Variable-length Field Extraction for Unknown Binary Network Protocols

Xiuwen Sun, Huiying Li, Yu Chen, Jie Cui, Hong Zhong
School of Computer Science and Technology, Anhui University
Email: mr.xiuwen@gmail.com

2024 IEEE 49th Conference on Local Computer Networks (LCN) | 979-8-3503-8800-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/LCN60385.2024.10639751

Abstract*—Protocol reverse engineering can infer the specification or behaviour of unknown network protocols, which is essential in analyzing and evaluating network functionality and performance. There are variable-length fields in many network protocols, and the field boundaries significantly impact subsequent analysis as well as the inferred results. The existing works focus on extracting protocol keyword fields without considering whether the fields' length is variable. In this paper, we propose BERRY for extracting variable-length field of unknown binary network protocols from static traces. At first, BERRY clusters the same type of messages from the input trace and extracts their headers with the help of the information entropy. Then, it combines the feature of message length and location-aware association analysis with to locate candidate variable fields. Finally, it infers the variable-length field by the sequence alignment. We evaluate BERRY with BinaryInferno using six groups of real network protocol traces. BERRY exhibits high accuracy and reliability on the metrics of *precision*, *recall*, and *F1-score* while extracting variable-length fields. It also performs similar results to the Binaryinferno on extracting all the fields.

***Index Terms*—Protocol reverse engineering, Network field extraction, Association analysis, Clustering.**

I. Introduction

A network protocol defines the format and the order of messages exchanged among communicating entities [1]. A protocol specification guides the entities in understanding the transmitted data as well-defined information. It is essential to analyze the network's functionality or detect the transmitted traffic [2]. However, many network applications, especially the malware, would not disclose their protocol specifications to preserve privacy or carry out malicious attacks. To infer the keyword fields of protocol specifications by observing the network traces or analyzing the entity software, known as protocol reverse engineering (PRE) [3], has become an effective technique.

The PRE aims to acquire the details of the format, semantics, and behaviour of unknown protocols. Format inference is the foundation for the following semantic deduction and behaviour reconstruction [4], [5], and most existing works focus on inferring fixed-length fields [6]–[8]. Many binary protocols usually use variable-length field to represent the types of protocol header, the various lengths of header, payload or entire message length. Identifying their location and boundaries significantly impacts the subsequent analysis of semantics and behaviour.

This work is partially supported by the National Natural Science Foundation of China (62102001).

We observe a certain association between the variable-length field and the header length or some other fields of this protocol. For example, the *Options* in IPv4 is a variable-length field, and the *IHL* is the length of the IP header [9]. Whether the *Options* field exists would impact the value of *IHL*, *i.e.* the header length of IPv4. However, the association of many protocols are not always as obvious and simple as that of the IPv4 when the length of their field is not fixed. We need deep analysis to disclose it and can use the association to extract the variable-length field.

In this paper, we propose a variaBle-length fiEld extRaction method for unknown binaRY network protocols from static traffic, named *BERRY*. It takes network traces of a specific protocol as input and automatically outputs the protocol's variable-length field. At first, BERRY clusters message types from the original network traces, classifies length features in each cluster and removes the futile payload according to the difference in information entropy between the message's header and payload. Then, it combines association analysis to locate candidate fields. Finally, it infers variable-length field through the sequence alignment algorithm.

To be specific, the contributions of this paper are:

(1) We propose BERRY, a general framework to extract the variable-length field of the protocol. The modules of BERRY are independent and can be replaced with similar functionality.

(2) BERRY uses k-means to cluster message types and information entropy to divide the header and payload of a message. Then, it uses location-aware association analysis to locate candidate fields. Finally, it uses sequence alignment to determine the inferred result.

(3) We evaluate BERRY with real-world datasets and compare it with the start-of-the-art approach, demonstrating that BERRY can deliver automatic, robust and high accuracy in extracting variable-length field across diverse protocols.

II. Related Work

Since we are going to extract the variable-length field from the static traces, we briefly survey some typical methods of inferring protocol format through these techniques.

A. Sequence-based Analysis

Sequence alignment compares byte values based on their fixed position in a message. The position-based correlation of bytes or n-grams reveals field boundaries by distinguishing between static and variable values.

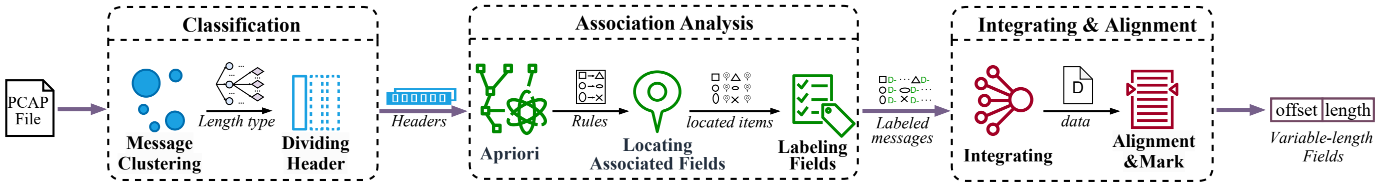


Fig. 1. The framework of BERRY.

PreDecoder [10] infers protocol format by grouping and clustering messages. Each message cluster uses the sequence alignment module to find invariant fields between messages. NETPLIER [7] uses multi-sequence alignment to align all messages. It determines which (aligned) field is the keyword by a probabilistic method and clusters them, restoring formatting directly. BinaryInferno(BI) [11] uses modular and extensible targeted detectors to identify atomic data types, find boundaries between adjacent fields using Shannon entropy, and discover variable-length sequences by searching for common serialization idioms. ROSE [12] use linear regression to extract the length field, which is a specific semantic field rather than all the fields extracted by the other methods.

B. *Graph-based Analysis*

Graph-based format reasoning methods include graph theory and images. This method constructs a graph model to observe the relationship between protocols and thus infer the format of the protocols.

Biprominer [13] uses a special state machine called prefix tree receptor, which can be used as a graph-based message format inference without alignment. The paper [14] uses a probabilistic graphical model to parse formats. It introduces vote-mechanism into the Hidden Markov Model using a protocol state transfer relationship. ProGraph [15] constructs a graphical model of the target protocol to describe the internal dependencies of packets and then classifies the traffic using the graphical model to deduce the protocol format. ProsegDL [8] uses image semantic segmentation technology, superimposes multiple binary messages with similar formats, takes the generated protocol image matrix is used as input to identify field boundaries.

C. *Algebraic-based Analysis*

In addition, the following algebraic format reasoning methods have also been proposed in the existing research. These methods are characterized by calculating the similarity or entropy of the protocol messages.

NEMETYL [6] uses three different segmentations [16] to handle message segments and combines it with clustering algorithms by calculating Canberra dissimilarity and Needleman Wunsch algorithm. The subsequent work [17] uses principle component analysis to improve the protocol clustering accuracy of NEMETYL. SPRA [18] delivers automatic and robust syntax inference across diverse protocols by building a parallel workflow to co-optimize both the generative model for keyword extraction and building a probability-based model for message clustering. FSIBP [19] designs multiple recurrent neural networks to analyze the fields and their corresponding contexts for field feature extraction.

The above methods provide sequence-based, graph-based, and algebraic techniques for inferring protocol formats. They extrapolate the protocol format from the perspective of fixed-length fields, ignoring variable-length field. Next, we will show how BERRY extracts variable-length field.

III. Approach

A. *Framework*

The network protocols are usually layered, which consists of a header and payload. The meaningful fields of protocols are located in the header and a pure, complete header is better for inferring the fields from static traces. Removing the payload is useful because the content of payloads is random and irregular compared to that of headers.

As mentioned, there is a certain relationship between the variable-length field and the header length or some other fields of this protocol. We can leverage the association analysis to disclose the relationship and use the sequence alignment to identify the possible boundaries of the variable-length field.

Hence, we propose BERRY to extract the variable-length field. It takes static network traces of a specific network protocol as input and outputs the boundaries of the inferred variable-length field. BERRY consists of three modules: *classification, association analysis* and *integrating&alignment*, which are shown in Fig. 1. The brief introductions are as follows:

Classification: The various types of messages have distinct structures and semantics, resulting in different correlations between the internal fields of each message type. *Message clustering* classifies different types of messages based on these characteristics and then reclassifies them based on their length feature. The key information of the protocol is usually in the header, so *dividing header* is also a necessary mission. This module outputs classification messages of different lengths after dividing the payload.

Association Analysis: BERRY uses the association analysis algorithm *Apriori* to find the association between fields in messages of the same length in clustering. To improve the accuracy of association analysis, it introduces positional attention in Apriori to locate association fields. Then, BERRY labels the possible fields based on the offset records of the bytes in each message. Finally, it outputs the message headers with associated field markers.

Integrating & Alignment: BERRY integrates labelled messages to reduce the computation in subsequent missions. Then, it aligns these integrated sequences and marks the target field.

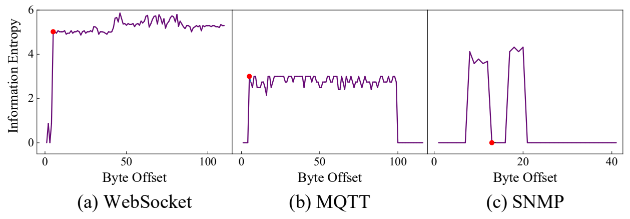


Fig. 2. Rapid change of information entropy of header and payload can be used to identify the boundaries of them.

Finally, it outputs the message header with variable-length field boundaries.

B. Classification

A protocol can produce various messages, and a message can be called an instance of the protocol. Each message can be treated as the connection of successive fields, and each field comprises a sequence of data segmented from the message. The protocol, message, and field are the logical concepts and they can be contained in the packets, namely the network traffic which is also called static trace.

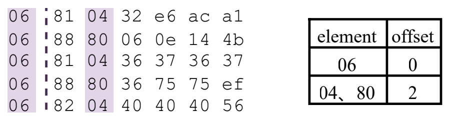
1) Message clustering: A protocol may contain plentiful types of messages resulting in different structures and semantics. So, clustering the messages by their characteristics or any other similarities would simplify the subsequent processing and improve the inferred performance. BERRY clusters the messages of a specific protocol into optimal different message types by the k-means algorithm with the Euclidean distance of the messages. It iteratively chooses the optimal number of clusters with the Silhouette Coefficient, an indicator that measures the quality of clustering results, combining the compactness of the sample within its cluster and the degree of separation from other clusters. Then, BERRY classifies the messages based on their length within the divided clusters at the end of this step.

2) Dividing payload: The header provides key information about a protocol specification, which is crucial for understanding the communication details. Therefore, the PRE typically only focuses on the header rather than the payload. BERRY uses information entropy to find the segmentation point between them and then cuts off the payload. It calculates the information entropy at each offset in the clustered messages using Eq. (1):

$$H(V_k) = -\sum_{v \in V_k} P(v)\log_2(P(v)) \tag{1}$$

V_k is the bag formed by the kth byte of each clustered message, and $P(v)$ is the probability of the v value in V_k, calculated as the number of times v appears in V_k divided by the size of V_k. For example, the information entropy in the second column of Fig. 3, $H([0x81, 0x88, 0x81, 0x88, 0x82]) = 1.52$.

There are significant differences in the information carried by the header and payload. The information entropy tends to stabilize after reaching a maximum point in the messages without an explicit delimiter between the header and the payload, as shown in Fig. 2(a) and Fig. 2(b). As to the messages



```
06 ¦81 04 32 e6 ac a1
06 ¦88 80 06 0e 14 4b
06 ¦81 04 36 37 36 37
06 ¦88 80 36 75 75 ef
06 ¦82 04 40 40 40 56
```

element	offset
06	0
04、80	2

Fig. 3. Example of inferring WebSocket messages. The left part represents five messages, the first column in shadow is the length of the message, and the rest is the bytes of each message. The right represents the infferred association rules.

with delimiters between them, the information entropy rapidly increases after experiencing some fluctuations and reaching the first minimum point, shown as the SNMP protocol in Fig. 2(c).

The information entropy of the payload may not always be large, as the payload data may carry a part of redundant information. Similarly, the information entropy in the message header may not be minimal, as certain fields like message ID and token may have approximately random values. This implies that the information entropy decreases rapidly, as illustrated in Fig. 2(b) and Fig. 2(c). The delimiter typically appears between the header and the payload. Despite appearing in the header, its value would be similar to others. Hence, the information entropy in the message header would not be extensive or fluctuate. Therefore, BERRY can divide the header and payload by using the maximum or minimum point.

C. Association Analysis

We notice some connections between the variable-length field and the length of the message header or other fields. BERRY can utilize association analysis with the length feature to uncover the relationship.

1) Apriori: The association analysis can be used to discover meaningful connections hidden in the messages. Frequent item sets and association rules can represent the discovered connections. The strength of association rules can be measured by their support s and confidence c levels. The support determines that a rule can be used for the frequency of a given dataset, that is, to effectively discover association rules. Confidence determines the frequency of Y in transactions containing X. The forms of support s and confidence c measures are shown in Eq. (2), where X and Y represent the candidate fields, i, j = $0, 1, 2,...,M_k$, the σ represents the support count, and M_k represents the total number of messages in this set.

$$\begin{aligned} s(X_i \to Y_i) &= \frac{\sigma(X_i \cup Y_i)}{M_k} \\ c(X_i \to Y_i) &= \frac{\sigma(X_i \cup Y_i)}{\sigma(X_j)} \end{aligned} \tag{2}$$

Discovering association rules involves identifying all rules with support and confidence equal to or greater than the specified thresholds, *minsup* and *minconf*, which selected in BERRY is 0.5 and 1 for the following reasons:

minsup: As shown in Fig. 3, the goal is to obtain the offsets of the frequent items that meet the conditions, namely the offset 0 and 2 corresponding to rules *06* $\to$ *04* and *06* $\to$ *80*, with a confidence level of 1 and support levels of 0.6 and 0.4, respectively. BERRY only need rule *06* $\to$ *04* as the offsets obtained by the two rules are the same. The sum of support at the same offset is 1, which verifies that our support threshold *minsup* setting of 0.5 is reasonable.

minconf: Confidence $c(X \to Y)$ represents the possibility of Y appearing in works containing X. In other words, it represents the degree of correlation between X and Y, with 1 indicating a certain correlation. Another explanation is that if this rule's improvement degree $lift$ is greater than 1, it indicates a convincing rule. The form of improvement degree $lift$ is shown in Eq. (3):

$$lift(X_i \to Y_i) = \frac{c(X_i \to Y_i)}{\sigma(Y_j)} \tag{3}$$

2) Locating associated fields: The classic association analysis only focuses on the relationship between the value of elements but ignores their positions. For example, a shopping transaction contains a unique identification and a corresponding set of products. The traditional algorithm would find some associations between the unordered products. However, the fields of a network protocol are strictly ordered even some fields are variable. As the example shown in Fig. 3, the bits of the first byte represent some flags followed by a length field. Even if the length field of the protocol is variable, the starting offset of the field is the second byte and remains unchanged. We introduce *location-aware* concept that keeps the fields' positions during the association analysis.

BERRY firstly locates elements in previously found rules. Then, it tracks and synchronizes the offsets of the association rules while searching for them. We calculate the *minsup* with the locational information. Without introducing it in association analysis, the associated rules may include some meaningless or incorrect rules. For example, there are two 06 and three 40 appearing in the second and fifth lines in Fig. 3, the classic association analysis may infer two independent elements $\{06, 40\}$ without any further information. The position of the elements is missing, which would lead to incorrect associated rules.

3) Labeling fields: BERRY records the number of occurrences and offset of elements in the rules derived from association analysis. In the example in Fig. 3, the relevant records for elements in rule *06* $\to$ *04* are $\{06, 6, 0, 0, 3, 0, 0, 0\}$ and $\{04, 3, 2, 2, 2\}$. In these records, the first element represents the target element, and the second is the number of occurrences, followed by each offset where the element appears.

The bytes in a message may not always be different, as in this rule, element 06 appears twice in the same message. However, based on association analysis and previous classification, the number of target offset positions we need is definitely greater than the others. So, BERRY only selects the offset with the highest number of occurrences after the second element in the record. Finally, BERRY uses'D-' to mark this offset, which is regarded as a marker for the associated field. The labelled results are used for subsequent alignment.

D. Integrating & Alignment

Berry integrates the labelled messages to avoid heavy computation requirements. Then, it aligns the integrated sequences

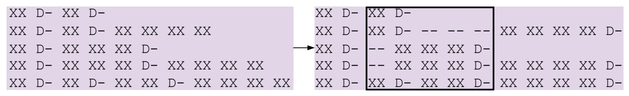

XX D- XX D-
XX D- XX D- XX XX XX XX
XX D- XX XX XX D-
XX D- XX XX XX D- XX XX XX XX
XX D- XX D- XX XX D- XX XX XX XX
XX D- XX D-
XX D- XX D- -- -- -- XX XX XX XX D-
XX D- -- XX XX XX D-
XX D- -- XX XX XX D- XX XX XX XX D-
XX D- XX D- XX XX D- XX XX XX XX D-

Fig. 4. Example of integrating & alignment. The left represents the integrated headers and the right represents the padding aligned according to the delimiter.

to get the boundaries of the target variable-length field.

1) Integrating: BERRY integrates clusters of different message types generated in the Classification module for subsequent alignment. Then, it removes the feature of message length and simplifies fields using 'XX' with consistent marker offsets, length ranges, and message lengths, as shown in the left of Fig. 4.

2) Alignment & mark: Padding fields with '--' to line them up at delimiter offsets, BERRY aligns the 'D-' in the sequence from the beginning towards the longest sequence. First, move all 'XX' between the current and previous 'D-'. If there is no 'D-' before the current 'D-', move the current 'D-' as a whole with its previous 'XX'. Then, add '--' to the gap after movement, as shown in the right of Fig. 4.

In sequences where a variable-length field is situated at the beginning or end, the difference in the position of the first 'D-' close to each '--' is minimal (0-2). BERRY identifies these two positions as the boundaries of the variable-length fields, with the bytes in between constituting the variable-length fields of the message. Otherwise, BERRY uses the position of the gap character '--' to determine the boundaries. Each couple of characters '--' appearing in the minimum and maximum positions in the sequences represents the boundaries of the variable-length field.

IV. Evaluation

A. Datasets and Metrics

1) Datasets: To exhibit the reliability and robustness of BERRY, we collect six application layer protocol traces to evaluate it, including WebSocket(WS), MQTT, CoAP [20], SNMP, STUN [21], and DNS. There are variable-length field in four binary protocols of the traces, and the others (STUN & DNS) do not exist. The number of messages of each protocol is shown in the second column of Table I. We use each type of message as the input and compare the results to the ground-truth about the protocol specification. The traces, source code, and brief instructions are released on Github .

2) Metrics: We evaluate the inferred field with the ground-truth to consider the *precision*, *recall*, and *F1-score*, which are classic metrics. What semantics they represent are a few differences in the scenarios of extracting variable-length fields or all fields. We will explain the details later. Given a message trace of a specific protocol, we define the sets shown in Fig. 5. Then, we can furthermore consider the definitions of the evaluation metrics, which are described and calculated as follows:

- *Precision*: The soundness of the inferences. It is calculated as the ratio of the number of matched boundaries by the total boundaries predicted to be true.

https://github.com/xiuwencs/berry

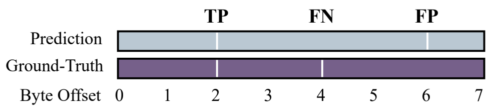


Fig. 5. Illustration of true positives (TP), false negatives (FN), and false positives (FP) between inferred field boundaries and the ground-truth.

- *Recall*: The coverage of the inferences. It is calculated as the ratio of the number of matched boundaries by the total number of boundaries of the true field in ground-truth.
- *F1-score*: The accuracy of the boundaries of extracted field by computing scores considering both the precision and the recall.

$$\begin{aligned} Precision &= \frac{TP}{TP+FP} \\ Recall &= \frac{TP}{TP+FN} \\ F1\text{-}score &= \frac{2 \times Precision \times Recall}{Precision + Recall} \end{aligned} \tag{4}$$

B. Results of Dividing Header from Payload

BERRY uses the information entropy to divide the header from the payload in the classification module and uses the divided headers to perform association analysis in the next step. We evaluate it first by collecting the number of correct or incorrect divided results and using Eq. (5) to calculate the accuracy of dividing, namely the proportion of correctly divided results, which are defined as:

- True: The inferred offset is equal to the ground-truth.
- False: The inferred offset is not equal to the ground-truth.
- Accuracy: The soundness of the inferred division offset between the header and the payload.

$$Accuracy = \frac{True}{Ture + False} \tag{5}$$

The evaluation results are presented in Fig. 6. BERRY can correctly divide the headers from most messages of CoAP and SNMP protocols. Due to the randomness of the *MASK* field at the end of the header in the WebSocket protocol, there is no significant difference in its information entropy between the header and payload, resulting in an accuracy of only 61.9%. The MQTT protocol has diverse message types. There are two types whose information entropy is 0 in the evaluated dataset. It means that the header and payload of these messages are identical. Hence, it is hard to divide the header and payload of these messages. However, these divided results slightly influence the subsequent extraction of the variable-length field, as the divided headers only have two more bytes of the number '0' compared to the ground-truth.

C. Results of Extracting Fields and Comparision

1) Results of Extracting Variable-Length Field: The inferred results and the ground-truth of variable-length fields in the datasets are shown in Table I. GT represents the ground-truth. The symbols - and / represent that there is no existence

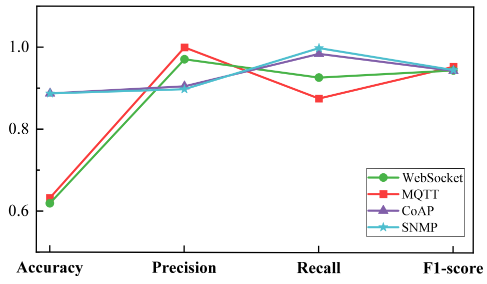


Fig. 6. Results of payload division and variable-length field extraction.

of ground-truth and inference results, respectively. BERRY correctly identifies the starting offsets of all the variable-length fields in the experiments.

TABLE I
SPECIFICATION FOR VARIABLE-LENGTH FIELD IN EXISTING DATASETS

Protocol	# Msgs.	Offset			Range of Length		
		GT	BERRY	BI	GT	BERRY	BI
WebSocket	11268	1	1	1	{1, 3}	{1-3, 5-8}	/
MQTT	4775	1	1	1	{1, 2}	{1, 2}	/
CoAP	3946	4	4	4	{1-8}	{1-8}	/
SNMP	4329	7	7	/	{4-8}	{4-7}	/
DNS	2000	-	/	12	-	/	/
STUN	2000	-	/	11	-	/	/

We first define the TP, FN and FP in the scenario of extracting variable-length fields. Then, we calculate the metrics according to Eq. (4). The definitions are as follows:

- TP(matched): The offset of an extracted *variable-length* field matches the ground-truth.
- FN(missed): BERRY does not locate the offset of a specific *variable-length* field.
- FP(unmatched): The offset of an extracted *variable-length* field does not match the ground-truth.

The average *precision*, *recall*, and *F1-score* of inferring the four protocols with variable-length field are 94.4%, 94.7%, and 94.7%, respectively as shown in Table II and Fig. 6. The STUN and DNS protocols do not exist any variable-length fields. BERRY does not produce any insertable '- -' separators during integration & alignment. It means that the fields are fixed-length, and BERRY is reliable in identifying variable-length fields.

As mentioned above, BinaryInferno infers binary message formats by atomic, field-boundary, and pattern-based detectors. It uses a pattern-based detector to discover variable-length sequences. However, it's limited to the patterns described in pre-defined grammar, which lacks generality in extracting variable-length field. Therefore, we compare BERRY with BI in extracting variable-length field and extracting all the fields.

2) Comparison of extracting variable-length field: Table II and Fig. 7 show the *precision*, *recall* and *F1-score* of extracting variable-length fields for each protocol. Obviously, BERRY is better than BI except for the result of WebSocket

TABLE II
COMPARISION WITH EXISTING APPROACH

Protocol	Evaluation results of inferred variable-length field						Evaluation results of inferred all fields					
	BERRY			BI			BERRY			BI		
	Prec.	Rec.	F1-score	Prec.	Rec.	F1-score	Prec.	Rec.	F1-score	Prec.	Rec.	F1-score
WebSocket	0.971	**0.927**	**0.945**	**1**	0.5	0.667	0.971	**0.92**	**0.945**	**1**	0.437	0.609
MQTT	**1**	**0.876**	**0.954**	**1**	0.5	0.667	**1**	**0.461**	**0.631**	**1**	0.398	0.569
CoAP	**0.905**	**0.985**	**0.944**	0.5	0.5	0.5	0.952	**0.993**	**0.972**	**1**	0.57	0.726
SNMP	**0.898**	**0.999**	**0.946**	0.333	0.429	0.4	**1**	0.25	0.4	**1**	**0.429**	**0.6**
DNS	-	-	-	-	-	-	0.646	**0.624**	**0.635**	**1**	0.333	0.5
STUN	-	-	-	-	-	-	**0.667**	**0.667**	**0.667**	0.5	**0.667**	0.571
Average	**0.944**	**0.947**	**0.947**	0.708	0.482	0.559	0.873	**0.652**	**0.708**	**0.917**	0.472	0.596

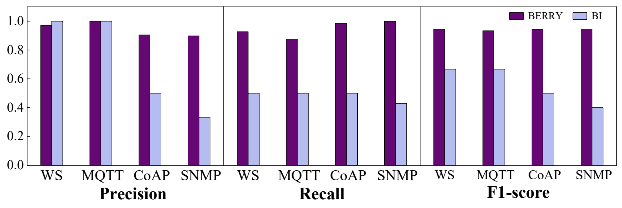


Fig. 7. Comparison of results for variable-length field with BinaryInferno.

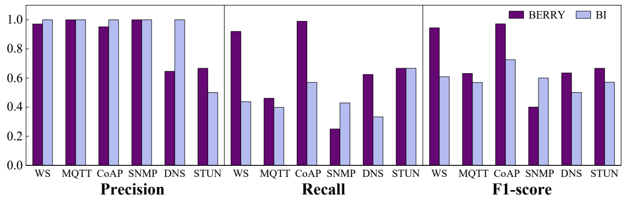


Fig. 8. Comparision of results for all fields with BinaryInferno.

protocol. The average of metrics of BI is about 60%, while that of BERRY is about 95%.

As shown in Table I and Table II, BI only correctly extracts the starting offset of the field from WebSocket and MQTT protocols. Since BI only extracts an offset in the experiments of these two protocols, *i.e.* the correct starting offset of a variable-length field, and it does not extract any other offsets, FP is 0. Hence, its precision is TP/(TP+0)=1. BI performs worse in the CoAP and SNMP protocols experiments. It only correctly extracts the starting offset of CoAP but identifies enormous incorrect ending offsets from both of them. As for the results of DNS and STUN without variable-length field, BI mistakenly believes that there are such fields. Moreover, BERRY can identify the boundaries and the ranges of the extracted variable-length fields. BI only infers the starting offset of the variable-length fields, but it does not provide the range of them.

From the analysis above, it can be found that BERRY not only accurately extracts the starting offset of the field but also shows high accuracy extraction results for the complete fields and ranges. BI does not extract the boundaries of the variable-length fields completely (both the starting and ending offsets) and the range of the fields.

3) Comparison of extracting all the fields: Before evaluating the performance of BERRY on extracting all the fields, we define the TP, FN, and FP in this scenario. Then, we calculate the metrics according to Eq. (4). The definitions are as follows:

- TP: The offset of any extracted fields matches their ground-truth.
- FN: BERRY does not locate the offset of specific fields.
- FP: The offset of the extracted fields does not match the ground-truth.

As shown in Table II and Fig. 8, Although the variable-length fields do not account for a large proportion of all the fields in the entire messages, BERRY outperforms BI in the metrics of *recall* and *F1-score*. It is not as good as BI on the precision metric. BERRY is a method for extracting variable-length fields; it is not good at extracting all the fields. As the start-of-the-art approach, BI can precisely identify the protocol fields. However, due to the influence of variable-length field, BI fails to extract such that fields completely and precisely. It also does not perform good results in the experiments of DNS, SNMP and STUN protocols.

V. Conclusion and Future Work

In this paper, we propose BERRY, the method of extracting variable-length field of binary network protocols from static traces. BERRY divides the header from the payload of the messages, analyzes the divided headers and locates the candidate fields by applying location-aware concept to the association analysis. Finally, it identifies the location and boundaries of the variable-length field using sequence alignment. The experiments shows that the average *precision*, *recall*, and *F1-score* can be 94.4%, 94.7% and 94.7%, respectively. BERRY also performs well in protocols without variable-length field. We also evaluate BERRY with the state-of-the-art approach BinaryInferno, showing that BERRY outperforms BinaryInferno in extracting variable-length field.

BERRY can identify the variable-length field boundaries well, it is little insufficient for extracting fixed-length fields compared to the state-of-the-art approach. We are going to explore a method of extracting fixed and variable fields of binary network protocol together in the future.

References

[1] J. F. Kurose and K. W. Ross, *Computer networking: A Top-Down Approach*, 8th ed. Pearson Education, Inc., 2020.

[2] A. Kondoro, I. B. Dhaou, H. Tenhunen, and N. Mvungi, "Real time performance analysis of secure iot protocols for microgrid communication," *Future Generation Computer Systems*, vol. 116, pp. 1–12, 2021.

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

[4] E. Hoque, O. Chowdhury, S. Y. Chau, C. Nita-Rotaru, and N. Li, "Analyzing operational behavior of stateful protocol implementations for detecting semantic bugs," in *IEEE/IFIP International Conference on Dependable Systems and Networks(DSN)*. IEEE, 2017, pp. 627–638.

[5] S. Kleber, L. Maile, and F. Kargl, "Survey of protocol reverse engineering algorithms: Decomposition of tools for static traffic analysis," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 526–561, 2019.

[6] S. Kleber, R. W. van der Heijden, and F. Kargl, "Message type identification of binary network protocols using continuous segment similarity," in *IEEE Conference on Computer Communications(INFOCOM)*. IEEE, 2020, pp. 2243–2252.

[7] Y. Ye, Z. Zhang, F. Wang, X. Zhang, and D. Xu, "Netplier: Probabilistic network protocol reverse engineering from message traces," in *Network and Distributed Systems Security Symposium(NDSS)*, 2021.

[8] 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 International Conference on Network Protocols(ICNP)*. IEEE, 2022, pp. 1–12.

[9] J. Postel, "Internet Protocol," RFC 791, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc791

[10] 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 International Conference on Network Protocols(ICNP)*. IEEE, 2012, pp. 1–10.

[11] C. Jared, W. Adam, and F. Kathleen, "Binaryinferno: A semantic-driven approach to field inference for binary message formats," in *Network and Distributed System Security Symposium(NDSS)*. ISOC, 2023, pp. 1–18.

[12] X. Sun, Z. Wu, J. Lin, P. Fu, J. Cui, and H. Zhong, "Extracting length field of unknown binary network protocol from static trace," in *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2023, pp. 462–469.

[13] Y. Wang, X. Li, J. Meng, Z. Zhao, and L. Guo, "Biprominer: Automatic mining of binary protocol features," in *International Conference on Parallel and Distributed Computing, Applications and Technologies*. IEEE, 2011.

[14] Y. He, Y. Wu, C. Zhang, J. Shen, K. Xiao, S. Keshav, and L. Sun, "A sparse protocol parsing method for iiot based on bpso-vote-hmm hybrid model," *IEEE/ACM Transactions on Networking*, vol. 31, pp. 485–496, 2023.

[15] H. Qun, L. Patrick P.C., and Z. Zhan, "Exploiting intra-packet dependency for fine-grained protocol format inference," in *IFIP Networking*. IEEE, 2015, pp. 1–9.

[16] S. Kleber, H. Kopp, and F. Kargl, "Nemesys: Network message syntax reverse engineering by analysis of the intrinsic structure of individual messages." in *USENIX Workshop on Offensive Technologies*, 2018.

[17] S. Kleber and F. Kargl, "Refining network message segmentation with principal component analysis," in *IEEE Conference on Communications and Network Security*. IEEE, 2022, pp. 281–289.

[18] W. Zhang, X. Meng, and Y. Zhang, "Dual-track protocol reverse analysis based on share learning," in *IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2022, pp. 51–60.

[19] M. Zhan, Y. Li, B. Li, J. Zhang, C. Li, and W. Wang, "Toward automated field semantics inference for binary protocol reverse engineering," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 764–776, 2024.

[20] Z. Shelby, K.Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7252

[21] M. Petit-Huguenin, G. Salgueiro, J. Rosenberg, D. Wing, R. Mahy, and P. Matthews, "Session Traversal Utilities for NAT (STUN)," RFC 8489, Feb. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8489