ProsegDL: Binary Protocol Format Extraction by Deep Learning-based Field Boundary Identification

Sen Zhao[†‡], Jinfa Wang[‡], Shouguo Yang[†‡], Yicheng Zeng[†‡], Zhihui Zhao[†‡], Hongsong Zhu[†‡*], Limin Sun[†‡]
[†]School of Cyber Security, University of Chinese Academy of Sciences, China
[‡]Beijing Key Laboratory of IOT Information Security Technology, Institute of Information Engineering, CAS, China
{zhaosen, wangjinfa, yangshouguo, zengyicheng, zhaozhihui0820, zhuhongsong, sunlimin}@iie.ac.cn

***Abstract*—Protocol reverse engineering can be applied to various security applications, including fuzzing, malware analysis, and intrusion detection. It aims to acquire an unknown protocol's format, semantic, and behavior specifications, where format extraction is the primary task. One subset of the mainstream research utilizes the network traffic for the reverse analysis. These approaches leverage various algorithms, such as multiple sequence alignment, frequent itemset mining, and information entropy to extract format information from messages. However, they are primarily intended to locate the keyword fields and have limitations in extracting contextual features or dealing with large data sets. This paper presents ProsegDL, a deep learning-based format extraction tool for binary protocol, with a specially designed method of generating training data sets. ProsegDL innovatively leverages image semantic segmentation and siamese network techniques, focusing on extracting the features of fields and identifying field boundaries for fixed format protocols. The tool is evaluated on six popular protocols. The results show that it has at most 13% higher precision, 23% higher recall than the comparison methods when inferring with a small data set, and at most 18% higher precision, 28% higher recall when inferring with a large number of messages.**

***Index Terms*—protocol reverse engineering, binary protocol reverse, image semantic segmentation, siamese network**

I. Introduction

Protocols are designed to regulate the communication between network nodes. With the development of 5G (5th Generation Mobile Networks) and IoT (Internet of Things), the types of network nodes have expanded beyond PCs, cell phones, and servers to include anything that can access the Internet. The number of newly proposed protocols is rising to meet various communication requirements. Particularly, they are becoming increasingly sophisticated to support various promising application scenarios such as Blockchain and IoV (Internet of Vehicles) [1]–[3]. Furthermore, malware also utilizes private protocols to hide its communication and activity traces. The botnet C&C server, for instance, relays commands to bot nodes via its private protocol [4]. ICS (Industrial Control System) protocol is another focus of security analysis in recent years due to its importance in the infrastructures [5]. However, many protocols are undocumented, making them difficult for security analysis, such as fuzzing [6]–[8], malware analysis [9] [10], and intrusion detection [11]. To this end, PRE (Protocol Reverse Engineering) technique is proposed to restore the unknown protocol specifications for further security research.

[*] Hongsong Zhu is the corresponding author.

PRE aims to acquire the syntax, semantic, and behavior information of an unknown protocol. Format extraction is an essential task for the following semantic deduction and behavior reconstruction [12]. The target protocols can be divided into two types, textual and binary protocols. Textual protocols are defined based on readable characters and generally consist of words and symbols, where the symbols can usually be used for field segmentation [13]. In contrast, binary protocols are composed of bytes or even bits, frequently used in ICS and IoT networks for efficient data transmission. Thus, binary messages are unreadable and bring challenges to PRE. Current PRE approaches mainly fall into two categories, execution trace and network trace. Execution trace-based methods [14]–[16] usually leverage dynamic taint analysis to extract format information from program binaries and are more accurate. Unfortunately, most private protocol programs are difficult to access [17], which makes these methods not practical. On the contrary, network trace-based methods are more applicable. Such methods require the network traffic between the nodes as analysis targets, which can be intercepted in various ways. This paper focuses on the format extraction of binary protocol based on the network trace-based technique.

To gain more insights, we stack multiple binary messages of similar format together and obtain inspiring observations (Section II-C) of binary protocols. This way of stacking allows us to judge the boundaries of the different fields very intuitively. Motivated by the observations, we propose a tool that primarily leverages the image semantic segmentation technique in deep learning [18] to segment the messages precisely and efficiently. Additionally, we embed the LSTM-FCN [19] model to the siamese network [20] for refining the segmentation results from the former step. We conduct the evaluations on NTP, DHCP, NBNS, SMB, S7Comm, and Modbus protocols. The experimental results show that our approach outperforms similar methods by large margins. Our contributions are summarized as follows:

- We discover the features of fields and their boundaries in fixed format binary protocols by stacking multiple traffic messages. These features are leveraged by image semantic segmentation and siamese network models to facilitate the format extraction. (Section III)
- We design a binary protocol format extraction tool,

ProsegDL, which stands for deep learning-based protocol field segmentation. Meanwhile, we proposed a method to generate data set for training, which only requires network traffic and little manual work. (Section IV)
- We train the models with the traffic of eleven protocols and test it with six other protocols. It outperforms the similar format extraction approaches, Netzob and NEMESYS (at most 18% higher precision and 28% higher recall), with acceptable time consumption. (Section V)

II. Motivation

Despite being proposed for over a decade, binary protocol format extraction is still considered unsolved due to its unique features. First, delimiter-based methods [13] [21] have poor performances because most binary protocols are of flat format and need no symbol to indicate the field boundaries [22]. Second, short field is a prominent characteristic of binary protocol, especially for ICS and IoT protocols. The length of control-related fields, such as function code, version, and message type, are typically only a few bytes or even bits. Several short fields usually appear in succession, making their boundaries obscurer for the deduction. Numerous works have been proposed in the past few years and the state-of-the-art binary protocol format extraction approaches generally take two technical routes: alignment-based and probability-based. Their ideas and limitations are as follows.

A. *Alignment-based Methods*

Originally derived from bioinformatics, MSA (Multiple sequence alignment) algorithms, such as NW (Needleman Wunsch) [23] and SW (Smith Waterman) [24], are the most frequently used methods for format extraction [25]–[28]. These algorithms compare several message sequences, find similar sub-sequences, and define them as keywords. The boundaries of these keywords are then marked as the field segmentation positions. Many researchers improved the MSA algorithms by modifying the scoring matrix [29]–[31] or optimizing the alignment rules [13] [32]. Compared with the original MSA algorithm, they can achieve better segmentation results. However, alignment-based methods are not resistant enough to the noises. The diversity of poorly clustered data degrades the alignment quality severely. Fields with a high degree of randomness, like encapsulation metadata, also affect the results. Meanwhile, though the MSA-based algorithms can produce relatively better results, they are inefficient for mass message calculation.

B. *Probability-based Methods*

These methods arose because of the observation that the frequencies of different values in the messages are not the same. If a candidate value frequently appears in protocol messages, it is more likely to be a protocol keyword field. Various algorithms, such as frequent itemset mining [33]–[35], information entropy [36] [37], and HMM [38] [39], are utilized for the format extraction. Meanwhile, position information [40] [41] and n-gram [42] [43] are used to introduce contextual information. Most works are capable of locating the frequent keyword fields in the messages precisely and efficiently and fit the requirement of downstream tasks perfectly. For example, fuzzing needs the inference of keyword fields as the injection points of the mutation values. However, most probability-based approaches do not aim at extracting the full format of the messages. They also require the adjustment of multiple parameters, which calls for specialist knowledge.

C. *Our Ideas*

A binary protocol message commonly consists of a header and a payload, where the payload comes after the header. During format extraction, we pay more attention to the header since it follows a certain specification and is commonly structured. It is worth noting that the header usually tends to be fixed in length. Based on such fact, we observe intuitive and enlightening findings by stacking multiple binary messages to form an image, as shown in Fig. 1. The yellow pixels represent the bits of value 1, and the green pixels in the background represent the bits of value 0. We summarize two discovered features inspired by the observations:

- Different types of fields show different textures in the image. For example, the red box is a sequential number field, and the blue box is a control field. They have distinctly different features. We describe this as the ***intra-field features***.
- Most adjacent fields tend to have different distribution characteristics, making their boundaries more evident than other positions. For example, the field distributions on the left and right sides of the black dot line are different, and the boundary is obvious. We describe this as the ***inter-field features***.

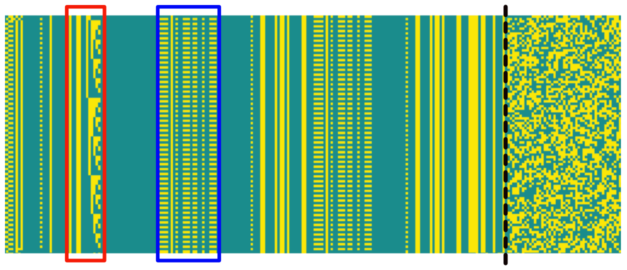


Fig. 1: Textures and boundaries of different fields in the image generated by stacking multiple binary messages.

Motivated by these two observations, format extraction can be done with the idea of extracting field features and identifying the field boundaries from protocol images.

To this end, we are able to handle the stacked messages with the image semantic segmentation technique for byte-level field segmentation. Image semantic segmentation can classify the parts belonging to the same object in an image to the same instance, which is capable of pixel-level labeling [44]. We utilize it to extract the features of different areas in the image and further label their boundaries. According to the second observation, another siamese network-based model is designed to strengthen the reliability of inference results. Siamese network [20] is a neural network architecture that uses the same weights while working in tandem on two different inputs to compute their similarity. We slice a few bytes of data from the left and right sides of the predicted segmentation position as the two inputs of the siamese network, then output the similarity between them. If they are of different types, the inferred boundary is usually correct; otherwise incorrect. The models will be detailed in Section III.

Meanwhile, supervised methods are also applicable to the task. On the one hand, though protocols are designed for different scenarios, their meta-fields are basically of several types, such as fixed value, incremental sequence, identifier, and data [45]. When the number of stacked messages is large enough, these fields have clear distribution and boundaries, which can be learned and used in the format extraction for unknown protocols. On the other hand, protocols are designed with inheritance, not only for public protocols (e.g., SSDP is proposed based on HTTP) but also for private protocols. For example, protocols used by the malware from the same family have similar formats [46] [47], and some private ICS protocols are also modified based on Modbus [48]. Once the knowledge of the known protocols is learned, models can use it to infer the format of similar potential protocols.

III. Format Extraction Models

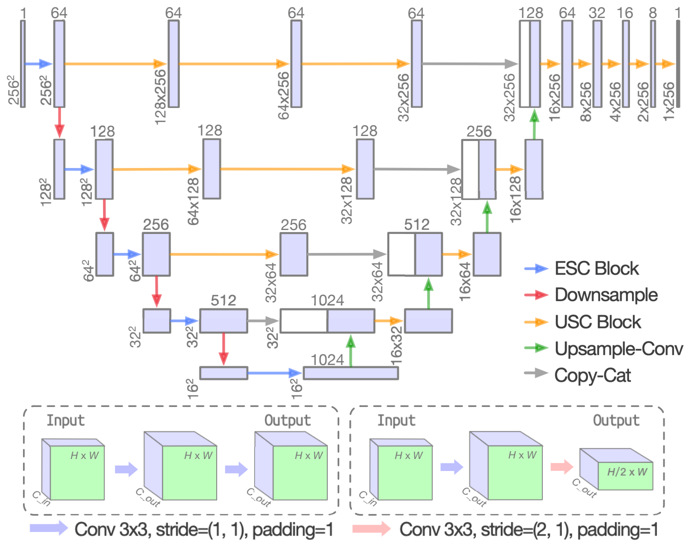
This section introduces how we leverage deep learning models to solve the protocol format segmentation problem. The first part is ProsegUNet (III-A), the image semantic segmentation based segmentation model. The second part is Proseg-Seiamese (III-B), which introduces the details of leveraging the siamese network to identify the boundaries. The training results of the two models are in the last part (III-C) .

A. *ProsegUNet*

We first propose the ProsegUNet based on the observation of inter-field features. We modified and improved the original structure of U-Net [49], a classical semantic segmentation model for medical grayscale images, to segment the protocol format. We define the protocol field segmentation as the model only needing to output the label sequence (1D vector) of each bit position in the binary protocol format. This is slightly different from the usual image semantic segmentation, where the model needs to output a segmentation map (2D matrix).

*1) **Inputs and Outputs**:* ProsegUNet takes the protocol image matrix generated from multiple messages as inputs, and outputs the field segmentation label vector. From the perspective of input and output shapes, letting $I \in R^{n\times1\times256\times256}$ denotes the input samples and $Y \in R^{n\times1\times1\times256}$ denotes the output results, where n denotes the batch size.

*2) **Model Details**:* As shown in Fig. 2, ProsegUNet basically follows the original U-Net, but with some special processing. It still consists of a convolution-downsampling path (left side) and an upsampling-convolution path (right side). However, instead of directly concatenating the corresponding



(a) Equal Strided Convolution (b) Unequal Strided Convolution

Fig. 2: The architecture of ProsegUNet. It takes grayscale image of size 256*256 as inputs, and uses USC blocks to gradually shrink the height of feature maps. The final output of the model is a vector of size 256, corresponding to the probabilities of segmentation position for each bit in the message format.

feature maps from the left side to the right side as in U-Net, we perform additional convolution operations to shrink the height of the feature maps, which will be detailed in the next paragraph. Moreover, we replace the max-pooling (2x2) with convolution (3x3 kernel, (2, 2) stride) in the downsampling blocks, making the model retain more features during processing [50].

Original U-Net convolution block consists of the repeated application of two 3x3 kernel-sized convolutions layers with the stride of 1 both in horizontal and vertical direction operation, illustrated as Fig. 2 (a) and named of ESC (Equal Strided Convolution) blocks. However, for binary protocol images in our work, we pay more attention to the data distribution characteristics between adjacent bits in the same messages rather than the same positional bits between different messages. So features extracted on vertical pixels of protocol images can be moderately discarded without introducing significant errors to the output results. Therefore, in the upsampling stage, we use USC (Unequal Strided Convolution) blocks to make the model gradually output the final desired vector-shaped segmentation sequence. The USC block consists of two convolution layers, as shown in Fig. 2 (b). The first convolution layer is set of 3x3 kernel, (1, 1) stride. The second convolution layer is of the same kernel size, but with (2, 1) stride. When the block works, it takes feature maps of shape $(Channel_{in}, Height, Width)$ as input and outputs feature maps of shape $(Channel_{out}, Height/2, Width)$. Meanwhile, to prevent overfitting, BN (Batch Normalization) layer is employed after the convolution layers and followed by a LeakyReLU (Leaky Rectified Linear Unit) activation layer.

*3) **Loss Function**:* We introduce the loss function that give extra weights to the boundaries of field segmentation positions, forcing the model to learn more boundary features. Initially, we squeeze the model's 3D outputs y_i to 1D and get $\hat{y}_i = Sigmoid(y_i)$, then the loss is calculated follow:

$$\mathcal{L} = \lambda \mathcal{L}_{FS} + (1-\lambda)\mathcal{L}_{NFS}$$

where $\mathcal{L}_{FS}$ denotes the loss of field segmentation positions, $\mathcal{L}_{NFS}$ denotes the loss of non-field segmentation positions and λ is the weight coefficient. Both $\mathcal{L}_{FS}$ and $\mathcal{L}_{NFS}$ are averaged over each batches' samples. To be more specific,

$$\mathcal{L}_{FS} = \sum_{i=1}^{n} t_i(t_i - \hat{y}_i)^2, \quad \mathcal{L}_{NFS} = \sum_{i=1}^{n} (1-t_i)(t_i - \hat{y}_i)^2$$

where t is the target ground-truth and n is the width of image (256 in our model). Since the ground-truth label only consists of 0 or 1, $\mathcal{L}_{FS}$ loss only has value when $t_i = 1$ and $\mathcal{L}_{NFS}$ loss only has value when $t_i = 0$. During the training phase, the loss function with boundary weights can make the model to learn more features of field boundaries and increase the convergence speed.

B. *ProsegSiamese*

Apart from segmenting fields by extracting their inter-field features, we proposed ProsegSiamese to judge the boundaries of fields with intra-field features. The main idea is based on our second observation described in Section II-C.

As shown in Fig. 3, the backbone of the ProsegSiamese is LSTM-FCN [19] network, which has been proved to have decent feature extraction capability for sequential data. We remove its original downstream classification layers to match the siamese network and add fully connected layers to output the result as the similarity. The two inputs are processed separately by the LSTM-FCN that are of shared parameters. The Euclidean distance of their outputs is calculated as the input of the subsequent FCN layers. Finally, we use the Sigmoid operation to map the final output to the value between $(0, 1)$. It is used as the similarity between the two sub-images extracted from the image, denoted as P_{sim}. High similarity usually means they are of the same data type and same field. Otherwise, their boundary is the field segmentation position. Therefore, $1 - P_{sim}$ can be regarded as the probability of the true field boundary.

The training and evaluation data sets of ProsegSiamese are generated from the same data sets of ProsegUNet. We extract 8-bits wide data slices (shape of $n \times 8$, n is the height of the images) from the protocol image on the left and right sides of each segmentation position (also used as the model ground-truth label Y). Then convert them into integer sequences like $[v_1, v_2, \cdots, v_n], v_i \in [0, 255]$, which are used as model inputs X_l and X_r, respectively. For non-segmentation positions, since their number far exceeds the number of segmentation positions, we need to balance the positive and negative samples in the data set. We perform the same data generation procedure but with probability P_{nsp} (set to 0.1 in our practice). The model is also relatively simple to be trained using BCE-loss without specially designed loss functions.

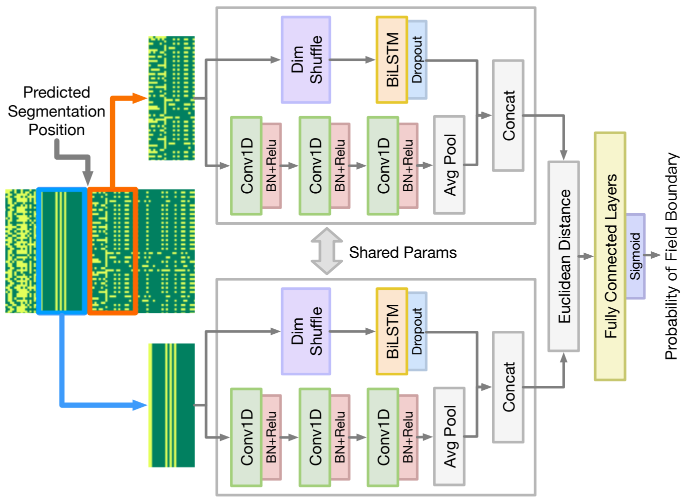


Fig. 3: The architecture of ProsegSiamese. LSTM-FCN is embedded as the shared backbone in the siamese network for the two input sequences. It consist of recurrent and convolution for the same sequence to generate the feature vector. Then the distance between two feature vectors of input sequences is calculated, and finally output their similarity.

C. *Model Training*

Both ProsegUNet and ProsegSiamese need training. We update all model parameters by backpropagation using RMSprop with an initial learning rate of 10^{-4}, decay weight of 10^{-7}, and momentum of 0.8. All experiments were conducted on a single GPU of Nvidia Tesla V100 16GiB under Ubuntu 16.04.6, and were implemented in PyTorch 1.10.0. We trained ProsegUNet with the batch size of 16, and trained ProsegSiamese with the batch size of 64. The training and evaluation were performed separately on the generated training and evaluation data sets in each epoch. Losses are shown in Fig. 4.

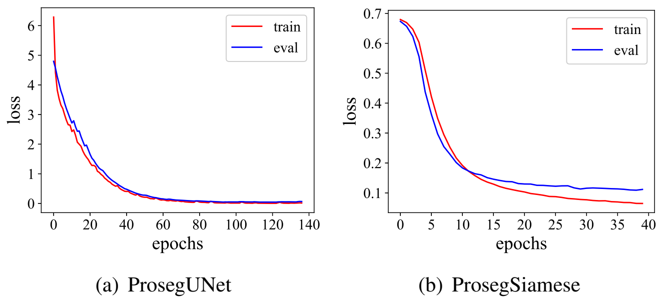


Fig. 4: The training and evaluation loss of the models. With the increase of epochs, the losses of the two models gradually decrease and finally converge after 80 epochs and 25 epochs, respectively. The loss curves of ProsegUNet is smoothed.

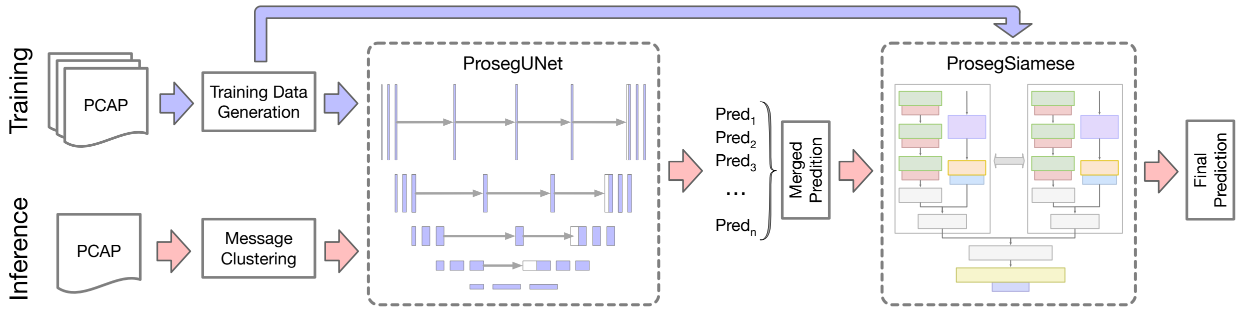


Fig. 5: The framework and workflows of ProsegDL. During training, it parses the collected PCAP files to generate the data sets and trains the ProsegUNet and ProsegSiamese models, respectively. During inference, we cluster packets messages to form the input images and then use well-trained models to do the prediction.

IV. System Implementation

Based on ProsegUNet and ProsegSiamese, we implement the ***ProsegDL***. In this section, we first analyze the characteristics of the two models and present the designing principles and the architecture of ProsegDL (IV-A). Then, we introduce the details of training data generation (IV-B).

*A. **ProsegDL Designs***

Both proposed ProsegUNet and ProsegSiamese have decent format extraction capabilities, but they have different mechanisms and effects. On the one hand, ProsegUNet is capable of extracting features of different texture parts in the images and then indicating their boundaries. However, due to the limitation of generated image size, only up to 256 network traffic messages can be analyzed at a time. Additional fusion step is required for multiple predictions when feeding a large quantity of data. On the other hand, we find that ProsegSiamese tends to over-segment the messages with many unnecessary predictions in practice, especially for long fields where multiple messages have common prefixes or paddings. These wrong predictions are difficult to delete, but we found its precision on true-positive predictions is good and usable. Therefore, we consider that ProsegUNet can extract features more comprehensively and extract protocol formats without excessive over-segmentation, making it a suitable module for the first-level inference. And ProsegSiamese focuses more on the local features in the images, making it a better choice for the second-level filtering and judging module.

Based on these characteristics, we propose a pipeline system, ProsegDL, as shown in Fig. 5. In the training phase, we use collected PCAPs to generate training data and trained the models. In the inference phase, it is noteworthy that the PCAPs are no longer of known protocols, which means they cannot be dissected directly for generating images. Therefore, we utilize Netplier [17] to cluster the messages accurately and generate the images following a similar process of training data generation. For the combination of the models, we designed it as a map-reduce-like procedure. ProsegUNet can infer the format of the protocol. However, it cannot infer complete and accurate enough results from one single image. If the input traffic data is sufficient, we can fuse multiple outputs into a more reliable result. The basic idea is shown in Fig. 6, which mainly consists of two stages: *merging* and *purging*.

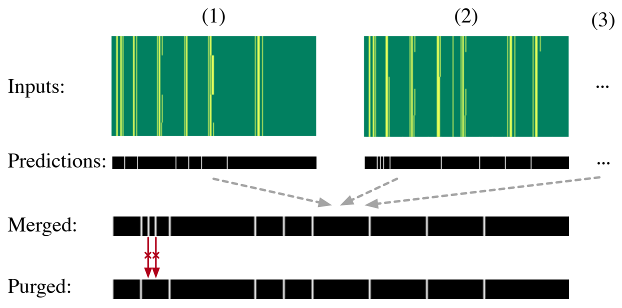


Fig. 6: The process of prediction fusion. Multiple predictions are inferred from several images of similar formats and merged into one prediction. Then each predicted segmentation position is judged to remove the erroneous positions.

*1) **Merging**:* Multiple predictions are merged into one prediction. First, for each bit Pos_i of every prediction, count the value of 1 (positive) as N_p and value of 0 (negative) as N_n. Then we calculate the positive ratio of every bit position $P_i = N_p/(N_p + N_n)$, where P_i can be regarded as the confidence of whether it is the correct segmentation position. When $P_i > P_{min}$, we denote the corresponding bit position $\hat{Pos_i}$ as 1 otherwise 0, where P_{min} is the minimum support threshold. Finally, reassemble every inferred $\hat{Pos_i}$ together to get the merged prediction of the target format. The threshold P_{min} needs to be adjusted according to the actual protocol traffic to make the predictions in line with the target recalls.

*2) **Purging**:* The merged prediction usually tends to be over-segmented, which means many predicted positions need to be deleted. In the purging stage, we leverage ProsegSiamese to filter out the false boundary predictions. For each predicted position that needs to be determined, we extract the sequence data on the left and right sides of it and input them to the ProsegSiamese. The output of the model is the probability of the true field boundaries. Those positions of low probability will be removed from the candidates.

Meanwhile, an essential characteristic of protocols is that they are binary-aligned, which means their field segmentation

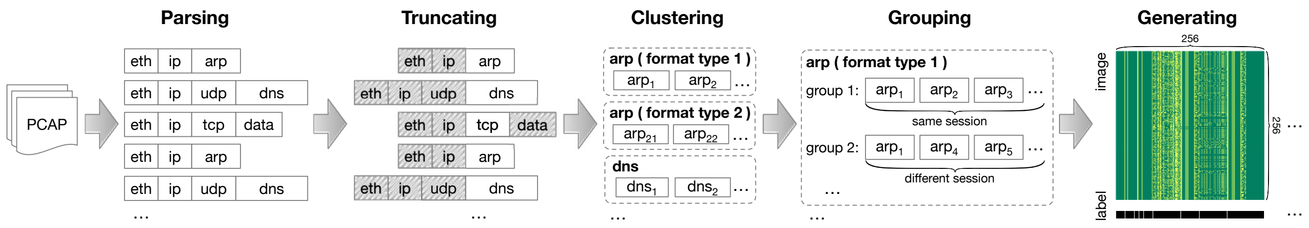


Fig. 7: The process of generating training data.

positions are commonly the multiples of 2 and mostly the multiples of 8. When short fields appear in the protocol format, they tend to come up consecutively with similar lengths and form one or several complete bytes. Thus, very few predicted positions that are not binary-aligned and or close to others will be considered unaligned. We remap these outliers to the nearby aligned positions.

B. Training Data Generation

Large amounts of protocol messages are needed to leverage the deep learning model in PRE and train the models. This subsection describes the construction details of the training data set. Traditional annotation task costs significant manual effort. Luckily, noticing that many DPI (Deep Packet Inspection) tools like *tshark* [51] have already utilized the knowledge of acquirable protocols, we propose an automatic method to generate training data using captured PCAP files. We utilize knowledge, such as the position, length, bitmask, and type of the fields to segment the protocols for constructing training data set. As shown in Fig. 7, the whole process consists of five steps, parsing, truncating, clustering, grouping, and generating.

We first collect network traffic from public sources, such as CAIDA [52], Netresec [53], and UNSW [54]. Most of them are provided in PCAP-like files, and we ended up collecting more than 100 files. In the *parsing* phrase, executing tshark with '-T jsonraw' option to handle the collected PCAP files and load the messages. In the *truncating* step, we remove the irrelevant layers and encapsulate metadata. Then in the *clustering* step, truncated data will be classified into clusters of the same format because the same format data tend to have similar characteristics in field patterns and value distributions. Clustering is just a rough classification. The data will then be distributed into smaller groups, called *grouping*. Data will be organized together so that the neural network can mine the features of different fields. There are two grouping principles for facilitating the features extraction:

- Grouping by the same sessions. This strategy tends to group packets of the same session together, facilitating the model to learn the features of fields of types such as sequence number and data. These fields have distinct time-sequential value distribution features.
- Grouping by different sessions. The values of identity or control-related fields tend to be fixed in the same session, resulting in the lack of diversity in generated image textures. This strategy can reveal the features of these fields, rather than make models only regard them as static fields.

During *generating*, we convert the first 32 bytes of each data to its binary representation of 256 bits. Then we stack every 256 items in the same group together to form single-channel grayscale images of size 256 × 256. If the data length is less than 32 bytes, or if the available items are less than 256, we use -1 as paddings. When the data length exceeds 32 bytes, we slide to the next 32 bytes of data and perform the same operations. Meanwhile, field position information from tshark is used to generate the ground-truth labels. For the corresponding bit positions in data, mark the starting positions of each fields as 1 and the other positions as 0. Additionally, some protocols like DNS are composed of both textual and binary fields, and the textual fields are usually variable and excessively lengthy. Additional digest replacement operations are performed on these textual fields to align the protocol format. First, we treat consecutive printable characters as the textual field and use the hash function to generate its fixed-length digest (MD5 is used in our method, and only the first 4 bytes reserved). The digest is then used to replace the original textual field in the raw message.

If ProsegDL is deployed in practice to segment an unknown protocol format, we need to train the model with as much data as possible. However, for experiments, we choose the most representative protocols for training, and use several other protocols for testing. We select eleven binary protocols to form the data set, as listed in TABLE I. Their formats can cover the most frequent field combinations. Meanwhile, traffic in the same session tends to be pattern-fixed, and the messages are of high similarity. Therefore, we filter out similar images from the automatically generated data set and retain the most unique ones. This step requires little manual involvement.

TABLE I: Generated training protocol images, and up to 500 images per protocol. During training, they are divided into two parts in a ratio of 8:2 for training and evaluation.

Protocol	Count	Protocol	Count	Protocol	Count
arp	500	dns	500	udp	500
tcp	500	ssl	500	ip	500
icmpv6	500	stun	215	eapol	168
esp	126	nbdgm	53	***Total***	4062

V. Experiments and Results

In this section, we first introduce the experiment setups (V-A) with the goals, specimen protocols, comparison approaches, and the testbeds, followed by the metric (V-B) for evaluating the results. Finally, we present the experimental results with corresponding interpretations (V-C).

A. *Experiment Setups*

Although protocol format extraction is intended for reverse analysis of unknown protocols, we must use known protocols to test the methods because the ground-truth knowledge of unknown protocols is not available. We set up the experiments with the following details:

*1) **Experimental Goals**:* The purpose of our experiments is to evaluate the performance of ProsegDL, and to discover its advantages and limitations. We outline the experiments from the following three goals:

- **Performance on protocols:** The field segmentation result is the most important indicator for measuring the performance of the format extraction method. By comparing the output of ProsegDL with the ground-truth, we can quantify its segmentation capability. The comparison is performed on multiple protocols and multiple methods.
- **Data scalability:** Different methods have different data amount requirements, corresponding to their processing performance's bottom and top lines. Methods may not work with small amounts of data or may have difficulty handling large amounts of data. We test ProsegDL and comparison methods to discover their data scalabilities.
- **Processing time:** The time consumption of ProsegDL is mainly from two parts: the runtime of multi-image inference using ProsegUNet, and the runtime of fused prediction judgment using ProsegSiamese. We test ProsegDL and other approaches with different amounts of messages and record the runtime.

Besides, memory usage is not our concern because ProsegDL is a deep learning-based tool which requires for CUDA memory. Its usage does not directly reflect the differences between the neural network model and the non-deep learning comparison algorithms.

*2) **Specimen Protocols**:* Six binary protocols are selected as the testing targets, including four Internet protocols (NTP, DHCP, NBNS, and SMB) and two ICS protocols (S7Comm, and Modbus). These protocols are classical, widely used, and representative enough. Most importantly, they can be dissected to get their field segmentation ground-truths (though s7comm is still a private protocol of Siemens, it can also be partially parsed). We collect the network traffic of these protocols and select the messages with the highest variance as the testing data. In particular, due to the lack of Modbus and S7Comm traffic from public sources, we collected their network traffic from our ICS experimental environment (built on 300/400 series Siemens, ABB, and Schneider PLCs, as well as the corresponding supervisor computers). We generate the protocol images for each protocol as the testing inputs.

*3) **Comparison Approaches**:* There are many proposed methods aiming at solving the unknown protocol format extraction problem. However, the code and experimental data of most methods are close-sourced. They cannot be used for comparative experiments directly. Meanwhile, many methods can not directly extract the format of the protocol. Some approaches like Netplier are dedicated to message clustering, which is another important step before the format extraction according to the basic procedure of PRE [12]. These methods require additional manual involvement during the segmentation phase. They are also incomparable. Therefore, NEMESYS and Netzob are selected as the final comparison methods. They are open-sourced, and are targeted at the format extraction. NEMESYS is a state-of-the-art probability-based format extraction method that focuses on the internal structure of the protocol messages [55]. Netzob is an alignment-based method that leverages the NW algorithm and semantic information for segmentation [29]. It is a milestone approach in PRE and is the most frequently used baseline.

*4) **Testbeds**:* The experiments of ProsegDL were carried out on the Intel i7-8700 3.2GHz with 32 GiB of memory and Nvidia GTX-1080ti running Ubuntu 16.04.3. NEMESYS and Netzob were also tested on the same PC with their Docker implementation versions, which can be accessed on the Github [56] [57].

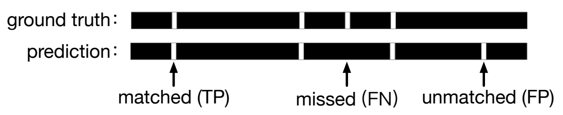


Fig. 8: Types of inferred field boundaries. We denote positions of value 1 both in prediction and ground-truth as *matched* (*TP*), only of value 1 in prediction as *unmatched* (*FN*) and only of value 1 in ground-truth as *missed* (*FP*).

B. *Metrics*

The outputs of ProsegDL are the segmentation vectors, similar to the results of sequence annotation tasks like NER (Named Entity Recognition). Thus, we can refer to those metrics commonly used in the NER result evaluations. The analysis results of NEMESYS and Netzob can also be converted into vector-like annotations. We compare each bit position in the predicted segmentation sequence with the corresponding ground-truth label sequence, and mark them as illustrated in Fig. 8. We furthermore consider precision(Prec.), recall (Rec.), and F1 as evaluation metrics.

- **Precision**: The soundness of the predictions. It is calculated as the ratio of that the number of *matched* positions by the total positions predicted to be true.
- **Recall**: The coverage of the predictions. It is calculated as the ratio of the number of *matched* positions by the total number of true segmentation positions in ground-truth.

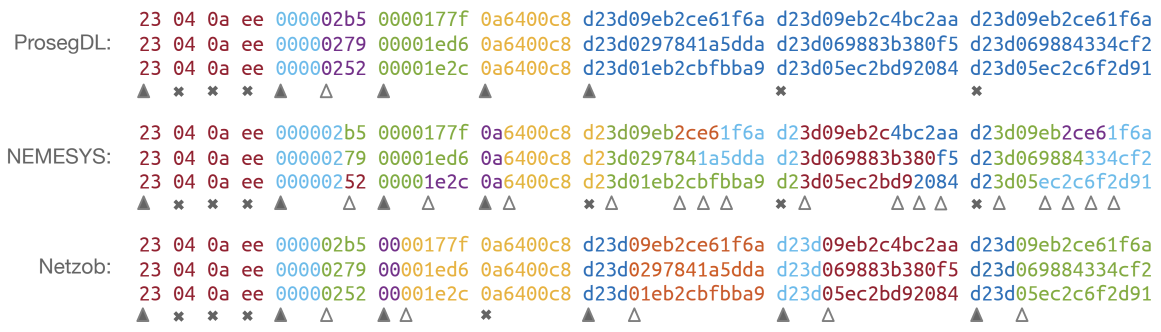
$$Prec. = \frac{TP}{TP+FP}, \quad Rec. = \frac{TP}{TP+FN}$$

- **F1**: The accuracy of the segmentation results by computing scores that consider both the precision and the recall.

$$F1 = \frac{2 \times Prec. \times Rec.}{Prec. + Rec.}$$

TABLE II: Format segmentation performances of different methods.

Protocol	$ProsegDL_{unfused}$			$ProsegDL_{fused}$			$NEMESYS$			$Netzob$		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
s7comm	0.905	0.578	0.705	**0.800**	**0.750**	**0.774**	0.349	0.303	0.325	0.534	0.525	0.529
modbus	0.600	0.290	0.391	0.696	**0.479**	**0.567**	0.833	0.313	0.455	**0.900**	0.281	0.429
ntp	0.635	0.257	0.366	**0.556**	0.416	0.476	0.199	0.215	0.207	0.453	**0.709**	**0.553**
dhcp	0.672	0.433	0.527	**0.667**	**0.500**	**0.571**	0.551	0.301	0.389	0.636	0.372	0.469
nbns	0.573	0.686	0.625	**0.688**	**0.733**	**0.710**	0.679	0.561	0.615	0.266	0.378	0.312
smb	0.412	0.218	0.285	0.597	**0.621**	**0.609**	0.594	0.354	0.444	**0.618**	0.254	0.360



```
23 04 0a ee 000002b5 0000177f 0a6400c8 d23d09eb2ce61f6a d23d09eb2c4bc2aa d23d09eb2ce61f6a
23 04 0a ee 00000279 00001ed6 0a6400c8 d23d0297841a5dda d23d069883b380f5 d23d069884334cf2
23 04 0a ee 00000252 00001e2c 0a6400c8 d23d01eb2cbfbba9 d23d05ec2bd92084 d23d05ec2c6f2d91
```

Fig. 9: (Best viewed in color.) An example of field segmentation results of NTP protocol between approaches. Inferred fields are marked in different colors, and the spaces denote the ground-truth segmentation positions, ▲ denotes the *matched* positions, △ denotes the *unmatched* positions, ✖ denotes the *missed* positions.

Typically, recall can reach high scores when the protocol format is over-segmented into too many unnecessary segments. The same goes for the precision score, which can be extremely high when the format is under-segmented. Therefore, a reasonable goal is to achieve a high score in both precision and recall, which is reflected by the F1 score.

C. Results and Interpretations

Following the pre-designed experiment goals (V-A1), our experiments are conducted in three parts: assessing the format extraction results of different protocols, performances with different sizes of measured messages, and the time consumption. The experimental results are as follows.

1) ***Results of Different Protocols:*** In this part, we conducted experiments to assess the format extraction abilities of different approaches. All tests were carried out with 800 messages of each protocol. It is mainly because the runtime and memory overhead of Netzob with more data are unacceptable for some protocols, which will be detailed in the following subsection. We concatenate the predictions and compare them with the concatenated ground-truth to calculate the Prec., Rec. and F1 scores.

To demonstrate the effectiveness of the prediction fusion step stage ProsegDL, we perform the experiments separately on the direct outputs of ProsegUNet only (denoted as $ProsegDL_{unfused}$) and the final outputs of the entire system (denoted as $ProsegDL_{fused}$). Meanwhile, The P_{min} parameter in the prediction fusion stage of $ProsegDL_{fused}$ was set to 0.1 empirically. $ProsegDL_{unfused}$ requires no parameter adjustment during experiments. The results of NEMESYS are affected by the σ parameter, which adjusts the Gaussian filter for noise reduction. We iterate σ from 0.6 to 1.2 with the interval of 0.1 and record the highest F1 score result as the final segmentation. It is the same for the similarity $thresh$ parameter in Netzob, which affect the message clustering results. We also iterate its value from 40 to 80 with the interval of 5 and record the result of the highest F1 score. All experiment results are listed in TABLE II.

Comparing the format extraction results of different approaches, we can find that ProsegDL achieves better performance than NEMESYS and Netzob in precision, recall, and F1 scores for most protocols. However, in some protocols, like NTP, ProsegDL is not good enough. The main reason is that the NTP has consecutive similar data fields in its messages, which is quite tricky for ProsegUNet to judge the field boundaries and more likely to treat them as one field. NEMESYS's results are also undesirable, which is caused by its over-segmentation when using statistical methods to handle the field values. On the contrary, Netzob can achieve a better result with a relatively broad clustering thresh. Fig. 9 shows the results of inferring a small set of NTP protocol messages, where the differences between the three methods are evident. Meanwhile, the results between $ProsegDL_{unfused}$ and $ProsegDL_{fused}$ in TABLE II also indicate that the prediction fusion step is critical for the ProsegDL. The importance of this step is mainly reflected in the improvement of the recall rate, and the results are in line with our expectations.

2) ***Data Sets of Different Sizes:*** We evaluated the performance of ProsegDL with different amounts of messages and compared it with NEMESYS and Netzob. The test data is organized with 100, 800, and 10,000 messages per protocol (600, 4,800, and 60,000 messages in total). The Prec., Rec, and F1 scores are calculated as the average of corresponding scores of all protocols. Netzob spent more than an hour without completing the inferences for DHCP and SMB when 1000 messages were measured, so 800 messages were used as the second comparison quantity. In addition, ProsegDL could not generate at least one usable protocol image with 100 messages for each protocol, so the scores were also discarded. The reusluts are listed in TABLE III.

TABLE III: Format extraction performances of different methods with different amounts of messages. n_images denotes the number of images that ProsegDL generated from messages.

Approaches		Messages measured of each protocol		
		100	800	10000
ProsegDL	n_images	3	71	637
	Prec.	-	0.579	0.706
	Rec.	-	0.568	0.643
	F1	-	0.573	0.673
NEMESYS	Prec.	0.517	0.515	0.521
	Rec.	0.343	0.360	0.363
	F1	0.412	0.424	0.428
Netzob	Prec.	0.431	0.493	-
	Rec.	0.420	0.428	-
	F1	0.425	0.458	-

We notice that ProsegDL can infer better predictions when fitting more messages due to its mechanism. However, if a generated image has too many empty padding areas, it will not work. According to our experimental results, an image must consist of at least 130 messages for the models to generate a valid result. Otherwise, the convolutional network may fail to extract features correctly and commonly output empty segmentation vectors. This indicates that ProsegDL has higher data volume requirements than the other two methods, which is a shortage. However, the results also illustrate the higher upper limit of its capabilities when processing a large amount of data. NEMESYS is a more stable method and can infer the results with an extensive range of data amounts. Netzob is not suitable for handling too many messages for its high overhead.

*3) **Time Consumption**:* We ran ProsegDL, NEMESYS, and Netzob with different numbers of messages from 100 to 1000 with an interval of 100, and all six protocols were tested to calculate the average runtime. In addition, the time for data loading and results preparation is not included. Though running NEMESYS and Netzob in Docker has little performance loss, it does not affect the overall results. Be noticed that the runtime of every approach is not only related to the number of messages but also related to message length. Here we only consider the former factor because the scores are calculated as the average of all protocols. The results are shown in Fig. 10.

We can find that there is a linear correlation between the runtime and the number of messages for ProsegDL and NEMESYS. The latter runs slightly faster than the former, but the gap is acceptable. They are all considered to have the advantage in scalability. The time consumption of ProsegDL is divided into two main parts, the prediction phase and the fusion phase. In the prediction phase, the processing time of ProsegUNet increases linearly with the number of messages. In contrast, the runtime of the fusion phase is almost fixed. Because the segmentation positions that need to be judged by ProsegSiamese are already merged, their number changes very little. Netzob spent much more time completing the calculation than the other two methods. The main reason is its application of the global MSA algorithm, which has an exponential runtime overhead.

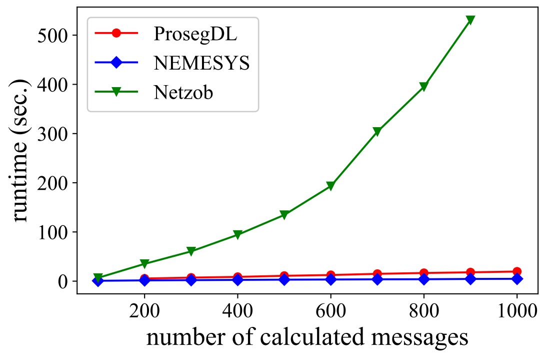


Fig. 10: Runtime of each approaches when calculating different amounts of messages. Netzob spent too many time when computing with more than 1000 messages, and ProsegDL failed to generate valid input images with 100 messages. Thus these two data points are discarded.

Summary: Experimental results above conclude that ProsegDL outperforms NEMESYS and Netzob in most circumstances. **It is important to note that** some works trained and tested their deep learning models on the same protocols (only divide the data set into training and testing parts at a specific ratio). However, we trained the models with some protocols and tested them with the others, though it led to a cliff fall in experimental scores. Our supervised models are based on the assumption that there are commonalities between different protocols in format designs, which are like the ”genes” within. We want to prove that the proposed methods are valid not only for the trained protocols but also for the unknown protocols.

VI. Discussion

ProsegDL is a novel deep learning-based approach for binary protocol format extraction. However, there are still several details that need further discussion. In this section, we want to discuss its limitations (VI-A) and future works (VI-B).

A. *Limitations of ProsegDL*

ProsegDL has shown its advantages in inference performance and scalability. However, there is no one-size-fits-all solution, and ProsegDL has limitations mainly as follows:

*1) **Data Requirements**:* There are two aspects of data limitations for ProsegDL. First, training data is the most significant obstacle to supervised deep learning PREs. And when deploying ProsegDL in practice, network traffic requirements are still large. We should collect traffic of as many protocols as possible, not just the experimental 11 protocols in the paper. Second, ProsegDL performs poorly when lacking target unknown protocol's traffic compared to NEMESYS and

Netzob. A large amount of data is a prerequisite for its better performance, and it is not a proper choice for little traffic PRE.

*2) **Robustness and Granularity**:* We randomly delete, switch or duplicate few fields in the raw messages for the six experimental protocols to test the robustness of ProsegDL. And the average results show no significant deviations in the format extraction performance. Thus our approach is considered to have certain robustness against small changes in the protocol format. Meanwhile, ProsegDL can segment fields at most at the half-byte level and still cannot achieve bit-level segmentation. We tried to train ProsegUNet with bit-level annotations, but it tends to ignore them. The main reason is that the bit-level contextual features are too fragmented for model to learn, and difficult to be retained during downsampling.

*3) **Variable-length or Adversarial Protocols**:* ProsegDL is targeted at binary protocols, which commonly have fixed format structures. However, there are protocols with variable formats, especially those with text fields. ProsegDL does not perform well when handling the data of these protocols directly. More pre-processing is needed, for example, digesting textual fields to fix length representations. For protocols used by malware, they are designed with some adversarial measures like encoding or obfuscation to make their messages harder to reverse. Expert knowledge is still necessary in these cases.

*B. **Future Works***

Our works propose a novel binary protocol format segmentation solution using deep learning models. There are still several works to do in the future.

*1) **Protocol Semantics Inference**:* ProsegDL currently focuses on binary protocol format extraction. However, the image semantic segmentation technique has abilities of pixel-level labeling, which can be leveraged for field type and semantic deduction.

*2) **End-to-end Model**:* ProsegDL is considered a cascade model. It combines ProsegUNet and ProsegSiamese sequentially to do the format extraction. However, cascade models may accumulate more errors due to the parameter transfer and the lack of sharing of information between models and. The end-to-end model may be able to achieve better performance in the future.

*3) **Message Clustering**:* When deployed in practice to infer the format of unknown protocols, ProsegDL requires accurate clustering messages as inputs, or its performance will be significantly degraded. Clustering is not the focus of our works and we only re-use Netplier as the clustering method during pre-processing. However, Netplier is difficult to handle larger amounts of data. Deep learning methods may achieve better clustering results within an acceptable time consumption.

VII. Related works

Protocol Security Analysis. Protocol security analysis is the most important application scenario for PRE. ICS protocols have become the focus of security analysis, and fuzzing is the mainstream analyzing method [6]–[8]. When using PRE techniques for ICS protocol fuzzing, keyword fields in the target protocol format need to be obtained. Based on this, we can construct the mutated test cases for fuzzing. Malware analysis is also the focus of network security [9], [10]. With the PRE approach of network tracing, we can depict malware traffic and behavior from low-level analysis instead of high-level program analysis techniques.

Traffic Identity and Intrusion Detection. Modeling unknown protocols and identifying their traffic is one of the basic applications of PRE. The knowledge of unknown protocols retrieved from PRE enables deep packet inspection tools to identify its network traffic in real-time [13]. Furthermore, the intrusion detection system can be constructed for malware defense and analysis [58] [11]. PRE provides a method to quickly build defenses against unknown intrusion attacks, especially for early industrial control systems that lack security mechanisms.

PRE meets Machine/Deep Learining. In recent years, machine learning and deep learning approaches have been proposed with better feature extraction for PRE. N-gram is aa important model from NLP for message feature extraction and is used in many methods [42] [43]. Wang et al. [59] leverage CNN to classify binary message bytes and further segment the format. Zhao et al. [60] and Yang et al. [45] also segment the format by classifying the type of every byte but with the LSTM-FCN model. PREUNN [61] proposed deep learning-based reversing architecture based on multiple types of neural networks. PRE with deep learning is still a rising and promising sub-field in the future.

VIII. Conclusion

Protocol reverse engineering is proposed to extract the specifications of protocols and facilitate the security research (e.g., fuzzing and malware analysis). Based on the intuitive observations of the fields and their boundaries‘ features in the stacked multiple messages, we propose ProsegDL, a novel deep learning-based binary protocol format extraction tool. We implement it by integrating image semantic segmentation model and siamese network sequentially. Powered by the extraordinary feature extraction capability of deep learning, our approach can perform the precise format extraction within acceptable runtime. Experimental results show that our tool outperforms the state-of-the-art approaches. Once well trained, it is able to infer unknown protocol formats, facilitating the PRE process and further security analysis.

Acknowledgment

We would like to thank all the anonymous reviewers for their constructive comments on the earlier version of this paper and our shepherd for helping us improve the presentation of the final version. This work is supported in part by the National Key Research and Development Program of China (Grant No. 2020YFB2103803), the Young Scientists Fund of the National Natural Science Foundation of China (Grant No. 62002342), and the Key Program of the National Natural Science Foundation of China (Grant No. 61931019).

References

[1] K. Afzal, R. Tariq, F. Aadil, Z. Iqbal, N. Ali, and M. Sajid, "An optimized and efficient routing protocol application for iov," *Mathematical Problems in Engineering*, vol. 2021, 2021.

[2] S. Yu, J. Lee, K. Park, A. K. Das, and Y. Park, "Iov-smap: Secure and efficient message authentication protocol for iov in smart city environment," *IEEE Access*, vol. 8, pp. 167 875–167 886, 2020.

[3] C. Goes, "The interblockchain communication protocol: An overview," *arXiv preprint arXiv:2006.15918*, 2020.

[4] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[5] Z. Luo, F. Zuo, Y. Shen, X. Jiao, W. Chang, and Y. Jiang, "Ics protocol fuzzing: Coverage guided packet crack and generation," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[6] H. Zhao, Z. Li, H. Wei, J. Shi, and Y. Huang, "Seqfuzzer: An industrial protocol fuzzing framework from a deep learning perspective," in *2019 12th IEEE Conference on software testing, validation and verification (ICST)*. IEEE, 2019, pp. 59–67.

[7] P. Fiterau-Brostean, B. Jonsson, R. Merget, J. De Ruiter, K. Sagonas, and J. Somorovsky, "Analysis of {DTLS} implementations using protocol state fuzzing," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2523–2540.

[8] S. Jero, M. L. Pacheco, D. Goldwasser, and C. Nita-Rotaru, "Leveraging textual specifications for grammar-based fuzzing of network protocols," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 9478–9483.

[9] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security 17)*, 2017, pp. 1093–1110.

[10] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: analysis of a botnet takeover," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 635–647.

[11] Y. Cao, Y. Shoshitaishvili, K. Borgolte, C. Kruegel, G. Vigna, and Y. Chen, "Protecting web-based single sign-on protocols against relying party impersonation attacks through a dedicated bi-directional authenticated secure channel," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2014, pp. 276–298.

[12] 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, 2018.

[13] W. Cui, J. Kannan, and H. J. Wang, "Discoverer: Automatic protocol reverse engineering from network traces." in *USENIX Security Symposium*, 2007, pp. 1–14.

[14] J. Caballero, H. Yin, Z. Liang, and D. Song, "Polyglot: Automatic extraction of protocol message format using dynamic binary analysis," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 317–329.

[15] C. Y. Cho, D. Babić, P. Poosankam, K. Z. Chen, E. X. Wu, and D. Song, "{MACE}:{Model-inference-Assisted} concolic exploration for protocol and vulnerability discovery," in *20th USENIX Security Symposium (USENIX Security 11)*, 2011.

[16] P. M. Comparetti, G. Wondracek, C. Kruegel, and E. Kirda, "Prospex: Protocol specification extraction," in *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 110–125.

[17] Y. Ye, Z. Zhang, F. Wang, X. Zhang, and D. Xu, "Netplier: Probabilistic network protocol reverse engineering from message traces." in *NDSS*, 2021.

[18] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew, "A review of semantic segmentation using deep neural networks," *International journal of multimedia information retrieval*, vol. 7, no. 2, pp. 87–93, 2018.

[19] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "Lstm fully convolutional networks for time series classification," *IEEE access*, vol. 6, pp. 1662–1669, 2017.

[20] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," *Advances in neural information processing systems*, vol. 6, 1993.

[21] H. Li, B. Shuai, J. Wang, and C. Tang, "Protocol reverse engineering using lda and association analysis," in *2015 11th International Conference on Computational Intelligence and Security (CIS)*. IEEE, 2015, pp. 312–316.

[22] X. Wang, K. Lv, and B. Li, "Ipart: an automatic protocol reverse engineering tool based on global voting expert for industrial protocols," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 35, no. 3, pp. 376–395, 2020.

[23] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[24] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[25] S. Gorbunov and A. Rosenbloom, "Autofuzz: Automated network protocol fuzzing framework," *Ijcsns*, vol. 10, no. 8, p. 239, 2010.

[26] C. Leita, K. Mermoud, and M. Dacier, "Scriptgen: an automated script generation tool for honeyd," in *21st Annual Computer Security Applications Conference (ACSAC'05)*. IEEE, 2005, pp. 12–pp.

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

[28] S. Tao, H. Yu, and Q. Li, "Bit-oriented format extraction approach for automatic binary protocol reverse engineering," *Iet Communications*, vol. 10, no. 6, pp. 709–716, 2016.

[29] G. Bossert, F. Guihéry, and G. Hiet, "Towards automated protocol reverse engineering using semantic information," in *Proceedings of the 9th ACM symposium on Information, computer and communications security*, 2014, pp. 51–62.

[30] F. Meng, C. Zhang, and G. Wu, "Protocol reverse based on hierarchical clustering and probability alignment from network traces," in *2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA)*. IEEE, 2018, pp. 443–447.

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

[32] O. Esoul and N. Walkinshaw, "Using segment-based alignment to extract packet structures from network traces," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2017, pp. 398–409.

[33] Y. Wang, N. Zhang, Y.-m. Wu, B.-b. Su, and Y.-j. Liao, "Protocol formats reverse engineering based on association rules in wireless environment," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2013, pp. 134–141.

[34] G. Li, Q. Qian, Z. Wang, X. Zou, X. Chen, and X. Wu, "Protocol keywords extraction method based on frequent item-sets mining," in *Proceedings of the 2018 International Conference on Information Science and System*, 2018, pp. 53–58.

[35] X. Hei, B. Bai, Y. Wang, L. Zhang, L. Zhu, and W. Ji, "Feature extraction optimization for bitstream communication protocol format reverse analysis," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2019, pp. 662–669.

[36] Z. Zhang, Z. Zhang, P. P. Lee, Y. Liu, and G. Xie, "Toward unsupervised protocol feature word extraction," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 10, pp. 1894–1906, 2014.

[37] M.-S. Lee, Y.-H. Goo, K.-S. Shim, S.-H. Yoon, S.-H. Ji, and M.-S. Kim, "A method for extracting static fields in private protocol using entropy and statistical analysis," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2019, pp. 1–6.

[38] J. Cai, J.-Z. Luo, and F. Lei, "Analyzing network protocols of application layer using hidden semi-markov model," *Mathematical Problems in Engineering*, vol. 2016, 2016.

[39] B.-C. Li and S.-Z. Yu, "Keyword mining for private protocols tunneled over websocket," *IEEE Communications Letters*, vol. 20, no. 7, pp. 1337–1340, 2016.

[40] J.-Z. Luo and S.-Z. Yu, "Position-based automatic reverse engineering of network protocols," *Journal of Network and Computer Applications*, vol. 36, no. 3, pp. 1070–1077, 2013.

[41] Y. ZHU, J. HAN, L. YUAN *et al.*, "Spfpa: A format parsing approach for unknown security protocols," *Journal of Computer Research and Development*, vol. 52, no. 10, pp. 2200–2211, 2015.

[42] 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 *2012 20th IEEE*

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

[43] P. Lin, Z. Hong, L. Wu, Y. Li, and Z. Zhou, "Protocol format extraction based on an improved cfsm algorithm," *China Communications*, vol. 17, no. 11, pp. 156–180, 2020.

[44] S. Asgari Taghanaki, K. Abhishek, J. P. Cohen, J. Cohen-Adad, and G. Hamarneh, "Deep semantic segmentation of natural and medical images: a review," *Artificial Intelligence Review*, vol. 54, no. 1, pp. 137–178, 2021.

[45] C. Yang, C. Fu, Y. Qian, Y. Hong, G. Feng, and L. Han, "Deep learning-based reverse method of binary protocol," in *International Conference on Security and Privacy in Digital Economy*. Springer, 2020, pp. 606–624.

[46] S. Fagerland, "The many faces of gh0st rat: Plotting the connections between malware attacks." [Online]. Available: http://download01.norman.no/documents/ThemanyfacesofGh0stRat.pdf

[47] Y. Liu and H. Wang, "Tracking mirai variants," *Virus Bulletin*, pp. 1–18, 2018.

[48] P. Liu, Y. Zheng, Z. Song, D. Fang, S. Lv, and L. Sun, "Fuzzing proprietary protocols of programmable controllers to find vulnerabilities that affect physical control," *Journal of Systems Architecture*, vol. 127, p. 102483, 2022.

[49] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[50] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[51] "Tshark.dev." [Online]. Available: https://tshark.dev

[52] "Caida." [Online]. Available: https://www.caida.org

[53] "Netresec." [Online]. Available: https://www.netresec.com

[54] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2018.

[55] S. Kleber, H. Kopp, and F. Kargl, "{NEMESYS}: Network message syntax reverse engineering by analysis of the intrinsic structure of individual messages," in *12th USENIX Workshop on Offensive Technologies (WOOT 18)*, 2018.

[56] "Nemesys." [Online]. Available: https://github.com/vs-uulm/nemesys

[57] "Netzob." [Online]. Available: https://github.com/netzob/netzob

[58] A. Abdou, D. Barrera, and P. C. v. Oorschot, "What lies beneath? analyzing automated ssh bruteforce attacks," in *International conference on PASSWORDS*. Springer, 2015, pp. 72–91.

[59] Y. Wang, B. Bai, X. Hei, L. Zhu, and W. Ji, "An unknown protocol syntax analysis method based on convolutional neural network," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 5, p. e3922, 2021.

[60] R. Zhao and Z. Liu, "Analysis of private industrial control protocol format based on lstm-fcn model," in *Proceedings of the 2020 International Conference on Aviation Safety and Information Technology*, 2020, pp. 330–335.

[61] V. Kiechle, M. Börsig, S. Nitzsche, I. Baumgart, and J. Becker, "Preunn: Protocol reverse engineering using neural networks." in *ICISSP*, 2022, pp. 345–356.