Computer Communications 182 (2022) 238–254

Contents lists available at ScienceDirect

Computer Communications

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

Protocol Reverse-Engineering Methods and Tools: A Survey

Yuyao Huang, Hui Shu [*], Fei Kang, Yan Guang

State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, 450000, China

ARTICLE INFO

Keywords:
Protocol reverse engineering
Communication security
Network analysis
Binary analysis

ABSTRACT

The widespread utilization of network protocols raises many security and privacy concerns. To address them, protocol reverse-engineering (PRE) has been broadly applied in diverse domains, such as network management, security validation, and software analysis, by mining protocol specifications. This paper surveys the existing PRE methods and tools, which are based on network trace (NetT) or execution trace (ExeT), according to features representation. The feature-based protocol classification is proposed for the first time in literature to describe and compare different tools more clearly from a new perspective and to inspire crossover approaches in future works. We analyze the rationale, genealogy, contributions, and properties of 74 representative PRE methods/tools developed since 2004. In addition, we extend the general process of the PRE from a feature perspective and provide a detailed evaluation of the well-known methods/tools. Finally, we highlight the open issues and future research directions.

1. Introduction

Online communications are becoming opaque, i.e., 87% of the web traffic was encrypted in 2019, which was only 53% in 2016 [1]. However, encryption does not always mean security. While some protocols protect privacy, a great portion of them supports unauthorized communications. Malware attacks, hidden within the encrypted traffic, constituted the majority of malicious attacks, 70%, in 2017 [2]. The need for incident response and threat intrusion detection promotes protocol reverse-engineering (PRE) to understand private protocol specifications. One of the main challenges, i.e., the reliance on manual analysis compounded by the rapid growth of various protocols, accelerates the development of automated PRE, which aims to infer protocol specification automatically without any (or extremely limited) a priori knowledge.

An overview of the organization of this survey is presented in Fig. 1. We first discuss the background of PRE and its related surveys. Section 2 begins with a summary description of protocol features. Section 3 classifies and describes the methods/tools according to the proposed feature-based PRE classification method. Section 4 discusses the common process and evaluation methods of PRE. Finally, Section 5 summarizes and introduces the future trends.

1.1. Background

PRE analyzes traffic or program traces data for protocol field (PF) segmentation and protocol finite state machines (PFSM) recovery. The core problems, including format extraction, field definition, inference and state machine recovery, correspond to the syntax, semantics, and timing of the protocols. Extracting the protocol state machine often relies on the inference of the protocol format, while the segmentation of the fields is a prerequisite of the semantics.

PRE analysis has different perspectives. From the control point of view, it can be classified into two groups: active and passive analyses. Active analysis directly controls and stimulates the system by constructing specific parameters, whereas passive analysis mainly relies on the observed data for calculation. From the input point of view, it can be divided into (i) execution trace (ExeT)-based inference, which analyzes the dynamic or static (symbolic) execution process of the network applications; and (ii) network trace (NetT)-based inference, which deals with the traffic data captured from the real environment.

NetT-based inference: The analysis based on NetT is easy to operate, time-sensitive, fast-running, and can get results quickly when there is a large number of message samples and protocol types. In some cases (e.g., non-generic architectures) where the program execution cannot be traced, NetT is the only method that can be used for analysis. The intercepted network streams are often analyzed by combining bioinformatics, statistical analysis, or data mining methods to perform clustering analysis on messages. Based on the similarity of format message values, the analysis obtains protocol syntax and semantic information and uses the timing relationships between messages to infer the PFSM. However, it is difficult to enumerate all possible protocol instances, and it lacks the capability for uncaptured samples due to the poor test coverage. The typical NetT-based PRE analysis methods include sequence comparison [3–5], n-gram [6–10], and hidden Markov models [11], which are platform-independent and simple to implement.

* Corresponding author.
E-mail address: shuhui123@126.com (H. Shu).

https://doi.org/10.1016/j.comcom.2021.11.009
Received 4 July 2021; Received in revised form 16 October 2021; Accepted 9 November 2021
Available online 26 November 2021
0140-3664/© 2021 Published by Elsevier B.V.

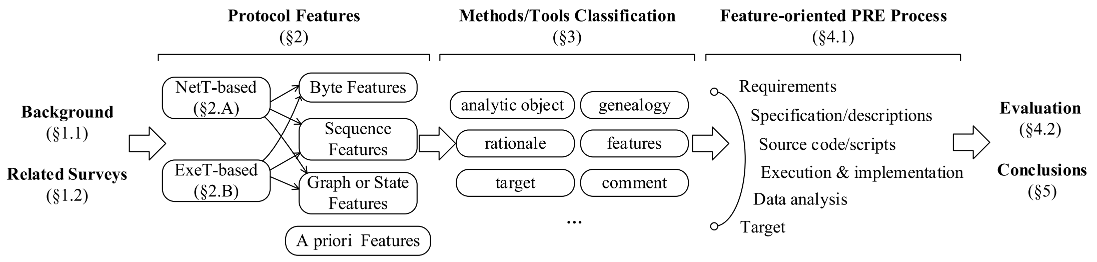


Fig. 1. Survey organization and overview.

Table 1
Comparisons of related surveys.

Surveys	Years	Contribution comparisons
Pan et al. [12]	2011	Classified and comparatively analyzed the existing PRE techniques from NetT/ExeT.
Zhang et al. [13]	2013	Improved work [12] in aspects of accurate definition, classification perspective, and protocol behavior specification mining.
Narayan et al. [14]	2015	Collected a total of 24 PRE tools categorized by PF/PFSM and proposed three indicators for evaluation of PRE tools.
Duchene et al. [15]	2018	Summarized a total of 19 PRE tools and provided the generic process of the PRE.
Sija et al. [16]	2018	Evaluated 39 approaches grouped into four categories according to their inputs, network, and execution traces.
Kleber et al. [17]	2018	Gave the state of art of the comprehensive algorithm analysis in the field of the static-traffic based PRE methods.
Wang et al. [18]	2020	Proposed a classification method based on the PRE output and divided tools into four categories.
(This paper)	2021	Presents the first protocol feature-based classification method for PRE covering 74 methods/tools, the first extended common process, as well as the first evaluation system for PRE.

However, these methods have the following inherent limitations: (i) the canonical language cannot be obtained with only positive examples. For the sample set of messages with all positive PFs, it is impossible to get accurate PFs by traffic analysis. (ii) Encryption and compression mechanisms may corrupt the value features of the message format. (iii) The effectiveness depends on the coverage of the sample set, and the PFSM cannot be completely recovered when the state does not exist.

ExeT-based inference: ExeT-based analysis captures information about the internal process of the binary executable and parses messages by program analysis techniques. It is especially necessary for the encrypted or compressed communication protocols, where the basic format of the traffic data is corrupted. This method uses the instruction execution traces in protocol processing and employs the dynamic taint analysis to track the data parsing process, thereby finding the context-based protocol specification and the way that the program parses the source. Since it is not limited by the completeness of the sample set, its accuracy is significantly higher than those of the NetT-based analysis techniques. However, the platform-specific instructions make it difficult to port, which requires access to protocol program executable and needs reverse-engineering expertise.

1.2. Related surveys and overview papers

There are various review studies on the PRE published so far, which summarize approaches, methods, and tools around different perspectives. Table 1 lists the contribution comparisons of these surveys.

Pan et al. [12] classified and comparatively analyzed the existing PRE techniques from the aspects of NetT and ExeT, besides providing the formal definitions, models, and requirements of the PRE. Zhang et al. [13] discussed the current technologies, evaluations, and application scenarios according to the two layers of specification scope, format, and semantic behavior mining in the packets. They improved this work towards three aspects in [12]: accurate definition, classification perspective, and protocol behavior specification mining. However, both of these studies lack the detailed classification of the PRE targets and main features. Narayan et al. [14] collected a total of 24 PRE tools and categorized them for different targets of PF or PFSM. Particularly, they proposed three independent indicators, namely correctness, conciseness, and coverage, to evaluate the performance of the PRE tools. Duchene et al. [15] summarized nine tools based on NetT and 17 tools based on ExeT. They also included a distinction between active and passive approaches and provided a brief overview of the generic steps of the PRE. However, the methods/tools described in this review were incomplete, i.e., missing some state-of-the-art studies. Sijia et al. [16] evaluated 39 approaches grouped into four categories according to their inputs, network, and execution traces. They further presented an in-depth discussion around manual-to-automated analysis, protocol categories (text, binary, hybrid, or other), and the seven layers of the OSI (Open System Interconnect) model corresponding to the protocols. Kleber et al. [17] decomposed the algorithms and solutions of the static traffic analysis tools, classified them in detail, and comparatively analyzed the underlying functional points. However, they neglected the ExeT-based methods/tools. Wang et al. [18] proposed a classification method based on the reverse engineering output of the protocol, divided the protocol reverse method into four categories, namely, focusing on protocol format extraction, focusing on protocol state machine inference, focusing on complete protocol specification description, focusing on other output results, and made analysis and comparison based on the above. But their work lacked a quantitative evaluation of existing PRE tools.

In this paper, we collect a wide range of research papers from various resources, such as SCI (Web of Science), IEEE/IET Electronic Library, and ACM Digital Library. The papers are selected based on two main criteria: whether they (i) present new techniques or theories on the PRE, or (ii) provide empirical data support. As a result, a total of 74 methods/tools (published since 2004) that meet the selection criteria have been selected. Fig. 2 is plotted based on the chronological hotness, where the horizontal axis shows the literature reviews on the PRE, while the vertical axis denotes the publication years of the methods/tools. The node size illustrates the number of methods/tools for that particular year. Fig. 2(a) reveals that the PRE methods/tools have been emerging in the last 15 years, with a greater impact since 2009. Fig. 2(b) identifies the tools of two different inputs, NetT and ExeT, shown in blue and green, respectively. There are more studies on NetT than ExeT, and the research trend of NetT is gradually increasing. However, a few new ExeT tools have emerged after they had a large impact during 2008–2010. The rightmost vertical columns of both subgraphs represent the contribution of this paper, which covers far more methods/tools than the similar studies in the literature.

This paper provides a comprehensive overview of the PRE research and summarizes the PRE techniques and workflows in terms of feature

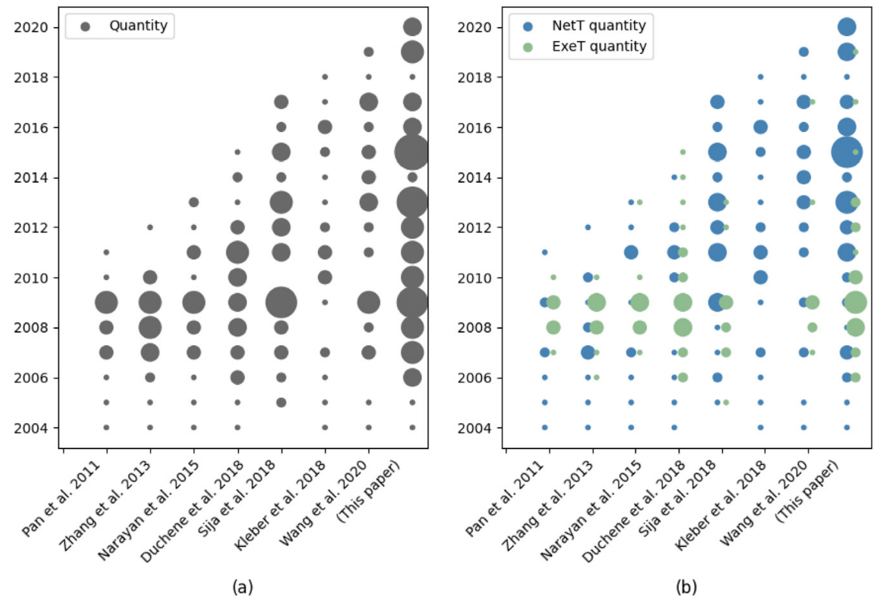


Fig. 2. Heat map of the methods/tools, (a) the number of the PRE methods/tools covered in each survey; and (b) category details on NetT and ExeT.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

data extraction. The existing works based on NetT or ExeT are classified and evaluated. The challenges and trends in this field are also highlighted. The main contributions of this paper are as follows:

(i) The most recent and comprehensive survey in the field of PRE, with 74 methods/tools covering all possible categories. A summary of tool genealogy relationships is also presented.

(ii) The first protocol feature-based classification method of the PRE methods/tools, describing the different principles of the tools in more detail and increasing the classification perspective compared to the previous methods based on the input alone.

(iii) The first introduction of protocol design and development process, further refining the discussion of the various stages of the PRE and giving many detailed statistical figures and tables.

(iv) The first evaluation system for the PRE methods/tools, providing an important reference for analysts to select tools according to their needs and also helping to improve the quality of PRE tool development.

2. Summary of protocol features

While the previous efforts often classify the PRE methods/tools based on their approaches or input–output types, this paper proposes a feature-oriented classification for the first time in the literature. We are motivated by the fact that protocols, as conventions for network data exchange, have relatively stable semi-structured features. Different protocols and properties (e.g., syntax, semantics, or timing) reflect different features, hence the intrinsic relationships vary. Besides, the increasing number of protocol data and the development of enabling technologies, such as automatic analysis and machine learning, have made it possible to utilize protocol features. Therefore, we classify the protocol features into three levels by granularity: (i) bit, byte, and single statistical value level; (ii) context and sequence level; and (iii) graph, model, or state level.

2.1. NetT-based features

Mining of a large number of sample features based on NetT methods facilitates the full use of techniques, such as multiple sequence comparison algorithms for intra-messages and sequence modeling for inter-messages, in inferring the PFs. The following types of feature representations are commonly extracted:

(i) Bit, byte, and single statistical value features: message byte frequency, mutation rate, the distance between any two packets, token patterns, related bits grouped into headers, the entropy of same type group messages, the number of one-byte character, keywords and their corresponding probabilities, support rates and variances of positions, repeatability, periodicity, frequency of gap, statistics of the reappearing frequency rate of a token in basic fields, information entropy, token format distance (TFD), message format distance (MFD), minimum support, and subsequence frequency.

(ii) Context and sequence features: message sequence alignment, the contextual information of sessions, n-grams, keyword sequences with maximum likelihood probability, intra-packet dependency of packet payloads, observation sequence, and contiguous sequential pattern.

(iii) Graph and distribution features: Markov process model, common substring graphs, the probability distribution of all feature sequences, and the distribution of value changes in the byte sequence.

Furthermore, since it is difficult to reconstruct accurate specifications based only on the above-given statistical features, some studies extract protocol a priori information as features, e.g., part of the protocol specification given by analysts, pre-defined characters for the text-based protocols, and manually assigned or rule-based features.

2.2. ExeT-based features

The methods using the above-summarized features for PF extraction have been widely studied in the literature. Some examples are bioinformatics-based sequence matching methods, statistical analysis-based n-gram methods, and probabilistic-based hidden semi-Markov model methods.

ExeT-based analysis achieves the identification of field semantic information and constraint relationships. This is based on the invocation of various memory data reflected by program instructions during protocol processing, which mainly contains the following types of features:

(i) Bit, byte, parameter, and statistical value features: parameters and return values of system calls, the number of times that the field is scanned by instructions, the percentages of arithmetic and bitwise operations, the names of accessed files, network parameters, and the frequency of each selected instruction.

(ii) Sequence, set, and other continuous value store features: function call stack frame, tainted bytes of protocol messages, instruction

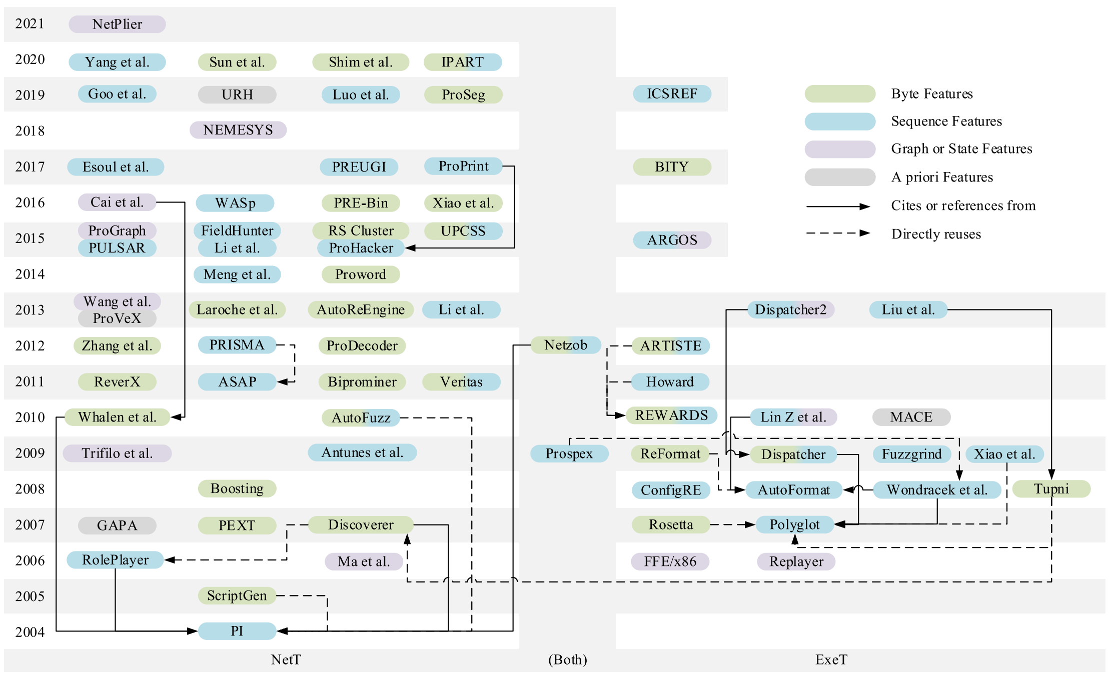


Fig. 3. Genealogy of the PRE methods/tools developed since 2004. Each node separated by rows represents a method or tool appearance in the named year with the color reflecting its main features. The arrows represent the citation relationships.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

addresses set, function call sequences, file operation sequences, executed instructions, the content of the operands, network-related system calls, memory access patterns, allocation log with information about allocation/deallocation operations, instruction dependency chain, and taint propagation relation of the API.

(iii) Graph and state features: control flow graph, protocol states satisfying certain constraints, dynamic control dependence graph, syntax trees, and the snapshots of a process state.

Since the ExeT-based analysis extracts static or runtime features that are closely related to the protocol handler, it retrieves more accurate message format, semantics, and other information. For example, the different percentages of arithmetic and bitwise instructions can tell the different stages of encryption and decryption, then divide the buffers for recording protocol plaintexts and cipher-texts. Another example is the taint propagation feature that expresses data and control dependencies of the protocol construction and the parsing process. Clear protocol processing can often be recovered in reverse by using program analysis techniques. However, we also observe that some studies do not adopt taint propagation to extract the field tree of the sent messages. In the construction of sent messages during message creation, only a small portion of potential tainted information sources, i.e., system calls and the output data, is used. On some occasions, it is observed that too many features may not always achieve high accuracy [19].

Fig. 3 illustrates the relationships and associated properties of the 74 PRE methods/tools considered in this paper in chronological order, while the overall statistics of the features are shown in Fig. 4. The following section will detail the PRE tools with their main principles according to the classification of the input types, highlighting the feature extraction methods and related attributes.

3. Methods and tools features classification

In this section, we first describe the PRE tools according to NetT-based inference analysis. Then, the same is done for and ExeT. Finally, we describe the features and rationale of the PRE methods/tools.

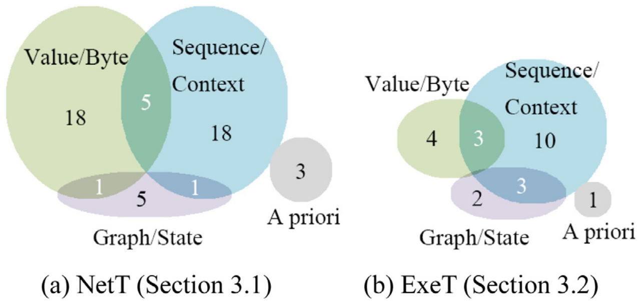


Fig. 4. Distributions of the 74 PRE tools features.

3.1. NetT-based feature representation

In the NetT-based inference analysis, clustering and statistical analysis methods are used to identify similar message field formats and boundaries, given the network capture of messages between hosts. Since there is no need to access protocol handling applications, this method is suitable for obtaining traffic information through intermediate devices, such as routers or gateways. For most protocol fields, each specific field carries its corresponding semantic information. Different fields present different data distributions depending on their semantics; hence, the amount of information carried by each field and the degree of variations between adjacent fields vary. The salient characteristic of the NetT technology is that it extracts the feature differences of the PFs, which can be used to divorce different fields, thereby inferring the state machine. These tools are listed in Tables 2–6 according to the classification of the different features.

Table 2 lists the traffic inference tools based on byte features. ScriptGen [20] calculates the basic properties of each byte based on the tool's PI [3] output. The byte sequences that satisfy the following conditions are treated as independent fields: (i) having the same types, (ii) having similar mutation rates, (iii) containing the same type of data, and (iv) having or not having gaps. PRE-Bin [33] and Xiao et al. [34]

Table 2
Traffic inference tools based on byte features.

Name	Year	Venue	Features	PF	PFSM
ScriptGen [20]	2005	ACSAC	Frequent type of data, frequent value, the mutation rate of the values, and the presence of gaps in that byte		✓
PEXT [21]	2007	WCRE	Distance between any two packets		✓
Discoverer [22]	2007	USENIX	Type of a token and token patterns	✓	
Boosting [23]	2008	NDSS	Simple byte-equality features corresponding to some window around fields	✓	
Whalen et al. [24]	2010	SecureComm	Related bits grouped into headers, the entropy of group messages of the same type	✓	
ReverX [25]	2011	WCRE	Byte-value variances	✓	✓
Biprominer [26]	2011	PDCAT	The statistical nature of protocol formats	✓	✓
ProDecoder [27]	2012	ICNP	Keywords and their corresponding probabilities	✓	
Zhang et al. [28]	2012	ICDMA	Length field constraints		✓
Laroche et al. [29]	2013	CEC	Protocol return message		✓
AutoReEngine [30]	2013	J. Netw. Comput. Appl.	Support rates and variances of positions	✓	✓
Proword [31]	2014	INFOCOM	Frequency and entropy	○	
RS Cluster [32]	2015	Mathematical Problems in Engineering	Message types	○	
PRE-Bin [33]	2016	IETC	Frequency of gap	✓	
Xiao et al. [34]	2016	IFS	Statistics of the reappearing frequency rate of token in basic fields	✓	
ProSeg [35]	2019	Comput. Commun.	Information entropy and mutual information	✓	
Sun et al. [36]	2020	Comput. Networks	Token format distance (TFD) and message format distance (MFD)	○	
Shim et al. [37]	2020	Int. J. Netw. Manag.	Minimum support	✓	

Note: "$\surd$" refers to complete support, and "○" refers to partial support.

also use frequency values as protocol features. The former calculates the gap frequency, while the latter divides the fields with the statistics of the reappearing frequency rate of tokens.

PEXT [21] finds the longest common subsequence (LCSS) to calculate the distance between any two packets. It employs agglomerate hierarchical clustering [38] to group packets into separate classes. Sun et al. [36], on the other hand, use token format distance (TFD) and message format distance (MFD) to define the distance between packets.

Discoverer [22] solves the main problem in the PI tools [3] by clustering messages of the same type and then determining the fields by comparing the contents of the clustered messages. This tool has three main stages: (i) tokenization and initial clustering, which decomposes longer messages into shorter tokens; (ii) recursive clustering, which forms clusters of message types and fields within each message; and (iii) format merging, which consolidates similar message types. Discoverer also identifies sequences of consecutive bytes, text or binary, in a message that may belong to the same message field. Although 90% of the reverse results in experiments conformed to the original protocol specification, the specification was still not completely covered due to the inherent poor capturing capability of the NetT-based method. Discoverer also uses a type of token and token patterns as features, similar to RS Cluster [32].

Boosting [23] utilizes a fixed-length sliding window to detect field boundaries, while Zhang et al. [28] focus on the message content associated with length semantic constraints. Many tools, such as Whalen et al. [24], Proword [31], and ProSeg [35], adopt message entropy for measuring protocol features.

ReverX [25] uses language recognition algorithms to identify delimiters, such as carriage returns and spaces in protocol messages, and finds protocol keywords in protocol messages by identifying byte sequence frequencies, which is very effective for text protocols, e.g., FTP. However, it relies on the delimiter identification of tokens, which makes it difficult to handle binary protocols with no distinct field boundaries. Compared to ReverX, Laroche et al. [29] can identify the existence of different protocol versions through meaningful interaction.

Biprominer [26] and ProDecoder [27] use statistical methods to find protocol keywords and possible keyword sequences. Biprominer utilizes variable-length pattern recognition to distinguish keywords, which is based on three stages: (i) identifying the statistically relevant patterns for a specific pattern length and recognizing them as keywords, (ii) defining messages via distinguishing keywords, (iii) calculating the transition probability between keywords to find the sequence of possible messages for the keywords. ProDecoder adopts the first two stages of Biprominer and then performs text alignment through clustering and Needleman–Wunsch algorithms [39]. Although PI uses Needleman–Wunsch algorithm, it can only recognize the protocol domain as constant domain or variable domain, and the specific semantics needs further analysis by hand. And ProDecoder adds the application of IB (information bottleneck) algorithm to the Needleman–Wunsch algorithm to cluster the similar messages, and achieves better results.

All preceding tools employ a left-to-right sequential format for protocols, but some protocol keywords are quoted from the end of the message, such as the SMTP mail protocol using '.' at the end to indicate termination. AutoReEngine [30] overcomes this problem by: (i) using data mining techniques to identify keywords in messages; (ii) vectorizing and sorting keywords; and (iii) using the positional variance referenced from the beginning and end of messages to classify vectors into message types. It uses a location-based approach to infer the message format as a sequence of the aligned keyword vectors. AutoReEngine achieves better accuracy compared to similar tools, such as Discoverer, by using message termination fields to identify protocols. Shim et al. [37] also support a similar feature (see Table 3).

Table 3 lists the traffic inference tools based on sequence features. PI [3] is a relatively old automatic protocol reverse analysis tool that relies on the Needleman–Wunsch algorithm [39] for long sequence pattern recognition in bioinformatics. It uses the TCPDump file as input and finds invariant keywords as well as variable-length fields in a message. The output is the mutation rate for a specific byte offset, where a mutation rate of 0% indicates the keyword data, and a mutation rate of 100% indicates the highly variable data. The main disadvantages of this tool are: (i) being useful only for messages containing similar byte sequences, while many protocols use different byte sequences within the same message format, (ii) being not suitable for variable-length packet messages. The other tools that use byte alignment features similar to PI are RolePlayer [40], Antunes et al. [4], and Meng et al. [5]

RolePlayer [40] uses extensions of byte-stream alignment algorithms from bioinformatics to compare different instances of a session to determine which fields it must change to successfully replay one side of the session. However, RolePlayer's understanding of network protocols is very limited—just the knowledge of a few low-level syntactic conventions, such as common representations of IP addresses and the use of length fields to specify the size of subsequent variable-length fields. Discoverer adopts its heuristics to identify the length and offset fields.

ASAP [41] identifies the components of network payloads by mapping them to a vector space and determines informative base directions

Table 3
Traffic inference tools based on sequence features.

Name	Year	Venue	Features	PF	PFSM
PI [3]	2004	–	Message sequence alignment	✓	
RolePlayer [40]	2006	NDSS	Low-level syntactic conventions, and contextual information of sessions	✓	
Antunes et al. [4]	2009	INFORUM	Message sequence alignment		✓
ASAP [41]	2011	ECML	An alphabet of relevant strings extracted from raw network payloads and used to map these payloads into a vector space	○	
PRISMA [42]	2012	AISec	Embedding with n-grams and tokens		✓
Li et al. [43]	2013	Journal of Software	Keyword sequences with maximum likelihood probability	✓	
Meng et al. [5]	2014	WARTIA	Message sequence alignment	✓	✓
FieldHunter [6]	2015	NETWORKING	Key–value singleton tokenizer for textual and n-gram tokenizer for binary protocols	✓	
PULSAR [44]	2015	SECURECOMM	String tokens separated by pre-defined characters for textual protocols and n-gram message for binary protocols	✓	✓
Li et al. [7]	2015	CIS	N-grams	✓	
ProHacker [45]	2015	ICNP	Relationship for n-grams of different orders		
WASp [46]	2016	WiSec	Repeatability, periodicity, monotonic increment or decrement, and constant field	✓	✓
ProPrint [8]	2017	JNCA	N-grams	○	
Esoul et al. [9]	2017	QRS	N-gram occurrences	✓	
PREUGI [47]	2017	IFS	A set of token sequences		✓
Goo et al. [48]	2019	ACCESS	Contiguous sequential pattern	✓	✓
Luo et al. [10]	2019	Sensors	N-grams	✓	✓
Yang et al. [49]	2020	SPDE	Specific field sequence coding method	✓	

Note: "$\surd$" refers to complete support, and "○" refers to partial support.

Table 4
Traffic inference tools based on byte or sequence features.

Name	Year	Venue	Features	PF	PFSM
AutoFuzz [51]	2010	IJCSNS	Extraction of generic message sequences, a representation of a message that separates static from variable data fields and associates variable data fields with type and length information	✓	✓
Veritas [52]	2011	ACNS	Expressed by a vector, with the ith (starting from 0) component counting the number of one-byte characters (values i) in that message.		✓
Netzob [53]	2012	AsiaCCS	Contextual information related to the application, like names of accessed files, network parameters, or system calls.	✓	✓
UPCSS [54]	2015	ICCC	Client-to-server number of packets and other 9 flow correlation features		
IPART [55]	2020	IJPEDS	Subsequence frequency and bytes variability	✓	

through matrix factorization techniques. It extracts an alphabet of relevant strings from the raw network payloads as vector features. The other tools that use strings as features similar to ASAP are PULSAR [44] and Goo et al. [48]

Many tools employ n-grams to characterize a certain feature, generally based on the observational assumption that for most of the field types, the n-grams in the field also show characteristics similar to the field. In other words, fields with different types change differently over specific sub-collections. The tools that use n-grams include PRISMA [42], FieldHunter [6], Li et al. [7], ProHacker [45], ProPrint [8], Esoul et al. [9], and Luo et al. [10]

Li et al. [43] first establishes a hidden semi-Markov model (HSMM) [50] for optimal segmentation and estimates the model parameters by using a sample set of message sequences transmitted by unknown application layer protocols during a network session. Then, it optimally segments the fields in the messages via the HSMM-based maximum likelihood probability segmentation method.

Unlike most tools that deal with application layer protocols, WASp [46] is designed specifically for unknown custom protocols over IEEE802.15.4. It identifies the meaningful byte positions, bounds their values, and narrows down the coverage for the efficient generation of spoofing packets.

PREUGI [47] uses an error-correcting grammatical inference method to automatically derive protocol state machines. After the combination of identical state tokens, negative example-driven revision, and reduction through similar semantics, it obtains a more generalized state machine of concise style.

By analyzing the change features of protocol fields, Yang et al. [49] propose a field sequence coding method for field change features. The main purpose of this coding method is to exclude the influence of field values on classification. Furthermore, it realizes division and type identification of unknown protocol fields based on the field classification model driven by deep learning (see Table 4).

Table 4 lists the traffic inference tools based on byte or sequence features. AutoFuzz [51] gathers similar messages and runs the algorithm described in the PI on each cluster. Then, it constructs the Generic Message Sequences (GMSs) for each cluster and associates them with the corresponding Finite State Automaton (FSA). As a result, it has the FSA of the protocol generated from a large sample of network traces.

PRISMA [42] uses ASAP's [41] statistical, test-driven, dimension reduction mechanism to tackle the feature space issue of the traffic data. It determines the message format and the state machine, besides the rules for propagating information between the states, by using machine learning techniques.

Veritas [52] extracts the features of each format message to group similar format messages and calculates the similarity between them. The features are expressed as a vector, with the ith (starting from 0) component representing the number of one-byte characters (values i) in that message. Finally, it uses the Partitioning Around Medoids (PAM) algorithm to group protocol format messages.

Netzob [53] subdivides action clusters by grouping messages with the same sequence of contextual data occurrences. Furthermore, it extends Needleman and Wunsch sequence alignment algorithms [39] and UPGMA [56] hierarchical clustering algorithms for format clustering.

Aiming at the problem of poor stability of the clustering results when labeling training samples, UPCSS [54] proposes a semi-supervised and learning-based unknown network protocol classification method. It performs sample label propagation, unknown protocol discovery, and protocol detailed classification, respectively.

IPART [55], an unsupervised tool, automatically reverses the format of the industrial protocol. It employs an extended voting expert algorithm to infer the boundaries of industrial protocol fields derived by statistical methods (see Table 5).

Table 5 lists the traffic inference tools based on graph or state features. To reduce the strict reliance on port mapping due to several

Table 5
Traffic inference tools based on graph or state features.

Name	Year	Venue	Features	PF	PFSM
Ma et al. [11]	2006	SIGCOMM	Different protocol models like product distributions, Markov processes, and common substring graphs		
Trifilo et al. [57]	2009	CISDA	The variance of the distribution of the variances (VDV) of each byte field in the protocol message	✓	✓
Wang et al. [58]	2013	TrustCom	The probability distribution of all the feature sequences, and the latent relationships among them	✓	
ProGraph [59]	2015	NETWORKING	Intrapacket dependency of packet payloads	✓	
Cai et al. [60]	2016	Mathematical Problems in Engineering	Observation sequence, where the emission probability matrix implies the relationship between observations and hidden states, and the state transition probability matrix implies the relationship of the field location	✓	
NEMESYS [61]	2018	USENIX	The distribution of value changes in the byte sequence	✓	
NetPlier [62]	2021	NDSS	A joint distribution of random variables and the observations of the messages	✓	✓

Table 6
Traffic inference tool based on a priori features.

Name	Year	Venue	Features	PF	PFSM
GAPA [63]	2007	NDSS	Protocol specification is given by the analyst	✓	✓
ProVeX [64]	2013	DIMVA	Previously inferred as probabilistic vectorized signatures		
URH [65]	2019	USENIX	Manually assigned or rule-based	✓	

factors, such as firewall port blocking and tunneling, dynamic port allocation, and a bloom of new distributed applications, Ma et al. [11] analyze three alternative methods using statistical and structural content models for automatic identification of traffic, which rely solely on the flow content. Those methods are the product distributions of byte offsets, Markov models of byte transitions, and common substring graphs of message strings. Some of the tools that use distributions as protocol features are Trifilo et al. [57], Wang et al. [58], and NEMESYS [61]. Also, Cai et al. [60] use Markov and ProGraph [59] graphical models as features.

ProGraph [59] characterizes intra-packet dependency by constructing a graphical model for a target protocol. It refines the graphical model with respect to statistical distributions of the values iteratively observed in different portions of payloads. It can operate on packet payloads at both byte- and bit-level granularities for protocol format inference.

Aiming at the lengths of protocol keywords and message fields, Cai et al. [60] present a hidden semi-Markov method to model the protocol message format by using a clustering technique based on an affinity propagation mechanism.

In NetPlier, a joint distribution is constructed among the random variables and the observations of the messages. Probabilistic inference is then performed to determine the most likely keyword field, which allows messages to be properly clustered by their true types and enables the recovery of message format and state machine (see Table 6).

Table 6 lists the traffic inference tool based on a priori features. GAPA [63] is a suite of protocol analyzers designed for Shield [66]. It conserves safety with a memory-safe language used for both textual and binary message parsing and analysis. Programmers can use a C-like interpreted language and embed code to specify how a certain field should be parsed based on the variables calculated according to the packet payloads.

PROVEX [64] is a system that automatically learns and matches the syntax of the C&C protocol messages using probabilistic vectorized signatures. It addresses the issue that many of the analyzed C&C protocols do not have invariant payload sequences (even in the plaintext C&C messages) or the invariant sequences are too short to be distinctive.

Similar to WASp [46], Universal Radio Hacker (URH) [65] is an open-source software suite with good scalability, which analyzes and attacks stateful wireless protocols. It reveals the protocol logic based on differential analysis and decoding bits, besides putting bits into context after mapping the received waves to the digital information.

3.2. ExeT-based feature representation

ExeT-based PRE techniques infer protocol specifications by analyzing executables. It is effective to perform static analysis if the program source codes are available. However, that rarely occurs in normal conditions; hence, reverse analysis must be performed for the protocol binary executable. Several PRE methods have been developed to extract protocol specifications from program execution. The general approach of these methods is that once an external message is inputted into the executable, the tool monitors the series of instructions invoked by the executable. By observing when the message handler executes a particular instruction and how the program accesses memory when receiving a protocol message, the field's format layout and message semantics can be inferred. ExeT-based analysis methods achieve higher simplicity and accuracy compared to NetT-based counterparts [19]. Table 7 shows the relevant tools and their features.

The File-Format Extractor for x86 (FFE/x86) [67] is the earliest static program analysis tool destined for the PRE, which works on the stripped executables. It constructs a Hierarchical Finite State Machine (HFSM) that over-approximates the output data format for annotating with Value-Set Analysis (VSA) [90,91] and Aggregate Structure Identification (ASI) [92].

Replayer [68] puts forward the first formal definition of the general replay problem. It tries modifying the previous input to reach the corresponding protocol state and handles protocol messages including specific fields. Replayer does not directly aim at extracting PFs or PFSMs but the protocol replay method.

Polyglot [69] and Dispatcher [75] collect more information about the message format of the target protocol by monitoring the execution of software binaries implementing the target protocol [93]. Polyglot executes the binary of the target protocol and captures instruction traces. For the incoming messages, it deduces the boundaries of fixed- and variable-length fields when identifying the keywords. Furthermore, it uses dynamic taint analysis techniques where the memory registers manipulating messages are marked as the source when an incoming protocol message arrives and is accessed by an executing instruction. For each executed instruction, it recognizes the boundaries of the tainted area and identifies: (i) the fields containing location information, inferring boundaries and information about other fields, (ii) the separators, and (iii) the fixed-length fields. Polyglot was later used by Xiao et al. [77]

Dispatcher [75] monitors the stack memory focusing on the sent message and assumes that it is a sequence of smaller message fields within the system memory buffer. It sets the memory containing the output message as sink points and recursively decomposes the memory buffer to obtain the initial source points, thereby revealing the format of the sent message. Dispatcher uses Rosetta [70] as the basis for semantic inference. Both tools have a high degree of correctness and simplicity but their message coverages are more limited due to the very single path of dynamic binary execution.

AutoFormat [71] handles different message fields at different running levels. It adopts dynamic taint analysis techniques to generate a hierarchical tree by feeding messages into a protocol binary executable and then monitoring the execution. AutoFormat groups the consecutive bytes running in the same context together and creates associations

Table 7
ExeT-based inference tools.

Name	Year	Venue	Features	PF	PFSM	Val/Byte	Sequence	Graph/State	A priori
FFE/x86 [67]	2006	WCRE	A reduced inter-procedural control-flow graph (hierarchical finite state machine)	○				✓	
Replayer [68]	2006	CCS	Protocol states satisfying certain constraints	○				✓	
Polyglot [69]	2007	CCS	Tainted register and memory location	✓			✓		
Rosetta [70]	2007	–	Parameters and return values of system calls and their dependencies to protocol fields			✓			
AutoFormat [71]	2008	NDSS	Function call stack frame	✓			✓		
Wondracek et al. [72]	2008	NDSS	Tainted bytes of protocol messages				✓		
ConfigRE [73]	2008	CCS	The propagations of the information flow tainted by individual fields				✓		
Tupni [74]	2008	CCS	Number of times the field was scanned by instructions	✓		✓			
Dispatcher [75]	2009	CCS	Executed instructions, the content of the operands, the associated taint information, and function's prototype with arguments	✓		✓	✓		
Prospex [76]	2009	SP	Instruction addresses set, function call sequence, file operation sequence	✓	✓		✓		
Xiao et al. [77]	2009	NSS	Similar to Polyglot [69]		✓		✓		
Fuzzgrind [78]	2009	SSTIC	Network related system calls				✓		
ReFormat [79]	2009	ESORICS	The percentages of arithmetic and bitwise operations	✓		✓			
MACE [80]	2010	CCS	User-provided input/output message abstraction function		✓				✓
Lin Z et al. [81]	2010	TSE	Dynamic control dependence graph, stack operations, and syntax trees	✓			✓	✓	
REWARDS [82]	2010	NDSS	(Same as Dispatcher [75]) And the in-memory layout of program data	✓		✓	✓		
Howard [83]	2011	NDSS	Memory access patterns, propagatable type sinks information				✓		
Netzob [53]	2012	AsiaCCS	Contextual information related to the application, e.g., names of the accessed files, network parameters, or system calls	✓	✓	✓	✓		
ARTISTE [84]	2012	–	Execution trace and the allocation log with primitive types stored in registers and memory	○		✓	✓		
Dispatcher2 [85]	2013	Comput. Networks	Executed instructions, allocation log with information on the allocation/deallocation operations, and the snapshots of the process states (memory and register contents)	✓			✓	✓	
Liu et al. [86]	2013	EIDWT	Instruction dependency chain of each buffer byte, and taint propagation relation of API	✓			✓		
ARGOS [87]	2015	RAID	Syscall and kernel API specifications with execution context. And reverse engineered data structure type graph	✓			✓	✓	
BITY [88]	2017	ICFEM	The frequency of each selected instruction and the extra useful information, e.g., memory size, is an argument of a function	✓		✓			
ICSREF [89]	2019	NDSS	Knowledge databases containing I/O operations and sequence of opcodes	✓			✓		

between fields at different levels by determining whether a message field inherits from a parent message field.

ReFormat [79] investigates the issues relating to the cipher fields in protocol messages and performs the dynamic analysis under two tasks: (i) identifying encryption and decryption instructions, and (ii) reverse-engineering the protocol syntax. By measuring the rate of arithmetic instructions to the total number of instructions within the complete execution trace, ReFormat divides the execution trace into two parts: (i) encryption or decryption, and (ii) parsing. After identifying the boundary, it determines the plaintext message buffer by finding the intersection of the memory buffers written in the decryption phase and read in the parsing phase. This type of reverse analysis task for encrypted messages can only be solved by ExeT-based methods.

Tupni [74] uses a taint analysis engine during program execution to detect loops and identify loop boundaries to locate a segment of records. After defining each record boundary, it clusters similar processing record processes together. Although Tupni attempts to infer semantics by associating system-level API calls with protocol fields, it is not implemented in the paper. Its novelty lies in the ability to aggregate multiple dynamic execution traces to improve the accuracy of the reversed specification.

ConfigRE [73] accommodates new techniques to detect semantic relationships between the configuration fields. Different from the traditional taint analysis, it studies the control flows among configuration fields to identify their semantic relations. Also, it analyzes how information flows interact with each other, revealing the key components of the policies. ConfigRE is tainted by two different sources, configuration files, and a service request.

Wondracek et al. [72] dynamically track the input of the tainted data and analyze the processing in detail, determining delimiters, length fields, file names, etc. On this basis, Prospex [76] is extended to the same type of message recognition and automata extracting.

Fuzzgrind [78] is not a PRE-specific tool but the authors mentioned the possibility of extending the approach to the network-related system calls, where the data constraints shaping the path execution reflect the message format.

While the aforementioned tools based on taint analysis focus on the top-down grammars, Lin et al. allow the reverse analysis of a specific set of protocols using the bottom-up grammars. They gather the sub-execution related to the stack to create the input syntax tree. The results revealed both the message format and the nature of the message hierarchy.

REWARDS [82] shares the same insights as Dispatcher [75] for type inference and semantics extraction. Besides, it extends to the general data structures in the program.

MACE [80] is similar to Prospex [76] but it is the first approach deducing the complete protocol state machines in a realistic, high latency network setting.

In Howard [83], memory access patterns constitute clues about the memory data layout. One of the array detection methods of Howard was influenced by that of Polyglot [69], which resulted in a completely new array detection technique.

Netzob [53] concludes that the signature of a successive system and the function calls can show a specific program action on a system. It uses sandboxes to capture network traffic in addition to any useful application-related contextual information, e.g., accessed file names, network parameters, and/or system calls. Netzob [53] is a relatively comprehensive and complete tool designed specifically for the PRE and is being updated regularly.

ARTISTE [84] is different from the previous dynamic techniques since it addresses the recovery process of the full data structure from the primitive field types to the recursive data structures.

J. Caballero et al. extend Polyglot [69] and Dispatcher [75] by taking executed instructions and allocation logs revealing the allocation/deallocation operations. They use snapshots of the process state (memory and register contents) as features, which leads to inferring protocol information more completely. Their method is also capable of analyzing the encrypted protocols.

Liu et al. [86] extend the use of system API calls to deduce protocol semantics. The provided tools monitor the execution traces and determine whether to write on the send socket or not. When the tool identifies the sending message in the memory buffer, it isolates all parts of the instruction trace, which generated the message. Then, it defines the field boundaries in the message from the byte groups inherited from the same segment of the instruction trace. They experimentally validated their method in terms of correctness by using two malicious programs, namely Agobot3 and SDBot, which were controlled via the IRC protocol.

Recent years have shown an increase in the research efforts targeting binary data structure recovery, which is generalized as a fundamental technique for the PRE. ARGOS [87] is the first system that focuses on the semantic reverse-engineering of data structures, making the first step towards the reverse-engineering of kernel data structures. Similar to existing network protocol reverse-engineering systems, ARGOS is also based on the data uses tell data types principle. BITY [88] learns the types of the recovered variables by using the related representative instructions. ICSREF [89], on the other hand, is a reverse-engineering framework for industrial binaries, which combines reverse analysis methods to build a priori knowledge database and performs automated analysis on the codesys binary files.

4. Discussion

In this section, we first give the general process description of feature-oriented PRE and extending the PRE steps in more detail. Then, we propose a more comprehensive PRE tool evaluation system and accordingly evaluate the methods/tools covered in this paper.

4.1. Feature-oriented common PRE process

Different from the existing literature, we extend the scope of the PRE—going back to the preceded protocol generation process containing specifications and conventions (see Fig. 5). Based on user requirements, the protocols are designed and developed to generate entity programs named client or server. The a priori features of the upstream process become the input of the PRE, along with the network or execution traces observed during the implementation phase. The analysis phase, which is the main focus of this paper, focus on modeling, inference, and reconstruction. The target output comes at the end of the PRE, and its coverage of protocol specifications, i.e., how well it reflects the original, is considered as one of the most important factors in evaluation.

Requirements: Regardless of targeting a completely structured development process, the initial phase of the protocol design should convert the designer's ideas into concrete requirements. Meanwhile, network protocols often need revisions and extensions due to changing requirements or the problems encountered during operation. Hence, the semantic outputs of the PRE should reflect the requirements describing the desired protocol behavior. The requirements outline the protocol's services and functions containing the purpose, constraints, assumptions, and rules, often using the natural language.

Specifications with outline descriptions: The analysis of the requirements leads to a specification, which is a formal statement revealing design details by using structural text or modeling language. Descriptions of the PFs or the PFSMs should satisfy the predefined logical properties of the protocol after validation.

Source code or scripts: Since the source code naturally retains the constants, variable names, data structures (including types), function prototypes, and other information of the developer intention, it reveals more semantic features compared to the compiled binary executable. At the same time, source code of the related protocols or the irrelevant ones but with the same capabilities tend to have a higher similarity, while for different functions may not. Particularly, the source code features seem to be more representative of the increase in the reuse of open-source protocols and codes. However, for most network protocols, such as commercially protected, dedicated private, and malware, the source code is often unavailable. We summarized 19 methods/tools in Table 8 revealing that the compiled ones are often programmed with C/C++ and the interpreted ones are with Python.

Execution & Implementation: The stripped protocol binary executable faces difficulties in the recovery due to exposing fewer semantic features compared to the source code. However, it is the only available data that completely reflects the protocol implementation when the source code is unreachable. Furthermore, the dynamic execution of the binary better reflects the accuracy of some specific protocol paths compared to the static analysis. Due to the limitations of compiling architecture, anti-analysis technique, uncooperative dependency environment, or lack of activation conditions, protocol binaries may not be properly or completely executed. Hence, some contexts like external input and random numbers, which are available only during the dynamic execution, are not obtainable. Sufficient features to recover semantics require the PRE support for guiding execution, allowing protocol details to be traversed and analyzed. Table 9 provides an overview of monitors and emulators used by the PRE methods/tools. Since the observer has access to neither the client nor the server, any details of the program execution cannot be revealed, which makes the network traffic the only information that is visible to the outside. The performance of the capturer (or probe) in the large complex networks, multi-layer forwarding networks, or multi-channel synchronous networks will affect the integrity of features, especially for the NetT inference. For modern telephone, telegraph, television, analog signal, and telecommunication networks, as well as IoT devices and other environments, the preparation of the capturer is crucial. Table 10 summarizes the capturers used by the PRE methods or tools.

Input Data: Involving more than just a binary executable but network traffic inputs distinguish PRE from the conventional reverse-engineering tools. Traffic and execution traces are respectively considered as the most important observed features to the inference of NetT and ExeT. Besides, some a priori knowledge of the protocol design process may contain several features if available.

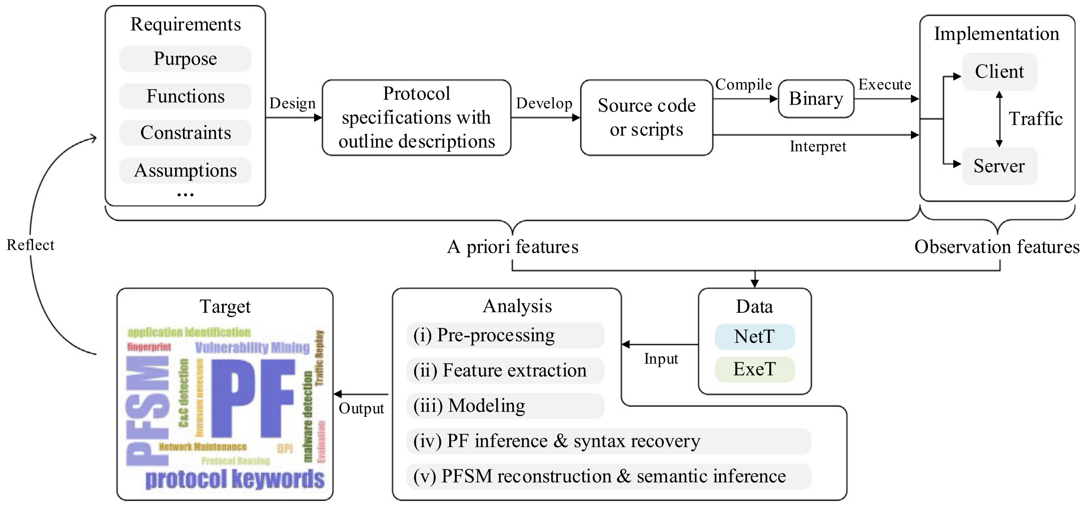


Fig. 5. Extended view of the PRE. The target word cloud reflects the statistics of the PRE output type.

Table 8
Overview of the developing languages used by the PRE methods or tools.

Type	Lang.	Num.	PRE methods or tools
Compiled	C	4	RolePlayer [40], GAPA [63], Netzob [53], ProVeX [64]
	C++	7	Replayer [68], ARTISTE [84], Ma et al. [11], GAPA, ProGraph [59], URH [65], Goo et al. [48]
	Java[a]	2	Antunes et al. [4], ReverX [25]
	Ocaml	1	ARTISTE
Interpreted	Python[a]	9	Fuzzgrind [78], PI [3], ScriptGen [20], Whalen et al. [24], Netzob [53], ProVeX [64], NEMESYS [61], URH [65], IPART [55]
	Ruby[a]	1	PEXT [21]

[a]The language can be executed as either a compiled program or an interpreted script in the interactive mode.

Table 9
Overview of the monitors/emulators used by the PRE methods or tools.

Monitor or emulator	Num.	PRE methods or tools
CodeSurfer/x86 [94]	1	FFE/x86 [67]
KLEE [95]	1	Howard [83]
Qemu [96]	3	Howard [83], Rosetta [70], ARGOS [87]
Panorama [97]	2	Polyglot [69], Xiao et al. [77]
Pin [98]	2	ConfigRE [73], ARGOS [87]
TaintCheck [99]	2	Dispatcher [75], ARTISTE [84]
Valgrind [100]	2	Replayer [68], Fuzzgrind [78]
Cuckoo [101]	1	Netzob [53]

Table 10
Overview of the capturers used by the PRE methods or tools.

Capturer	Num.	PRE methods or tools
Wireshark [102]/Ethereal[a]	15	ScriptGen [20], Boosting [23], Ma et al. [11], Netzob [53], Zhang et al. [28], Wang et al. [58], Li et al. [43], Meng et al. [5], Li et al. [7], RS Cluster [32], Luo et al. [10], ProSeg [35], Yang et al. [49], PRISMA [42], Xiao et al. [34]
tshark[b]	3	Cai et al. [60], Esoul et al. [9], NEMESYS [61]
tcpdump/libpcap [103]	8	Xiao et al. [77], Goo et al. [48], RolePlayer [40], PEXT [21], Trifilo et al. [57], ReverX [25], PRISMA [42], Xiao et al. [34]
Netmon [104]	1	Goo et al. [48]
Scapy [105]	1	Sun et al. [36]
BRO [106]	1	Ma et al. [11]

[a]Ethereal was the original name of the open-source Wireshark packet analysis software.
[b]Tshark (Terminal wire SHARK) is the command line tool (CLI) that has most, but not all, of the features of Wireshark.

Analysis:

(i) Pre-processing: This step cleans the input data and removes the noise. Furthermore, it layers, reorganizes, aggregates, and categorizes the traffic as necessary. Execution traces should be separated from application and kernel spaces or different processes/threads according to the logging methods (e.g., system-level virtualization, binary instrumentation). For hybrid analysis, the traffic data should be divided by the process or even by the program function.

(ii) Feature extraction: The way that the program processes the data, together with the data itself, reflects the protocol properties. Statistical and correlation analysis apply to these features to explore several parameters, such as span, direction, and density of streams, sending and receiving intervals of packets, metadata distribution of payload, input and output ranges, and entropy of complexity. According to the payload value domain, network protocols are typically classified into two groups, namely textual and binary, having different features and inference methods. Fig. 6 illustrates the protocol coverage of 41 PRE tools.

(iii) Modeling: The model abstracts the extracted protocol features and computes the results. It is often represented as algorithms or processes of mathematical analyses, such as the transition probability model [26], Hidden Semi-Markov model [60], Bayesian model [33], and the PTA [25].

(vi) Field inference and syntax recovery: The protocol field format defines a lexical and syntactic constraint for messages, reflecting the field's symbolic features and the organization rules. Common field types include delimiter, keyword, length, padding, and payload data. Parallelism, sequence, and hierarchy are the common structural relationships. NetT-based field inference usually assumes that the messages of the same format are similar, which can be derived by clustering samples. ExeT-based methods, however, supposes that the fields with different meanings are divided by different program functions through

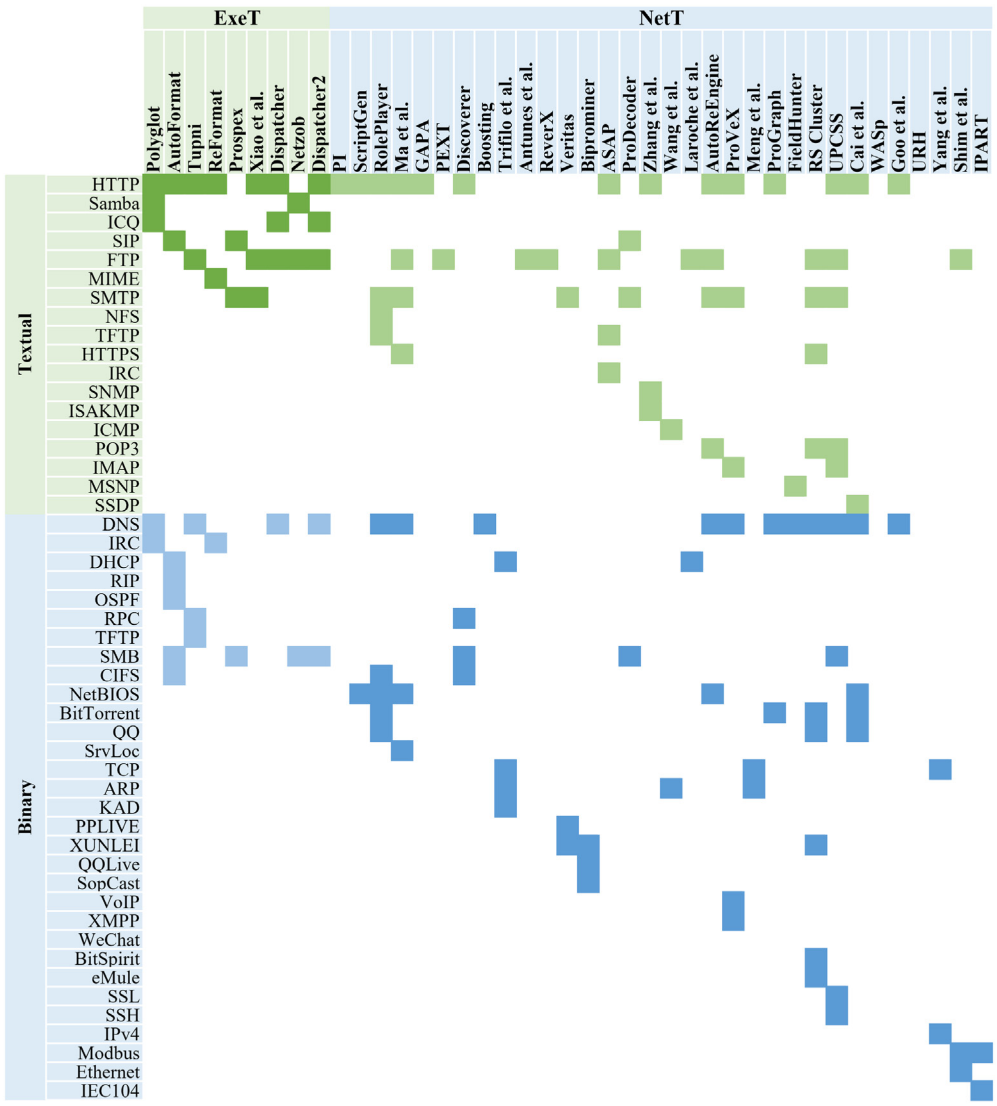


Fig. 6. Protocol coverage of the PRE tools, provided under four categories: textual/binary protocols and ExeT/NetT-based inference.

data and control flows, analysis tracking construction, parsing, and encryption or decryption process of protocols.

(v) Finite state machine reconstruction and semantic inference: The state machine reflects the logical rules for protocol entities' state transitions and the behavioral semantic information. Taking the clarity of fields as a prerequisite, state machine inference first abstracts messages into types, then performs state annotation, fusion, conversion, and correction to form a complete model.

Target: Besides PF and PFSM, the output of PRE also includes protocol keywords, vulnerability mining, deep packet inspection (DPI), fingerprint, malware detection, command-and-control (C&C) detection, application identification, evaluation, protocol reusing, traffic replay, network maintenance, and intrusion detection, some of which are summarized in Table 11.

Table 11
Overview of the target output of the PRE methods/tools besides PF or PFSM.

Output	Num.	PRE methods or tools
Protocol keywords	8	Replayer [68], ProDecoder [27], AutoReEngine [30], Li et al. [7], ProHacker [45], RS Cluster [32], Cai et al. [60], Xiao et al. [34]
Vulnerability mining	3	Fuzzgrind [78], AutoFuzz [51], PULSAR [44]
DPI	2	UPCSS [54], FieldHunter [6]
Fingerprint	1	ProPrint [8]
Malware detection	2	FFE/x86 [67], ProVeX [64]
C&C detection	2	MACE [80], ProVeX [64]
Application identification	2	Ma et al. [11], ProHacker [45]

4.2. Evaluation

The early PRE evaluation methods proposed by Narayan et al. are very well-known, where correctness measures how well the results match the specification; conciseness determines the number of messages or states representing a single one; coverage is the measure of completeness. In response to the increasing numbers of private protocols, as well as the widespread use of encryption and different architectures in various network environments, this paper establishes a multidimensional and quantitative evaluation approach using the Fuzzy Comprehensive Evaluation (FCE) method.

Table 12
Details of the three types of weight vectors.

Evaluation factor	Correctness	Coverage	Level of automation	Conciseness	Decryption capability	Efficiency	Multi-platform	Scalability
A_1 (NetT)	0.23[a]	0.22	0.09	0.09	0.09	0.18	0.05	0.05
A_2 (ExeT)	0.25	0.10	0.10	0.15	0.20	0.10	0.05	0.05
A_3 (All)	0.26	0.15	0.11	0.11	0.16	0.11	0.05	0.05

[a]The values indicate the relative importance of each factor.

First, we create a factor set

$$\begin{aligned} U &= \{U_1, U_2, \ldots, U_8\} \\ &= \left\{ \begin{aligned} &Correctness, Coverage, Level\ of\ automation, Conciseness, \\ &Decryption\ capability, Efficiency, \\ &Multi-platform, Scalability \end{aligned} \right\} \end{aligned}$$

Table 12 presents the three types of weight vectors, A_1, A_2, A_3, with different factors respectively representing the importance on NetT-based, ExeT-based, or all of the tools, where:

$$A = \{a_1, a_2, \ldots, a_n\}, \sum_{i=1}^{n} a_i = 1 \left(0 < a_i < 1\right)$$

We considered based on the fact that different evaluation weight criteria are implemented for different PRE tasks. A_1, A_2, and A_3 reflect our different weighting of the 8-dimensional metrics in the NetT/ExeT/All conditions.

Then, we assume that the comment set

$$\begin{aligned} V &= \{V_1, V_2, V_3\} \\ &= \{Good\ performance, Fair\ performance, \\ &\quad Poor\ performance\ or\ unknown\} \end{aligned}$$

We evaluate each factor based on the following principles:

(i) *Correctness*: Accuracy of results in the protocol specification.

(ii) *Coverage*: Percentage of specifications that can be covered. Approaches inferring more fields should score higher than the ones inferring only a few ones. Approaches supporting both sides of send/receive protocols should score higher than the ones supporting only one-way.

(iii) *Level of automation*: NetT-based approaches score higher than the ones based on ExeT. PFs-focused approaches score higher than the ones focused on PFSMs. Passive PFSM inference scores are higher than that of the active ones [107].

(vi) *Conciseness*: The number of results required to reflect the true protocol specification. The more redundant, the less concise.

(v) *Decryption capability*: Support for analysis against encryption protocols or traffic.

(vi) *Efficiency*: The number of analyses completed in the unit interval, considered by stages. NetT and ExeT-based inference methods require the collection and parsing stages, respectively. For the offline independent static analysis, it is easy to calculate the time separately, which is hard to achieve for the simultaneous dynamic analysis. Here, we simplify by considering only the total time for accomplishment.

(vii) *Multi-platform*: Being either platform-specific or portable to other environments or architectures. NetT-based methods tend to be more portable than ExeT-based methods, and the passive methods are more portable than the active ones.

(viii) *Scalability*: Being not only PRE-specific but also supporting conventional reverse-engineering, such as data structure identification, type inference, and object recovery.

In this paper, we selected 32 well-known tools either published in top conferences (e.g., SP, CCS, USENIX, and NDSS) or with more than 40 citations. Table 13 summarizes the evaluation ranks, which will then be transformed into the evaluation matrix:

$$R = \left(r_{ij}\right)_{n\times m}, r_{ij}\ (i = 1, 2, \ldots, m, j = 1, 2, \ldots, n)$$

The evaluation result matrix B is a fuzzy subset on the domain of V, which can be expressed as:

$$B = A \circ R$$

To fully use R, we choose the fuzzy operator '$\circ$' to be:

$$M(\bullet, \oplus)$$

which implies that:

$$B_k = \sum_{j=1}^{m} a_j r_{jk},\ k = 1, 2, \ldots, n$$

Assuming that $m = 32$ and $n = 8$, we rank B by calculating the comprehensive evaluation value according to the maximum subordination principle. Fig. 7 illustrates the performance of the top three PRE tools, which are individually evaluated based on NetT (Fig. 7(a)) and ExeT (Fig. 7(b)). Table 14 provides the top ten PRE tools.

Considering that the weight vector is not unique, the analyst should select the appropriate evaluation vector according to tasks. For example, ProVeX and Dispatcher2 rank first if *decryption capability* weight increases. Based on the evaluation, we find: (i) NetT-based tools are generally weaker than the ones based on ExeT in terms of *conciseness*. (ii) NetT-based tools using mature algorithms, such as multiple sequence comparison, unsupervised clustering, and statics approach, tend to be more efficient and automated, whereas the complex ExeT-based inference is relatively inefficient. (iii) Most NetT-based tools are weak in terms of decryption of the cipher payload not having clear features of the oral specification.

5. Conclusions

In recent years, researchers working in the field of PRE have developed numerous security applications, such as intrusion detection, vulnerability mining, and malicious code analysis. Although the PRE has produced several tools for specification extraction, there still exist some problems in automation, efficiency, and accuracy due to the complexity of private protocols and the limitations of the method. The loss of a priori information makes it difficult to derive the joint probability distribution of fields, hence the PRE can only infinitely approximate the correct position under the observed posterior probabilities.

The existing methods, generally combining the empirical inference with experimental verification and heuristically based on some engineering grounds, still lack systematic theory support and generalization ability. Especially for the encrypted protocols, the NetT-based tools are difficult to figure out fields semantic. The traditional PRE techniques focus on the syntactic, such as field division and attributes, while a deeper understanding of protocol semantics is the key.

Potential research opportunities include the following:

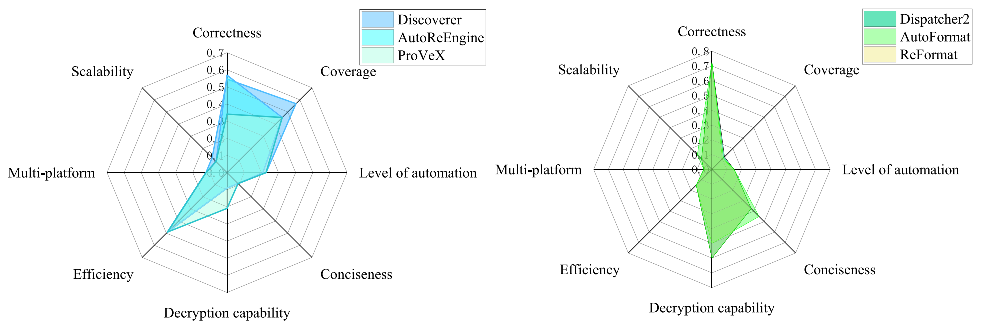
(i) Since widespread encryption makes it hard to parse private protocols only through the network traffic and to access the protocol binary executables, side-channel methods may be exploited to capture higher dimension features for analysis, particularly for firmware devices.

(ii) From a design perspective, more newborn protocols are reusing existing public specifications for agile development. That is helpful to address the problem of how to translate the design logic, the features of the public protocol program's source code or binary, and the analytical experience into the automated reverse process.

Table 13
Evaluation of the 32 PRE methods or tools.

Methods or tools	Year	Correctness	Coverage	Level of automation	Conciseness	Decryption capability	Efficiency	Multi-platform	Scalability
PI [3]	2004	▲	▲	●	○	○	●	●	●
ScriptGen [20]	2005	▲	●	●	○	○	●	●	○
RolePlayer [40]	2006	▲	●	●	○	○	●	●	●
Ma et al. [11]	2006	▲	●	●	○	○	●	●	▲
FFE/x86 [67]	2006	●	○	▲	▲	○	●	○	●
Replayer [68]	2006	●	○	▲	▲	○	○	○	▲
GAPA [63]	2007	▲	●	●	○	○	●	●	▲
Discoverer [22]	2007	●	●	●	○	○	●	●	●
Polyglot [69]	2007	●	○	▲	●	○	○	○	●
Boosting [23]	2008	▲	●	●	○	○	●	●	▲
ConfigRE [73]	2008	●	○	▲	▲	○	○	○	▲
AutoFormat [71]	2008	●	○	▲	●	●	○	○	●
Tupni [74]	2008	●	○	▲	▲	○	○	○	●
Wondracek et al. [72]	2008	●	○	▲	▲	○	○	○	●
ReFormat [79]	2009	●	○	▲	▲	●	○	○	▲
Prospex [76]	2009	●	○	▲	●	○	▲	○	▲
Dispatcher [75]	2009	●	○	▲	●	▲	○	○	●
REWARDS [82]	2010	●	○	▲	▲	○	○	○	●
Lin Z et al. [81]	2010	●	○	▲	▲	○	○	○	●
MACE [80]	2010	●	▲	▲	▲	○	▲	○	▲
ReverX [25]	2011	▲	▲	●	▲	○	●	●	▲
Howard [83]	2011	●	○	▲	▲	○	○	○	●
ProDecoder [27]	2012	▲	▲	●	○	○	●	●	▲
Netzob [53]	2012	●	○	▲	▲	○	▲	●	▲
ProVeX [64]	2013	▲	●	●	○	▲	●	●	▲
AutoReEngine [30]	2013	●	●	●	○	○	●	●	▲
Dispatcher2 [85]	2013	●	○	▲	▲	●	○	○	▲
Proword [31]	2014	▲	●	●	○	○	●	●	▲
NEMESYS [61]	2018	▲	●	●	○	○	●	●	▲
URH [65]	2019	▲	●	●	○	○	●	●	○
ICSREF [89]	2019	●	▲	▲	▲	○	○	▲	○
Shim et al. [37]	2020	▲	●	●	○	○	●	●	▲

●: Good performance, ▲: Fair performance, ○: Poor performance or unknown.



(a) NetT (b) ExeT

Fig. 7. Performance of the top three PRE tools based on NetT and ExeT.

Table 14
List of the top ten PRE tools.

Rank	1	2	3	4	5
Tool Score	Dispatcher2 0.723	AutoFormat 0.718	ReFormat 0.718	Discoverer 0.707	Dispatcher 0.700
	6	7	8	9	10
	AutoReEngine 0.670	Netzob 0.668	ProVeX 0.658	Polyglot 0.651	MACE 0.649

(iii) Innovation grows from the combination of PRE and other technologies, such as structure recovery, fuzzy test, deep packet inspection, machine learning, and so on. Meanwhile, the integration of NetT- and ExeT-based techniques stands as a key research direction in finding a balance between accuracy and coverage.

(vi) A universal hybrid analysis platform is required to support traffic manipulation and guide the execution of protocol executables with high efficiency. Ongoing research focuses on the combination of static, online, and offline analyses, and improving the performance of symbolic execution solvers.

Table A.15

Name	Year	NetT /ExeT	Contributions
PI [3]	2004	N	The sequence matching algorithm became the basis of many later approaches based on NetT inference
ScriptGen [20]	2005	N	The first tool to build the protocol state machine from NetT
Ma et al. [11]	2006	N	A generic architectural and mathematical framework for unsupervised protocol inference
RolePlayer [40]	2006	N	A system to replay both sides for a variety of network applications
FFE/x86 [67]	2006	E	The earliest static program analysis tool for PRE
Replayer [68]	2006	E	Developed the first sound solution to the protocol replay problem
GAPA [63]	2007	N	The first system allows rapid development of generic application-level protocol analyzers
PEXT [21]	2007	N	Clusters similar packets, and extracts FSM from them
Discoverer [22]	2007	N	Many subsequent tools are derived from it
Polyglot [69]	2007	E	Discovers information about the message format by monitoring the execution of the binaries that implement the target protocol
Rosetta [70]	2007	E	The semantic inference technique basis of Dispatcher
Boosting [23]	2008	N	An initial implementation of an active learning system for field identification
AutoFormat [71]	2008	E	Creates the association of fields by entering the message into the binary and monitoring the execution over the build hierarchy tree
Wondracek et al. [72]	2008	E	Tracks the input of dynamic taints and analyzes the process in detail to identify delimiters, length fields, etc
ConfigRE [73]	2008	E	Proposes a suite of new techniques for detecting the semantic relations among configuration fields
Tupni [74]	2008	E	Aggregates multiple execution tracks by dynamic analysis method and improves the accuracy of the PRE specification
Trifilo et al. [57]	2009	N	Present an approach for capturing the logic of the protocol by deriving the protocol's features and reconstruction of the state machine
Antunes et al. [4]	2009	N	Present greedy and conservative approach to build an automaton of a network protocol from NetT
Dispatcher [75]	2009	E	The tool recursively decomposes the memory buffer to reveal the format of the message being sent
Prospex [76]	2009	E	Extends the work [72] to the same type message recognition and automata extracting
Xiao et al. [77]	2009	E	Divides message fields with statistics of the reappearing frequency rate of token
Fuzzgrind [78]	2009	E	Supports data constraints performed by execution paths to reflect message categories
ReFormat [79]	2009	E	Uses dynamic analysis to identify encrypted fields in messages
AutoFuzz [51]	2010	N	A framework intended to automatically extract specifications
Whalen et al. [24]	2010	N	A novel HMM-based approach for inferring the state machine of network protocols using NetT
MACE [80]	2010	E	The first technique to infer complete protocol state machines in the realistic highlatency network setting
Lin Z et al. [81]	2010	E	Proposed hierarchical field analysis
REWARDS [82]	2010	E	Shares the same insight as Dispatcher [75] for type inference and semantics extraction and extend to general data structures in a program
Veritas [52]	2011	N	Formalize a new model, probabilistic protocol state machine (P-PSM), for describing the PFSM
ASAP [41]	2011	N	A framework identifies components of network payloads by mapping them to a vector space and determining informative base directions
ReverX [25]	2011	N	Presents a new methodology and a tool to derive a protocol specification from NetT
Biprominer [26]	2011	N	Gives a probabilistic description of the PFs
Howard [83]	2011	E	Infers the layout of data in memory by analyzing the memory access patterns
Netzob [53]	2012	N	The most friendly open source tool to build specifically for PRE
PRISMA [42]	2012	N	Extracts a state machine model with associated templates and rules using machine learning techniques
ProDecoder [27]	2012	N	Works with asynchronous application protocols and sampled NetT and does not assume any a priori knowledge
Zhang et al. [28]	2012	N	Proposed a novel method to explore positive packet sequence which could complement the initial sample sets and improved QSM technology for applying mining protocol specifications
ARTISTE [84]	2012	E	Addresses the full data structure recovery process from primitive field types up to recursive data structures
ProVeX [64]	2013	N	Detects C&C flows in arbitrary network traffic
Wang et al. [58]	2013	N	A new algorithm to identify unknown protocols from binary data by utilizing feature sequences and association rules
Li et al. [43]	2013	N	Uses a sample set of message sequences to estimate the parameters of the model
Laroche et al. [29]	2013	N	Used linear genetic programming as a method
AutoReEngine [30]	2013	N	Infers the message format to a series of aligned keyword vectors with good accuracy
Dispatcher2 [85]	2013	E	Has good handling of encrypted message formats
Liu et al. [86]	2013	E	Proposed backward program slicing approach
Meng et al. [5]	2014	N	Automatically generate the state models for binary protocol by listening to NetT
Proword [31]	2014	N	An unsupervised approach extracts protocol feature words from NetT
UPCSS [54]	2015	N	An unknown network protocol classification method based on semi-supervised learning
ProGraph [59]	2015	N	A protocol format inference tool operates on packet payloads at both byte-level and bit-level granularities
FieldHunter [6]	2015	N	Extracts and infers fields from both binary and textual protocols
PULSAR [44]	2015	N	A method for stateful black-box fuzzing of proprietary network protocols
Li et al. [7]	2015	N	A method for automatically reverse engineering the protocol message formats by using LDA and association analysis
ProHacker [45]	2015	N	Performs better results than Proword [31] for online protocol traffic analysis
RS Cluster [32]	2015	N	Improved the accuracy of PRE by applying a rough set theory-based method
ARGOS [87]	2015	E	The first system focuses on the semantic reverse engineering of data structures, and makes the first step towards reverse engineering of kernel data structures
Cai et al. [60]	2016	N	Infer message fields and keywords based on hidden semi-Markov model by maximizing the likelihood probability of message segmentation
WASp [46]	2016	N	A PRE tool developing for wireless protocol analysis and spoofing packet generation
PRE-Bin [33]	2016	N	Extracts bit-oriented boundaries from bit positions with frequency of gaps
Xiao et al. [34]	2016	N	Presented a method of identifying field boundary and extracting protocol token and designed a logic feature selector to filter the logic feature keywords
ProPrint [8]	2017	N	A protocol fingerprint inference system which uses Bayesian statistical model to discover temporal structure of protocol words
Esoul et al. [9]	2017	N	Proposed a segment-based alignment algorithm for the purpose of detecting packet structures
PREUGI [47]	2017	N	Infers protocol state machine by using a scheme of error-correcting grammatical inference (GI)
BITY [88]	2017	E	An approach to learn types for recovered variables from related representative instructions
NEMESYS [61]	2018	N	Presented a novel method of format inference with achieving results comparable to sequence alignment by analyzing only isolated messages
URH [65]	2019	N	The first software to provide a complete suite for investigating stateless and stateful wireless protocols
Goo et al. [48]	2019	N	Proposed a novel PRE method by using contiguous sequential pattern (CSP) algorithm
Luo et al. [10]	2019	N	Developed a type-aware approach by hiring topic generative models for protocol messages reversing

(*continued on next page*)

Table A.15 (*continued*).

Name	Year	NetT /ExeT	Contributions
ProSeg [35]	2019	N	Developed by introducing and optimize statistics (self-information and mutual information) from information theory
ICSREF [89]	2019	E	A reverse engineering framework for industrial binaries
Yang et al. [49]	2020	N	A field sequence coding method for field change features
Sun et al. [36]	2020	N	Proposed a method of format-oriented message clustering of unknown protocols
Shim et al. [37]	2020	N	A PRE tool for industrial protocols
IPART [55]	2020	N	An unsupervised tool for automatically reverse the format of the industrial protocol from NetT
NetPlier [62]	2021	N	Unique advantages of technique in IoT protocol reverse engineering and malware analysis

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This paper is supported by the National Key Research and Development Project (2016YFB08011601). The authors would like to acknowledge them.

Appendix. Chronological contributions of pre methods or tools

See Table A.15.

References

[1] Mary Meeker, Internet Trends 2019, 2019, https://www.bondcap.com/report/itr19/ (accessed March 23 2021).

[2] Cisco, Cisco 2018 Annual Cybersecurity Report, 2018, https://www.cisco.com/c/dam/m/digital/elq-cmcglobal/witb/acr2018/acr2018final.pdf (accessed 23 March 2021).

[3] M. Beddoe, The protocol informatics project, 2004, http://www.4tphi.net/~awalters/PI/PI.html.

[4] J. Antunes, N. Neves, Building an automaton towards reverse protocol engineering, 2009, http://www.di.fc.ul.pt/~nuno/PAPERS/INFORUM09.pdf.

[5] F. Meng, Y. Liu, C. Zhang, T. Li, Y. Yue, Inferring protocol state machine for binary communication protocol, in: Proceedings of the IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA '14), 2014, pp. 870–874.

[6] I. Bermudez, A. Tongaonkar, M. Iliofotou, M. Mellia, M.M. Munafo, Automatic protocol field inference for deeper protocol understanding, in: Proceedings of the 14th IFIP Networking Conference (Networking '15), 2015, pp. 1–9.

[7] H. Li, B. Shuai, J. Wang, C. Tang, Protocol reverse engineering using LDA and association analysis, in: 2015 11th International Conference on Computational Intelligence and Security (CIS), Shenzhen, China, Dec. 2015, 312–316. http://dx.doi.org/10.1109/CIS.2015.83.

[8] Y. Wang, X. Yun, Y. Zhang, L. Chen, G. Wu, A nonparametric approach to the automated protocol fingerprint inference, J. Netw. Comput. Appl. 99 (2017) 1–9.

[9] O. Esoul, 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), Prague, Czech Republic, Jul. 2017, pp. 398–409. http://dx.doi.org/10.1109/QRS.2017.49.

[10] X. Luo, D. Chen, Y. Wang, P. Xie, A type-aware approach to message clustering for protocol reverse engineering, Sensors 19 (3) (2019) 716, http://dx.doi.org/10.3390/s19030716.

[11] M. Justin, L. Kirill, K. Christian, S. Stefan, V. Geoffrey, Unexpected means of protocol inference, in: Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC, 2006, pp. 313–326, http://dx.doi.org/10.1145/1177080.1177123.

[12] Pan Fan, et al., Overviews on protocol reverse engineering, Appl. Res. Comput. 28 (Aug) (2011) 2801–2806.

[13] Z. Zhang, Q. Wen, W. Tang, Survey of mining protocol specifications, Comput. Eng. Appl. 49 (9) (2013) 1–9.

[14] J. Narayan, S.K. Shukla, T.C. Clancy, A survey of automatic protocol reverse engineering tools, ACM Comput. Surv. 48 (2015) http://dx.doi.org/10.1145/2840724.

[15] J. Duchene, C. Le Guernic, E. Alata, V. Nicomette, J. Duchene, C. Le Guernic, E. Alata, V. Nicomette, M. Kaâniche, State of the art of network protocol reverse engineering tools, 2018, HAL Id : hal-01496958.

[16] B.D. Sija, Y. Goo, K. Shim, H. Hasanova, M. Kim, A Survey of Automatic Protocol Reverse Engineering Approaches, Methods, and Tools on the Inputs and Outputs View, 2018 (2018).

[17] S. Kleber, Survey of protocol reverse engineering algorithms : Decomposition of tools for static traffic analysis, IEEE Commun. Surv. Tutor. PP (2018) 1, http://dx.doi.org/10.1109/COMST.2018.2867544.

[18] X. Wang, J. Shen, H. Liu, et al., Review of research on automatic protocol reverse engineering, Appl. Res. Comput. 37 (9) (2020).

[19] J. Caballero, D. Song, Automatic protocol reverse-engineering: Message format extraction and field semantics inference, Comput. Netw. 57 (2013) 451–474, http://dx.doi.org/10.1016/j.comnet.2012.08.003.

[20] C. Leita, K. Mermoud, M. Dacier, Scriptgen: An automated script generation tool for honeyd, in: Proc. - Annu. Comput. Secur. Appl. Conf. ACSAC. 2005, 2005, pp. 203–214, http://dx.doi.org/10.1109/CSAC.2005.49.

[21] M. Shevertalov, S. Mancoridis, A reverse engineering tool for extracting protocols of networked applications, in: Proc. - Work. Conf. Reverse Eng. WCRE, 2007, pp. 229–238, http://dx.doi.org/10.1109/WCRE.2007.6.

[22] W. Cui, J. Kannan, H.J. Wang, Discoverer: Automatic protocol reverse engineering from network traces, in: 16th USENIX Secur. Symp., 2007, pp. 199–212.

[23] K. Gopalratnam, S. Basu, J. Dunagan, H.J. Wang, Automatically extracting fields from unknown network protocols, in: Proceedings of the 15th Symposium on Network and Distributed System Security (NDSS '08), 2008.

[24] S. Whalen, M. Bishop, J.P. Crutchfield, Hidden Markov models for automated protocol learning, in: S. Jajodia, J. Zhou (Eds.), Security and Privacy in Communication Networks, Vol. 50, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 415–428.

[25] J. Antunes, N. Neves, P. Verissimo, Reverse engineering of protocols from network traces, in: Proceedings of the 18th Working Conference on Reverse Engineering (WCRE '11), 2011, pp. 169–178.

[26] Y. Wang, X. Li, J. Meng, Y. Zhao, Z. Zhang, L. Guo, Biprominer: automatic mining of binary protocol features, in: Proceedings of the 12th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '11), 2011, pp. 179–184.

[27] Y. Wang, X. Yun, M.Z. Shafiq, et al., A semantics aware approach to automated reverse engineering unknown protocols, in: Proceedings of the 20th IEEE International Conference on Network Protocols (ICNP '12), IEEE, Austin, Tex, USA, 2012, pp. 1–10.

[28] Z. Zhang, Q.-Y. Wen, W. Tang, Mining protocol state machines by interactive grammar inference, in: Proceedings of the 2012 3rd International Conference on Digital Manufacturing and Automation (ICDMA '12), 2012, pp. 524–527.

[29] P. Laroche, A. Burrows, A.N. Zincir-Heywood, How far an evolutionary approach can go for protocol state analysis and discovery, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13), 2013, pp. 3228–3235.

[30] J.-Z. Luo, S.-Z. Yu, Position-based automatic reverse engineering of network protocols, J. Netw. Comput. Appl. 36 (3) (2013) 1070–1077.

[31] Z. Zhang, Z. Zhang, P.P.C. Lee, Y. Liu, G. Xie, ProWord: An unsupervised approach to protocol feature word extraction, in: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, Toronto, ON, Canada, Apr. 2014, 1393–1401. http://dx.doi.org/10.1109/INFOCOM.2014.6848073.

[32] J.-Z. Luo, S.-Z. Yu, J. Cai, Capturing uncertainty information and categorical characteristics for network payload grouping in protocol reverse engineering, Math. Probl. Eng. 2015 (2015) 962974, 9 pages.

[33] S. Tao, H. Yu, Q. Li, Bit-oriented format extraction approach for automatic binary protocol reverse engineering, IET Commun. 10 (6) (2016) 709–716, http://dx.doi.org/10.1049/iet-com.2015.0797.

[34] M.-M. Xiao, S.-L. Zhang, Y.-P. Luo, Automatic network protocol message format analysis, IFS 31 (4) (2016) 2271–2279, http://dx.doi.org/10.3233/JIFS-169067.

[35] F. Sun, S. Wang, C. Zhang, H. Zhang, Unsupervised field segmentation of unknown protocol messages, Comput. Commun. 146 (2019) 121–130, http://dx.doi.org/10.1016/j.comcom.2019.06.013.

[36] F. Sun, S. Wang, C. Zhang, H. Zhang, Clustering of unknown protocol messages based on format comparison, Comput. Netw. 179 (2020) 107296, http://dx.doi.org/10.1016/j.comnet.2020.107296.

[37] K. Shim, Y. Goo, M. Lee, M. Kim, Clustering method in protocol reverse engineering for industrial protocols, Int. J. Netw. Manage. (2020) http://dx.doi.org/10.1002/nem.2126.

[38] W. Day, H. Edelsbrunner, Efficient algorithms for agglomerative hierarchical clustering methods, J. Classification 1 (7) (1984) 7–24.
[39] S.B. Needleman, C.D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, J. Mol. Biol. 48 (1970).
[40] W. Cui, V. Paxson, N.C. Weaver, R.H. Katz, Protocol-independent adaptive replay of application dialog, in: Proceedings of the 13th Symposium on Network and Distributed System Security (NDSS '06), 2006.
[41] T. Krueger, N. Krmer, K. Rieck, ASAP: automatic semantics-aware analysis of network payloads, in: Proceedings of the ECML/PKDD, 2011.
[42] T. Krueger, H. Gascon, N. Krmer, K. Rieck, Learning stateful models for network honeypots, in: Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence, AISec'12, ACM, New York, NY, 2012, pp. 37–48, http://dx.doi.org/10.1145/2381896.2381904.
[43] M. Li, S.Z. Yu, Noise-Tolerant and optimal segmentation of message formats for unknown application-layer protocols, J. Softw. 24 (3) (2013) 604–617, http://www.jos.org.cn/1000-9825/4243.htm.
[44] H. Gascon, C. Wressnegger, F. Yamaguchi, D. Arp, K. Rieck, Pulsar: Stateful black-box fuzzing of proprietary network protocols, in: B. Thuraisingham, X. Wang, V. Yegneswaran (Eds.), Security and Privacy in Communication Networks, Vol. 164, Springer International Publishing, Cham, 2015, pp. 330–347.
[45] Y. Wang, X. Yun, Y. Zhang, Rethinking robust and accurate application protocol identification: A nonparametric approach, in: IEEE Proceedings of the 23rd International Conference on Network Protocols (ICNP), 2015, pp. 134–144.
[46] K. Choi, Y. Son, J. Noh, H. Shin, J. Choi, Y. Kim, Dissecting customized protocols: automatic analysis for customized protocols based on IEEE 802.15.4, in: Proceedings of the 9th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Darmstadt, Germany, 2016, pp. 183–193.
[47] M.-M. Xiao, Y.-P. Luo, Automatic protocol reverse engineering using grammatical inference, IFS 32 (5) (2017) 3585–3594, http://dx.doi.org/10.3233/JIFS-169294.
[48] Y.-H. Goo, K.-S. Shim, M.-S. Lee, M.-S. Kim, Protocol specification extraction based on contiguous sequential pattern algorithm, IEEE Access 7 (2019) 36057–36074, http://dx.doi.org/10.1109/ACCESS.2019.2905353.
[49] C. Yang, C. Fu, Y. Qian, Y. Hong, G. Feng, L. Han, Deep learning-based reverse method of binary protocol, in: S. Yu, P. Mueller, J. Qian (Eds.), Security and Privacy in Digital Economy, Vol. 1268, Springer Singapore, Singapore, 2020, pp. 606–624.
[50] S.Z. Yu, Hidden semi-Markov models, Artificial Intelligence 174 (2010) 215–243.
[51] S. Gorbunov, A. Rosenbloom, Autofuzz: Automated network protocol fuzzing framework, IJCSNS 10 (8) (2010) 239.
[52] Y. Wang, Z. Zhang, D.D. Yao, B. Qu, L. Guo, Inferring protocol state machine from network traces: a probabilistic approach, in: Proceedings of the 9th Applied Cryptography and Network Security International Conference (ACNS '11), 2011, pp. 1–18.
[53] G. Bossert, F. Guihéry, G. Hiet, Towards automated protocol reverse engineering using semantic information, in: Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, Kyoto, Japan, 2014.
[54] R. Lin, O. Li, Q. Li, Y. Liu, Unknown network protocol classification method based on semi supervised learning, in: Proceedings of the IEEE International Conference on Computer and Communications (ICCC '15), Chengdu, China, 2015, pp. 300–308.
[55] X. Wang, K. Lv, B. Li, IPART: an automatic protocol reverse engineering tool based on global voting expert for industrial protocols, Int. J. Parallel Emergent Distrib. Syst. 35 (3) (2020) 376–395, http://dx.doi.org/10.1080/17445760.2019.1655740.
[56] R.R. Sokal, C.D. Michener, A Statistical Method for Evaluating Systematic Relationships, University of Kansas Scientific Bulletin, 1958.
[57] A. Trifilo, S. Burschka, E. Biersack, Traffic to protocol reverse engineering, in: Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–8.
[58] Y. Wang, N. Zhang, Y.-M. Wu, B.-B. Su, Y.-J. Liao, Protocol formats reverse engineering based on association rules in wireless environment, in: Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom '13), Melbourne, Australia, 2013, pp. 134–141.
[59] Q. Huang, P.P.C. Lee, Z. Zhang, Exploiting intrapacket dependency for fine-grained protocol format inference, in: Proceedings of the 14th IFIP Networking Conference (NETWORKING '15), Toulouse, France, 2015.
[60] J. Cai, J. Luo, F. Lei, Analyzing network protocols of application layer using hidden Semi-Markov model, Math. Probl. Eng. 2016 (2016) 9161723, 14 pages.
[61] S. Kleber, H. Kopp, F. Kargl, NEMESYS: Network message syntax reverse engineering by analysis of the intrinsic structure of individual messages, 2018.
[62] Y. Ye, Z. Zhuo, F. Wang, X. Zhang, D. Xu, NetPlier: probabilistic network protocol reverse engineering from message traces, in: Proceedings of the Symposium on Network and Distributed System Security (NDSS '21).
[63] N. Borisov, D.J. Brumley, H.J. Wang, J. Dunagan, P. Joshi, C. Guo, Generic Application-Level Protocol Analyzer and Its Language, MSR Technical Report MSR-TR-2005-133, 2005.
[64] C. Rossow, C.J. Dietrich, PROVEX: detecting botnets with encrypted command and control channels, in: Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2013.
[65] J. Pohl, A. Noack, Universal radio hacker: A suite for analyzing and attacking stateful wireless protocols, Baltimore, MD, Aug. 2018, [Online].
[66] H. Wang, C. Guo, D. Simon, A. Zugenmaier, Shield: Vulnerability-driven network filters for preventing known vulnerability exploits, in: Proceedings of ACM SIGCOMM, 2004.
[67] J. Lim, T. Reps, B. Liblit, Extracting output formats from executables, in: 13th Working Conference on Reverse Engineering, 2006. WCRE '06, IEEE, Benevento, 2006, pp. 167–178, http://dx.doi.org/10.1109/WCRE.2006.29.
[68] J. Newsome, D. Brumley, J. Franklin, D. Song, Replayer: Automatic protocol replay by binary analysis, in: Proc. ACM Conf. Comput. Commun. Secur., 2006, pp. 311–321, http://dx.doi.org/10.1145/1180405.1180444.
[69] J. Caballero, H. Yin, Z. Liang, 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 (CCS '07), ACM, 2007, pp. 317–329.
[70] J. Caballero, D. Song, Rosetta: Extracting Protocol Semantics using Binary Analysis with Applications To Protocol Replay and NAT Rewriting, Technical Report CMU-CyLab-07-014, Carnegie Mellon University, Pittsburgh, 2007.
[71] Z. Lin, X. Jiang, D. Xu, X. Zhang, Automatic protocol format reverse engineering through context-aware monitored execution, in: Proceedings of the 15th Symposium on Network and Distributed System Security (NDSS '08), 2008.
[72] G. Wondracek, P.M. Comparetti, C. Kruegel, E. Kirda, S.S.S. Anna, Automatic network protocol analysis, in: Netw. Distrib. Syst. Secur. Symp., 2008, pp. 1–18.
[73] R. Wang, X. Wang, K. Zhang, Z. Li, Towards automatic reverse engineering of software security configurations, in: Proceedings of the 15th ACM Conference on Computer and Communications Security, CCS'08, ACM, Limerick, 2008, pp. 245–256, http://dx.doi.org/10.1145/1455770.1455802.
[74] W. Cui, M. Peinado, K. Chen, H.J. Wang, L. Irun-Briz, Tupni: automatic reverse engineering of input formats, in: Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS '08), ACM, Alexandria, Va, USA, 2008, pp. 391–402.
[75] J. Caballero, P. Poosankam, C. Kreibich, D. Song, Dispatcher: enabling active botnet infiltration using automatic protocol reverse-engineering, in: Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09), ACM, Chicago, Ill, USA, 2009, pp. 621–634.
[76] P.M. Comparetti, G. Wondracek, C. Kruegel, E. Kirda, Prospex: protocol specification extraction, in: Proceedings of the 30th IEEE Symposium on Security and Privacy, Berkeley, Calif, USA, 2009, pp. 110–125.
[77] M.M. Xiao, S.Z. Yu, Y. Wang, Automatic network protocol automaton extraction, in: Proceedings of the 3rd International Conference on Network and System Security (NSS '09), 2009, pp. 336–343.
[78] G. Campana, Fuzzgrind: un outil de fuzzing automatique, in: Symposium sur la Scurit des Technologies de l'Information et de la Communication, SSTIC, SSTIC, Rennes, France, 2009.
[79] Z. Wang, X. Jiang, W. Cui, X. Wang, M. Grace, ReFormat: automatic reverse engineering of encrypted messages, in: M. Backes, P. Ning (Eds.), Computer Security—ESORICS 2009. ESORICS 2009, in: Lecture Notes in Computer Science, vol. 5789, Springer, Berlin, Germany, 2009, pp. 200–215.
[80] C.Y. Cho, D. Babi, E.C.R. Shin, D. Song, Inference and analysis of formal models of botnet command and control protocols, in: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS '10, ACM, New York, NY, 2010, pp. 426–439, http://dx.doi.org/10.1145/1866307.1866355.
[81] Z. Lin, X. Zhang, D. Xu, Reverse engineering input syntactic structure from program execution and its applications, IEEE Trans. Softw. Eng. 36 (5) (2010) 688–703, http://dx.doi.org/10.1109/TSE.2009.54.
[82] Z. Lin, X. Zhang, D. Xu, Automatic reverse engineering of data structures from binary execution, in: Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS), Internet Society, San Diego, 2010.
[83] A. Slowinska, T. Stancescu, H. Bos, Howard: a dynamic excavator for reverse engineering data structures, in: Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS), Internet Society, San Diego, 2011.
[84] J. Caballero, G. Grieco, M. Marron, Z. Lin, D. Urbina, ARTISTE: Automatic Generation of Hybrid Data Structure Signatures from Binary Code Executions, Technical Report TR-IMDEA-SW-2012-001, IMDEA Software Institute, Madrid, 2012.
[85] J. Caballero, D. Song, Automatic protocol reverse-engineering: message format extraction and field semantics inference, Comput. Netw. 57 (2) (2013) 451–474.
[86] Min Liu, Chunfu Jia, Lu Liu, Zhi Wang, Extracting sent message formats from executables using backward slicing, in: 2013 4th International Conference on Emerging Intelligent Data and Web Technologies (EIDWT), pp. 377–384.
[87] J. Zeng, Z. Lin, Towards automatic inference of kernel object semantics from binary code, in: 18th International Symposium, RAID 2015, Vol. 9404, Springer, Kyoto, 2015, pp. 538–561, http://dx.doi.org/10.1007/978-3-319-26362-5.
[88] Z. Xu, C. Wen, S. Qin, Learning Types for Binaries, Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), in: LNCS, vol. 10610, 2017, pp. 430–446, http://dx.doi.org/10.1007/978-3-319-68690-5_26.

[89] K. Anastasis, M. Michail, ICSREF: A Framework for Automated Reverse Engineering of Industrial Control Systems Binaries, 2019, http://dx.doi.org/10.14722/ndss.2019.23271.

[90] G. Balakrishnan, T. Reps, Analyzing memory accesses in x86 executables, in: International Conference on Compiler Construction, CC 2004. Lecture Notes in Computer Science, vol. 2985. Springer, Berlin, Heidelberg. http://dx.doi.org/10.1007/978-3-540-24723-4_2.

[91] T. Reps, G. Balakrishnan, J. Lim, Intermediate representation recovery from low-level code, in: PEPM, 2006.

[92] G. Balakrishnan, T. Reps, Recovery of Variables and Heap Structure in X86 Executables, Tech. Rep. TR-1533, Comp. Sci. Dept. Univ. of Wisconsin, Madison, WI, 2005.

[93] H. Yin, D. Song, E. Manuel, C. Kruegel, E. Kirda, Panorama: capturing system-wide information flow for malware detection and analysis, in: ACM Conference on Computer and Communications Security, Alexandria, VA, 2007.

[94] T. Reps, G. Balakrishnan, J. Lim, A next-generation platform for analyzing executables, in: APLAS, 2005.

[95] C. Cadar, D. Dunbar, D. Engler, KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs, in: USENIX Symposium on Operating Systems Design and Implementation (OSDI 2008), 2008.

[96] F. Bellard, QEMU, a fast and portable dynamic translator, in: Proc. of the USENIX Annual Technical Conference, 2005.

[97] H. Yin, D. Song, E. Manuel, C. Kruegel, E. Kirda, Panorama: Capturing system-wide information flow for malware detection and analysis, in: ACM Conference on Computer and Communications Security, Alexandria, VA, 2007.

[98] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V.J. Reddi, K. Hazelwood, Pin: building customized program analysis tools with dynamic instrumentation, in: PLDI '05: Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation, 2005, pp. 190–200.

[99] J. Newsome, D. Song, Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software, in: NDSS, 2005.

[100] Nicholas Nethercote, Julian Seward, Valgrind: A program supervision framework, in: Proceedings of the Third Workshop on Runtime Verification (RV'03), Boulder, Colorado, USA, 2003.

[101] C. Guarnieri, M. Schloesser, J. Bremer, A. Tanasi, Cuckoo sandbox - open source automated malware analysis, in: Black Hat USA, 2013.

[102] Wireshark: The word's most popular network protocol analyzer. https://www.wireshark.org/.

[103] Tcpdump, a powerful command-line packet analyzer, and libpcap, a portable C/C++ library for network traffic capture. https://www.tcpdump.org/.

[104] Network Monitor 3.4 is the archive versioned tool for network traffic capture and protocol analysis. https://www.microsoft.com/en-us/download/details.aspx?id=4865.

[105] Scapy is a powerful interactive packet manipulation program. https://scapy.net/.

[106] V. Paxson, Bro: a system for detecting network intruders in real-time, Comput. Netw. 31 (23–24) (1998) 2435–2463, (Amsterdam, Netherlands: 1999).

[107] X. Wang, J. Shen, H. Liu, et al., Review of research on automatic protocol reverse engineering, Appl. Res. Comput. 37 (09) (2020) 2561–2570+2585.