Protocol Reverse Engineering via Deep Transfer Learning

YANYANG ZHAO, School of Software, Tsinghua University, China
ZHENGXIONG LUO[*], National University of Singapore, Singapore
WENLONG ZHANG, Central South University, China
FEIFAN WU, School of Software, Tsinghua University, China
YUANLIANG CHEN, School of Software, Tsinghua University, China
FUCHEN MA, School of Software, Tsinghua University, China
QI XU, School of Software, Tsinghua University, China
HEYUAN SHI, Central South University, China
YU JIANG[*], School of Software, Tsinghua University, China

Protocol reverse engineering infers the specification of proprietary or poorly documented protocols and serves as the foundation for security analysis such as fuzz testing. While many existing techniques achieve this by mining statistical features from network traces, they face increasing challenges due to incomplete field pattern information available in the traces. While protocol development has accumulated rich prior knowledge about protocol design, this knowledge remains largely untapped in protocol reverse engineering.

This paper introduces TransRE, a protocol reverse engineering tool that leverages prior syntax knowledge from standardized protocols through deep transfer learning to better understand proprietary protocols. TransRE first selects optimal source domains by analyzing inter-domain differences between the existing knowledge base and the target protocol. It then employs a neural network to extract representation features and applies domain adaptation techniques to optimize the syntax transfer model, enabling accurate inference of protocol formats. Our evaluation on 12 widely used protocols shows that TransRE identifies fields with a perfection score of 0.43, which is 1.48×-3.07× the performance achieved by five state-of-the-art methods. Furthermore, to demonstrate practical applicability, we enhanced an existing protocol fuzzer with TransRE for testing proprietary protocols in real-world network cameras and discovered four bugs.

CCS Concepts: • **Networks** → **Protocol testing and verification**; • **Computing methodologies** → *Transfer learning*.

Additional Key Words and Phrases: Protocol Reverse Engineering, Deep Transfer Learning

ACM Reference Format:
Yanyang Zhao, Zhengxiong Luo, Wenlong Zhang, Feifan Wu, Yuanliang Chen, Fuchen Ma, Qi Xu, Heyuan Shi, and Yu Jiang. 2026. Protocol Reverse Engineering via Deep Transfer Learning. 1, 1 (February 2026), 21 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

[*]Zhengxiong Luo and Yu Jiang are the corresponding authors.

Authors' Contact Information: Yanyang Zhao, zhaoyanyoung@163.com, School of Software, Tsinghua University, China; Zhengxiong Luo, National University of Singapore, Singapore, Singapore, fouzhe15@gmail.com; Wenlong Zhang, Central South University, Changsha, China; Feifan Wu, School of Software, Tsinghua University, Beijing, China; Yuanliang Chen, School of Software, Tsinghua University, Beijing, China; Fuchen Ma, School of Software, Tsinghua University, Beijing, China; Qi Xu, School of Software, Tsinghua University, Beijing, China; Heyuan Shi, Central South University, Changsha, China; Yu Jiang, School of Software, Tsinghua University, Beijing, China, jiangyu198964@126.com.

ACM XXXX-XXXX/2026/2-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Protocol reverse engineering analyzes proprietary or poorly documented protocols and serves as the foundation for various security analyses, such as fuzzing [17, 23, 29, 30, 48], security reinforcement [3, 37], and model checking [21, 33]. For example, protocol fuzzing requires constructing test packets according to protocol specifications to detect bugs. However, in many critical scenarios such as Industrial Control Systems (ICS), protocol specifications are often unavailable [37, 40].

Protocol reverse engineering involves inferring protocol formats through program analysis or network trace analysis. Program-based methods [7, 12, 22, 32] use techniques like taint analysis to monitor protocol program execution and track message processing, achieving high accuracy empowered by rich runtime semantics. However, access to source code and binary firmware is often restricted in proprietary scenarios such as ICS or IoT.

Alternatively, network trace-based approaches [4, 6, 8, 13, 27, 50] perform statistical analysis on captured network traces using techniques like message clustering, sequence alignment, and heuristic rules to identify fixed patterns. Though easier to deploy, these methods depend on message completeness and suffer from low accuracy due to inflexible field patterns. First, they employ fixed patterns derived from heuristic rules to identify fields, limiting their applicability to protocols incompatible with predefined patterns. For example, TCP timestamps [18] do not conform to standard formats like NTP or Unix timestamps, making them unrecognizable by BinaryInferno's [8] timestamp detector. Second, fixed patterns with thresholds are inaccurate for protocol-specific field recognition. Existing techniques using uniform approaches to standardize all protocols can lead to low accuracy. Learning-based approaches that rely on large datasets to learn inherent patterns of proprietary protocols [19, 36] face difficulties in obtaining sufficient and high-quality labeled data for training, which is often impractical and labor-intensive.

Protocol design and implementation typically follow functional application requirements and are grounded in specifications and knowledge from existing protocols [1, 39]. This shared foundation results in common patterns and similar structures across different protocols. For example, *Length* fields in Modbus, AMQP, and IEC104 protocols all follow a linear relationship between message length and field value [35]. Leveraging prior knowledge of these consistent patterns can significantly improve protocol reverse engineering. However, existing approaches primarily rely on statistical analysis to infer protocol formats using general field patterns observed from some protocols. The substantial differences between protocols often hinder the application of these general patterns, limiting analysis accuracy and effectiveness.

To address this problem, we introduce TransRE, a protocol reverse engineering tool that leverages prior syntax knowledge from standardized protocols [1] to more accurately understand proprietary protocols through deep transfer learning. TransRE first identifies optimal source domains by analyzing inter-domain differences between the knowledge base and the protocol under analysis. It then extracts syntax patterns from standardized protocols through deep transfer learning to infer proprietary protocol syntax. **To achieve this approach, we need to address two challenges**: (i) How to measure protocol differences and similarity without specifications for effective source domain selection. Achieving standardized difference evaluation among diverse protocols is challenging because different protocols have varying message types and structures. (ii) How to design a general transfer learning strategy for protocol format recovery. Ensuring accurate pattern transmission across protocols is challenging because different protocols may have distinct field characteristics.

[1]This work only focuses on message syntax, and does not support the inference of state transitions. We refer to protocols with specifications as standardized protocols, which are typically public, and protocols without specifications as proprietary protocols, which are often less-documented or custom-built.

For the first challenge, we transfer the multi-message sequence transpose to the distance of the metric distribution in the calculable space through the mapping transformation due to the different feature spaces among protocols. We then establish a transfer-oriented metric to identify the optimal mapping relationship for maximizing the mean difference. This difference estimation is feasible under simple protocols with the same length and syntax. However, it fails when protocols differ significantly, as messages across different protocols vary widely. Using fixed mapping rules to construct protocol features may cause errors, resulting in significant bias when estimating the distance of different protocol distributions. To achieve this goal, we utilize a two-phase approach. Initially, we cluster protocol messages using heuristic rules and integrate the constructed feature data. Building on this, we perform mapping transformations on the processed message data, analyze the differences in feature distribution between the target protocol and multiple standardized protocols, and implement adaptive source domain selection for proprietary protocols based on these inter-domain differences.

To tackle the second challenge, we leverage the powerful representational learning of deep networks to facilitate the adaptation between standardized and proprietary protocols. Traditional learning methods rely on labeled data with proportional partitioning for training and evaluation, while proprietary protocols lack prior knowledge, making transfer adaptation challenging. Therefore, we have designed a dual-stream transfer learning structure, including shared layers and transfer layers, with the base functions being fully connected neural networks standardized for their strong representational capabilities. To accelerate deep learning, in the shared portion, bottleneck layers with batch normalization are added to reduce the dimensions of the deep features of messages, facilitating the calculation of distances between standardized and proprietary protocols. For inter-protocol knowledge transfer, the transfer layer employs a deep subdomain transfer network method based on probabilities to flexibly carry out deep transfer of protocol patterns. During back-propagation, the learning loss for the supervised part is calculated using the syntactic labels of standardized protocols and the network's predicted labels, and then the transfer parameters are jointly optimized with the transfer loss calculated at the bottleneck layer. Moreover, to optimize model performance and prevent over-fitting, we have also set the training cut-off condition. Ultimately, this model learns message inference logic on standardized protocols to infer the proprietary protocol format.

We implement and evaluate TransRE against five state-of-the-art protocol reverse engineering tools on 12 widely used protocols. The experimental results demonstrate that TransRE identifies fields with a perfection score of 0.43, significantly outperforming state-of-the-art methods such as Netzob, Netplier, FieldHunter, BinaryInferno, and Nemesys, which achieve an average of 0.21, 0.29, 0.14, 0.21, and 0.22, respectively. Additionally, we conducted ablation experiments on both source selection and syntax transfer modules, highlighting their effectiveness. Finally, to demonstrate the practical application of TransRE, we adapted it to enhance existing fuzz testing for proprietary protocols in four real-world network cameras, including two from Hikvision and two from Honeywell. Equipped with TransRE, the fuzzer successfully identified four vulnerabilities. Our main contributions are as follows:

- We propose the concept of protocol syntax transfer, first designing a source selection method to analyze inter-domain differences in order to match the optimal prior syntax knowledge domain for the protocol under test, and then using deep transfer learning to mine the syntax patterns of standardized protocols and infer proprietary protocol syntax.
- We implement TransRE and evaluate it on 12 protocols. The results show that TransRE outperforms state-of-the-art approaches.

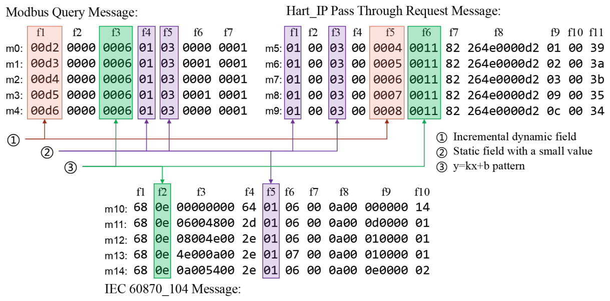
- We enhance the current fuzzing approach by applying TransRE to proprietary protocols of 4 real-world cameras, uncovering four vulnerabilities.

2 Motivation

In this section, we first show the limitations of traditional network traffic-based methods. Then, we show the basic ideas and challenges in our work.

2.1 Motivating Example

Traditional heuristic rule-based traffic analysis methods suffer from field pattern incompleteness and inflexibility.



```
Modbus Query Message:
     f1   f2   f3   f4 f5 f6   f7
m0: 00d2 0000 0006 01 03 0000 0001
m1: 00d3 0000 0006 01 03 0001 0001
m2: 00d4 0000 0006 01 03 0000 0001
m3: 00d5 0000 0006 01 03 0001 0001
m4: 00d6 0000 0006 01 03 0000 0001

Hart_IP Pass Through Request Message:
    f1 f2 f3 f4 f5   f6   f7 f8       f9 f10 f11
m5: 01 00 03 00 0004 0011 82 264e0000d2 01 00 39
m6: 01 00 03 00 0005 0011 82 264e0000d2 02 00 3a
m7: 01 00 03 00 0006 0011 82 264e0000d2 03 00 3b
m8: 01 00 03 00 0007 0011 82 264e0000d2 09 00 35
m9: 01 00 03 00 0008 0011 82 264e0000d2 0c 00 34

      f1 f2 f3       f4 f5 f6 f7 f8   f9     f10
m10: 68 0e 00000000 64 01 06 00 0a00 000000 14
m11: 68 0e 06004800 2d 01 06 00 0a00 0d0000 01
m12: 68 0e 08004e00 2e 01 06 00 0a00 010000 01
m13: 68 0e 4e000a00 2e 01 07 00 0a00 010000 01
m14: 68 0e 0a005400 2e 01 06 00 0a00 0e0000 02
IEC 60870_104 Message:
```

①
②
③
① Incremental dynamic field
② Static field with a small value
③ y=kx+b pattern

Fig. 1. Example of common pattern between protocols

Pattern Incompleteness. Traditional methods employ limited fixed patterns or heuristic rules to identify protocol formats. These methods may fail to cover all protocol field patterns or automatically adapt to the evolution of protocols, especially when dealing with proprietary or dynamically changing protocols. Over time, protocols are modified and extended in response to changing requirements, with newly added fields potentially not adhering to preset rules, or the meanings of existing fields changing. For instance, Netplier uses four heuristic rules to identify key fields, while BinaryInferno has five built-in fixed pattern recognition typical fields. Unsupervised learning-based approaches rely on large amounts of protocol data to learn the characteristics of proprietary protocols themselves to obtain protocol type or protocol syntax information [19, 36]. However, this approach struggles to handle noisy data or complex protocol changes and may inaccurately capture field patterns. Existing protocols exhibit a wide range of field patterns, making the exploration of predefined patterns between these fields complex and labor-intensive. For instance, Figure 1 shows the syntax of Modbus Query messages [41], Hart_IP Pass Through Request Message [9], and IEC 60870_104 Message slices and syntax [34]. The f1 field in the Modbus message and the f5 in the Hart_IP Message can be accurately identified with simple and obvious patterns. Still, the inherent mode of Modbus protocol's f3, f4, and f5 fields is cryptic and difficult to capture. Consequently, heuristic rule-based methods cannot account for all field patterns.

Pattern Inflexibility. The fixed rules of traditional methods can be based on the length, position, value range of protocol fields, and even the behavior characteristics of network traffic. For example, the sequence of commands for the RTSP protocol (e.g., SETUP, PLAY, PAUSE) is important. The server's response to these commands depends heavily on the order in which they are received. These methods work well for recognizing known protocols, but their accuracy and adaptability are limited when faced with varying or unknown protocols. Because different protocols have different field patterns, these tools usually rely on preset and fixed patterns to identify and analyze data packets, which lack flexibility, resulting in limited benefits of field segmentation. For example, because the timestamps used by the TCP to enhance the performance and reliability of network communications do not conform to standard time formats like NTP or Unix timestamps, they cannot be recognized using the custom timestamp detector from BinaryInferno [8]. Moreover, some heuristic-based methods employ threshold value to recognize fields, which can increase identification errors when dealing with proprietary protocols, as these thresholds are not applicable in all situations. Although all three protocols include the $y = kx + b$ mode in Figure 1, there are differences among the three modes. In Hart_IP and IEC 60870_104, 'x' represents the length of the subsequent field containing the field itself, while Modbus represents the sequence length after the field. FieldHunter uses a fixed threshold value to adjust for the linear correlation between the field value and the actual message size, which cannot accurately identify its field.

2.2 Basic Idea of TransRE

The basic idea of TransRE comes from inter-protocol shared foundation and the principle of transfer learning.

Shared Foundation. The design and implementation of new protocols usually follow the requirements of functional applications and are based on the specifications and knowledge of existing protocols. This shared basis leads to the emergence of common patterns or comparable structures across various protocols. For example, the Length field in the Modbus protocol follows a linear relationship between message length and field value; this pattern also appears in AMQP and IEC104 protocols [35]. Considering the similar functional requirements or industry standards, protocols in the same application scenario usually exhibit more similar syntax patterns. For example, in ICS, protocols from different vendors may all need to support functions such as device communication, status monitoring, and command transmission. As a result, there may be protocols with similar message formats for data exchange or device communication. There are rich common patterns between industrial protocols, which can be transferred to the format inference task of unknown protocols as prior knowledge. Figure 1 provides three potential common syntax patterns among Modbus [41], Hart_IP protocol [9], and IEC 60870_104 [34] by different colors, involving incremental dynamic pattern (①), static fields with a small value (②), and y=kx+b pattern for identifying length fields (③). Besides, there are some more hidden features, e.g., the f6 field of the Modbus and the f4 and f6 fields of the IEC 60870_104 change slightly.

Transferability. The core advantage of transfer learning lies in its ability to leverage a large amount of labeled data or known structural information from the source task to overcome challenges such as insufficient data or high complexity in the target task. In protocol format inference, the source protocol may have clear field structures, data flow patterns, and communication rules. The target protocol, even if it is a proprietary one without formal specifications, may still share similar functional requirements or application scenarios with the source protocol. For instance, while the specific field definitions of a proprietary protocol may not be publicly available, they could align closely with other known protocols in terms of communication purposes, data transmission methods, etc. Thus, transfer learning can exploit these commonalities. Since different protocols employ different field patterns, the similarity in field patterns between protocols varies. For example, in Figure 1, the Modbus protocol and the Hart_IP protocol exhibit three field patterns, whereas the IEC104 does not include a fixed-step increment dynamic field pattern. From the perspective of transfer learning, should the source and target protocols exhibit a greater degree of similarity in field structure, the transfer learning model will yield more promising results in inferring the proprietary protocol.

2.3 Challenges

To leverage prior knowledge of protocols with specifications through deep transfer learning to improve the inference of message formats for proprietary protocols, we must address two challenges.

C.1 Inter-protocol Correlation. The same type of messages of protocol can compute distribution distances through mapping transformations, but evaluating inter-protocol correlation on cross-protocol messages provides an additional challenge.

The relevance between the source and target domains is crucial for effective transfer learning. A highly relevant source domain can significantly improve learning efficiency and model performance. Conversely, dissimilar source and target domains may result in poor transfer or negative outcomes. Achieving a standardized difference evaluation between different protocols, like length and weight, is not feasible, because messages of different protocols have various types and field patterns. Face numerous publicly available protocols, superficial choices are insufficient for determining effective transfer. Therefore, in the absence of specific protocol specifications, accurately assessing the differences between different protocols and selecting an appropriate source protocol for transfer learning to target protocols is a highly complex and challenging task.

C.2 Protocol Diversity. Designing a generalized transfer learning strategy capable of facilitating protocol format recovery across protocol diversity presents significant challenges.

In network security and data communication, it is very difficult to manually obtain syntax-labeled data for proprietary protocols. These areas often deal with diverse and evolving protocols, which may overturn existing protocol field pattern logic. Different protocols may have their unique data patterns and structures, with significant variations in their structure and semantics. The continuously evolving new protocols may introduce new field patterns that overturn existing protocol field pattern logic. Transfer learning can improve the model's performance in understanding proprietary protocols by introducing prior syntax knowledge from standardized protocols. However, standard deep learning architectures lack the flexibility to effectively handle such diverse data sources, making knowledge adaptation and transfer difficult. Modifying and optimizing deep learning networks to better handle standardized and proprietary protocols and capture key common features across various protocols remains a significant challenge.

3 System Design

Overview. Figure 2 provides TransRE framework, including preprocessing, source selection, and syntax transfer modules: after protocol preprocessing, combined with syntax labels generated by Tshark [44] to build a knowledge base. The source selection module selects the optimal source domain for the target protocol from the knowledge base using adaptive matching. Then, by combining feature extraction and domain adaptation techniques, the syntax transfer module optimizes the transfer model to infer the message format of proprietary and poorly documented protocols.

Converting and Labeling. In this phase, network traffic of both standardized and proprietary messages is collected to train, validate, and test the model. First, we pre-process the incoming network traffic to extract the target protocol packets and group them by protocol type. We then use Tshark [44] to traverse the various layers of the target packets and use a depth-first search to extract information from the target layer by deduplicating and sorting to obtain the field values and positions of the original packet bytes. At the same time, we extract the protocol messages and then convert the field values to decimal integers ranging from 0 to 255. Sequence labels are generated on the basis of field positions and syntax, with the last byte of a message field marked positive and all other bytes marked negative. A standardized protocol requires calling the entire process, while for proprietary protocols only the conversion from sequences to numeric values is performed.

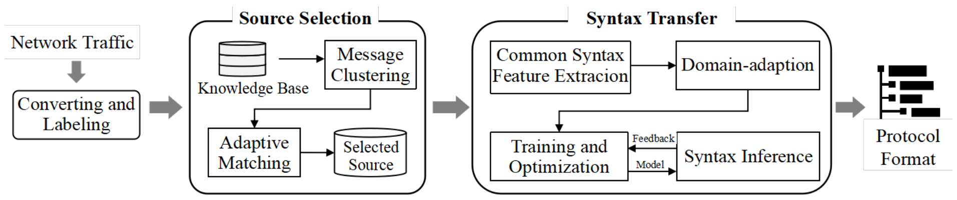


Fig. 2. TransRE Overview. The tool uses the source selection module to identify the optimal source protocol with message syntax for the proprietary protocol. It then uses a Neural Network to extract common syntax features. These features are processed by domain adaptation techniques to train and optimize the syntax transfer model, enabling inference of the proprietary protocol format.

Source Selection. To enhance the accuracy of transfer learning, this method extracts the high-dimensional features of protocol messages through the fully connected neural network according to the distribution of unknown protocol messages and various known protocols. It maps the protocol features into Hilbert space through transformation. Based on the assumptions of transfer learning, the smaller the difference between the source and target domain, the more knowledge can be transferred between them. Therefore, this study selects the protocol message set with the smallest domain difference from the unknown protocol as the training set.

Syntax Transfer. This module primarily implements the inference of proprietary protocol syntax using prior knowledge from known protocols. It has designed a deep transfer model for domain adaptation and transfer tasks between standardized and proprietary protocols, incorporating components such as a base network model, bottleneck layer, and transfer loss to effectively learn representations with good generalizability from the source domain to the target domain. The model trains on source and target domain data, striving to minimize a combination of transfer loss and model parameter optimization loss, adapting to various protocol data. The model is also regularly evaluated based on training conditions and employs early stopping to prevent overfitting, ultimately yielding a transfer model with known protocol knowledge to infer proprietary protocol format.

3.1 Source Selection

For protocol $\mathcal{P}_{pty}$ without prior knowledge, this module first determines similarity degree in data distribution between $\mathcal{P}_{pty}$ and standardized protocol $\mathcal{P}_{std} \in \{\mathcal{P}_{std}\}$, and selects the most similar standardized protocol data packet $\mathcal{P}^*_{std}$ to improve the accuracy of protocol reverse task for $\mathcal{P}_{pty}$.

Algorithm 1 provides an overview of the source selection module. This module first sets an empty list: $\{C\}$ (Line 2) to store the covariance results of multiple source domains. The covariance distance C between two data domains is obtained by the `CovCalculate` procedure (Lines 8-20) and when the source domain selection module calls `CovCalculate`, the covariance distance between the target domain and the candidate source domain is written into $\{C\}$.

To obtain matrix information of protocol messages in the data domain, by examining the publicly available Industrial Control Systems (ICS) protocol dataset[1], we found that 12 protocols consist entirely of homogeneous clusters, and other protocols all contain more than 75% of homogeneous clusters. Consequently, the `CovCalculate` algorithm first computes the message length data associated with both standardized and proprietary protocols, and then performs a clustering analysis based on this information. Given the label information generated by the protocol within the cluster, a transpose calculation is performed in conjunction with feature concatenation between clusters, thereby generating matrix information and decomposing it into feature vectors. The proprietary protocol is directly transposed to generate matrix information and decompose it into feature vectors.

Algorithm 1: Select Source

```
Input  : Standardized protocol sets $\{\mathcal{P}_{std}\}$
Input  : Proprietary protocol $\mathcal{P}_{pty}$
Output : Optimal source $\mathcal{S}^*$
Algorithm
    $\{ C \} \leftarrow \{\}$
    for $\mathcal{P}_{std}$ in $\{\mathcal{P}_{std}\}$ do
        $C$ = CovCalculate($\mathcal{P}_{std}, \mathcal{P}_{pty}$)
        $\{ C \} \leftarrow$ append($\{ C \}, C$)
    $\mathcal{S}^* \leftarrow \underset{S}{\text{argmin}}\, \{ C_S \mid C_S \in \{ C \} \}$
    return $\mathcal{S}^*$
Procedure CovCalculate($\mathcal{P}_A, \mathcal{P}_B$),
    $\mathcal{L}_A \leftarrow$ length of $\mathcal{P}_A$
    $\mathcal{L}_B \leftarrow$ length of $\mathcal{P}_B$
    $C \leftarrow \{\}$
    for $l_1 \leftarrow 1$ to $100$ do
        $n \leftarrow \min(\mathcal{L}_A, \mathcal{L}_B)$
        for $l_2 \leftarrow 1$ to $n$ do
            $D_s, c_s \leftarrow$ next ($\mathcal{P}_A$)
            $D_t, \_ \leftarrow$ next($\mathcal{P}_B$)
            zero_grad($opt$)
            $M_{cov} \leftarrow \mathcal{L}_{cov}(D_s, D_t, c_s)$
            $C \leftarrow$ append($C, M_{cov}$)
    return $C$
```

Finally, CovCalculate generates source domain data D_s with label c_s and target domain data D_t to calculate their covariance M_{cov}.

To achieve the domain difference calculation between standardized and proprietary protocol message data, calculate the covariance matrix using the obtained D_s and D_t combined with the identity matrix as shown in the formula 1, where D is the matrix after feature transformation in the data domain; n is the size of the data domain; E is the identity matrix.

$$C = \frac{1}{n-1}(D^T D - \frac{1}{n}(E^T D)^T (E^T D)) \tag{1}$$

$$\mathcal{L}_{cov} = \frac{||C_s - C_t||}{4d^2} \tag{2}$$

Then, the covariance matrices C_s and C_t of the source and target protocols, respectively, are used in the formula 2 to compute the data domain differences. Here, d represents the number of packets selected for protocol feature construction. The goal is to generate the optimal difference calculation result M_{cov} by gradient optimization. To reduce the generated error, perform 100 covariance computations in a loop and construct a covariance list $\{C\}$ to select the source protocols.

In the select source module, by calling the CovCalculate, it is feasible to calculate the inter-domain difference C between several standardized candidate protocols and the target protocol and construct a covariance list $\{C\}$. Traverse the position of each element and find the optimal element for the current position in each inter-domain difference list. If a list has the smallest value at that position, increase the count of that list. Finally, find the column with the highest counts of the smallest values and use that as a good source domain for the target protocol selection.

3.2 Syntax Transfer

Algorithm 2: Syntax Transfer

```
Input  : Standardized protocol message $\mathcal{P}_{std}$
Input  : Proprietary protocol message $\mathcal{P}_{pty}$
Input  : Label of source protocol $c_s$
Output : Proprietary protocol syntax $\mathcal{S}_{pty}$
Algorithm
    $D_s, D_t \leftarrow$ processing($\mathcal{P}_{std}, \mathcal{P}_{pty}$)
    TransNet $\leftarrow (N_{base}, N_{bottleneck}, N_{cls})$
    $M \leftarrow$ TransNet($D_s, D_t, c_s$)
    Procedure $M$.trian()
        $\mathcal{F} \leftarrow M.N_{base}(D_s, D_t, c_s)$
        $\hat{\mathcal{F}_s}, \hat{c} \leftarrow M.N_{cls}(\mathcal{F})$
        $\mathcal{L} \leftarrow$ criterion($\hat{\mathcal{F}_s}, c_s$)
        logits$_t \leftarrow$ softmax($\hat{c}$, dim=1)
        $w_{ss}, w_{tt}, w_{st} \leftarrow$ weights($c_s$, logits$_t$)
        kernels $\leftarrow$ gaussian_kernel($D_s, D_t$)
        $ss, tt, st \leftarrow$ kernels[:i,:i], kernels[i:,i:], kernels[:i, i:]
        $\mathcal{L}_{trans} \leftarrow \{ss, tt, st\} \times \{w_{ss}, w_{tt}, w_{st}\}$
        $\mathcal{L}_{tot} \leftarrow \mathcal{L} + \lambda * \mathcal{L}_{trans}$
    Procedure $M$.eval()
        $\mathcal{F}_{unk} \leftarrow M.N_{base}(D_t)$
        _, $\hat{c} \leftarrow M.N_{cls}(\mathcal{F}_{unk})$
        $\mathcal{S}_{pty} \leftarrow$ processing($D_t, \hat{c}$)
    return $\mathcal{S}_{pty}$
```

The source selection module selects optimal data sources from publicly accessible source protocols for the target proprietary protocol, providing prior syntactic knowledge for TransRE to understand proprietary protocols. For the proprietary protocol, the TransRE clusters the messages of the proprietary protocol, then conducts homogeneity tests within each cluster. If a cluster contains heterogeneous messages, the system will remove and re-cluster the messages. This process iterates until each cluster contains only homogeneous messages. After converting the bytes of the messages in the cluster to decimal integers, the base network of TransRE performs syntax feature extraction. Subsequently, the subdomain adaptation algorithm trains and optimizes the transfer learning model, utilizing the prior syntactic knowledge of standardized protocols to infer the message format of the proprietary protocol.

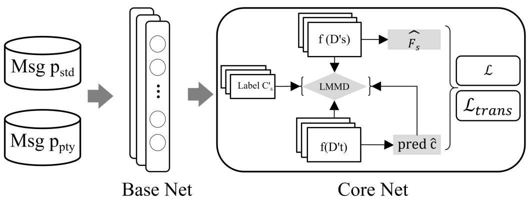


Fig. 3. TransNet's network structure and workflow, including two parts, the base network N_{base} and the core network involving local minimization of $\mathcal{L} + \lambda\mathcal{L}_{trans}$

The syntax transfer algorithm 2 is a deep transfer learning model TransNet. Figure 3 provides the network structure and workflow for TransRE model. TransNet consists of two parts (Line 3), the base network N_{base} and the core network involving bottleneck layer $N_{bottleneck}$ and classification layer N_{cls} to minimize inter-domain variability: the activations $f(D)$, the ground-truth label C_s, and the predicted label $\hat{C}$.

Base net N_{base} is a fully connected neural network. It extracts features from input message data and transforms them into higher dimensional syntax representations. N_{base} network architecture consists of linear layers and ReLU activation functions that convert the input vector into hidden features. A ReLU activation function follows this layer to introduce non-linearity. Finally, it outputs a higher dimensional representation for the protocol syntax, which helps TransNet to learn the predefine field patterns in the protocol with the message format.

The architecture of the transfer learning model includes a bottleneck layer $N_{bottleneck}$ designed to reduce the feature dimensionality. This helps to avoid overfitting and improves the ability of the model to transfer across domains. The $N_{bottleneck}$ consists of a linear transformation, a batch normalization (to stabilize learning), a ReLU activation function (to introduce nonlinearity), and a dropout layer (for regularization, configured to drop 50% of activations to prevent co-adaptation).

TransNet's N_{cls} classification layer is designed to map the extracted bottleneck features to the required number of output classes. This classifier includes two sequential linear transformation and ReLU activation blocks, with a dropout layer in between to enhance the generalization capability of the transfer model.

The characteristic of TransNet is its ability to handle domain adaptive of different protocol types. The loss of TransNet is divided into two parts: one part is the loss generated from training the model with standardized protocol, and the other part is the loss that evaluates the distribution difference between the feature representations of the standardized and proprietary protocols. The objective of TransNet is to optimize two complementary objective functions (Line 14): (1) minimize the training error $\mathcal{L}$ of the model, and (2) minimize the distribution difference $\mathcal{L}_{trans}$ between the standardized protocol and the proprietary protocol. The final optimization objective is:

$$\min_{M} \ \mathcal{L} + \lambda \mathcal{L}_{trans} \tag{3}$$

where λ is a positive regularization parameter that takes the value 0.01. The model learning error $\mathcal{L}$ (Line 8) of TransNet trained on the standardized protocol data is as follows:

$$\mathcal{L} = \frac{1}{m} \sum_{i=1}^{m} J(\Theta(D_s, c_s)) \tag{4}$$

where $\mathcal{J}(\cdot)$ is the cross-entropy loss function, $\Theta(\cdot)$ is the conditional probability that the TransNet assigns D_s to c_s, and m represents the size of dataset. Regarding the distribution divergence (Line 13), we adopted local maximum mean discrepancy (LMMD) [42], as shown in formula 5, which could compare different distributions based on the distance between standardized and proprietary protocol datasets in an RKHS.

$$\mathcal{L}_{trans} = \frac{1}{C} \sum \parallel \sum \omega_{sc} \Phi(D_s) - \sum \omega_{tc} \Phi(D_t) \parallel_{\mathcal{H}}^{2} \tag{5}$$

where $\omega.$ and $\Phi(\cdot)$ are the adaptive weight of syntax labels and the feature mapping function, respectively. During training, the network processes source and target data through the same pipeline, using cross-entropy to compute the loss between labels and predicted structures. Features of the target data are then used by the network to derive logical values for the target protocol. Weight parameters are obtained by combining packet data and syntax labels from standardized protocols, followed by the calculation of transfer loss using a Gaussian kernel. This determines the final learning loss of the transfer model. To optimize the transfer model, TransNet also uses a stochastic gradient descent (SGD) optimizer, setting different learning rates for different network layers to effectively balance feature extraction and domain adaptation during training.

When faced with proprietary or poorly documented protocols, TransNet's base network extracts high-dimensional features of the target protocol and further infers the syntax format of proprietary protocols using a model trained on prior syntax formats of standardized protocols.

Compared to traditional neural networks, the key difference is that TransNet adds a measure of the difference in data distribution between standardized protocol and proprietary protocol data domains after the fully connected layer. This measure of difference is included in the loss calculation during network training. In other words, when TransNet is trained with standardized protocol messages and their syntax-tagged vectors, TransNet also requires that the data distribution difference between the standardized protocol and the target protocol be minimized in the hidden representation of the base network.

3.3 Implementation

We implement a prototype of TransRE using Python 3 based on TensorFlow. It consists of two modules: the source selection module and the syntax transfer module. The source selection module implements the inter-domain difference analysis and the source domain selection algorithm. This module does not limit the number of candidate protocols, allowing manual assignment of candidate protocols based on expert analysis of proprietary protocols. It also supports manual inspection of protocol analysis results based on selections made by the source selection module. Based on the selected source domain, the syntax transfer module learns the protocol syntax knowledge and infers proprietary protocol syntax through domain difference analysis. Specifically, the knowledge transfer module is built on top of LMMD to provide inter-protocol difference analysis.

4 Evaluation

We implement and evaluate TransRE to answer the following four research questions.

- **RQ1** How does TransRE's performance compare to state-of-the-art tools?
- **RQ2** Is TransRE's source selection module effective?
- **RQ3** How does TransRE's transfer approach compare to other transfer techniques?
- **RQ4** How effective is TransRE in real-world application?

4.1 Experiment Setup

Subjects. Table 1 shows the 12 publicly available protocols[2] selected for evaluation. We selected them by referring to prior research [27, 50] and considering various characteristics. These protocols cover a variety of applications, including industrial automation and control protocols, power and utility industry protocols, proprietary protocols, IoT message delivery protocols, and publish/subscribe protocols. The diversity of these protocols demonstrates the versatility of our approach.

BACnet [46] and Lon [20] are primarily used in building automation, industrial control, and transportation systems, with the latter being a proprietary protocol developed by Echelon Corporation; DNP3 [16, 31] and IEC 104 [34] are used for remote monitoring, control, and data acquisition in power systems; S7comm [51] is a proprietary protocol from Siemens used for communication in S7 series PLCs (Programmable Logic Controllers), typically utilizing serial communication or Ethernet. HART_IP [5] is a variant of the HART protocol widely used in industrial automation

[2]The repository contains Pcap files related to Operational Technology (OT) and Information Technology (IT) protocols used in ICS. This dataset can be found at the following link: https://github.com/ICSDefense/ICS-Pcaps.

for data transmission between devices and sensors over IP networks. AMQP and MQTT [35] are open-source protocols: AMQP is used for message queuing middleware, primarily in financial and communication industries, while MQTT-QoS1 & MQTT-QoS2 are lightweight messaging protocols widely used for communication between IoT devices [43, 45, 56]. RTPS [2] is a real-time data distribution protocol based on the publish/subscribe model, mainly applied in distributed real-time systems such as autonomous driving and aerospace. The network communication protocol COTP [47] is used to establish reliable transmission connections, especially in German industrial control systems.

Table 1. Results of TransRE against five compared methods (bold indicates the best)

Protocol	**TransRE**			**BinaryInferno**			**Netzob**			**FieldHunter**			**Nemesys**			**Netplier**		
	Corr.F	Corr.B	Perf	Corr.F	Corr.B	Perf	Corr.F	Corr.B	Perf	Corr.F	Corr.B	Perf	Corr.F	Corr.B	Perf	Corr.F	Corr.B	Perf
Amqp	0.89	0.61	0.16	0.33	0.15	0.13	0.40	0.15	0.13	0.67	0.15	0.00	0.75	0.64	**0.49**	**0.96**	**0.85**	0.38
Bacnet	**1.00**	**0.99**	**0.75**	**1.00**	0.36	0.25	0.84	0.09	0.04	**1.00**	0.16	0.05	0.87	0.47	0.06	0.96	0.63	0.40
COTP	0.97	**0.48**	0.22	**1.00**	0.37	0.16	0.96	0.00	0.00	**1.00**	0.37	0.16	0.74	0.45	**0.24**	0.98	0.08	0.00
DNP3	0.84	0.61	0.23	0.52	0.37	0.23	0.97	0.61	**0.42**	0.61	0.23	0.00	0.48	0.33	0.19	**0.99**	**0.94**	0.37
HART_IP	**1.00**	**1.00**	**0.66**	0.69	0.31	0.09	0.85	0.51	0.21	**1.00**	0.38	0.28	0.85	0.54	0.24	0.92	0.77	0.41
IEC 104	**1.00**	**1.00**	**0.67**	0.90	0.58	0.48	**1.00**	0.67	0.34	**1.00**	0.32	0.26	0.71	0.46	0.24	0.99	0.74	0.38
Lon	**1.00**	**1.00**	**0.65**	0.67	0.35	0.30	0.87	0.41	0.17	**1.00**	0.06	0.06	0.98	0.50	0.13	0.94	0.65	0.50
Modbus	**1.00**	**1.00**	**0.36**	**1.00**	0.50	0.00	0.83	0.50	0.00	0.88	0.33	0.21	0.51	0.38	0.28	0.90	0.33	0.07
MQTT-QoS1	0.99	**0.94**	0.36	0.88	0.70	0.50	0.98	0.72	**0.61**	**1.00**	0.35	0.25	0.86	0.43	0.23	0.88	0.76	0.34
MQTT-QoS2	**0.99**	**0.96**	**0.35**	0.58	0.43	0.29	0.62	0.44	0.31	0.58	0.42	0.29	0.95	0.48	0.29	0.74	0.43	0.25
RTPS	0.94	0.62	**0.13**	**1.00**	0.00	0.00	**1.00**	0.01	0.00	**1.00**	0.00	0.00	0.67	0.44	0.08	**1.00**	**0.94**	0.11
S7comm	**1.00**	**1.00**	**0.64**	**1.00**	0.10	0.05	0.91	0.53	0.24	**1.00**	0.10	0.18	0.79	0.53	0.22	0.93	0.63	0.32
Average	**0.97**	**0.85**	**0.43**	0.80	0.35	0.21	0.85	0.39	0.21	0.89	0.24	0.14	0.76	0.47	0.22	0.93	0.65	0.29

Compared Tools. To verify our TransRE, we select five advanced protocol syntax inference tools as the control baseline, including Netzob [6], Netplier [50], FieldHunter [4], BinaryInferno [8] and Nemesys [24]. These publicly available methods are all network trace based reverse engineering tools, but the implementation principles are diverse. Netzob and Netplier represent the alignment-based approach, with different message clustering algorithms. The former employs a message similarity-based clustering, while the latter uses keyword-based clustering. Nemesys adopts a heuristic approach that identifies field boundaries based on the bit-level congruence of consecutive byte pairs. FieldHunter, a public re-implementation tool, identifies fields by leveraging statistical characteristics specific to general fields, such as host ID. BinaryInferno ensembles various detectors suitable for different field types and applies heuristics to identify field boundaries. We use the publicly available versions of these tools for comparison.

To evaluate the efficiency of the transfer module, we modularized two popular transfer learning methods: the Kernel Mean Matching method (KMM) [11, 15], the Correlation Alignment method (CORAL) [10] and SynRE [55]. The former improves learning performance by adjusting the distribution differences between the source and target domain samples, while the latter aims to achieve domain adaptation by minimizing the differences between the covariance matrices of the source and target domains.

Metrics. We evaluate the accuracy of the syntax inference result of the TransRE by whether each byte produces the correct syntax recognition result, involving three indicators: field correctness, boundary correctness, and Perfection. These evaluation metrics are employed in recent similar work [8, 27, 50]. We utilize Tshark [44] to obtain the syntax ground truth of the target protocol. Then, we compare each inferred field's boundaries and values with true fields.

Field correctness, named *correctness_F*, considers an inferred field as a correct one if the inferred field is part of a single true field or combines several consecutive true fields. The *correctness_F* is computed to measure the inferred formats, defined as follows:

$$Correctness\_F = \frac{Correct\ \ Field\ \ Number}{Total\ \ Inferred\ \ Field\ \ Number} \tag{6}$$

Boundary correctness, named *correctness_B*, considers an inferred field boundary as a correct one if the inferred field boundary is a true field or true sub-field boundary. The *correctness_B* is computed to measure the inferred protocol syntax, defined as follows:

$$Correctness\_B = \frac{Inferred\ True\ Field\ Boundary\ Number}{True\ Field\ Boundary\ Number} \tag{7}$$

Perfection is used to evaluate whether the inferred field matches the basic truth value exactly. Specifically, the field is *accurate* if it perfectly matches a true field. Therefore, perfection is the ratio of perfectly inferred field number to the total number of true fields. Their calculation is as follows:

$$Perfection = \frac{Perfectly\ Inferred\ Field\ Number}{Total\ True\ Field\ Number} \tag{8}$$

4.2 Effectiveness of TransRE

This section shows the syntax inference results of our TransRE against five control methods across different type protocols. We positioned the target method from three metrics *correctness_F*, *correctness_B*, and perfection.

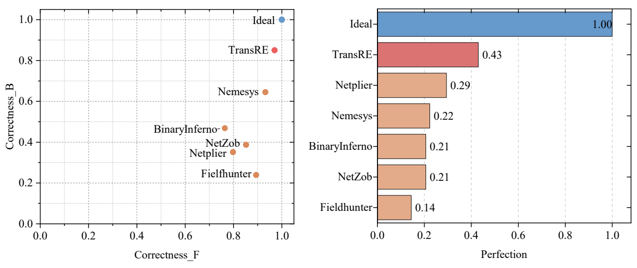


Fig. 4. Plot of average correctness_F, correctness_B, and perfection on 12 protocols for each tool. Results closer to ground truth (i.e., *Ideal*) are better.

Figure 4 plots the average performance for each tool on various protocols. TransRE substantially outperforms the other tools on field boundary division and perfect field extraction and achieves the upper bound on correctness_F, correctness_B, and perfection metrics across all protocols. On average, TransRE achieves a correctness_F of 0.97, a correctness_B of 0.85, and a perfection of 0.43 across the 12 protocol data, compared to (0.85, 0.39, 0.21) for Netzob, (0.93, 0.65, 0.29) for Netplier, (0.89, 0.24, 0.14) for FieldHunter, (0.80, 0.35, 0.21) for BinaryInferno, and (0.76, 0.47, 0.22) for Nemesys. Concerning the correctness_B, TransRE achieved 2.18×, 1.31×, 3.54×, 2.43×, and 1.81× relative to Netzob, Netplier, FieldHunter, BinaryInferno, and Nemesys, respectively. Notably, on the most important metric perfection, TransRE achieves 2.05×, 1.48×, 3.07×, 2.05×, and 1.95× of that achieved by Netzob, Netplier, FieldHunter, BinaryInferno, and Nemesys, respectively.

Table 1 presents the evaluation results of TransRE against five comparison tools across 12 protocols on correctness_F, correctness_B, and perfection metrics. TransRE demonstrates outstanding performance across multiple protocols. For correctness_F, TransRE correctly infers the message field for all seven protocols, where the inferred field is part of a single true field or combines several consecutive true fields, whereas BinaryInferno and Netplier achieve this for only four and one protocol, respectively. Although FieldHunter achieved a score of 1.00 on correctness_F, combining boundary inference and perfection metrics shows that most of its correct results are combinations of consecutive true fields. In contrast, TransRE excels not only in correctness_F but also performs very well in boundary inference and perfection. For instance, with the IEC_104 protocol, TransRE matched FieldHunter correctness_F score but achieved a perfection score of 0.67 in inferring perfect fields, compared to FieldHunter (0.26), outperforming it by 2.85×. Similar performance was noted with BacNET, HART_IP, and S7Comm protocols.

Regarding correctness_B, TransRE achieves an inference of field boundaries accuracy of over 94% for eight protocols, with a better correctness_B score of 1.00 for five protocols (i.e., HART_IP, IEC_104, Lon, Modbus, and S7comm). In comparison, only Netplier, a high-performance comparison tool, achieves 94% field boundary inference correctness for DNP3 and RTPS, with no other four tools reaching this level of performance. This indicates that our method for identifying field boundaries is highly accurate and outperforms five comparison tools, which is crucial for protocol understanding, error detection, and security analysis.

TransRE exhibits excellent performance across all protocols in perfection, achieving the best perfection score in field inference for eight protocols. Compared to the Netzob and Nemesys that demonstrated outstanding performance in two protocols, TransRE improved this metric mean by 105% and 207%, respectively. Considering that the Netplier (i.e., 0.29) with the highest perfection among the comparison tools, TransRE achieved better field inference results than Netplier in 10 projects. For example, S7comm, TransRE's perfection score is 0.64, which is double that of Netplier's 0.32. Similar phenomena are observed in other protocols as well. This demonstrates that TransRE has a significant advantage in extracting perfect fields from proprietary protocols, enabling it to more effectively identify and extract key field information.

The comprehensive analysis shows that TransRE performs excellently in most protocols, especially Bacnet, HART_IP, and IEC_104 protocols. Unlike tools that use heuristic rules to extract fixed patterns, TransRE leverages neural networks to deeply explore the field patterns of standardized protocol syntax, allowing it to learn embedded fixed models from the standardized protocols. This enhances its ability to understand the syntax of proprietary protocols and improves generalization for understanding them.

4.3 Accuracy of Source Selection

To evaluate the source selection module's accuracy, we ranked all 11 candidate source domains by their syntax inference performance (i.e., perfection) for each target protocol, then determined where TransRE's selected source domain ranked among these candidates, as shown in Table 2. Experimental results show that the source selection module can select relatively appropriate source domains in most cases. The source selection module selected the best source domains (ranked 1/11) for six target protocols, namely Bacnet, HART_IP, IEC_104, Lon, Modbus, and S7comm, and selected the suboptimal source domains for the other six target protocols. This indicates that the source selection module has high accuracy in selecting source domains.

Table 2. Ranking of TransRE's selected source domains among all candidates

Protocol	Amqp	Bacnet	COTP	DNP3	HART_IP	IEC 104	Lon	Modbus	MQTT-QoS1	MQTT-QoS2	RTPS	S7comm
Ranking	2/11	1/11	2/11	2/11	1/11	1/11	1/11	1/11	2/11	2/11	2/11	1/11

For the COTP protocol, which is a transport protocol, although it has a long message length, the data payload occupies most of it and contains relatively little field information. For example, a COTP protocol message has a length of 98 bytes, while the payload length is 95 bytes, leaving only 3 valid fields with three bytes. The syntax transfer module combines the entire message information to extract syntax knowledge elements from the message. Still, target protocol data payload interferes with knowledge migration, resulting in TransRE being unable to effectively learn and transfer the syntax knowledge of the source protocol during the transfer learning process. Similar message structures exist in the MQTT protocol and the content body messages of the AMQP protocol.

The characteristics of proprietary protocols have a significant impact on the source selection. The unsupervised source selection module evaluates the similarity between source protocols and the target protocol by analyzing and comparing the features of different source protocols. Factors present in proprietary protocols, such as the payload in COTP protocol messages, may affect the evaluation of source protocols, leading to inaccurate judgments about the similarity between the selected source protocol and the target protocol. The source selection module chooses the optimal and suboptimal source protocols for the target protocol, and this impact on the source selection module is negligible.

4.4 Validity of Syntax Transfer

To validate the transfer capability of TransRE, we implemented both KMM, CORAL, and SynRE transfer modules embedded into the Syntax Transfer module to evaluate our transfer efficiency. Table 3 shows the experimental results of KMM, CORAL, and SynRE transfer methods across 12 protocol data, along with the gap ratio of TransRE's transfer capability compared to the two transfer modules.

Table 3. Results of two transfer-learning techniques and TransRE's improvements

Protocol	**KMM**						**CORAL**						**SynRE**					
	Corr.F		Corr.B		Perf.		Corr.F		Corr.B		Perf.		Corr.F		Corr.B		Perf.	
Amqp	0.90	-1.47%	0.29	111.11%	0.10	66.67%	0.63	39.67%	0.23	171.43%	0.02	900.00%	0.70	26.06%	0.34	81.00%	0.15	9.04%
Bacnet	0.94	6.52%	0.77	28.94%	0.51	46.58%	0.93	7.14%	0.64	53.63%	0.39	92.07%	0.95	5.41%	0.34	192.12%	0.21	259.26%
COTP	1.00	-2.62%	0.68	-28.57%	0.13	75.00%	0.98	-1.05%	0.52	-6.25%	0.00	-	0.95	2.23%	0.34	12.90%	0.21	4.41%
DNP3	0.21	290.10%	0.08	683.13%	0.00	-	0.14	485.16%	0.08	683.13%	0.00	-	0.91	-8.40%	0.51	19.46%	0.18	22.28%
HART_IP	1.00	0.00%	0.30	233.33%	0.09	625.53%	0.83	20.00%	0.40	150.00%	0.18	262.77%	0.88	13.83%	0.60	66.48%	0.32	103.40%
IEC 104	0.62	62.50%	0.58	71.43%	0.38	73.33%	0.57	75.00%	0.25	300.00%	0.15	333.33%	0.95	5.09%	0.46	116.86%	0.12	436.38%
Lon	0.93	7.69%	0.29	246.15%	0.05	1074.73%	1.00	0.00%	0.00	-	0.00	-	0.99	0.727%	0.31	217.62%	0.15	345.02%
Modbus	1.00	0.00%	0.15	550.00%	0.00	-	1.00	0.00%	0.08	1200.00%	0.00	-	0.90	10.55%	0.44	125.83%	0.16	121.59%
MQTT-QoS1	0.99	0.44%	0.87	7.88%	0.45	-19.77%	0.45	122.54%	0.21	345.00%	0.03	1280.00%	0.72	38.27%	0.43	118.27%	0.25	45.34%
MQTT-QoS2	0.95	4.86%	0.62	54.48%	0.42	-18.07%	0.55	81.34%	0.20	378.14%	0.02	1743.48%	0.81	23.27%	0.47	105.39%	0.29	19.38%
RTPS	0.78	20.00%	0.32	94.55%	0.14	-8.00%	0.79	18.89%	0.36	72.58%	0.13	0.00%	0.76	23.29%	0.16	298.52%	0.00	-
S7comm	1.00	0.00%	0.39	156.25%	0.14	350.00%	1.00	0.00%	0.46	115.79%	0.19	237.50%	1.00	0.07%	0.58	72.56.43%	0.32	103.18%
Average	0.86	12.67% ↑	0.44	91.17% ↑	0.20	112.01% ↑	0.74	30.84% ↑	0.29	198.12% ↑	0.09	363.20% ↑	0.88	10.37%↑	0.41	105.16%↑	0.20	119.24%↑

Although the KMM has better transfer capability than the CORAL and is comparable to SynRE for most protocols, it is not as effective as our transfer method. Especially in the correctness_B and perfection metrics, the improvement of TransField is particularly significant. On average, the KMM outperforms the CORAL method in Correctness_F, Correctness_B, and Perfection, with average values of 0.86, 0.44, and 0.20, respectively, while the CORAL's values are 0.74, 0.29, and 0.09; the SynRE's values are 0.88, 0.41, and 0.20. TransRE were higher than KMM by (12.67%, 91.17%, and 112.01%), higher than CORAL by (30.84%, 198.12%, and 363.20%), higher than SynRE by (10.37%, 105.16%, and 119.24%) on correctness_F, correctness_B, and perfection.

For perfection, TransRE outperformed KMM in format inference for eight protocols, with a range of (46.58%-1074.73%). For the DNP3 and Modbus protocols, the KMM and CORAL did not infer perfect fields (i.e.,0.00), but TransRE achieved encouraging inference results (i.e.,0.23 and 0.36). For the MQTT1 and MQTT2 protocols, the KMM outperforms TransRE and CORAL in inferring perfect fields, with values of 0.45 and 0.42, respectively. TransRE (0.36 and 0.35) falls short of the KMM by 19.77% and 18.07%. The KMM uses kernel methods to weight samples to adjust and balance the distribution between the two domains, while TransRE directly measures and optimizes the similarity of sample distributions by calculating the mean differences of samples from the source and target domains in the feature space. For MQTT, some of its messages (i.e., publish messages) contain long payloads, which interfere with TransRE's optimization of data distribution similarity.

4.5 Real-World Fuzzing Application

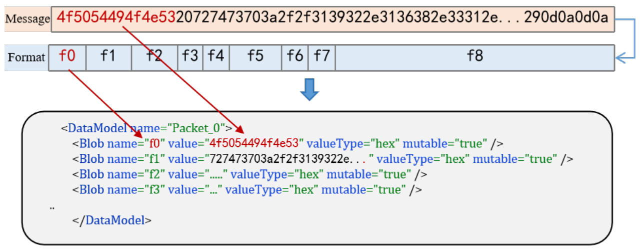
The protocol specification derived through protocol reverse engineering can be applied to security tasks, such as protocol fuzzing [28, 52] and model checking [21, 33]. In this section, we demonstrate how this model can enhance the fuzzing of proprietary protocols to uncover additional vulnerabilities, highlighting the real-world effectiveness of TransRE.

To further validate the effectiveness of TransRE in real-world application scenarios, we apply it to proprietary protocols that are widely used in commercial electronic devices. Due to the difficulty of obtaining official specifications and ground truth for proprietary protocols, we designed a fuzzing experiment specifically tailored for proprietary protocols, enabling us to assess the effectiveness of both TransRE and other available tools.

Considering the practical application in real-world cases, we selected both network cameras manufactured by Hikvision (DS-2SC3Q140MY-TE) and Honeywell (HVCD-4300I) as experimental subjects. These devices are widely used across industries such as security, IoT, autonomous driving, and intelligent transportation, making them ideal candidates to evaluate the effectiveness and generalizability of TransRE. As core devices for real-time data collection, network cameras predominantly rely on modified versions of multiple protocol prototypes (e.g., RTSP, RTP, RTCP, ONVIF, etc.) for video streaming, remote control, and device integration. This results in complex communication processes that are highly susceptible to security and privacy vulnerabilities. To thoroughly assess these aspects, we designed a three-step fuzzing evaluation process:

(1) *Experimental environment construction.* We configured both the Honeywell and Hikvision network cameras by setting their IP addresses, usernames, passwords, and other relevant parameters via the manufacturers' configuration tools. These cameras were then integrated into a network monitoring system, with all default settings retained. We verified that the devices successfully connected to the network and were capable of transmitting video streams.

(2) *Protocol reverse engineering.* In the configured environment, we first activated the cameras using the respective IPC tools from Honeywell and Hikvision. Next, we connected to the cameras' IP addresses via VLC to generate network traffic. Using Tshark, we captured the raw, protocol-agnostic packet payloads from the designated IP addresses during these operations. To ensure real-world applicability, we did not collect any additional network packets. We then applied Nemesys, BinaryInferno, Netzob, FieldHunter and TransRE to reverse engineer the payloads and infer packet formats [3].


Message
4f5054494f4e5320727473703a2f2f3139322e3136382e33312e. . . 290d0a0d0a
Format
f0
f1
f2
f3
f4
f5
f6
f7
f8

```
<DataModel name="Packet_0">
  <Blob name="f0" value="4f5054494f4e53" valueType="hex" mutable="true" />
  <Blob name="f1" value="727473703a2f2f3139322e. . . " valueType="hex" mutable="true" />
  <Blob name="f2" value="....." valueType="hex" mutable="true" />
  <Blob name="f3" value="..." valueType="hex" mutable="true" />
..
  </DataModel>
```

Fig. 5. Examples of data model transformation from message and its inferred format.

(3) *Application of the inferred formats.* To evaluate each method's effectiveness, we first converted each inferred packet format into a data model and then applied a uniform startup configuration to ensure compatibility with Peach Fuzzer [17]. Figure 5 provides an example of the conversion from

[3]Since Netplier categorizes traffic based on interaction information before inferring protocol formats, and the captured traffic from the cameras did not meet its requirements, it could not evaluate this real-world case.

messages and their inferred formats to data models. We used Peach Fuzzer to run fuzz tests on the IP cameras, monitoring for crashes, anomalous behavior, or vulnerabilities. For fairness, each experiment was repeated ten times, with each campaign lasting 24 hours.

Table 4. Vulnerabilities discovered by Peach when enhanced with different protocol reverse engineering tools

#	Device	TransRE	Nemesys	BinaryInferno	Netzob	FieldHunter
1	Honeywell HVCD-4300	✓	✗	✗	✓	✗
2	Hikvision DS-2SC3Q140MY-TE	✓	✗	✗	✓	✗
3	Honeywell HVCD-4300	✓	✗	✗	✗	✗
4	Hikvision DS-2SC3Q140MY-TE	✓	✗	✗	✗	✗

Following the described procedure, TransRE uncovered four vulnerabilities—two in Hikvision devices and two in Honeywell devices—while Netzob identified only two, and no vulnerabilities were discovered by any other tools, as shown in Table 4. After communicating with the vendors, it was confirmed that these vulnerabilities had also been found by other researchers and have since been addressed. The details of each vulnerability are as follows:

Vulnerabilities 1 and 2 (Sequence Field Overflow). TransRE discovered a sequence field that, when fuzzed beyond 1024, triggered a vulnerability. Exploiting this issue caused the network camera to freeze briefly before forcibly disconnecting, resulting in a denial-of-service (DoS) condition. Because no authentication is required and the flaw can be exploited remotely, it presents a heightened security risk. Both the Honeywell HVCD-4300I and Hikvision DS-2SC3Q140MY-TE devices were affected, and developers confirmed they were already aware of the issue. Identifying this vulnerability required isolating the specific sequence field; attempts to replicate it using the packet formats inferred by Nemesys, BinaryInferno, and FieldHunter did not succeed within 24 hours or over 10 trials.

Vulnerabilities 3 and 4 (Variable-Length Field Overflow). TransRE also identified a variable-length payload field. When Peach Fuzzer mutated this field to an abnormal length (4096), it caused the host machine to freeze and crash, preventing VLC from connecting to the video source. This, too, was confirmed as a DoS vulnerability, and its severity was amplified by the lack of authentication requirements and the possibility of remote exploitation. Both the Honeywell HVCD-4300I and Hikvision DS-2SC3Q140MY-TE devices were affected, and developers again acknowledged that it was a known issue. Reproducing these vulnerabilities required accurately pinpointing the variable-length field and mutating its size field; once more, other methods all failed to expose these flaws.

By confirming these vulnerabilities with the vendors, TransRE demonstrated its ability to accurately infer proprietary protocol formats, thereby making a valuable contribution to fuzzing initiatives focused on real-world proprietary protocols.

5 Discussion

Effectiveness by Payload. The core of TransField is a deep transfer learning network, which, like all learning techniques, is sensitive to noisy data. Some protocols may lack relevant documentation or specifications, leading to ambiguities in their specific structure, field types, and data formats. If a protocol contains a payload (i.e., the actual data content of the protocol), it may reduce the accuracy of TransRE's source domain selection module. This is because different payload structures and types can affect the features used by TransRE, thereby impacting the effectiveness of knowledge transfer, as seen in the case of the COTP protocol. One potential solution is to filter and process interference in the data payload by identifying message headers and data payload separators.

Encrypted/Compressed/Obfuscated Traffic Handling. Encrypted/compressed/obfuscated traffic is a common limitation for most network trace-based reverse engineering approaches [4, 6, 50].

Addressing these challenges requires handling issues orthogonal to protocol syntax recovery: cryptographic aspects for decryption, algorithm identification for decompression, and encoding scheme reversal for de-obfuscation. A possible solution is to use man-in-the-middle proxies (e.g., Charles [49]) to capture cleartext traffic.

6 Related Work

Heuristic Rule-Based Reverse Engineering [14, 25] infers protocol formats by employing fixed field patterns, where field patterns are derived from grouping, structuring, or summarizing based on message types, such as byte frequency, information entropy, linear/non-linear relationships, etc. For example, Netzob [6] applies pairwise alignment to derive field length, order, and inter-field relations. Netplier [50] integrates a probability-based keyword recognizer with clustering to identify static and dynamic fields. BinaryInferno [8] employs heuristic-based atomic detectors (e.g., float, timestamp, length) organized via a directed acyclic graph. SPRA [53] adopts a parallel workflow for keyword extraction and clustering. MDIplier [26] addresses the field information loss caused by global clustering by leveraging the hierarchical structure of protocol messages, performing iterative inference with message delimiter identification for layer separation. REInPro [38] determines the industrial protocol structure by controlling fields, and then infers the key semantics of the industrial protocol by distinguishing the fields characteristics. Different heuristic methods integrate different field patterns, which can recognize some typical fields, such as type or length fields, but their ability is limited in the face of new patterns.

Static Analysis-Based Reverse Engineering [24, 40] analyses standard protocol field to extract typical fields or relationship thresholds. Discoverer [13] tokenizes messages by simple semantics (textual or binary), clusters them, and applies statistical analysis (e.g., byte frequency, entropy) to infer field relations; Nemesys [24] detects field boundaries from value change distributions and proposes the Format Match Score, though limited in handling diverse protocols; and FieldHunter [4] leverages common field semantics (e.g., length, host ID) to identify fields via statistical correlations with byte values. These tools usually rely on preset and fixed field patterns to identify and analyze data packets, which lack flexibility, resulting in limited benefits of field segmentation.

Learning-Based Reverse Engineering employs deep learning to infer protocol types or protocol formats [19, 36]. SynRe [55] exploits accumulated syntax knowledge to infer new/extension protocol format. PREIUD [36] employs a BiLSTM-AM-CRF model that integrates unsupervised learning with deep neural networks to efficiently reverse most industrial control protocols. CNNPRE [19] applies CNNs to classify messages for protocol reverse engineering; DL-ProS^2 [54] utilizes LLMs to extract protocol specifications and simulate the traffic of different formats.

Dynamic Inference-Based Reverse Engineering employs dynamic interaction with the protocol server to infer protocol specifications. DynPRE [27] infers protocol message formats through dynamic server interaction, leveraging session identifier detection and adaptive message rewriting to craft exploratory requests for semantic extraction from server responses.

Main Difference: Unlike these approaches, our TransRE does not limited to the data size of the target protocol. It leverages deep transfer learning to extract prior knowledge of standardized protocols to break through the incompleteness and inflexibility of inherent heuristic rules for traditional protocol reverse engineering approaches. TransRE extracts field patterns and deep understanding of protocol syntax from standardized protocols and adapts them to the characteristics of the target protocol message structure, thereby improving the accuracy of proprietary protocol format inference. In addition, TransRE enhances its generalization ability on proprietary protocols by assessing the relevance between proprietary protocols and standardized protocols to match the optimal source domain, allowing it to more flexibly adapt to new field patterns within the protocols. TransRE therefore can dynamically adapt to different protocol characteristics, demonstrating greater adaptability when facing diverse protocol field patterns. It can also leverage existing knowledge for reasoning, thereby overcoming the limitations imposed by fixed field patterns.

7 Conclusion

This paper presents TransRE, a protocol reverse engineering tool that introduces the prior syntax knowledge of standardized protocols to achieve more accurate analysis for understanding proprietary protocols. Rather than relying on fixed rules to recognize protocol formats, TransRE integrates an automated method that intelligently extracts field patterns based on the characteristics of proprietary protocols by incorporating prior syntax knowledge from standardized protocols, thereby improving the accuracy of understanding proprietary protocols. Our experiments show that TransRE substantially outperforms five state-of-the-art protocol reverse engineering tools in protocol syntax inference across 12 widely used protocols. Furthermore, to demonstrate the practical usage of TransRE, we adapted it to enhance existing fuzzing for proprietary protocols of network cameras and successfully detected four vulnerabilities.

References

[1] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. 2015. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials* 17, 4 (2015), 2347–2376.

[2] Basem Almadani. 2005. RTPS middleware for real-time distributed industrial vision systems. In *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)*. IEEE, 361–364.

[3] Tiffany Bao, Ruoyu Wang, Yan Shoshitaishvili, and David Brumley. 2017. Your Exploit is Mine: Automatic Shellcode Transplant for Remote Exploits. *2017 IEEE Symposium on Security and Privacy (SP)* (2017), 824–839.

[4] Ignacio Bermudez, Alok Tongaonkar, Marios Iliofotou, Marco Mellia, and Maurizio Matteo Munafò. 2015. Automatic protocol field inference for deeper protocol understanding. *2015 IFIP Networking Conference (IFIP Networking)* (2015), 1–9.

[5] Anish Uday Bhurke and Faruk Kazi. 2021. Methods of Formal Analysis for ICS Protocols and HART-IP CPN modelling. In *2021 Asian Conference on Innovation in Technology (ASIANCON)*. IEEE, 1–7.

[6] Georges Bossert, Frédéric Guihéry, and Guillaume Hiet. 2014. Towards automated protocol reverse engineering using semantic information. *Proceedings of the 9th ACM symposium on Information, computer and communications security* (2014).

[7] Juan Caballero, Pongsin Poosankam, Christian Kreibich, and Dawn Xiaodong Song. 2009. Dispatcher: enabling active botnet infiltration using automatic protocol reverse-engineering. In *Conference on Computer and Communications Security*.

[8] Jared Chandler, Adam Wick, and Kathleen Fisher. 2023. BinaryInferno: A Semantic-Driven Approach to Field Inference for Binary Message Formats. *Proceedings 2023 Network and Distributed System Security Symposium* (2023).

[9] Kaustubh Chavan and Soham Methul. 2024. Comparative Analysis of HART and MODBUS Communication protocols. In *Communication Protocols*. https://doi.org/10.15680/IJIRCCE.2024.1203179

[10] Chao Chen, Zhihang Fu, Zhihong Chen, Sheng Jin, Zhaowei Cheng, Xinyu Jin, and Xian-Sheng Hua. 2020. Homm: Higher-order moment matching for unsupervised domain adaptation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 3422–3429.

[11] Miao Cheng and Xinge You. 2021. Adaptive matching of kernel means. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2498–2505.

[12] Paolo Milani Comparetti, Gilbert Wondracek, Christopher Krügel, and Engin Kirda. 2009. Prospex: Protocol Specification Extraction. *2009 30th IEEE Symposium on Security and Privacy* (2009), 110–125.

[13] Weidong Cui, Jayanthkumar Kannan, and Helen J. Wang. 2007. Discoverer: Automatic Protocol Reverse Engineering from Network Traces. In *USENIX Security Symposium*.

[14] Weidong Cui, Jayanthkumar Kannan, and Helen J. Wang. 2007. Discoverer: Automatic Protocol Reverse Engineering from Network Traces. In *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, August 6-10, 2007*, Niels Provos (Ed.). USENIX Association. https://www.usenix.org/conference/16th-usenix-security-symposium/discoverer-automatic-protocol-reverse-engineering-network

[15] Lixin Duan, Ivor W Tsang, and Dong Xu. 2012. Domain transfer multiple kernel learning. *IEEE transactions on pattern analysis and machine intelligence* 34, 3 (2012), 465–479.

[16] Samuel East, Jonathan Butts, Mauricio Papa, and Sujeet Shenoi. 2009. A Taxonomy of Attacks on the DNP3 Protocol. In *Critical Infrastructure Protection III: Third Annual IFIP WG 11.10 International Conference on Critical Infrastructure Protection, Hanover, New Hampshire, USA, March 23-25, 2009, Revised Selected Papers 3*. Springer, 67–81.

[17] Michael Eddington. 2023. *Peach Fuzzing Platform*. https://gitlab.com/gitlab-org/security-products/protocol-fuzzer-ce.

[18] Paul Fiterău-Broştean, Ramon Janssen, and Frits Vaandrager. 2016. Combining model learning and model checking to analyze TCP implementations. In *Computer Aided Verification: 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II 28*. Springer, 454–471.

[19] Javad Garshasbi and Mehdi Teimouri. 2023. CNNPRE: A CNN-Based Protocol Reverse Engineering Method. *IEEE Access* 11 (2023), 116255–116268. https://doi.org/10.1109/ACCESS.2023.3325391

[20] Steve Heath. 1992. Echelon-networking control. *IEE review* 38, 10 (1992), 363–367.

[21] Md. Endadul Hoque, Omar Chowdhury, Sze Yiu Chau, Cristina Nita-Rotaru, and Ninghui Li. 2017. Analyzing Operational Behavior of Stateful Protocol Implementations for Detecting Semantic Bugs. *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2017).

[22] Jiayi Jiang, Xiyuan Zhang, Chengcheng Wan, Haoyi Chen, Haiying Sun, and Ting Su. 2024. BinPRE: Enhancing Field Inference in Binary Analysis Based Protocol Reverse Engineering. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security* (Salt Lake City, UT, USA) *(CCS '24)*. Association for Computing Machinery, New York, NY, USA, 3689–3703. https://doi.org/10.1145/3658644.3690299

[23] jtpereyda. 2023. *BooFuzz: Network Protocol Fuzzing for Humans*. https://github.com/jtpereyda/boofuzz.

[24] Stephan Kleber, Henning Kopp, and Frank Kargl. 2018. NEMESYS: Network Message Syntax Reverse Engineering by Analysis of the Intrinsic Structure of Individual Messages. In *WOOT @ USENIX Security Symposium*.

[25] Stephan Kleber, Lisa Maile, and Frank Kargl. 2018. Survey of protocol reverse engineering algorithms: Decomposition of tools for static traffic analysis. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 526–561.

[26] Kai Liang, Zhengxiong Luo, Yanyang Zhao, Wenlong Zhang, Ronghua Shi, Yu Jiang, Heyuan Shi, and Chao Hu. 2024. MDIplier: Protocol Format Recovery via Hierarchical Inference. In *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*. 547–557. https://doi.org/10.1109/ISSRE62328.2024.00058

[27] Zhengxiong Luo, Kai Liang, Yanyang Zhao, Feifan Wu, Junze Yu, Heyuan Shi, and Yu Jiang. 2024. DYNPRE: Protocol Reverse Engineering via Dynamic Inference. In *NDSS*.

[28] Zhengxiong Luo, Junze Yu, Qingpeng Du, Yanyang Zhao, Feifan Wu, Heyuan Shi, Wanli Chang, and Yu Jiang. 2024. Parallel Fuzzing of IoT Messaging Protocols Through Collaborative Packet Generation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43, 11 (2024), 3431–3442.

[29] Zhengxiong Luo, Junze Yu, Feilong Zuo, Jianzhong Liu, Yu Jiang, Ting Chen, Abhik Roychoudhury, and Jiaguang Sun. 2023. Bleem: Packet Sequence Oriented Fuzzing for Protocol Implementations. (2023).

[30] Zhengxiong Luo, Feilong Zuo, Yuheng Shen, Xun Jiao, Wanli Chang, and Yu Jiang. 2020. ICS Protocol Fuzzing: Coverage Guided Packet Crack and Generation. *ACM/IEEE Design Automation Conference (DAC)* (2020).

[31] Munir Majdalawieh, Francesco Parisi-Presicce, and Duminda Wijesekera. 2006. DNPSec: Distributed network protocol version 3 (DNP3) security framework. *Advances in Computer, Information, and Systems Sciences, and Engineering* 1 (2006), 227–234.

[32] Ron Marcovich, Orna Grumberg, and Gabi Nakibly. 2023. PISE: Protocol Inference using Symbolic Execution and Automata Learning. *Proceedings 2023 Workshop on Binary Analysis Research* (2023).

[33] Kenneth L. McMillan and Lenore D. Zuck. 2019. Formal specification and testing of QUIC. *ACM Special Interest Group on Data Communication* (2019).

[34] mz automation. 2023. *Official repository for libIEC61850, the open-source library for the IEC 61850 protocols*. https://github.com/mz-automation/libiec61850.git.

[35] Nitin Naik. 2017. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *2017 IEEE international systems engineering symposium (ISSE)*. IEEE, 1–7.

[36] Bowei Ning, Xuejun Zong, Kan He, and Lian Lian. 2023. PREIUD: An Industrial Control Protocols Reverse Engineering Tool Based on Unsupervised Learning and Deep Neural Network Methods. *Symmetry* 15, 3 (2023). https://doi.org/10.3390/sym15030706

[37] Maria Leonor Pacheco, Max von Hippel, Ben Weintraub, Dan Goldwasser, and Cristina Nita-Rotaru. 2022. Automated Attack Synthesis by Extracting Finite State Machines from Protocol Specification Documents. In *2022 IEEE Symposium on Security and Privacy (SP)*. 51–68.

[38] Zhen Qin, Zeyu Yang, Yangyang Geng, Xin Che, Tianyi Wang, Hengye Zhu, Peng Cheng, and Jiming Chen. 2024. Reverse Engineering Industrial Protocols Driven By Control Fields. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*. 2408–2417. https://doi.org/10.1109/INFOCOM52122.2024.10621405

[39] Pallavi Sethi and Smruti R Sarangi. 2017. Internet of things: architectures, protocols, and applications. *Journal of electrical and computer engineering* 2017, 1 (2017), 9324035.

[40] Qingkai Shi, Xiangzhe Xu, and Xiangyu Zhang. 2023. Extracting protocol format as state machine via controlled static loop analysis. In *32nd USENIX Security Symposium (USENIX Security 23)*. 7019–7036.

[41] stephane. 2023. *A Modbus library for Linux, Mac OS, FreeBSD and Windows.* https://github.com/stephane/libmodbus.git.

[42] Yiheng Sun, Tian Lu, Cong Wang, Yuan Li, Huaiyu Fu, Jingran Dong, and Yunjie Xu. 2022. Transboost: A boosting-tree kernel transfer learning algorithm for improving financial inclusion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 12181–12190.

[43] Martin Tappler, Bernhard K. Aichernig, and Roderick Bloem. 2017. Model-Based Testing IoT Communication via Active Automata Learning. *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)* (2017).

[44] TShark. 2023. TShark - Dump and analyze network traffic. https://www.wireshark.org/docs/man-pages/tshark.html.

[45] Qinying Wang, Shouling Ji, Yuan Tian, Xuhong Zhang, Binbin Zhao, Yu Kan, Zhaowei Lin, Changting Lin, Shuiguang Deng, Alex X. Liu, and Raheem A. Beyah. 2021. MPInspector: A Systematic and Automatic Approach for Evaluating the Security of IoT Messaging Protocols. In *USENIX Security Symposium*.

[46] Steffen Wendzel, Benjamin Kahler, and Thomas Rist. 2012. Covert channels and their prevention in building automation protocols: A prototype exemplified using BACnet. In *2012 IEEE International Conference on Green Computing and Communications*. IEEE, 731–736.

[47] Jonathan Adi Wirawan and Husni Fahmi. 2008. *Testing of COTP (Connection Oriented Transport Protocol) in ATN (Aeronautical Telecommunication Network)*. Ph. D. Dissertation. Swiss German University.

[48] Feifan Wu, Zhengxiong Luo, Yanyang Zhao, Qingpeng Du, Junze Yu, Ruikang Peng, Heyuan Shi, and Yu Jiang. 2024. Logos: Log guided fuzzing for protocol implementations. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 1720–1732.

[49] xk72. 2023. *Charles: Web Debugging Proxy.* https://www.charlesproxy.com/.

[50] Yapeng Ye, Zhuo Zhang, Fei Wang, X. Zhang, and Dongyan Xu. 2021. NetPlier: Probabilistic Network Protocol Reverse Engineering from Message Traces. *Proceedings 2021 Network and Distributed System Security Symposium* (2021).

[51] Yerseg. 2023. *Implementation of the S7comm protocol.* https://github.com/yerseg/s7comm_investigation.

[52] Junze Yu, Zhengxiong Luo, Fangshangyuan Xia, Yanyang Zhao, Heyuan Shi, and Yu Jiang. 2024. SPFuzz: Stateful path based parallel fuzzing for protocols in autonomous vehicles. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*. 1–6.

[53] Weiyao Zhang, Xuying Meng, and Yujun Zhang. 2022. Dual-track Protocol Reverse Analysis Based on Share Learning. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 51–60. https://doi.org/10.1109/INFOCOM48880.2022.9796964

[54] Sen Zhao, Shouguo Yang, Zhen Wang, Yongji Liu, Hongsong Zhu, and Limin Sun. 2024. Crafting Binary Protocol Reversing via Deep Learning With Knowledge-Driven Augmentation. *IEEE/ACM Transactions on Networking* 32, 6 (2024), 5399–5414. https://doi.org/10.1109/TNET.2024.3468350

[55] Yanyang Zhao, Zhengxiong Luo, Kai Liang, Feifan Wu, Wenlong Zhang, Heyuan Shi, and Yu Jiang. 2025. Protocol syntax recovery via knowledge transfer. *Computer Networks* 258 (2025), 111022. https://doi.org/10.1016/j.comnet.2024.111022

[56] Feilong Zuo, Zhengxiong Luo, Junze Yu, Ting Chen, Zichen Xu, Aiguo Cui, and Yu Jiang. 2022. Vulnerability Detection of ICS Protocols via Cross-State Fuzzing. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* (2022).