhao, P. Cheng, and J. Chen, "Reverse engineering physical semantics of plc program variables using control invariants," in *Proc. 20th ACM Conf. Embed. Netw. Sens. Syst.*, 2022.

[9] Y. Chen, D. Mu, J. Xu, Z. Sun, W. Shen, X. Xing, L. Lu, and B. Mao, "Ptrix: Efficient hardware-assisted fuzzing for cots binary," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2019.

[10] Pereyda, Joshua, "Boofuzz: Network Protocol Fuzzing for Humans," https://github.com/jtpereyda/boofuzz, 2020, [Online; Accessed July 2023].

[11] H. Pu, L. He, P. Cheng, M. Sun, and J. Chen, "Security of industrial robots: Vulnerabilities, attacks, and mitigations," *IEEE Netw.*, vol. 37, no. 1, pp. 111–117, 2023.

[12] E. López-Morales, C. Rubio-Medrano, A. Doupé, Y. Shoshitaishvili, R. Wang, T. Bao, and G.-J. Ahn, "HoneyPLC: A Next-Generation Honeypot for Industrial Control Systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020.

[13] J. Caballero, H. Yin, Z. Liang, and D. Song, "Polyglot: Automatic extraction of protocol message format using dynamic binary analysis," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007.

[14] Z. Lin, X. Jiang, D. Xu, and X. Zhang, "Automatic protocol format reverse engineering through context-aware monitored execution." in *Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2008.

[15] R. Ma, H. Zheng, J. Wang, M. Wang, Q. Wei, and Q. Wang, "Automatic protocol reverse engineering for industrial control systems with dynamic taint analysis," *Front. Inform. Technol. Electron. Eng.*, vol. 23, no. 3, pp. 351–360, 2022.

[16] Siemens, "SIMATIC NET Industrial Ethernet switches SCALANCE XB-200/XC-200/ XF-200BA/XP-200/XR-300WG Web Based Management Configuration Manual," 2019, [Online]. Available: https://docs.rs-online.com/efa0/A700000009217752.pdf.

[17] G. Bossert, F. Guihéry, and G. Hiet, "Towards automated protocol reverse engineering using semantic information," in *Proc. 9th ACM Symp. Inf. Comput. Commun. Secur.*, 2014.

[18] Y. Wang, X. Yun, M. Z. Shafiq, L. Wang, A. X. Liu, Z. Zhang, D. Yao, Y. Zhang, and L. Guo, "A semantics aware approach to automated reverse engineering unknown protocols," in *IEEE Int. Conf. Netw. Protoc. (ICNP)*, 2012.

[19] W. Cui, J. Kannan, and H. J. Wang, "Discoverer: Automatic Protocol Reverse Engineering from Network Traces." in *USENIX Secur. Symp. (USENIX Security)*, 2007.

[20] D. F. Feng and R. F. Doolittle, "Progressive sequence alignment as a prerequisitetto correct phylogenetic trees," *J. Mol. Evol.*, vol. 25, pp. 351–360, 1987.

[21] O. Esoul and N. Walkinshaw, "Using segment-based alignment to extract packet structures from network traces," in *IEEE Int. Conf. Softw. Qual. Reliab. Secur. (QRS)*, 2017.

[22] S. Kleber, R. W. van der Heijden, and F. Kargl, "Message type identification of binary network protocols using continuous segment similarity," in *Proc. Conf. Comput. Commun. (INFOCOM)*, 2020.

[23] Y. Ji, T. Huang, C. Ma, C. Hu, Z. Wang, A. Fu *et al.*, "IMCSA: Providing Better Sequence Alignment Space for Industrial Control Protocol Reverse Engineering," *Secur. Commun. Netw.*, vol. 2022, 2022.

[24] Q. Wang, Z. Sun, Z. Wang, S. Ye, Z. Su, H. Chen, and C. Hu, "A practical format and semantic reverse analysis approach for industrial control protocols," *Secur. Commun. Netw.*, vol. 2021, pp. 1–11, 2021.

[25] I. Bermudez, A. Tongaonkar, M. Iliofotou, M. Mellia, and M. M. Munafo, "Automatic protocol field inference for deeper protocol understanding," in *Int. Fed. Inf. Process. Netw. (IFIP)*, 2015.

[26] Z. Zhang, Z. Zhang, P. P. Lee, Y. Liu, and G. Xie, "ProWord: An unsupervised approach to protocol feature word extraction," in *Proc. Conf. Comput. Commun. (INFOCOM)*, 2014.

[27] X. Wang, K. Lv, and B. Li, "IPART: an automatic protocol reverse engineering tool based on global voting expert for industrial protocols," *Int. J. Parallel. Emergent. Distrib. Syst.*, vol. 35, no. 3, pp. 376–395, 2020.

[28] Y. Ye, Z. Zhang, F. Wang, X. Zhang, and D. Xu, "NetPlier: Probabilistic Network Protocol Reverse Engineering from Message Traces." in *Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2021.

[29] J. Beale, A. Orebaugh, and G. Ramirez, *Wireshark & Ethereal network protocol analyzer toolkit*, 2006.

[30] S.-S. Choi, S.-H. Cha, and C. C. Tappert, "A survey of binary similarity and distance measures," *J. Syst. Cybern. Inform.*, vol. 8, no. 1, pp. 43–48, 2010.

[31] O. Liu, B. Zheng, W. Sun, F. Luo, Z. Hong, X. Wang, and B. Li, "A data-driven approach for reverse engineering electric power protocols," *J. Signal Process. Syst.*, vol. 93, pp. 769–777, 2021.

[32] G. Ládi, L. Buttyán, and T. Holczer, "Message format and field semantics inference for binary protocols using recorded network traffic," in *26th Int. Conf. Softw. Telecommun. Comput. Netw. (SoftCOM)*, 2018.

[33] Modbus Organization, "Modbus Application Protocol Specification, v1.1b3," http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf, 2012, [Online; Accessed July 2023].

[34] "A DNP3 Protocol Primer," https://www.dnp.org/Portals/0/AboutUs/DNP3%20Primer%20Rev%20A.pdf, 2005.

[35] Anonymous, "Supplemental Material of Submission #103," https://drive.google.com/file/d/1yemJDKtTC63VtkgGyZ3ExzrmmKnDNtTT/view?usp=drive_link, 2023, [Online; Accessed July 2023].

[36] M. S. I. Mamun, A. A. Ghorbani, and N. Stakhanova, "An entropy based encrypted traffic classifier," in *17th Int. Conf. Inf. Commun. Secur.*, 2016.

[37] Z. Yang, L. He, P. Cheng, J. Chen, D. K. Yau, and L. Du, "PLC-Sleuth: Detecting and Localizing PLC Intrusions Using Control Invariants," in *Proc. 23rd Int. Symp. Res. Attacks, Intrus. Defenses. (RAID)*, 2020.

[38] J. Meng, Z. Yang, Z. Zhang, Y. Geng, R. Deng, P. Cheng, J. Chen, and J. Zhou, "Sepanner: Analyzing semantics of controller variables in industrial control systems based on network traffic," in *Proc. 39th Annual Comput. Secur. Appl. Conf.*, 2023.

[39] M. A. Beddoe, "Network protocol analysis using bioinformatics algorithms," *Toorcon*, vol. 26, no. 6, pp. 1095–1098, 2004.

[40] W. Zhang, X. Meng, and Y. Zhang, "Dual-track protocol reverse analysis based on share learning," in *Proc. Conf. Comput. Commun. (INFOCOM)*, 2022.

[41] H. Zhao, Z. Li, H. Wei, J. Shi, and Y. Huang, "SeqFuzzer: An industrial protocol fuzzing framework from a deep learning perspective," in *12th IEEE Conf. Softw. Test. Valid. Verif. (ICST)*, 2019.

[42] R. Zhao and Z. Liu, "Analysis of private industrial control protocol format based on LSTM-FCN model," in *Proc. Int. Conf. Aviation Saf. Inf. Technol. (ICASIT)*, 2020.

[43] C. Yang, C. Fu, Y. Qian, Y. Hong, G. Feng, and L. Han, "Deep learning-based reverse method of binary protocol," in *Int. Conf. Secur. Priv. Digit. Econ. (SPDE)*, 2020.

[44] S. Zhao, J. Wang, S. Yang, Y. Zeng, Z. Zhao, H. Zhu, and L. Sun, "ProsegDL: Binary Protocol Format Extraction by Deep Learning-based Field Boundary Identification," in *IEEE Int. Conf. Netw. Protoc. (ICNP)*, 2022.

[45] J. Chandler, A. Wick, and K. Fisher, "BinaryInferno: A Semantic-Driven Approach to Field Inference for Binary Message Formats." in *Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2023.