Survey of Protocol Reverse Engineering Algorithms: Decomposition of Tools for Static Traffic Analysis

Stephan Kleber Lisa Maile Frank Kargl

***Abstract*— Knowledge about a network protocol to understand the communication between entities is necessary for vulnerability research, penetration testing, malware analysis, network reconnaissance, and network modeling. Traffic analysis is one approach to infer a protocol, and this approach has specific challenges, tasks, methods, and solutions. In this survey, we collect tools presented by prior research in the field of protocol reverse engineering by static traffic trace analysis. We dissect each tool to discern the individual mechanisms and the algorithms on which they are based, then categorize and contrast the mechanisms and algorithms used in static traffic trace analysis to discuss how successfully they were applied in each case.**

To structure our discussion about the tools, we compared classification schemes for protocol reverse engineering. We present and discuss an explicit process model for static traffic trace analysis to reveal the common structure of the decomposed tools and frameworks from previous research. Via discussions of the algorithms applied within each tool, we show relations between tools, methods, and the process for each process task. We validate our model by applying it to each of the tools, then provide an outline of the utility of protocol reverse engineering. Beginning with the process description, we deduce which solutions and algorithms have already been investigated and where challenges remain to determine how new solutions may be researched in the future. Across the entire field of protocol reverse engineering, few implementations of tools and frameworks are publicly available, which remains a prevalent problem.

***Index Terms*—Protocol reverse engineering, Communication networks, Pattern recognition, Statistical analysis, Machine learning, Network security.**

I. Introduction

Network protocols are intended to enable communication between nodes. The communicating nodes are entities which implement a shared protocol specification. In this context, a specification defines how the receiver must interpret transmitted data as well-defined information. It is necessary to understand how systems are interconnected, what interdependencies exist in a deployed network, and whether implementations of a protocol work as expected in order to analyze any implications that the communication has on functionality, performance, and security of the network. Any analysis of a network is based on knowledge of the occurring protocols' inner workings; however, being confronted with communication using an unknown specification impairs the work of the network researcher and practitioner. An analyst bound to work with such a protocol will need to reverse engineer its specification by monitoring network traffic or the communicating entities. Numerous examples, such as security analyses of implantable medical devices [1] or inference of botnet communication [2], show that this reverse engineering task is tedious and time consuming.

A. Basic Terms

Protocol reverse engineering (PRE) is intended to infer an approximation of the protocol specification by observing the communication. This inference goal sets PRE apart from reverse engineering of executable program binaries (software), which focuses on obtaining source code or an understanding of a program's implementation. PRE may, however, employ reverse engineering of software, but it is not limited to these kinds of analyses to infer the communication [3]. Therefore, two main methods for learning the inner workings of a protocol can be discerned: entity and trace analysis.

Entity analysis infers the protocol by applying software reverse engineering to a node implementation (e. g., [4, 5]). Entity analysis requires access to the program and its execution environment in a way that allows the use of techniques for software control flow analysis and memory introspection. Often it proves impossible to either obtain the program or a suitable execution environment for it; in both cases, techniques for entity analysis are not applicable. In contrast, traffic **trace analysis** resorts solely to analysis of the communication that is observable in the link between entities; traffic trace analysis remains possible in cases where reverse engineering of the executable program is not feasible. Whereas such trace analysis can only gain information from what can be observed on the communication link, it is non-invasive and does not require control over any entity.

The security assessment of an implantable cardioverter defibrillator (ICD) requires several steps [6] and stands as an example of a trace analysis task. Since the communication is wireless with a proprietary physical layer, the radio frequency transmission first needs to be recorded and decoded. The actual PRE process begins after obtaining the transmitted bits, via inspection of the trace to determine similar messages of the same format, e. g., carrying patient data. By recognizing patterns in the messages, like ASCII-encoded characters denoting the patient name, the common format of the message type may be revealed, and data fields in the message may thus be inferred. Finally, valid message exchange sequences, like the acknowledgment of the ICD's telemetry data, need to be determined.

Software reverse engineering techniques are fundamentally different from traffic analysis. They require different methods, a different set of tools, and a different kind of analysis process; software reverse engineering techniques also require of the analyst a specific type of experience. Moreover, the methods necessary for entity analysis are common and well understood beyond PRE [7]. We therefore focus our discussion on traffic

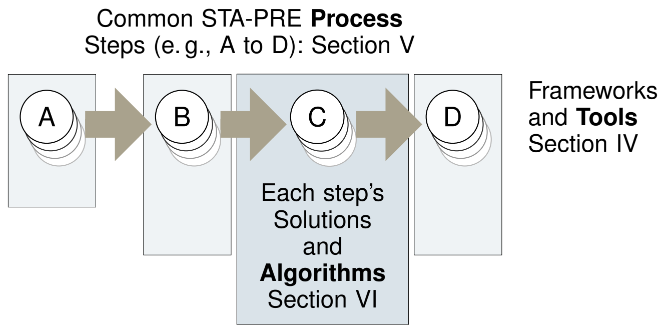

Common STA-PRE **Process** Steps (e.g., A to D): Section V
A
B
C
D
Frameworks and **Tools** Section IV
Each step's Solutions and **Algorithms** Section VI

Fig. 1: Strategy of this survey.

trace analysis, which is based only on captured network traffic. Since this analysis is performed offline after recording, and therefore without any interactive or dynamic analysis, we call this approach **static trace analysis** (STA).

B. Lacking Automation of Protocol Reverse Engineering

The PRE process requires that an analyst brings a large spectrum of knowledge and experience [8, 9, 10] to solve its tasks. The approaches we discuss in this survey have been proposed to support the PRE process by increasing the level of automation, which reduces the requisite expertise and increases coverage of the available information about a protocol from traces. The field of automated mechanisms has been attracting researchers for the last fourteen years, starting with Beddoe [11] in 2004. Despite this constant effort, however, no approach automates fully the whole PRE process. Automation or semi-automation of individual tasks has been proposed by each respective approach we included in this survey; however, the discussion has mostly been applied to each approach in isolation. Throughout the PRE community, there is no detailed definition of the PRE process steps and the challenges each presents. This hinders further discussion, comparison, and research, but most importantly, it prevents the widespread application of PRE automation and cooperation among experts. This is due to the notion that intuitive strategies, though casually accepted, tend to be baseless, incomplete, and subjective. The subtasks of the PRE process can most often be dealt with individually, but there exist dependencies between steps; therefore, a structured process and method description is of great interest for PRE as a common basis.

C. Strategy of this Survey

The overall strategy of this survey, as visualized in Figure 1, is to examine the workflows of all proposed STA-PRE frameworks and tools to deduce a common sequence of process steps. We decomposed all surveyed tools into their contained solutions and algorithms. From this decomposition approach, for each process step, we then identify and discuss the utilized solutions and algorithms.

After relating our survey to existing work on the systemization of PRE and general process descriptions in Section II, we revisit known classifications of (semi-)automated static traffic analysis tools in Section III as basis of our discussion. Section IV highlights the PRE workflow tasks, which each of the previously proposed tools addresses. Deducing from this task analysis, we introduce a process model implicitly common to all (semi-)automated tools and discuss various known adaptations of this model in Section V. Applying our model to all revisited tools, we dissect each into their underlying solutions and algorithms, map these to the tools, classify them, and discuss their utility for the use case in Section VI.

In this survey, we focus on automated and semi-automated approaches and tools since manual analysis is cumbersome and based mostly on analyst intuition, rendering it highly individual, burdened by the drawbacks inherent to a subjective approach. The analysis of software has a completely different set of challenges, and thus the processes are not comparable. We therefore leave the discussion of software analysis techniques for PRE to a future discussion.

II. Related Surveys and Overview

In 2006, Rauch [12] discussed automation of PRE and identified the steps necessary to reduce repetitive work by humans in the PRE process. The tool he presented at BlackHat was never published, so that no perpetual improvements for the PRE field could be gained. Over a decade later, the typical PRE process is still most often performed manually, although in the meantime several approaches with different levels of automation were proposed (e.g., Cui et al. [13], Antunes et al. [14], Krueger et al. [15], and Bossert et al. [16]). These new approaches have received only isolated discussion, and a more general understanding of the underlying common process and the algorithmic foundation of their functionality were never discussed. We argue that a structured identification of interdependencies of common procedures, algorithms, and solutions is needed. Several approaches have been proposed to improve static trace analyses by increasing the amount of automation in the process (e.g., [13, 14, 15, 16, 17]). However, to the best of our knowledge, the present contribution marks the first extensively structured process description and method discussion.

There do exist a few previous works that survey the PRE field, each covering specific aspects. We discuss all known surveys and compare them in the following paragraphs and Table I. As first of its kind, the survey by Li and Chen [8] provides a selective view of PRE by revisiting seven analysis tools, four static traffic analysis approaches [11, 13, 18, 29], referring to them as network-based, and four entity analysis approaches [36, 37, 38, 39, 40], naming them hybrid and binary-based. Li and Chen classify goals of PRE into obtaining knowledge about functions, messages, syntax, and the state machine of a protocol; but the study lacks an in-depth comparison of the listed concepts.

Narayan et al. [9] are considerably more verbose and provide a thorough classification of approaches. Their survey is broad and provides many valuable insights, but contains few details on the employed algorithms. In addition, not all existing tools for static traffic analysis are covered. On the other hand,

Surveyed STA-Tools	Li and Chen [8] (2011)	Narayan et al. [9] (2015)	Duchêne et al. [10] (2018)	Bossert et al. [16] (2014)	Kleber et al. (2018)
PI [11] (2004)	●	●	●		●
RolePlayer [18] (2006)	●	●	●		●
Discoverer [13] (2007)	●	●	●	●	●
Whalen et al. [19] (2010)					●
Biprominer [20] (2011)		●			●
ProDecoder [21] (2012)		●			●
Li et al. [22] (2015)					●
FieldHunter [23, 24] (2015, 2016)					●
Cai et al. [25] (2016)					●
PRE-Bin [26] (2016)					●
Xiao et al. [27] (2016)					●
NEMESYS [28] (2018)					●
ScriptGen [29] (2005)	●	●	●	●	●
PEXT [30] (2007)		●			●
Trifilo et al. [31] (2009)		●			●
Veritas [32] (2011)		●	●		●
PREUGI [33] (2017)					●
AutoFuzz [34] (2010)					●
ReverX [14] (2011)		●	●		●
ASAP [35] (2010)			●	●	●
PRISMA [15] (2012)			●		●
AutoReEngine [17] (2013)		●			●
Netzob [16] (2014)			●	●	●
Goal/Contribution					
describe tools	●	●	●	●	
discuss tools		●	●		
compare and categorize tools		●	●		●
evaluate tools				●	
decompose tools to algorithms					●
discuss and compare algorithms					●
synthezise process model			○		●
future work for algorithms					●

TABLE I: Comparison with related survey articles

Narayan et al. address not only static traffic analysis but also entity analysis; therefore, their work is broader than ours. Their survey discusses eleven trace and thirteen entity analysis approaches, naming them network trace and dynamic analysis tools, respectively. They define syntax and state machine as components of a protocol specification and categorize approaches accordingly. The survey offers a compact outline of the three aspects "correctness," "conciseness," and "coverage" as metrics for the quality of PRE results, as adopted from Cui et al. [13]; an inference is correct if each format is not recognized as multiple different ones. An inference is concise if an inferred format matches not more than one true type or state; finally, the coverage of an inference is high if each message in the available input data is reflected by at least one format or state in the inference. In the following discussion we use the terminology for protocol elements suggested by Narayan et al.

Duchêne et al. [10] provide a very recent survey that discusses nine traffic-based ("network based") and seventeen entity-based ("application based") tools. Their survey distinguishes "passive" and "active" approaches, where passive is the analysis of observed static data and active performs the analysis by controlled execution. The previous publication of Duchêne et al. [41] had already introduced the steps "observation," "pre-processing," and "inference," which the authors use here [10] to discuss the main challenges of PRE. They classify approaches by the message type inference method, the message format model, the behavior ("grammar") model, and whether the inference is active or passive. The survey presents at a glance the main contributions of each tool for PRE as a research field at one glance. Although Duchêne et al. introduce multiple steps of PRE, they classify and discuss the tools only by the methods used in the inference step. In comparison, we extend the definition of steps to a full process model of eight steps and characterize all surveyed tools according to the mechanisms and algorithms that each applies in each step individually. Thus, our discussion focuses on the concrete specific algorithms and methods employed by each tool for each step, while Duchêne et al. provides a high-level comparison of the different main types of solutions available for each inference step.

Bossert et al. [16] introduce and evaluate their own approach Netzob, but also compare it empirically to Discoverer [13], ScriptGen [29], and ASAP [35]. They do include a detailed analysis of the inner workings and employed algorithms of all three tools, but they do not discuss the other approaches in the same manner.

The present work provides a detailed, step-by-step process description, going well beyond the tasks in existing surveys, which all are limited to message type identification, format inference, and state machine construction. While Duchêne et al.'s three-step model of PRE is based on an idea similar to ours, it is not detailed enough to enable a comparison of the methods based on an underlying common process model. We provide this detailed process description, step-by-step, then discuss it and base the comparison of methods on it. In our work, we discuss thoroughly the PRE process, which we compiled from all surveyed works.

It is important to note that our focus does not lie on the monolithic tools themselves, their overall contributions, and their applicability, as this has been done extensively in the surveys of Li and Chen, Narayan et al., and Duchêne et al. Instead, we decompose each tool into its employed algorithms and methods, since those are the foundation of the tools' functionality. We classify, discuss, and compare these algorithms and methods per step. We argue that this more detailed look at PRE enables researchers to better understand the algorithms and methods used in the different steps of PRE, which should inspire new approaches via new combinations or by introductions of new algorithms.

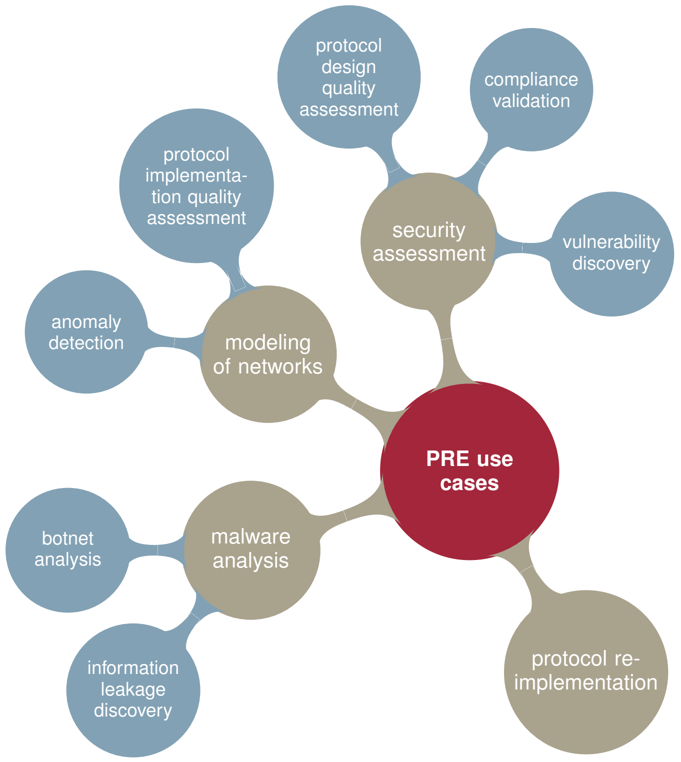


Fig. 2: Use cases for the analysis of unknown communication.

III. Classification of Protocol Reverse Engineering Tools

Several concurring classifications of PRE are possible. The general discrimination that we also used to define the scope of this survey is derived from Li and Chen [8]. They differentiate between PRE based on static traffic and entity analysis. They also offer the notion of a hybrid approach, which we, however, interpret as entity analysis that additionally incorporates traffic analysis. Therefore, we do not include these hybrid approaches in this survey, since they use distinctly different executable analysis methods, as argued above.

Another classification scheme is based on the **target use case** of the result, for which four general use cases of PRE are common, illustrated in Figure 2. First, during *malware analysis* it is required to understand whether and how malware communicates to its masters, especially for *information leakage discovery* and in *botnet analysis* where bots are capable of two-way communication with a Command and Control Server [2]. Second, alternative *protocol re-implementations* of proprietary or legacy protocols require the protocol specification, which is unavailable and can only be deduced through observation of genuine communication partners. Samba was developed this way [42] as a substitute for an SMB/CIFS-server. PRE also helped in recovering shutdown online-services [43].

Third, proprietary protocols with undisclosed specification are commonly used in network and point-to-point connections for industrial control systems and the Internet of Things. It is common for business customers to require *security assessments* of purchased products before they are deployed in their network. To estimate the *protocol design quality* [1, 44], to *validate the compliance* of the implementation to legal and organizational requirements [1, 45], and to *discover vulnerabilities* [1, 46, 47] requires a depth of understanding of the protocol, which is typically not intended by the product vendor, i. e., the vendor does not entrust the original protocol specification to the customer. Wireless control links for medical implants are an example of proprietary communication protocols that need to be assessed for a thorough security analysis of an IT-infrastructure or device [1]. A common task in security assessments is to specify a honeypot- or fuzzer-configuration. The manual modeling of a protocol, even for a known specification, may be too great an effort. Therefore, resorting to PRE helps to instead generate the model [15, 47].

Finally, various means for *modeling of networks* are used to identify the source of network problems in terms of functionality or performance and to discern between normal and irregular behavior during network monitoring with the goal of *anomaly detection* [48]. *Protocol implementation quality assessment* investigates discrepancies between specification and implementation of known protocols that may lead to incompatibilities and security vulnerabilities [44]. Obtaining a meaningful model requires the inference of the actual protocol primitives. The type of target use case subsequently determines the kind of necessary information about the protocol and the desired level of detail.

In this survey we classify tools based on the **portion of the protocol model** that is the **analysis goal**: Protocol parts that are of interest denote our first two categories (A) **message format** (Section IV-A) and (B) **behavior** (Section IV-B). In addition, there are tools that include both aspects of PRE, which we subsume into a third category, (C) **format and behavior** (Section IV-C). This classification scheme, which we apply to STA frameworks and to tools, is illustrated in Figure 3. To support this classification, we relate use cases to the named categories. The use case of each PRE approach also governs the portion of the protocol under test that is required to be reverse engineered. Since a protocol specification consists of multiple parts, protocol reverse engineering tools may target a specific part depending on the use case. Common targets are either the *message format* or the *protocol behavior*. The format describes the syntax of the byte sequence of a message, typically by denoting protocol fields and thereby structuring the contained data. The protocol behavior characterizes sequences

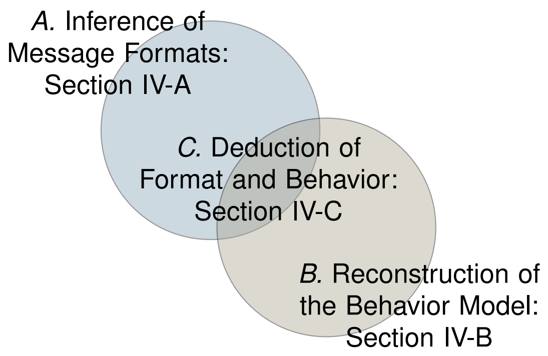

A. Inference of Message Formats: Section IV-A
C. Deduction of Format and Behavior: Section IV-C
B. Reconstruction of the Behavior Model: Section IV-B

Fig. 3: Classification of Static Traffic Analysis Tools

of messages to be valid and meaningful, like a message for retrieving data only being successful after a valid login message. Whether format or behavior is relevant is a central aspect of the analysis; known in advance, this may be used to decide which tools to choose. Therefore, we selected the supported tasks of the approaches as high-level categorization, as Narayan et al. [9] and Duchêne et al. [10] also proposed.

IV. Frameworks and Tools

In this section, we present an overview of all existing tools or frameworks of static traffic analysis and elaborate on the tasks that are covered by each approach. For each tool, we outline in brief its assumptions, the tasks the tool addresses, and the tool's functionality. We focus on a decomposition approach to (1) derive common steps from all surveyed tools (Section V), and (2) identify and describe the algorithms and methods implemented in each tool (Section VI). We therefore do not provide a general discussion about the tools themselves but only a brief introduction. Thorough reviews of their overall contributions and applicability is provided in Narayan et al. [9] and Duchêne et al. [10].

As stated in the previous sections, static traffic analysis (STA) approaches work solely based on offline analysis of recorded network traffic traces. Within our scope of STA, we differentiate all STA approaches into those addressing only message formats, those addressing only the behavior model of the protocol, or those addressing both.

All mentioned approaches typically require the identification of *message types* as essential information even before the different formats of each type are inferable. Some approaches also provide solutions for additional analysis tasks, such as semantic deduction. We will highlight in detail these internal concepts and functionalities in the subsequent sections and then draw further conclusions.

A. Inference of Message Formats

In protocol reverse engineering, the following tools concentrate on the message format. This part of the protocol specification may be regarded on its own, since the message format is already sufficient, depending on the use case. Such use cases may require only an understanding of the message format, without knowing the meaning of the message, its contents, and the circumstances that determine them. For instance, this is true for fuzzing or for deep packet inspection by intrusion detection systems.

1) Protocol Informatics (2004): Beddoe [11] is considered the pioneer of automated PRE, due to proposing Protocol Informatics[1] (PI). He first applied sequence alignment algorithms, designed to align amino-acid sequences for bioinformatics, in static traffic analysis to infer message formats. A remarkable number of subsequent approaches base their analysis at least in part on the general ideas of PI; these include RolePlayer, Discoverer, the tool by Whalen et al., ProDecoder, ScriptGen, PEXT, AutoFuzz, and Netzob, all of which will be presented within this section. PI is intended to be applied to a large number of network traces of one protocol under test. It identifies message types by clustering and infers each independent message format by sequence alignment.

2) RolePlayer (2006): RolePlayer's [18] purpose is the adaptive replay of application communication. Basically, it allows for a previously seen portion of communication to be replayed in a different context. To that end, RolePlayer uses single messages from traces to determine a template for replying to a seen message. Since no statistical variance in values can be observed in comparing only two messages, heuristics and human effort are employed to determine message fields of interest. Finally, sequence alignment is applied to a pair of messages to find variable fields.

3) Discoverer (2007): Discoverer was a project conducted by Cui et al. [13] at Microsoft Research[2] for reverse engineering of application-level message types and formats. Discoverer's design goal is a fully automated analysis from network traffic traces only, without the need for any additional information about the protocol. The concepts of RolePlayer were employed in Discoverer, and its sequence alignment techniques are closely related to Beddoe's work [11]. Since Cui et al. argue that parameters for sequence alignment are difficult to determine, especially for variable-length fields, they propose an alternative for both: clustering and field inference. As a solution, they introduce a type-based alignment algorithm using the distinction between sequences of bytes in the range of ASCII-encoded values in contrast to arbitrary byte values to determine message formats and types.

4) Whalen et al. (2010): This an alternative approach to specifically analyze message formats of static traffic traces. Whalen et al. promote the use of probabilistic graph models to enhance the inference of message formats. The tool requires an initial separation of sequences common throughout similar messages, which can be provided by sequence alignment. The authors give a comprehensive introduction to the utility of hidden Markov models (HMMs) for automated protocol learning, thus they derive ϵ-machines, which provide deterministic transitions between states, while the states then represent parts of a message format in graph representation.

5) Biprominer (2011): Biprominer (BInary PROtocol MINER) [20] specializes in the format extraction of binary protocols. The analysis process performed by Biprominer consists of a learning phase, iteratively alternating with a labeling phase, finalized by the generation of a transition probability model. It realizes a statistical analysis of the messages to identify common byte values sequences as format building blocks, representing protocol field candidates.

6) ProDecoder (2012): ProDecoder [21] was designed in 2012 to address the limitations of Discoverer in inferring the message types and formats of communication protocols. Compared to Discoverer, it is intended to be applicable to asynchronous protocols; it does not assume that a fixed number of leading bytes of a message are distinctive for a message type, and it does not assume the existence of delimiters.

[1]see http://www.4tphi.net/~awalters/PI/PI.html for PI's implementation

[2]The same team developed the entity analysis tools Polyglot [4, 36], Dispatcher [49], and Replayer [50]

Proposed by the same team of authors, ProDecoder is conceptually similar but internally unrelated to Biprominer [20]. At the same time, ProDecoder outperforms Biprominer due to its support for keyword-based, human-readable protocols. The identification of message types is accomplished by clustering messages according to common keywords across a subset of messages. Afterwards, within each cluster, the message format is derived using progressive sequence alignment.

7) Li et al. (2015): Li et al. [22] present a method similar to ProDecoder that determines common message parts for then deducing message types that derive from them. Afterwards, they use association analysis to discover relations between the extracted keywords. Thus, they extract words that occur frequently and that are strongly related, and thus may represent the message type; but the authors do not, however, investigate the contents of other parts of the messages nor do they use a clustering strategy.

8) FieldHunter (2015 and 2016): Bermudez et al. [23, 24] developed a tool to specifically extract fields and their types. Depending on the protocol type, they extract fields by identifying delimiters as frequent non-alphanumeric sequences or by dividing messages into tokens according to the inferred field type. Afterwards, they identify message type fields and other semantic meanings by calculating the entropy and then check for causal relations of fields. As a foundation for their assumptions about typical field characteristics, the authors investigated common protocols. They deduced design decisions for fields in these protocols to obtain hypotheses on field sizes and behavior.

9) Cai et al. (2016): An idea similar to Whalen et al. [19] for extracting message formats was proposed by Cai et al. [25], relying completely on Markov Models. In contrast to Whalen et al., the new approach requires no previous knowledge of field boundary candidates. Therefore, Cai et al. transfer all messages into a complete Hidden Semi-Markov Model of single bytes and map the underlying Markov process to fields. Afterwards, message types are inferred by clustering messages according to their message format.

10) PRE-Bin (2016): Tao et al. [26] developed the tool PRE-Bin, which extracts the message format of specific common protocols. They proposed a completely new approach for identifying different message formats: PRE-Bin uses clustering for format inference and proposes a silhouette coefficient to determine the optimal number of clusters. Then it derives indications on field boundaries by sequence alignment which the authors adapted to the special characteristics of their target protocols. In the last step, a Bayesian model determines the most likely positions of field boundaries by analyzing the frequency of gaps in the message alignments.

11) Xiao et al. (2016): Xiao et al. [27] discussed the challenge of format extraction. They describe a method for keyword identification by the frequency analysis of field contents. The approach relies on the separation of fields by delimiter characters.

12) NEMESYS (2018): Kleber et al. [28] propose NEMESYS as a method of structure inference for binary protocols. It derives field boundaries of individual messages from the distribution of value changes in the byte sequence.

B. Reconstruction of the Behavior Model

The following approaches focus on the reconstruction of the behavior model of a protocol, which is also known as protocol grammar or state machine. For the emulation of a protocol, e. g., as a honeypot, the understanding of the behavior is sufficient to reply to a message, in a way similar to an already observed message exchange. Use cases requiring replay of message sequences can also be served this way.

1) ScriptGen (2005, 2008): ScriptGen's [29] main purpose is to generate honeypot scripts by analyzing static traffic traces. The generated scripts may then be deployed on a honeypot to serve a protocol not even known to the security tester. ScriptGen was the first tool tailored to building the state machine of a communication protocol from traffic traces. Leita et al. embedded their work into SGnet [51], a distributed honeypot deployment. ScriptGen builds a state machine of observed message sequences, which is simplified and generalized afterwards. ScriptGen therefore merges common transitions and states according to clusters determined by means of PI and Region Analysis, one of its own contributions.

2) PEXT (2007): Shevertalov and Mancoridis [30] designed PEXT to infer the behavior model of the actual operation of a protocol by static traffic analysis. PEXT clusters messages to identify message types, which are used as states for the behavior model; it derives and minimizes a protocol state machine from the states cited above.

3) Trifilo et al., (2009): An approach to extract a state machine of the behavior for binary protocols from static traffic traces was proposed by Trifilo et al. [31]. The approach compares byte values at the same position across messages for their variance and thus identifies bytes, which then determine the message types. The basic idea is that only certain fields of a protocol message define the message type, and those fields can be used to create and refine a behavioral model.

4) Veritas (2011): Veritas [32] is an approach to automatically infer protocol behavior from network traces; it was conceived in 2011 by Wang et al. Veritas extracts the protocol state machine, without any knowledge about the protocol, using statistical methods for clustering and defining states. Veritas introduced its own state model to describe the protocol state machine in a probabilistic manner.

5) PREUGI (2017): Xiao and Luo [33] proposed PREUGI, a scheme to derive a protocol state machine. The performed grammatical inference is based on the concept of error-correction in combination with the statistical probability of message sequences. The approach contains a method for simplification of the state machine to remove redundant states using negative samples.

C. Deduction of Format and Behavior

The following approaches are designed to infer all main protocol specification parts: Message types, message formats, and the behavior model. Some approaches also presented ways to support basic semantic deduction of message content.

1) AutoFuzz (2010): The first tool which aims at deriving both message format and state machine of unknown protocols is AutoFuzz [34]. It was not designed primarily as a PRE approach, but for smart-fuzzing the implementations of an unknown protocol to identify unexpected behavior and potential security vulnerabilities. AutoFuzz therefore works as a proxy server that needs to be configured at an active client and server to extract the state machine of a protocol and to infer the message formats. Using the inferred protocol model, AutoFuzz acts as a smart fuzzer to test protocol implementations of communication nodes by manipulating messages and communication sessions.

2) ReverX (2011): Antunes et al. [14] introduced ReverXto infer format and state machine of a protocol from network traces; it does so by constructing and generalizing one state machine, representing the formats, and another state machine for the protocol behavior. The approach is limited to text-based protocols, while the authors proposed a set of small changes to support binary protocols in the future.

3) PRISMA (2012), extending ASAP (2010): PRISMA ("PRotocol Inspection and State Machine Analysis") [15] presents a method to infer a stateful model of a protocol based on the analysis of recorded traffic. This model is intended to simulate valid communication of the inferred protocol for use in malware analysis and the deployment of honeypots. The approach includes the inference of the protocol behavior, message format, and field semantics. Whereas Whalen et al. apply a probabilistic graph model for syntax inference, PRISMA contains a probabilistic approach to model the protocol behavior. It was developed by Krueger et al., incorporating their earlier work called ASAP [35] ("Automatic Semantics-aware Analysis of network Payloads", 2010). The inference of field semantics is addressed by the discovery of information propagated between protocol states.

4) AutoReEngine (2013): AutoReEngine [17] is based on keyword extraction to identify message types and formats, thus it derives states of the protocol behavior. Since this tool's analysis method is to identify high-frequency strings, only the most prevalent message contents are considered as keyword. Likewise, only message sequences which prove most frequent are retained to describe the behavior. Thus, the resulting model may be considered an intermediate result on which to base a more targeted protocol analysis.

5) Netzob (2014): Netzob is a tool crafted specifically to support an analyst throughout the process of protocol analysis. It is the most versatile and production-ready tool publicly available, capable of message type and format inference, behavior model reconstruction, some semantic deduction, and even protocol simulation. Netzob's architecture offers different methods and provides an easily extensible framework to match the great diversity of real network traffic. Moreover, Bossert et al. [16] presents its own enhanced clustering and sequence alignment approach, based mostly on Beddoe's concept [11], and an innovative dynamic traffic analysis for the behavior model reconstruction.

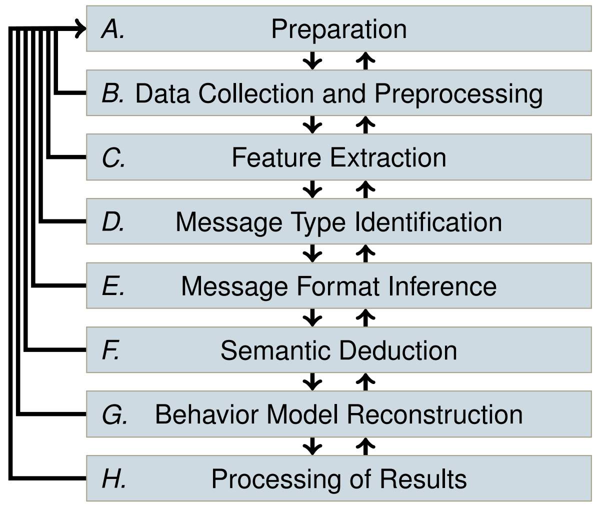

A. Preparation
B. Data Collection and Preprocessing
C. Feature Extraction
D. Message Type Identification
E. Message Format Inference
F. Semantic Deduction
G. Behavior Model Reconstruction
H. Processing of Results

Fig. 4: Illustration of the process steps

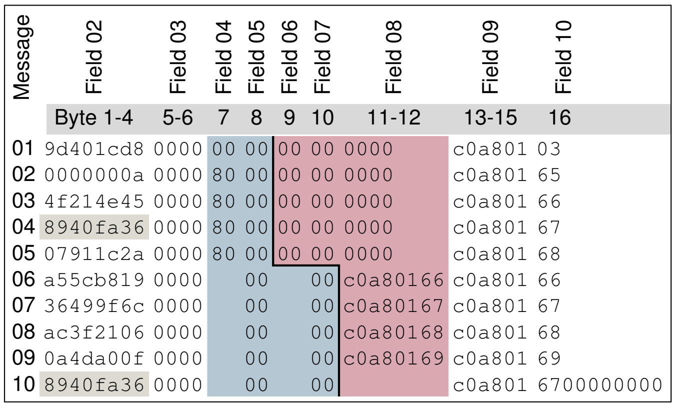
V. Static Trace Analysis Process Model

In the previous section, we reviewed tools concerning static traffic analysis. By identifying and correlating all the tools' building blocks, we determined the common process model, which we present in the current section. The Appendix provides graphical representation of all the discovered tools' and frameworks' building blocks and the mapping to each of the process steps of our model.

The protocol reverse engineering process model, which we present in the current section, is one of the contributions of our paper. It is deduced from the surveyed literature as the common foundation of the sequence of tasks to be performed during PRE using static traffic analysis. Although it may appear to be the natural thing to do, no explicit process model has ever been established as a reference for all approaches. Consequently, no classification according to a common process model has been possible, which hinders comparison and understanding.

We define the *purpose* of each step of the process and conceptually describe *methods* that are applicable to each step. At the end of each subsection (in *Challenges*), we discuss caveats, lessons learned, and practical experiences that an analyst will need to perform the analysis step; this discussion is intended to provide higher-order insights about the challenges of each step. After we have introduced and discussed all eight steps, we provide, at the end of this section, a mapping of the functions of the surveyed state-of-the-art tools to the steps they perform within the newly outlined process model.

An outline of the steps—also illustrated in Figure 4—is as follows: Before starting a new analysis, *preparations* must be made. Data, particularly traffic traces, must be *collected and preprocessed*. *Features* must be *extracted* from each message to determine similarities between messages. *Message types* must be identified and their *formats inferred*. Observed messages are abstracted towards general message types to represent them as groups of similar messages. *Semantic* comprehension, deduced from correlations in messages, profits from correctly inferred message formats. Finally, the recon-



Message	Field 02	Field 03	Field 04	Field 05	Field 06	Field 07	Field 08	Field 09	Field 10
	Byte 1-4	5-6	7	8	9	10	11-12	13-15	16
01	9d401cd8	0000	00	00	00	00	0000	c0a801	03
02	0000000a	0000	80	00	00	00	0000	c0a801	65
03	4f214e45	0000	80	00	00	00	0000	c0a801	66
04	8940fa36	0000	80	00	00	00	0000	c0a801	67
05	07911c2a	0000	80	00	00	00	0000	c0a801	68
06	a55cb819	0000		00		00	c0a80166	c0a801	66
07	36499f6c	0000		00		00	c0a80167	c0a801	67
08	ac3f2106	0000		00		00	c0a80168	c0a801	68
09	0a4da00f	0000		00		00	c0a80169	c0a801	69
10	8940fa36	0000		00		00		c0a801	6700000000

Fig. 5: Different kinds of misinterpretations in the field structure of an excerpt of messages of the binary protocol DHCP.

struction of the protocol *behavior model* identifies sessions and associates message types with protocol state transitions. The *results* need to be *processed* to prove suitable for the respective use case. Several approaches propose repeating certain steps multiple times to refine findings.

A. *Preparation*

1) Purpose

The purpose of this first step is to define the *desired result* of the analysis. The analysis goal may be a required level of correctness, conciseness, and coverage of any of these aspects: message type identification; message format inference; semantic deduction; or state machine generalization. This guides the selection of methods and their parameters in subsequent steps. Moreover, available *knowledge about the protocol* and the environmental *context* in which it is observed, are assessed. This facilitates a more efficient analysis and aids in estimating whether the defined analysis goal can actually be reached.

2) Methods

The preparation can be categorized into two aspects: assessment of information and the scope of the analysis.

a) Assessment of Information: A protocol might completely lack specification or it may be sparsely documented. Furthermore, an informal specification in natural language (e. g., RFC) may be available, while the effort to formalize it is too high. On the other hand, the specification may be available but the implementation may be unknown [44].

Another source of information is the diversity produced by the context of the communication. Diversity in the message content is necessary to find structure, and it provides additional information, especially for semantic deduction of values and behavior. Consequences of missing diversity are illustrated in Figure 5: In the example, the series of zeros (red boundary) are erroneously aligned with the unrelated 8th byte of other messages also having a value of zero. Actually, these zeros of bytes 9 to 12 are part of an unset IP address field. Considering Figure 5, all of the IP addresses are from a local subnet (192.168.1.0/24 reading `c0a801` in hex), leading to the interpretation of a static field 09 and one subsequent variable byte.

b) Define Scope: A well-defined goal determines the desired level of detail, e. g., if it is necessary to know the address semantic of fields or if it suffices to know that these are 4-bytes long. IP addresses are examples for fields with such semantics and are shown in Figure 5 at bytes 9 to 12 and 13 to 16. Furthermore, the goal needs to include measurable criteria of the quality of the results. For a protocol with at least a partly known specification, like the number of message types, fields per message type, or interrelations of states, it is possible to define quantitative criteria. As such criteria, Narayan et al. [9] adopt the evaluation metric of Cui et al. [13] to compare the inferred protocol with the specification to determine correctness, conciseness, and coverage aspects for the identification of message types.

3) Challenges

To prevent futile analysis attempts, general limitations of STA should be considered. For instance, compression or encryption may be prohibitive. If a first inspection of the raw trace data exhibits discernible patterns (e. g., Figure 5), no encryption or compression is expected. Both encryption and compression prevent a superficial preliminary analysis: If the message is compressed and the employed algorithm is unknown, patterns can be too obstructed to extract information from them, although the information is present in the trace. Encrypted messages can only be analyzed if the clear-text message can be obtained. This can be accomplished in two ways: If it is possible to manipulate the sending or receiving binary, messages may be gathered before or after encryption in the entity [10]. This approach requires memory introspection during the runtime of a program and is therefore not a method of static traffic analysis. Alternatively, a proxy can be used to record the decrypted messages by performing as Man-in-the-Middle between two genuine entities [46]. These methods require control over either the entities' execution environment or over the network topology.

Although the quality of the subsequent analysis steps depends considerably on the diversity of the context, especially the networking environment, no decisive set of parameters can be outlined here since there has been no investigation on the implications of the Preparation step. Instead, current approaches still rely much on human intuition. However, working with a distinct set of preparation parameters while considering the analysis goal would prove helpful in deciding about how much effort to make. Moreover, a structured preparation is beneficial in selecting methods and process steps to employ for the analysis.

B. *Data Collection and Preprocessing*

1) Purpose

This step of the PRE process provides and prepares specimens of the unknown protocol for the analysis. The data on which the analysis is based is either a previously recorded network trace, or if possible, traffic recorded in a network by the analyst himself. Therefore, this step comprises of setting up a network environment and recording traces, if this is possible, and furthermore reconstructing or defragmenting, filtering, and selecting parameters for further steps.

2) Methods

Typically, packet sniffers, like Wireshark[3] or tcpdump[4], are used to capture network traffic. To gain enough diversity for good coverage of valid messages and of content, the more data that can be gathered, the better the inference can become. However, capturing duration and performance considerations will limit the amount of obtainable data.

The identification of individual messages of the target protocol may not be obvious, depending on whether encapsulation of the protocol provides a virtual connection or allows for fragmentation. Therefore, if present, fragmented messages must be reassembled. If fragmentation information from an encapsulating protocol is unavailable, time gaps between messages can be used as a heuristic [15].

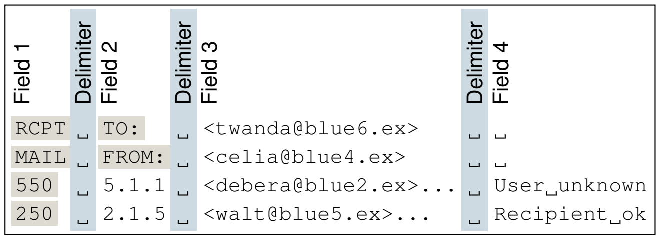
Unique messages that are too special and dissimilar from other messages must be eliminated since they tend to bias pattern recognition and thereby reduce the quality of the result. Figure 5 illustrates this in the last bytes of the last message, which were misaligned in this example since the last message was previously misclassified to be of a non-matching DHCP message type. Duplicates, retransmissions, and obviously invalid messages must be removed. Finally, interfering foreign communication and lower layer artifacts like acknowledgments must be filtered (e. g., using Wireshark). This works well in a controlled environment with no other traffic but might otherwise be incomplete.

3) Challenges

Although unknown protocols are the target of reverse engineering, some information is necessary as parameters for further analysis steps. Whether known, derivable, or guessed, the analyst has to obtain these parameters. One important information for the selection of the suited tool and its parameters is whether the protocol type is binary or textual. A protocol designed to be human-readable is referred to as a textual protocol, because in this case it consists mainly of encoded characters, such as ASCII. In contrast, many protocols are designed to be optimally machine-readable, using arbitrary byte values to encode any type of data; this is commonly referred to as a binary protocol. Further, a large number of ASCII byte values suggests the unknown protocol to be textual, otherwise it may be binary. Parameters that are determined from the protocol type include threshold values for clustering and field delimiters that are almost only relevant for textual protocols.

Filtering can remove large irrelevant portions of payload data, e. g., the mail body in SMTP, if such a distinction can be made. To enable this manipulation, and for the clustering and the state machine construction steps, sessions need to be identified from meta-data or the context, e. g., by larger gaps in the communication or sender-receiver role changes. Sessions are multiple message exchanges between entities denoted by protocol state transitions.

[3]https://www.wireshark.org/. All URLs last accessed on January 24, 2018.
[4]http://www.tcpdump.org/



Field 1	Delimiter	Field 2	Delimiter	Field 3	Delimiter	Field 4
RCPT	␣	TO:	␣	<twanda@blue6.ex>	␣	␣
MAIL	␣	FROM:	␣	<celia@blue4.ex>	␣	␣
550	␣	5.1.1	␣	<debera@blue2.ex>...	␣	User␣unknown
250	␣	2.1.5	␣	<walt@blue5.ex>...	␣	Recipient␣ok

Fig. 6: Keywords and delimiters of textual protocol SMTP.

C. Feature Extraction

1) Purpose

The essence of the analysis of traffic traces is to determine similarities and patterns within and across messages. As a precondition for finding groups of similar messages or correlating message patterns, features that quantify the similarity of messages and pattern sequences need to be extracted.

2) Methods

Due to the unknown structure of the byte sequences to be analyzed for features, applicable methods for Feature Extraction need to be able to efficiently correlate data. It is inefficient to conduct a plain value comparison of every possible byte combination in the trace, assuming a reasonable number M of 5000 messages in a trace with an average length N of 1000 bytes already takes $\frac{M(M-1)}{2}N^2 \approx 1.25 \cdot 10^{13}$ iterations just to compare this number of message bytes. Therefore, more sophisticated approaches need to be chosen. Guided by the assumption that differences in values are due to different message types, the typical strategy is to search for distinguishing n-grams or keywords (see Section VI-C). Tokenization and keyword extraction with statistical frequency and variance analysis of byte values are methods for this task, which are related to natural language processing. There are classes of automated feature extraction more suitable for binary or for textual protocols. Keyword extraction from n-grams, as well as general frequency and variance analysis of byte values, are applicable to binary protocols. For textual protocols, as illustrated in Figure 6, tokenization and keyword identification according to known field delimiters, like whitespace characters, have been established.

3) Challenges

Feature extraction is an integral prerequisite of a clustering algorithm to group messages. According to features determining the similarity of messages, message types are determined during the following Message Type Identification step. Defining features that reflect the similarity of messages of a protocol in a correct and efficient way is not trivial. For textual protocols, statistically determining keywords that distinguish message types from each other may be applied. Defining similarity between messages by recurring values throughout the content is typically used for binary protocols. There is no known definition of similarity which works for all kinds of protocols.

D. Message Type Identification

1) Purpose

The Message Type Identification step creates an abstraction of syntactically similar messages to find groups of semantically related messages. Assuming ideal correctness of the analysis, after grouping similar messages, each group contains many messages of only one distinct message type.

2) Methods

Manual message type identification is mostly based on intuition. Automated clustering, which applies methods from machine learning, uses similarity metrics extracted as features to relieve the analyst's manual workload. However, typical clustering methods require trial and error for suitable features and parameters since almost no information on the clustering result can be anticipated from the data in a trace or other sources. An identification of message types that is as correct as possible aids the analysis process, because then only related messages need to be compared. This reduces the complexity of a manual analysis, since groups of similar messages exhibit the same structure. Using the previously extracted features, either clustering or analytical methods are available to identify message types. Approaches are typically classified by whether they are suited for textual or for binary protocols.

3) Challenges

Depending on the use case, either groups of messages which exhibit the same syntactical structure, or groups of messages with similar meaning, may be desired. This can be accomplished by clustering according to the syntactical similarity or by keywords. By finding common keywords, a set of messages can be grouped according to their semantical similarity. Using syntactical similarity is in favor of the subsequent Message Format Inference step since the step's input is guaranteed to consist of clusters of similarly formatted messages.

Determining the structural similarity between messages typically results in a more fine-grained comprehension of the protocol; in contrast, finding semantic message types promotes the understanding of message purposes. Moreover, semantically similar messages may not be of the same message type and format, e. g., request and response messages may contain the same data, recognized as keywords, but at different positions in the messages. Figure 5 illustrates this in the last message, being a DHCP reply among DHCP request messages, which have a different address field containing a valid IP.

E. Message Format Inference

1) Purpose

A core step of the PRE process is the Message Format Inference. Common patterns and structure in the messages of each separate type are searched for in order to derive and characterize fields. Each field is defined by its position and length in the message and whether it is variable, ephemeral, or static. Moreover, each field can be characterized by its value domain, which can be numeric, ASCII, or a distinct list of enumeration values. Enumeration values have a specific implicit meaning, which is not contained in the message itself but rather predefined in the protocol. For example, `0x01` in DHCP's hardware type field stands for "Ethernet," `0x12` for "Fibre Cannel," and `0x20` for "Infiniband". This step results in known positions and lengths of presumed data fields in messages.

2) Methods

A multitude of methods to automate the message format inference have been proposed. They range from pattern recognition by delimiters, to sequence alignment, to n-gram analysis or keyword recognition. All results quality is highly dependent on the analysis type and how well the chosen analysis type matches the individual analysis situation.

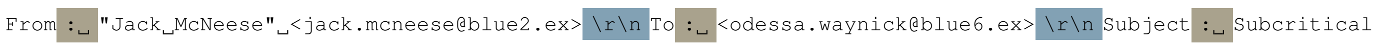
The most prevalent approaches are based on sequence alignment [11]. Sequence alignment lines up single occurrences of a certain byte sequence throughout two messages. To fully align all bytes of two messages by inserting gaps into either is called global sequence alignment. Multiple sequence alignment extends pairwise alignment to the global alignment of an arbitrary number of messages. Progressive sequence alignment is a kind of multiple alignment where the alignment gets iteratively refined by walking a hierarchical data structure. This may be achieved, for example, by propagating the alignment of subsequently merged clusters towards the root of a tree [52]. Finally, local sequence alignment only aligns the longest common sub-sequence (LCSS) of two messages. Applications of sequence alignment are discussed in Section VI.

3) Challenges

Static values may have certain functions in the protocol, be it delimiter, protocol version, or flags, such as how the first bit of byte 11 of a DHCP-message is defined to denote whether the message is unicast or broadcast. Coincidently, a field also may be of the same value for just the set of test traces, thereby not revealing any information. This confirms the importance of diversity in traces.

Known context allows for semantics-aware field identification: With known payload data, values originating in the context can be recognized and messages can be associated to sender-performed or to client-triggered actions. This way, values of a commonly known format, like email addresses, URIs, and file or host names reveal fields of the message format. When the meta-data is known, IPs or other addresses and timestamps can be pinpointed from the context information. For example, knowing the addresses of the involved entities allows one to conclude that Figure 5 shows these IPs in bytes 9 to 16. Connections can be associated with session identifiers and the communication direction. The peculiarity of such an approach is that it reverses the order of the Message Format Inference and Semantic Deduction steps since semantic relations need to be explored first to derive fields thereof.

Frequency and distribution analysis of characters allows for the identification of keywords or field delimiters. One big challenge is to determine the correct delimiters if multiple levels of them are present, such as separators for values inside of one field. To illustrate such hierarchical use of separators, Figure 7 is showing `\r\n` as field delimiter and a colon followed by a whitespace (`:␣`) as separator between type



```
From:␣"Jack␣McNeese"␣<jack.mcneese@blue2.ex>\r\nTo:␣<odessa.waynick@blue6.ex>\r\nSubject:␣Subcritical
```

Fig. 7: Single SMTP message showing two hierarchies of delimiter characters: `:␣` and `\r\n`

and value of the field. By constant and high frequency and distribution, a character becomes a candidate for a delimiter, while constant and low frequency may indicate a keyword. Semantics-aware fields identification marks the transition to the following step of Semantic Deduction.

The step of Message Format Inference provides first results that may be exploitable for a use case depending on the introspection of single fields of the message content. If no further information is required or expected from the analysis, the process may stop here.

F. Semantic Deduction

1) Purpose

A protocol is designed for transmitting information from one entity to another. To fulfill this purpose, semantic meaning must be associated with the protocol syntax. Semantic deduction is intended to reveal the semantic meaning of fields and messages in order to gain actual comprehension of the protocol. Common field semantics that can be derived automatically include, for example, length and offset fields, sequence numbers, address fields, and file names [13, 16, 21].

2) Methods

More than for any other of the PRE steps, semantic deduction is a time consuming and vague task. Due to the number of possible dependencies, the complexity of semantic inference is high; manual solutions can therefore only cover a small fraction of the search space. Only a few methods have been devised to ease the task by automation, and all approaches ultimately rely on the interpretation of the analyst. Considerations for this step include which kind of semantic dependencies to search for and whether a semantic field inference is preferable over a purely syntactic one. We discuss the challenges of discovering intra-message, inter-message, and environmental dependencies after explaining the difference between syntactic and semantic field inference.

Syntactic versus Semantic Field Inference: Message Format Inference and Semantic Deduction can be intertwined to promote the quality of the result by all available information. Two general approaches at the interconnection of syntactic and semantic field inference can be discriminated: Either the analyst ventures to (1) find semantics for fields previously inferred by format inference; or, (2) find fields by semantic deduction of expected values. On that account, the analyst uses context to identify information contained in messages and from that determines position, length, and representation of fields. Information that is expected to be found in the trace may be determined manually, or via automated dependency analysis of the meta-data from encapsulating protocols. Using semantic field inference exchanges the sequence of Message Format Inference and Semantic Deduction [16] in the PRE process.

3) Challenges

Semantic relations are classified into three types [16]. Environmental dependencies are caused by the communication context, like entity implementation, settings, and network topology. An inter-message dependency is a correlation between fields in different messages, like a sequence number. The term intra-message dependency refers to relations between multiple fields within one message, such as a field denoting the length of another.

Environmental dependencies: relate payloads or message meta-data to fields. Examples of these kinds of dependencies are addresses and port numbers from encapsulation, transmission direction, and node names. Environmental information can be searched for in traces by extraction from encapsulating layers, directly from the capturing process, and from knowledge about the context. Time of message transmission or receiving that are recorded with the traces helps identify timestamp fields. However, relating values in messages to environmental keywords must be regarded carefully, as short tokens often lead to false positives in binary data. For example, a netmask of 255.0.0.0 (`ff 00 00 00`) may be mistaken for an "End"-delimiter in DHCP (`ff`) typically followed by a `00`-byte field padding.

Inter-message dependencies: are detected by comparing different messages. Searching for fields which are incremented or values that are echoed allows for the detection of sequence numbers and ephemeral cookie or session-ID fields. Correlating message directions reveals additional information for the interpretation.

Intra-message dependencies: are detected by investigating whether the value of a field is reflecting a property of another field or the whole message, e. g., its length. Besides field or message length, checksums can be recognized by applying known checksum and hashing algorithms in brute force manner on portions of the message. Encodings of data are indicated by special patterns, like the lack of coherent zeros, as is typical for base64 and XOR encodings. For example, three zero-bytes are encoded as `AAAA\n` in base64 and using an XOR-pattern of `0x42 0x23`, the message would contain `0x42 0x23 0x42` in place of the zeros. Such patterns are recognizable if no encryption or compression is present, which both pose a show-stopper as explained in Section V-A.

G. Behavior Model Reconstruction

1) Purpose

The behavior of a protocol over time is typically expressed as a "Protocol State Machine" by assigning valid transmission sequences of message types to states and transitions. Assignment of protocol states and transitions to nodes (Moore machine) and edges (Mealy machine) is representation-specific and can vary between use cases, methods, and implementations. Types of states that are used to determine a protocol's condition are global protocol states or states of either entity. Global protocol states are representing the state of the whole protocol-run from an observer's point of view. For its representation message types are typically assigned to nodes and allowed message successions are denoted by edges.

2) Methods

Request and response cycles, semantic relations, and implicit connections (e. g., TCP or serial lines) allow for the reconstruction of valid message sequences represented by a state machine. State machine types used for PRE so far are Prefix Tree Acceptors and Markov Models. This can be automated using identified message types and can be correlated to their observed transmission sequences.

After the deduction of a state machine from observed message sequences, it needs to be generalized. The FSM initially accepts all observed sessions, it should, however, be able to accept other messages of the same message type as a state, even if it is not literally contained in the analyzed trace. This is accomplished by the merger of redundant and generalized states. Further generalization is achieved by finding potential loops, optional states, and states which can be accessed in arbitrary order. For automated approaches, heuristics determine what can be merged and generalized, and therefore the procedure is exposed to the risk of misinterpretation [30]. It is an inherent property of any state machine reconstruction from static traces that the necessary generalization can only be probabilistic.

3) Challenges

Building a state machine that can be generalized validly requires huge data-sets and exhaustive test cases. Each path through the FSM must be of, at the most, one session. Otherwise, the assumption of causal relationships between messages inside a session is wrong. For example, if interleaved messages—exchanged by one server with a client A that is in a state before having logged in and a client B that already did authenticate now starting to send data—are erroneously combined in one FSM, the result is a mixed up state machine with transition paths wrongly deduced as being valid inside a single session. Therefore, it is imperative to identify sessions from meta-data or context as explained in previous steps (Section V-B).

H. *Processing of Results*

1) Purpose

The defined use case of the analysis typically requires further manual and automatic processing of the analysis results. Therefore, processing of the analysis output initially consists of interpreting and arranging the results in a way that proves useful for the final goal of the analysis. Examples of what may be a useful representation are Regular Expressions derived from a list of aligned messages, a graphical depiction of a protocol state machine, or a fuzzer configuration to do vulnerability testing with the inferred protocol. The processing step includes the representation of the data in a suitable output format for the respective use case.

We provide use cases in the security context, their associated PRE goals, and the requirements about the kind of results as examples in the remainder of this section. A security assessment of deployed devices, common in penetration tests and fuzzing, needs a basic semantic overview of messages and fields and the behavior of the protocol. However, it is not necessary to know exact value domains or valid lengths for fields, even though, missing information, like undetected message formats, might lead to undiscovered vulnerabilities. Obtaining Intrusion Detection Systems' (IDS) training data requires discrimination between benign and malicious traffic. For this, the IDS typically does not need semantic information about fields, their value domains, the format of complete messages, or the protocol's behavior; identification of common syntax patterns is sufficient.

2) Methods

Output formats of the employed tools need to allow for the representation of data in rich formats to comprise all information. Challenges to correctly generalize field values to regular expressions or represent field variance information makes designing exporters non-trivial.

3) Challenges

To determine whether the analysis has been successful and whether the result meets the requirements defined in the Preparation step, measuring the quality of the result is necessary, although any metric faces an Oracle Problem. Since we typically cannot assume an Oracle for an unknown protocol and will not have enough information about the specification, estimations about the true protocol must be made. These estimations are highly protocol-dependent and no general solution has so far been proposed. Manual investigation and validation are possible due to human intuition, but this present state is unsatisfactory.

I. *Summary of the Process and Mapping to Tools' Functions*

The PRE process has eight distinct steps, of which seven can be automatized. The exception is Preparation, which inherently requires human decisions and manual setting up of the test context. As we presented in this section, each of the steps can be dealt with individually, making possible combinations of tools and methods; this way, the most effective methods can be selected for each step. To provide a solid basis for this selection of a method for each step, we provide an overview of the available methods that have been applied to PRE in the surveyed work. This following discussion allows to determine requirements and benefits of each method, either to apply the method or to identify tasks that have not been automated sufficiently and should gain attention in future research.

Based on the descriptions of tools and frameworks for static traffic trace analysis presented in Section IV, we reveal a mapping between these and their supported analysis steps based on the process model in Section V. We present this mapping in Table II. Our investigation of related work to find common PRE process characteristics, eventually resulting in the presented process model, shows most approaches consider

Message Type Identification and Message Format Inference. Behavior Model Reconstruction is regarded as stand-alone by some tools; in others, message format, type, and state machine are addressed by one combined approach. By this mapping, we conclude the presentation of our process model in the current section with illustration of the placement of all PRE tools and frameworks within this single PRE general process model and its steps.

VI. Solutions and Algorithms

In this section, we extensively describe all individual solutions and algorithms employed by all presented tools (Section IV); we extracted the process model (Section V) as common foundation. Knowing the universally applicable process steps, we are able to abstract from the individual tools and explain the different algorithms and solutions used by these tools to implement each of the various steps. In this section, the methods and algorithms applied in each of the tools are presented and explained, and each different case of applicability is compared. We categorize each of the proposed solutions into their nature and origin by placing them into respective subsections.

At the end of each section, we conclude the application of the presented methods and algorithms for the respective process step. This discussion is intended to provide lessons learned from the application of the methods. During each discussion we also attempt to provide higher-order insights about how the research on methods effected the step and what issues remain open for future work. The conclusions for each step discuss how the methods and algorithms can be applied practically to the step using the surveyed tools. This includes a survey of the tools' shortcomings and what challenges remain unsolved and need attention.

Tool/Approach	Data Collection and Preprocessing	Feature Extraction	Message Type Identification	Message Format Inference	Semantic Deduction	Behavior Model Reconstruction	Processing of Results
PI [11]		●	●	●			●
RolePlayer [18]	○			●	○		●
Discoverer [13]		●	●	●	●		
Whalen et al. [19]	○			●			
Biprominer [20]	○	●	●	●			
ProDecoder [21]		●	●	●			
Li et al. [22]		●					
FieldHunter [23, 24]		●	●	●	●		
Cai et al. [25]	●	●	●	●			
PRE-Bin [26]		●	●	●			
Xiao et al. [27]	○			●	●		
NEMESYS [28]		●		●			
ScriptGen [29]	●	●	●		○	●	●
PEXT [30]	○	●	●			●	●
Trifilo et al. [31]	○	●	●			●	
Veritas [32]	○	●	●			●	
PREUGI [33]			●			●	
AutoFuzz [34]	●	○	●	●		●	●
ReverX [14]	●	●	●	●		●	●
ASAP [35]		●	○	●			
PRISMA [15]	●	●	●	●	○	●	●
AutoReEngine [17]	○	●	●	●		●	
Netzob [16]	●	●	●	●	●	●	●

TABLE II: Mapping of Tools' Functions to Steps (● support; ○ limited support)

A. *Preparation*

For the Preparation step, there exists in the surveyed literature only sparse coverage of any specific methods. Nevertheless, all tools rely on the analyst's preparation of the PRE process, even if only the selection of the appropriate tool. Therefore, we cannot give a simple mapping of tools to the applied or required preparation tasks, like Table II or Table III. Instead, in this section we discuss in detail the merits of the surveyed approaches.

1) Assessment of Information

Some tools require a fairly detailed preliminary assessment of information before the approach can be applied in the first place. Therefore, all available sources of information about the protocol need to be analyzed. RolePlayer is an example for particular requirements during preparation, since it only compares two hand-selected messages. PEXT also specifically requires the preliminary selection of messages from a trace with representative input variations. AutoFuzz relies on so-called abstraction functions, which must provide a mapping of input messages to their message types to be applicable to a specific protocol. The analyst manually implements a Java-interface in order to specify such an abstraction function. How exactly to obtain the information necessary to provide suitable input and parameters for RolePlayer, PEXT, and AutoFuzz remains unspecified and, therefore, is entrusted completely to the analyst.

Most approaches assume that which kind of encapsulation is present is known, in other words, whether any enclosing layer adds header, trailer, or padding elements. Discoverer, ScriptGen, Trifilo et al., Veritas, PRISMA, and AutoReEngine require the encapsulation to be either a TCP/IP or a UDP/IP-stack. RolePlayer, PEXT, AutoFuzz, and ReverX are only

Notation in Tables: For a better overview, we accompany each section about a single step with a table which associates the tools to the solutions they implement. We use the following notation:

●	applied method
ⓐ	alternative option
ⓑ	used for binary protocols
ⓣ	used for textual protocols
ⓘ	used to yield an intermediate processing method during the current step in this approach
+F	method **also** used for Message **Format** Inference in this approach

applicable to TCP-encapsulated application protocols. In contrast, Whalen et al. propose to use entropy-analysis to discern between header and payload data and thereby to aid in the preparation.

Likewise, determining a protocol to be binary or textual is a completely manual process for most approaches. The selection of a tool suitable to either kind of protocol, such as Trifilo et al. for binary or AutoFuzz for textual, depends on knowing the protocol type before the actual analysis starts. PRISMA offers methods for textual and binary protocols, but requires the analyst to select beforehand which approach to choose. Discoverer is applicable to textual and binary protocols and determines the protocol type itself during its tokenization step for Feature Extraction.

Another kind of information necessary for all textual protocol analysis methods is to know the delimiter-chars used to separate fields. ReverX and PRISMA allow the analyst to specify the characters, which requires manual preparation.

Further parameters to be determined apriori may be similarity thresholds for clustering and the count of distinctive message types. The tools applying UPGMA and Single Linkage as clustering methods require a similarity parameter, and ReverX requires the number of distinctive commands of the protocol as a parameter. A possible approach for dealing with this is for an analyst to exploratorily apply the analysis to the same trace multiple times and then adjust the parameters repeatedly until the best result is yielded.

2) Availability of Entities

The availability of communicating entities governs the flexibility of the analysis and which approaches may be applied effectively. Three scenarios can be discerned for the kind of entity-availability. First, only *static trace files* are available, which is sufficient for most tools. Nevertheless, they still rely on preliminary filtering for only the subject protocol, without noise by unrelated network traffic. Second, if *passive network capturing* in a controllable environment is possible, captured traces are always from a desired known context and only of the target protocol. Additional semantic information can therefore be gathered, as is the case for Netzob's semantics-aided format inference or for RolePlayer's reliance on manually specified context values. Likewise, AutoFuzz requires capturing traces on its own, acting as a proxy, which is only possible having two running entities available. Finally, it may be the case that the *entities themselves are controllable* via some accessible interface. This allows for resetting them into a known initial state from which to start the analysis.

If a captured protocol is encrypted or compressed, actions to circumvent the security or compression measure are necessary. If information sources are available for obtaining encryption keys and for determining encryption or compression algorithms, the corresponding decryption and decompression method may be applied before analyzing the protocol. For example, Wireshark[5] offers such functionality for WPA and TLS among other common encryption layers. Other means to obtain encryption keys may be possible if two entities are available in a controlled network environment: By altering the forwarding or routing rules, a Man-in-the-Middle can be set up that yields the decrypted messages. Measures to bypass security may fail if they have been implemented properly for the protocol's entities. Correctly applied security measures may completely prevent static traffic analysis as discussed in this survey and as noted by all presented tools. However, unencrypted traces of protocols that need to be reverse engineered can most often be obtained.

3) Conclusions

The Preparation step **inherently** consists of **manual tasks** that provide decisions, information, and parameters for the conduct of further steps. Of the surveyed tools, only a few are concerned with possible methods and sources to **obtain the information necessary to start** the analysis. Moreover, one significant challenge is **how to deal with encrypted messages**, for which none of the surveyed tools propose an evaluated solution. The methods to investigate toward this solution depend on the availability of entities and should consider how to hook into an entity to gather decrypted messages and then compare it to the Man-in-the-Middle approach.

B. Data Collection and Preprocessing

Capturing and preprocessing of network traffic traces are considered standard tasks, not specifically regarded by automated approaches. However, some PRE approaches may require specifically prepared traffic traces, e. g., to take care of connection and session handling, and may therefore provide an integrated solution for capturing and preprocessing raw messages. Approaches which propose such specific capturing and preprocessing methods are discussed below and summarized in Table III.

1) Recording and Filtering

To capture network traffic, several so-called network packet analyzers are well established, foremost among them being Wireshark and tcpdump. The tools RolePlayer, Xiao et al., PEXT, Trifilo et al., ASAP, and PRISMA describe specific ways to apply tcpdump[6] for recording and filtering traces. tcpdump's library module libpcap is directly integrated into ReverX and Netzob for recording traces. For the analysis of application protocol messages, re-ordering and re-assembling of message fragments from packet flows, such as TCP-segments assuring ordered data transfer, are required. ScriptGen proposes libnids[7] for this kind of pre-processing.

gt ("ground truth"): Most PRE tools expect their input to be of one protocol only. How mixed traces should be filtered for the unknown protocol is not addressed specifically, except for Biprominer and Veritas. Both approaches propose to filter traces for a specific protocol by the tool *gt*, proposed by Gringoli et al. [53]. *gt* assigns labels to messages corresponding to the process which handled the message. This requires monitoring of a communicating host's kernel, and therefore requires full access to the entities' runtime environment.

[5]wiki.wireshark.org/HowToDecrypt802.11 and wiki.wireshark.org/SSL

[6]www.tcpdump.org

[7]libnids.sourceforge.net

Entropy analysis: Typically, arbitrary payload-data is not a target of PRE; that is, determining the coding of a compressed picture within the payload section of a protocol is beyond the task of PRE. To optimize PRE to the protocol format at hand and to exclude highly entropic payload data, such as images and movies, Whalen et al. propose to use *entropy analysis* for the discrimination of payload-data from the protocol header. To accomplish this, Whalen et al. calculate the Kullback-Leibler divergence [54] of message content. The Kullback-Leibler divergence originally measures the distance between probability distributions and is also known as relative entropy.

2) Context Inspection

The context of the communication may reveal information that cannot be obtained in any other way. The collection and analysis of context information is used for session distinction and the revelation of environmental semantic dependencies.

Session distinction: To be able to handle not only fragmented messages, but also multiple sessions of conversations between one or multiple pairs of entities, the analysis needs to discern individual sessions. In general, two approaches for this can be differentiated: using meta-data from protocol-encapsulation or conducting temporal analysis of the message transmissions. Cai et al., PEXT, Veritas, ReverX, and AutoReEngine perform session identification based on *context data*. They use tuples of addresses, direction, transport layer protocol, and ports in different combinations. Cai et al., Xiao et al., Veritas, ReverX, and PRISMA use statistics of *temporal gaps* between message arrival times to separate sessions from each other for further analysis; by the same method, applied within one flow, the tools separate distinct messages from each other.

A similar idea is the basis for the action-frame detection based on Gargi [55] and used by Netzob for session slicing. An action is a single purposeful event: it can either trigger an entity to send a message or it is the event that a message triggers in the entity. An example for the first kind of action is the event triggered by a user when clicking on a button while a message is sent. The second kind of action may be the application updating a UI-element with data received in a message. Action-frames may reveal semantic correlation between single purposeful entity actions and message content sent during this action-frame.

Environment semantics: The automated action-frame identification is not the only option for Netzob to obtain context information for further analysis. Netzob allows and RolePlayer requires the input of manually collected context information to relate actions to the content of messages. The semantics introduced into the protocol content by the environment can then be traced. While Netzob uses the additional information to improve clustering by semantics, RolePlayer requires manual input of this very specific context information in order to work.

Approach	tcpdump (manual)	libpcap	libnids	gt [53]	Entropy analysis [54]	Context data	Temporal gaps [55]	Collect context (manual)
PI								
RolePlayer	●							●
Discoverer								
Whalen et al.					●			
Biprominer				●				
ProDecoder								
Li et al.								
FieldHunter								
Cai et al.						●	●	
PRE-Bin								
Xiao et al.	●						●	
NEMESYS								
ScriptGen			●					
PEXT	●					●		
Trifilo et al.	●							
Veritas				●		●	●	
PREUGI								
AutoFuzz								
ReverX		●				●	●	
ASAP	●							
PRISMA	●						●	
AutoReEngine						●		
Netzob		●					●	●

TABLE III: Solutions for Data Collection and Preprocessing

3) Conclusions

Capturing network traces is a fairly standard task, therefore, most tools do not include their own functionality for this. Nevertheless, **filtering is a relevant challenge** since most PRE tools expect their input to be of one protocol only. Their performance and result quality are therefore highly dependent on the noise produced by other protocols. Only Biprominer and Veritas address this challenge and apply gt. Neither tool, however, gives details about gt's application to meet either of their input requirements.

Besides a noisy trace, analysis performance is also negatively impacted by lacking variance between messages and large irrelevant payloads that do not contain information about the protocol itself. Besides Whalen et al., no approach considers how to **deal with irrelevant payloads**. However, they always need to be taken care of before the analysis, otherwise the large amount of data without information about the protocol itself severely reduces performance and result quality. Typically, though, no conclusive information to discriminate payload can be obtained. Additional means of payload detection have not been proposed in any of the literature surveyed.

RolePlayer requires and Netzob supports the collection of context information besides the traffic traces. In both cases, the entity application's execution needs to be observed and messages need to be labeled by correlating actions and messages in time. While the benefit for the analysis has been shown, no method to **automate the collection of context** has been discussed in the surveyed literature. The analyst determines these actions through literally taking notes of the application's execution. Automated action-frame distinction by instrumentation is only possible for entities that can be placed inside a debugger, or inside a virtual environment, to be observed and is, strictly speaking, no longer a static traffic analysis method. There is no distinct research about how to apply binary instrumentation to this end.

To **summarize our conclusions** on the Data Collection and Preprocessing-step, only very few authors discuss its challenges, and even fewer tools sufficiently support it. Several authors only give a vague idea of what the analyst needs to do when preprocessing the obtained traces to effectively apply the tool. Only session identification is addressed specifically by several tools, either by regarding information from the encapsulating protocol or by assessing context data, like action frames or temporal gaps between messages. The other tools presented in Table II only propose general advice for the use of external preprocessing methods. This poses an obstacle for beginners, as well as researchers' and practitioners' adaption of a tool for a new use case.

C. Feature Extraction

To identify patterns in the recorded messages for further analysis, especially to derive message types thereof, the similarity between messages needs to be determined. The most prevalent sub-steps to complete this task are tokenization and the calculation of similarity or distance metrics for the subsequent Message Type Identification step. Table IV provides an overview about the application of algorithms by the surveyed tools.

Abstraction function: An abstraction function generalizes messages by extracting the distinguishing properties of one message using *apriori* information about the protocol. Instead of applying an automated approach for measuring the similarity of messages, an analyst manually provides an abstraction function. AutoFuzz uses this simple manual approach to identify similarities in its subsequent identification of the message types. In contrast, all further methods in this section provide some kind of automation to abstract messages.

1) Tokenization

Breaking down protocol messages into smaller parts is a common first step. This renders feasible the analysis and enables the application of methods that expect small basic analysis units as input. A common term for this splitting into parts is *tokenization*, for which two methods can be discerned: generating n-grams and splitting on delimiters [56].

n-grams: In natural language processing, the generation of n-grams is a common tokenization method. The generated n-grams for a byte-sequence contain all possible subsequences of length n of that byte-sequence. ProDecoder, Li et al., FieldHunter, and Veritas perform this kind of tokenization; Discoverer, ASAP, and PRISMA do as well as an option for binary protocols. However, Discoverer uses a simplified application of this tokenization scheme by regarding only single bytes for binary protocols.

Splitting on Delimiters: Delimiting message fields by reserved characters is a design pattern of textual protocols only. Therefore, splitting messages on delimiters is a kind of tokenization applicable to only textual protocols and requires the apriori knowledge about which characters or byte values qualify as delimiters for a specific protocol. This approach is available as an option to analyze textual protocols in Discoverer, FieldHunter, ASAP, and PRISMA.

2) Keyword-based and Statistical Analysis

One possibility to define the similarity of messages is to identify keywords, which sometimes are just called words. In this context, keywords are agglomerated tokens which are characteristic for a specific set or type of messages. Identifying discriminative keywords is typically accomplished by statistical tests on the keyword distribution across the protocol messages. A statistical (hypothesis) test in the context of PRE is a means to determine statistically significant deviations of the occurrences of tokens within the analyzed messages. This way, the null-hypothesis of whether a message is of a specific type is checked. The following statistical tests and methods have been applied to STA.

Latent Dirichlet Allocation: ProDecoder and Li et al. employ the established Latent Dirichlet Allocation (LDA) [57] algorithm to extract keywords from n-grams. LDA is an approach from natural language processing to identify words which appear together frequently and coherently. The identification of a set of keywords from the LDA-results is a classic Bayesian inference problem, posing the drawback that one parameter cannot be solved directly. Therefore, ProDecoder and Li et al. apply LDA in conjunction with Gibbs sampling [58], which is an iterative Markov Chain Monte Carlo method able to efficiently find an approximate solution. The benefit of using LDA for feature extraction is that not every token needs to be regarded individually for clustering, which significantly increases its performance.

Kolmogorov-Smirnov test: The Kolmogorov-Smirnov test (KS) [59] measures the similarity of two samples, e. g., two sets of n-grams. KS is a statistical test, which accepts if two sets of samples have a similar probability distribution of their items. Veritas uses KS to find common n-grams according to their distribution of the occurrence frequency in messages. KS is repeated while increasing the frequency threshold until the test accepts. Veritas uses the result to determine message parts as being type-distinguishing keywords, which are distinct for each message type.

Veritas uses this method in conjunction with the Jaccard index [60] to calculate a similarity metric. The Jaccard index determines the number of elements that two sets of items have in common. In this case, the message parts are the set of items that the Jaccard index intersects.

Statistical t-test: A well-known statistical method for measuring the similarity of a set of samples is the t-test [61]. It is a hypothesis test to determine if two sets of data are significantly different from each other. ASAP's and PRISMA's application for PRE determines if a string is neither constant nor completely volatile, while simultaneously coping with data diffusion and noise. Thus, frequency and volatility of n-gram

Approach	Abstraction function	n-grams [56]	Splitting on delimiters [56]	LDA [57]	Gibbs sampling [58]	Kolmogorov-Smirnov test [59]	Jaccard index [60]	t-Test [61]	Holm [62]	Pearson [63]	Apriori [64]	FP Growth [65]	Variance Distribution	Bit Congruence [66]	Viterbi [67]	k-means clustering [68]	Needleman-Wunsch [69]	Smith-Waterman [70]
PI																		●
RolePlayer Discoverer		ⓑ	ⓣ															
Whalen et al.																		
Biprominer													●					
ProDecoder		●		●	●													
Li et al.		●		●	●							●						
FieldHunter		ⓑ	ⓣ															
Cai et al.															●			
PRE-Bin																●		
Xiao et al.																		
NEMESYS														●				
ScriptGen													●				●	●
PEXT																		●
Trifilo et al.													●					
Veritas		●				●	●											
PREUGI																		
AutoFuzz	●																	
ReverX													●					
ASAP + PRISMA		ⓑ	ⓣ					●	●	●								
AutoReEngine											●							
Netzob																	●	

TABLE IV: Solution Application for Feature Extraction

values can be deduced. ASAP and PRISMA combine the statistical t-test with Holm's correction [62] and the Pearson correlation coefficient [63]. In this case, Holm's correction allows multiple testing at the same time for a string to be non-constant and non-volatile. The Pearson coefficient integrates multiple correlating tokens into a single one. Thus, the t-test becomes a similarity metric, based on frequencies of keywords with token correlation.

Apriori: By using keyword frequency analysis, words that distinguish message types may be identified. To this end, the Apriori data-mining algorithm [64] assumes that a frequent string's substrings must also be frequent. Therefore, the algorithm identifies frequent individual strings and expands them to a maximum coherent string. AutoReEngine extracts keywords by a modified Apriori using the keywords to identify message types.

FP Growth: Besides Apriori, FP Growth is one basic algorithm for association analysis. It uses an extended prefix-tree structure to efficiently analyze frequent patterns, and it outputs the complete set of found patterns and their relation. These frequent and related keywords are assumed to determine the message type for both binary and textual protocols in the approach of Li et al.

Distribution of Variances: Besides the five presented statistical keyword analysis methods, a purely statistical metric is the analysis of variances in the message sequences. The following PRE tools introduce their own method for such an analysis. ReverX uses byte-value variances as a similarity metric, while Trifilo et al. introduce the analysis of the "Variance of the Distribution of Variances" of bytes. The same basic idea of byte-value variances is applied in ScriptGen's Region Analysis, essentially being tokenization based on the value frequency, variance, and other byte characteristics. Likewise, Biprominer performs a coherence analysis of the byte value frequency to identify keywords.

Bit Similarities: Not completely unlike the Distribution of Variances, NEMESYS uses value patterns of each single message to identify the message's structure. These value patterns are calculated from bit similarities between subsequent message bytes. The specific similarity measure applied by NEMESYS is based on Sokal and Michener [66] and extended to a feature called Deltas of Bit Congruence.

Viterbi Algorithm: To determine keywords, Cai et al. use a probabilistic modeling approach. The tool initially uses a Markov model (see Section V-E) to infer the message format. The tool thereby swaps Message Format Inference and Message Type Identification of the typical PRE process. To deduce keywords from this Markov model, Cai et al. use the Viterbi algorithm [67]. This recursive dynamic programming algorithm yields the most probable sequence of hidden states of a statistical process. Cai et al. use it to extract the keywords from the observed bytes of messages and to assign the keywords to the hidden nodes of a hidden Markov model.

k-means Clustering: A different approach to determine a message characterization is to use a clustering algorithm on its own. A clustering algorithm proposed for this kind of approach

is k-means [68]. k random data points, called centroids, are defined, and each message in the dataset is associated to its nearest centroid. The algorithm therefore indicates the similarities of messages, which PRE-Bin uses in place of a feature extraction method. The difficulty of this clustering approach is in defining the optimal value for k. PRE-Bin addresses this problem with a silhouette coefficient [68]. It iteratively repeats the k-means clustering with random centroids and increases k each time. For each result, the silhouette coefficient is calculated, which validates the consistency within clusters. The value for k with the highest silhouette coefficient will be used to derive the optimal number of subspaces with similar messages.

3) Sequence Alignment

An early metric applied in PRE to determine the similarity of messages is sequence alignment, where alignments of multiple byte-sequences are conducted to determine the amount of difference between the sequences. For the two alignment algorithms applied in PRE, Smith-Waterman and Needleman-Wunsch, this difference is defined in terms of byte-matches, mismatches, or the necessity to insert a gap in one sequence to achieve alignment of the rest of the messages. To decide which byte-values should be considered as similar, a score-function ϕ is defined which assigns positive or negative penalty values for match, mismatch, and gap. The similarity between two messages, then, is determined during a traceback step, after the actual alignment by the maximum of the scores for a match, mismatch, or gap. Thereof, an $N \times N$ similarity matrix for each pair of messages $n_{i,j} \in N$ is generated.

Needleman-Wunsch: Needleman-Wunsch [69] is a dynamic programming approach, guaranteed to find an optimal global alignment of two messages. Netzob uses progressive multiple sequence alignment based on Needleman-Wunsch as a similarity metric for the following Message Type Identification. Progressive multiple sequence alignment allows for iterative alignment of more than two sequences, but at the cost of high computational complexity: $\mathcal{O}(N^M 2^M)$, for M sequences with a maximal length of N [71]. ScriptGen proposes a kind of token-type alignment, based on Needleman-Wunsch, for determining the similarity of messages during a phase it calls *Region Analysis*. Needleman-Wunsch-based sequence alignment is more commonly used for Message Format Inference than for Feature Extraction.

Smith-Waterman: Compared to the global alignment algorithm Needleman-Wunsch, Smith-Waterman [70] is a local sequence alignment method, determining the longest common subsequence (LCSS) between sequences. It is not suitable if message types are expected to differ in multiple positions, as is the case for multiple variable length fields; however, it has a slightly lower complexity of $\mathcal{O}(N^M)$ for M sequences with a maximal length of N. Smith-Waterman is used by PI, ScriptGen, and PEXT as similarity metric for message types.

4) Conclusions

The use of abstraction functions in the fashion of AutoFuzz does not help solve the challenge of feature extraction and cannot be regarded as an automated method. Automated approaches first require the breaking down of messages into smaller parts. Governed by the assumption that differences in byte values at specific positions of messages are due to different message types, **different approaches have been designed for textual and binary protocols**. While for binary protocols keyword extraction from n-grams and sequence alignment work well, for textual protocols, tokenization and keyword identification according to known field delimiters have proven effective. To be able to split on delimiters, their byte value needs to be known in advance. Most approaches assume white-space characters as delimiters, but this is not necessarily true for all protocols, rendering these tools ineffective.

Statistical methods, typically employed for natural language processing, like LDA (ProDecoder), the Kolmogorov-Smirnov test (Veritas), the t-test (ASAP and PRISMA), and Apriori (AutoReEngine) in principle work for both textual and binary protocols. However, the results of tools using statistical methods for binary protocols are not as convincing as for textual ones. **Frequency and variance analysis** of message byte patterns have been applied by Biprominer, ScriptGen, and Trifilo et al. to reveal properties of binary protocols. As an exception, ReverX successfully applies a similar kind of byte value analysis to textual protocols. **Bit Similarity** analysis is similar to the analysis of byte values with the exception that it does not require comparison of multiple messages to extract the feature. This comes in favor of the analysis performance.

Using sequence alignment to determine the similarity of messages is much less common than finding structure in similar messages. This is due mainly to the lack of any kind of template for the alignment, since we are dealing with unknown protocols. Without a template, multiple sequence alignment is necessary but has a high complexity, prohibitively for even medium-sized sets of several hundred messages. Therefore, **sequence alignment** using Needleman-Wunsch or Smith-Waterman is **not well suited for feature extraction**.

Handling feature extraction by a clustering algorithm is an unorthodox approach taken by PRE-Bin in using k-means. According to Tao et al. [26], the result quality of this approach for binary protocols is superior to variance distribution, sequence and type-based alignment, and keyword frequency analysis.

To summarize the Feature Extraction step, it is relevant how well the method matches to the specific properties of the unknown protocol—in particular textual or binary. This is important, since the result quality of the subsequent PRE steps is highly dependent on the kind of feature extraction. Larger traces generally lead to better results—especially in the Message Format Inference—since more diverse examples of messages can be included in the analysis. At the same time, with most current methods, the performance deteriorates exponentially with the trace's size. Therefore, a critical challenge is to find suitable criteria about the similarity and structure of messages—most prevalently for binary protocols. The surveyed literature has no conclusive solution to selecting the most appropriate method for the individual analysis situation.

D. Message Type Identification

The identification of message types is based on message features extracted in the previous step. By determining similarities, tools have attempted to group messages into their types. Available algorithms for this task are clustering approaches from machine learning and self-contributed ad-hoc grouping methods. The application of clustering with and without distance metric, as well as grouping with novel ad-hoc methods, was contributed by several of the PRE tools. Table V shows at one glance which tool applies which of the algorithms.

$$
\begin{array}{c|ccccc}
 & m_1 & m_2 & m_3 & m_4 & m_5 \\
\hline
m_1 & 0 & 4 & 6 & 2 & 9 \\
m_2 & 4 & 0 & 3 & 6 & 3 \\
m_3 & 6 & 3 & 0 & 8 & 6 \\
m_4 & 2 & 6 & 8 & 0 & 3 \\
m_5 & 9 & 3 & 6 & 3 & 0
\end{array}
$$

Fig. 8: Distance matrix: 0 denotes equality; the higher the value, the more dissimilar the messages are.

1) Ad-hoc Methods without Distance Metric

To identify message types, the following PRE tools contributed different straightforward ad-hoc methods.

Keyword analysis: To analyze messages based on keywords which were derived in the Feature Extraction step, some methods that differ slightly were proposed. The corresponding approaches Discoverer, Trifilo et al., Veritas, and AutoReEngine have in common that they determine the characteristic sets of tokens or keywords for each message type. Terminology for the distinctive keywords, however, differs. Discoverer calls the keywords “Format Distinguisher.” In Veritas, the “Protocol Format String” contains the sequence of keywords specific to a type. AutoReEngine calculates the “Minimum Position Variance” of keywords as distinctive property. FieldHunter calculates the entropy for every field. Thus, if the same fields in different messages are neither constant nor random, the mutual information of corresponding response and request message pairs are examined. This indicates that the fields represent the semantic of the whole message and are therefore considered as Message Type fields. In PREUGI, how to determine keywords characteristic to a message type is completely up to the analyst. PREUGI expects to be provided with a known set of tokens to search for in the trace.

Search for context data: If knowledge about the environment of the protocol-run is available as discussed in Section V-A, it can be utilized for message type identification. RolePlayer and Netzob search the messages for context data which was derived from the environment, e. g., a known username. If it is known that a value only occurs during a specific functionality, like a username is only transmitted within a login message, the according type of that message is discernible. Additional message types can be identified when recognizing implicit information in values, e. g., associating the broadcast-flag of DHCP with the Ethernet addressing mode used to disseminate any frame of this type. RolePlayer also uses the result of the context search for the inference of the message format.

Approach	Keyword analysis	Search context data	Path through format-graph	UPGMA [66]	Single Linkage [72]	Partitioning around Medoid [73]	Affinity Propagation [74]	Non-negative Matrix Factorization [75, 76]	Information Bottleneck [77, 78]
PI				●					
RolePlayer		●+F							
Discoverer	●								
Whalen et al.									
Biprominer			●						
ProDecoder									●
Li et al.									
FieldHunter	●								
Cai et al.							●		
PRE-Bin				●					
Xiao et al.									
NEMESYS									
ScriptGen				●					
PEXT					●				
Trifilo et al.	●								
Veritas	●					●			
PREUGI	●								
AutoFuzz				●					
ReverX			●						
ASAP + PRISMA					ⓑ			ⓣ	
AutoReEngine	●								
Netzob		●		●					

TABLE V: Solutions for Message Type Identification

Path through format-graph: A completely different approach for type identification and format inference is the graph-based method. This method represents a model of all message formats by a state machine. Each path from the starting to the final state is one message type, while the traversed states, labeled with n-grams or words, determine the message format. For format inference, the type identification basically is a by-product of the generation of the graph from the traffic trace. Therefore, there is no explicit type identification, but each type is a distinct path through the format graph. The tools that use this method are Biprominer and ReverX.

2) Clustering with Distance Metric

Most of the clustering methods from machine learning in use for PRE are hierarchical agglomerative clustering approaches with a similarity metric of messages. This approach recursively merges (“agglomerates”) pairs of clusters in a hierarchical generalization, up to a defined similarity threshold. This requires a metric for the similarity or dissimilarity of messages. A commonly used representation of pairwise similarities for i messages m_i is a distance matrix as shown in Figure 8. Such clustering approaches are susceptible to noise in the input data and therefore require thorough filtering of input messages during the Preprocessing step. Although not specifically named, the clustering algorithm used by Discoverer and

ProDecoder very likely is a hierarchical agglomerative one, according to the explanation given. Other approaches explicitly name UPGMA and Single Linkage.

Unweighted Pairwise Mean by Arithmetic Averages (UPGMA): UPGMA [66] is a hierarchical agglomerative clustering method, having the optimal complexity of $\mathcal{O}(n^2)$ for this class of problems. It progressively locates the smallest entry in the distance matrix and merges all entries which intersect in this cell by calculating their mean values. In this course, a phylogenetic tree is generated, representing the affinity between clusters. This approach is taken by PI, PRE-Bin, ScriptGen, AutoFuzz, and Netzob for message type identification.

Single Linkage: Like UPGMA, Single Linkage clustering [72], also called nearest neighbor clustering, is a hierarchical agglomerative method of complexity $\mathcal{O}(n^2)$. Single linkage recursively merges clusters based on the distance between the closest pair of members of each cluster. PRISMA uses it as a clustering method for binary and PEXT uses it for binary and textual protocols.

Partitioning around Medoid (PAM): In contrast to hierarchical clustering, PAM [73] tries to find partitions in the set of input data to cluster it according to a distance metric. PAM iteratively assigns one arbitrary sample per cluster as its center, the medoid. In continuously reassigning the medoid, the recalculated distances between the other samples and the medoids are minimized for each cluster. Finally, the medoid has the minimal average dissimilarity to all other objects in the cluster. This approach requires the expected number of clusters as input, which typically is not known for unknown protocols. To overcome this, and to therefore be able to utilize PAM, Veritas uses a generalization of the Dunn index. The Dunn index is a quality metric of the clustering result. By maximizing this index, Veritas determines the number of clusters.

Affinity Propagation: Affinity Propagation [74] is a non-hierarchical clustering algorithm. It finds members of an input set that are representative for their cluster. The algorithm iteratively passes information about their properties between data-points based on a similarity matrix until the clusters emerge. The advantage of this approach is that no assumptions on the internal structure of the data are needed, which means that this approach does not need the number of clusters in advance, nor does it require a specific form of the similarity matrix. Cai et al. use this approach to identify message types by clustering all derived messages with similar message formats. To create the required similarity matrix, keywords are derived during Message Format Inference, which in this tool is performed before Message Type Identification.

3) Grouping without Distance Metric

Two methods of identifying message types have been proposed in the surveyed literature that are not conventional clustering approaches using a distance metric for similarity. Instead, they rely on algebraic solutions for information compression to reduce the problem dimensionality.

Non-Negative Matrix Factorization (NMF): Dimensionality reduction to allow for efficient pattern analysis can be achieved by NMF [75]. This method is applied in the related tools ASAP and PRISMA as an option for the analysis of textual protocols. A matrix X is generated, which contains a mapping of messages-to-token vectors. Similar messages are represented by geometrically close vectors, whereas different content results in larger distances. Applying the NMF, X can be represented by two smaller matrices, with $X \approx B \times C$; $B, C \geq 0$, which are easier to handle. Moreover, clusters can be derived directly from matrix C, as the values indicate how strongly a message belongs to a certain cluster. NMF is working well in PRE since the nature of the message-token association guarantees each entry in the matrices to be positive. To reduce the matrix size, the initial input for NMF can already be reduced by merging the tokens that always occur together [76].

Information Bottleneck method (IB): Slonim and Tishby [77] originally proposed this approach to automatically reveal topics of texts. IB provides a suitable trade-off between complexity reduction of input data and retaining relevant information. The data compression is accomplished by removing redundancy in word occurrences while revealing which data is relevant [78]. Therefore, ProDecoder uses IB to group messages on the distribution of relevant words. Given a joint word-message probability distribution, IB groups messages and words into clusters. By clusters of messages, clusters of words can be predicted, since messages of similar type contain similar words. Now, instead of regarding every word in a message trace, it can be filtered to only regard relevant keywords that are decisive for the message types.

4) Conclusions

The presented methods to implicitly and explicitly identify message types have been applied to STA. The tools share common ground, in that similarities are explored, and one or multiple dimensions of shared properties are searched throughout the set of messages. The differences among presented methods, and their strengths and weaknesses for the STA use case, are discussed in the following paragraphs.

Searching for and grouping by data originating from the communication **context** is a simple and efficient method from an algorithmic point of view. However, it requires a **large amount of manual effort** to gather relevant context in advance. It is impossible to automate the extraction of the context without extensive entity analysis to collect payload values and communication parameters from the communicating software process.

Typically, **clustering needs to be unsupervised** for application in PRE, since no known samples of correct message types are available as labeled data. Moreover, the **number of message types is unknown** in advance. Among conventional clustering approaches from machine learning, therefore, hierarchical agglomerative methods are often chosen. To allow a tool to discern message types based on similarity, it needs a threshold up to which messages should be regarded similar enough to be grouped. Despite the fact that the according clustering methods are unsupervised, they still need this **threshold to be specified by the analyst** in advance. Setting these values too low results in underspecified messages types and, thus, groups of unrelated messages. Choosing the threshold too high tends to over-specify clusters, splitting one message type into multiple clusters. Deciding whether clusters are over- or under-specified and deducing the optimum in between requires human introspection. Empirical trial-and-error evaluations show typical effective similarity thresholds are ranging from 30 % to 60 %, more dependent on the protocol than on the method. An exception to the application of hierarchical agglomerative clustering are PAM and Affinity Propagation. While Affinity Propagation works without an expected number of clusters, PAM requires the count of clusters as parameter. Therefore, Veritas **runs PAM multiple times** with a different cluster-count-parameter, assessing the results and choosing the best. With the exception of Affinity Propagation, **neither proposed clustering** method from machine learning is applicable in an **efficient** way, because many trial-and-error iterations are necessary to optimize the result.

Using paths through a format-graph, NMF, and IB each provide a different viewpoint for identifying message types in an advanced way. **NMF proves well-suited for textual protocols**, since keywords need to be extracted beforehand. IB was evaluated with both binary and textual protocols but has a large performance footprint. Following a path through a format-graph is easy and efficient but shifts the challenge to the graph generalization (see Section VI-E).

To summarize, there are multiple working options to group messages according to their type for textual protocols. However, coverage of grouping binary protocols is sparse. An important improvement for the type identification in future research would be to increase the overall performance and the automation efficiency for determining the optimal clustering result.

E. Message Format Inference

The inference of formats is intended to yield as its result the position and length of message fields. The methods available in the surveyed frameworks are presented in short in Table VI.

1) Sequence Alignment

Sequence alignment has been used as a similarity metric by some tools, as we laid out in Section VI-C. Since the computational complexity is high, this usage has been adopted widely. Instead, for the inference of the message format from clusters, comprising considerably fewer messages and with a presumably similar message syntax, sequence alignment has been applied numerous times in various ways.

Simple alignment: A simple kind of alignment is to just compare byte values based on their fixed position in a message. The position-based correlation of bytes or n-grams reveals field boundaries by distinguishing between static and variable values. This approach works only for protocols without variable-length fields. While the applicability to only fixed-length protocol messages is its main limitation, simple alignment works well for binary protocols which exhibit this message format paradigm. The tool by Trifilo et al. uses this kind of alignment, and it is also an optional approach in Netzob.

Delimiter-based alignment: Some protocols use delimiters to separate message fields instead of relying on a specific position of a value inside the message. Thus, fields are inferred by aligning each delimiter byte across all messages of one type. Byte sequences in between two separators are interpreted as a field, which can assume the respective values of a field in the same relative position in other messages. Only textual protocols use this paradigm, which limits the applicability of delimiter-based alignment to textual protocols only. To be able to align on delimiters, most existing approaches require the delimiter-char or -char-sequence for the protocol to be known apriori. Discoverer and Netzob use delimiter-based alignment as an option for text tokens. ReverX and Xiao et al. use only this approach, thereby specializing solely in textual protocols. FieldHunter uses alternative Message Format Inference steps for binary and textual protocols and extracts the delimiters automatically; they thus distinguish field delimiters from delimiters between keys and values, for example, `Port: 80, UID: 1234`. Field delimiters are identified as frequent non-alphanumeric sequences of one or two characters. Key-value delimiters are located between field delimiters and are identified by grouping the fields with the Longest Common Prefix algorithm and extracting the common suffix within the groups as a delimiter.

Needleman-Wunsch-based: In contrast to the application in Feature Extraction, Needleman-Wunsch (NW) [69] is applied for Message Format Inference to actually infer field

Approach	Simple Alignment	Delimiter-based Alignment	Needleman-Wunsch [69]	Type-based alignment	State-splitting (ϵ-machine) [79]	HsMM [80] by Baum-Welch [81]	Prefix Tree Acceptor [82]	“Matrix C^n”-template [83]	Delta Bit Congruence
PI			●						
RolePlayer			●						
Discoverer		ⓣ		●					
Whalen et al.					●				
Biprominer							●		
ProDecoder			●						
Li et al.									
FieldHunter		●		●					
Cai et al.						●			
PRE-Bin			●						
Xiao et al.		●							
NEMESYS									●
ScriptGen									
PEXT									
Trifilo et al.	●								
Veritas									
PREUGI									
AutoFuzz			●						
ReverX		●					●		
PRISMA								●	
AutoReEngine				●					
Netzob	ⓐ	ⓐ	●						

TABLE VI: Solutions for Message Format Inference

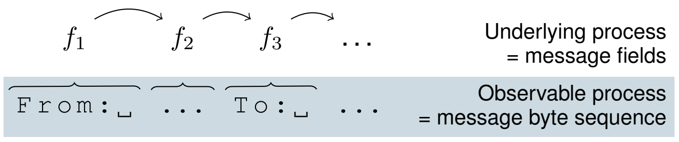

f_1
f_2
f_3
...
Underlying process = message fields
From:␣ ... To:␣ ...
Observable process = message byte sequence

Fig. 9: Field inference of a SMTP message by a hidden semi Markov model.

boundaries of one message type. For format inference, several Needleman-Wunsch-based multiple-progressive sequence alignment algorithms have been applied. The alignment is performed in a separate analysis step for each message type cluster. NW is applied this way by PI, RolePlayer, ProDecoder, PRE-Bin, AutoFuzz, and Netzob.

Type-based alignment: An alternative application of NW is not to align on single byte values but on tokens that have been previously identified as being a word of the message format. Such progressive alignment of tokens by NW, called type-based alignment, is proposed by Discoverer. FieldHunter uses this kind of alignment in a similar way and also regards the entropy of tokens as criteria for the kind of field type on which to align. The concept is related to ScriptGen's "Region Analysis," which this tool uses only for type identification. Moreover, AutoReEngine employs format inference by regarding keyword-field-patterns, which is also remotely similar to type-based alignment and Region Analysis.

2) Graph Generalization

In contrast, the following few alignment-free graph-based format inference methods have been proposed. They apply techniques similar to those of Behavior Model Reconstruction, discussed in Section VI-G, to derive message formats. Since graph representations are much more common for Behavior Model Reconstruction, we explain most of the general operations to manipulate graphs in that section. This method is unique in that it does not require an explicit type identification but provides an implicit one by itself as we explain in Section VI-D.

State Splitting algorithm: Whalen et al. apply the State Splitting algorithm by Shalizi and Shalizi [79] to messages for determining the format. The State Splitting algorithm is designed to reconstruct a hidden Markov model. Hidden Markov models, which are a kind of statistic graph models, are much more common for the representation of protocol behavior than message format. Therefore, we provide more details about them in Section VI-G. The State Splitting algorithm splits one initial state of such a model as often as necessary to satisfy a given statistical test. The specific kind of hidden Markov model, used by Whalen et al. is called an ϵ-machine. This ϵ-machine models the message format as causal states and a set of transition matrices. Its reconstruction therefore obtains the model without requiring any sequence alignment. The occurrence probability of specific message formats is retained and can be used to synthetically generate traffic of the inferred protocol.

Hidden Semi-Markov Model: Building a Hidden Semi-Markov Model (HsMM) [80] from messages also can derive the complete message formats. Messages represent the observable process, while each character is an observation. Compared to Whalen et al.'s application of the similar ϵ-machines, fields are blocks of observations as illustrated in Figure 9, not states. The hidden underlying Markov process indicates the lengths of fields as the state duration. Cai et al. introduce this approach and emphasize the advantage that these models can distinguish between strictly sequential fields and fields that have a juxtaposition location in the message. For example, in HTTP protocols, `GET` needs to be sequentially before `HTTP/1.1`, whereas the fields `Server:` and `Host:` can exchange their order. Whalen et al. cannot model this property of a field. To retrieve this information, Cai et al. propose to use the Baum-Welch algorithm [81], an established method to estimate the necessary parameters of a hidden Markov model by maximizing the likelihood of a given observation.

Prefix Tree Acceptor (PTA): In a PTA [84], a special type of state machine, all descendants of one vertex have a common prefix, leaving only the root starting without one. It can be used as another alignment-free graph-based message format inference. All byte-sequences of a message are assigned to different subsequent vertices. By adding state sequences for all observed messages, a PTA is generated, accepting all observed messages. By simplification and generalization (see Section VI-G), variable and optional subsequences are revealed and the message format is inferred. Particular for this method is that by inferring the message format, the message type is identified simultaneously. The only applications of this method in PRE are known in Biprominer and ReverX.

3) Algebraic and Analytical Solution

One algebraic and one analytical solution for format inference are known:

Non-negative Matrix Factorization: PRISMA applies the Non-negative Matrix Factorization (NMF) [83] for its protocol analysis (see VI-D). It uses the properties of the resulting factorized matrices to infer multiple aspects of the protocol. Because the template of each message is derived from matrix C, fields can implicitly be inferred from the NMF-result.

Delta Bit Congruence: NEMESYS determines field boundaries heuristically at recognizable positions in the value pattern of its feature Bit Congruence. These recognizable positions are inflection points at the rising edges of the noise-reduced Delta Bit Congruence; they denote field transitions with high probability.

4) Conclusions

For format inference of specific and simple protocols, byte-wise and delimiter-based alignments suffice. The application to a wider range of protocols requires more sophisticated methods, like the prevalent sequence alignment. However, **aligning multiple sequences is NP-complete** [85], therefore, Needleman-Wunsch offers a global optimal alignment at the price of an exponential runtime. On the other hand, protocols are designed by humans, so that structural patterns are bound to recur in multiple protocols. **Searching for design patterns** as heuristic for alignments has not been evaluated by any of the works surveyed. However, NEMESYS denotes a first step in this direction.

One alternative to alignment-driven format inference is the generalization of a format graph. It works well for textual protocols, and first attempts to apply it on binary protocols have been made. This method does not require an explicit type identification, but it provides an implicit one by itself. The challenge here is to **choose the appropriate generalization assumptions** for the type of protocol to be analyzed. These assumptions need to be made from knowledge about protocol design. Another alignment-free method is constituted by generating a template from an NMF. Although templates have a different goal in PRISMA, they also yield message format descriptions.

To summarize, the generic methods, like NW, are highly inefficient for large sets of messages. Assumptions about message formats are inherently necessary for graph generalization or heuristic pattern recognition. For example, ReverX and AutoReEngine expect messages to always exhibit keyword-value pairs. This pattern is true only for a small number of protocols, and therefore is too specific a limitation. NEMESYS makes a more balanced set of generic assumptions about protocol format patterns. Additional research on hypothesis-driven inference methods is needed to create an improved trade-off between efficiency and universality.

F. Semantic Deduction

The sparse coverage reflected in Table VII shows that deducing semantic information has gained only limited attention. The existing methods can be categorized as statistical and user-centric methods, as well as generic search.

1) Statistical Analysis

For the automated deduction of semantic relations between values and fields of messages, statistical means of analysis have been introduced.

Maximal Information Coefficient (MIC): MIC [86] is a metric for the strength of linear and non-linear associations between two variables. The resulting values of MIC range between statistical independence and strong relation between variables, making it possible to identify and characterize important relationships in data, thus discovering inter- and intra-message dependencies between field values (see Section V-F). To deduce the different kinds of dependencies, Netzob employs MIC in combination with the Pearson correlation coefficient.

Pearson correlation coefficient: While MIC reveals relations between variables, the Pearson product-moment correlation coefficient [63] differentiates between linear and non-linear correlation. Netzob uses both coefficients in combination to detect the kind of correlation of fields. It specifically supports linear inter-message dependencies in cookie and sequence number fields. Moreover, linear intra-message dependencies are discovered in size and offset fields. The only non-linear correlations Netzob supports are the intra-message dependencies of SHA-1-hashed or CRC32-checksummed content and the count field for the number of subsequent option fields in the message.

Mutation rate: The mutation rate is calculated by comparing the values at associated positions in multiple messages. It can be used to discriminate constant and variable tokens and is a very basic semantic of whether a value is variable when compared within its message type. This method is employed by Discoverer and by ScriptGen during its Region Analysis. Xiao et al. describe a simple frequency-rate analysis of field contents to identify command and parameter fields within the message format.

2) User-centric Analysis

With only few applied statistical analysis methods, semantic deduction strictly remains a human skill, at least up to the present day; it can, however, be supported by the suitable presentation of complex data and relations.

ϕ penalty score: Beddoe [11] suggested the penalty-score-function ϕ [52] of Needleman-Wunsch to be used for environment dependency detection. ϕ allows to define how similar differing byte-values are considered. By applying semantic tags to message bytes, Netzob allows to influence ϕ towards aligning known context values. As a by-product, these semantic tags can be interpreted after the alignment to

Approach	Maximal Information Coefficient [86]	Pearson correlation coefficient [63]	Mutation or frequency rate	ϕ penalty score [52]	Manual interpretation of *field types*	Manual interpretation of *message purpose*	Intra-message dependency search	Inter-message dependency search	Environment dependency search
PI									
RolePlayer								●	●
Discoverer			●				●	●	●
Whalen et al.									
Biprominer									
ProDecoder									
Li et al.									
FieldHunter								●	●
Cai et al.									
PRE-Bin									
Xiao et al.			●						
NEMESYS									
ScriptGen			●						
PEXT									
Trifilo et al.									
Veritas									
PREUGI									
AutoFuzz									
ReverX									
PRISMA						●			
AutoReEngine									
Netzob	●	●		●	●				

TABLE VII: Solution Applications for Semantic Deduction

recover semantics of fields. The actual interpretation, and the assignment of labels, however, is left as a manual task for the human analyst.

Content representation: The tedious manual interpretation of field types may be supported by a suitable and analyst-friendly content representation. This includes the application of value-encodings, since raw byte-data may obstruct an analyst from recognizing characters or numbers. Highlighting of field alignments and correlating values of these fields helps in manually revealing semantics. Netzob offers multiple configurable presentation modes for data to support manual semantic deduction.

Matrix B (NMF): The matrix B, generated during NMF, reveals the "topic" of messages. B is a representation of the messages' content that is semantically relevant for each specific message type. PRISMA's application of NMF supports the deduction of message purposes by manually determining the characteristic combinations of elements of the matrix B. Interpreting a message's purpose from these element combinations remains the role of the analyst.

3) Generic Search

If the Preparation step comprised an investigation of the environment of the protocol-run, or some parameter values are known which are reflected in the message content, a search for the known or derived values in messages can be performed. Conceptually, there are three options regarding what to search for to deduce semantics:

First, a generic search for *intra-message* dependent fields, such as length and offset, has been attempted. Discoverer searches for these by correlating field properties to other field's values. Second, a generic search for *inter-message* dependent fields, such as cookies, is proposed by RolePlayer, Discoverer, and FieldHunter. The tools correlate aligned values across messages to reveal dependencies between the messages. Finally, a generic search for *environment*-dependent fields recognizes values that are known from context. RolePlayer, Discoverer, and FieldHunter offer this semantic analysis method, which requires the extraction or manual input of relevant environmental context data.

4) Conclusions

Only few approaches have concerned themselves with semantics. Thus, with current approaches, only very simple semantics can be deduced automatically. More complex and less direct dependencies need human interpretation.

Among the few applied automated methods, the most sophisticated are the Maximal Information Coefficient, which Netzob uses to detect **dependencies between pieces of data within messages**, and PRISMA's application of Non-Negative Matrix Factorization to assist semantic interpretation of the **purpose of whole messages**. Moreover, highlighting differences between chronologically ordered messages is a common technique for manual interpretation of semantics by RolePlayer, Discoverer, and ScriptGen.

Non-linear dependencies need additional means of correlation. Such non-linearity occurs, for instance, when length fields are expressed as a multiple of bytes or when field values are an enumeration of predefined values. While the first case can be solved algorithmically, the second needs a lookup table that typically is neither known nor contained in the messages. Semantic interpretation is thereby generally limited by the available information.

Regarding the kind of **semantics of message parts**, the **coverage by tools** is as follows.

Environmental dependencies can be derived automatically, using RolePlayer, Discoverer, Netzob, or FieldHunter, searching for meta-information that may be reflected in field values. Encapsulating protocols can be sources of information that, e. g., contain source and destination IP addresses.

Inter-message dependencies, such as subsequently increasing or repeating values in consecutive messages, are identified as candidates for sequence- or session-numbers with Netzob and FieldHunter.

Intra-message dependencies revealed by Netzob and Discoverer are field lengths correlated with the preceding field value to determine their relation as length indicator, offset fields and counter fields for array-like field structures, as well as SHA-1 hashes and CRC32 checksums.

In summary, semantic deduction is vague in coverage, methods, and results. Although some promising first steps have been made by RolePlayer, Discoverer, Netzob, and FieldHunter, there are also severe limitations about how much of this step can currently be automated. Effectively, only Netzob's specific application of the ϕ penalty score and the Maximal Information Coefficient can be called general algorithmic solutions to search for semantics within messages. More domain-specific empirical research is needed about the appearance of protocol messages in transfer to deduce more complex and less direct dependencies. A large survey of message contents of known protocols may determine common properties of semantically similar fields, to recognize them more reliably in unknown protocols.

G. Behavior Model Reconstruction

One decision to make for the implementation of the Behavior Model Reconstruction step is the kind of state machine that can represent the protocol's behavior. Another central aspect of this step is the method of reconstructing the protocol state machine from the observed trace. Table VIII summarizes the existing graph-based representations and the reconstruction algorithms that have been employed for a protocol's behavior model.

1) Probabilistic Graph-based

Finite State Machines (FSMs) are the typical representation of protocol behavior to express the sequences of message types that are observed in multiple valid protocol runs between entities. State labels define the observed message types, and state transitions resemble their sequences.

Prefix Tree Acceptor (PTA): A PTA is a special type of state machine to represent valid message sequences. As mentioned in Section VI-E, ReverX also uses it for Message Format Inference. A PTA generated from trace observations

Approach	Prefix Tree Acceptor [82]	SMMDT	Protocol State Machine	Hidden Markov model [87]	Moore reduction [88]	Merge by negative samples	Generalization of PTA [82]	"State Splitting"	Apriori [64]	Angluin L* [89]
ScriptGen	●									
PEXT	●						●			
Trifilo et al.			●					●		
Veritas	ⓘ			●						
PREUGI	●					●				
AutoFuzz	●				●		●			
ReverX	●				●		●			
PRISMA	ⓘ			●						
AutoReEngine									●	
Netzob		●								●

TABLE VIII: Solutions for Behavior Model Reconstruction (omitting tools that explicitly do not address this step)

offers a complete model of all sequences of message-type occurrences. ScriptGen, PEXT, AutoFuzz, and ReverX use this kind of representation of the model. Veritas and PRISMA use a PTA for initial representation of the information contained in the traces, but ultimately transform it to a hidden Markov model. PREUGI represents the behavior of a protocol as sequences of characteristic keywords in different subsequent messages. Each of the keywords are then transformed into an Augmented Prefix Tree Acceptor (APTA). This is then refined recursively by an unnamed error correction mechanism in PREUGI. Initially, after its generation, a PTA is unable to distinguish between mandatory and optional messages and loops in the state machine. The PTA, therefore, is simplified or minimized, and generalized as follows.

Stochastic Mealy Machine with Deterministic Transitions (SMMDT): SMMDT is a specific extended kind of FSM, which Netzob introduced as an alternative to the PTA. It explicitly supports the representation of transition probabilities observed in the traffic.

Moore reduction: The Moore reduction [88] is a well-known minimization operation on PTAs. It merges non-distinguishable states, thereby simplifying the state machine. AutoFuzz and ReverX apply Moore reduction for simplifying their behavior model, while ReverX also uses it for its specific method of format inference. PREUGI minimizes its state machine using negative samples of similar protocols to merge states which denote the same operation.

Generalization of PTAs: The PTA needs to be generalized to accept not only the exact messages as they were observed in the analyzed trace, but any message of the corresponding message format. Generalization further simplifies a PTA by merging multiple vertices or edges [82] that are considered to represent the same transitions between message types. Generalization needs assumptions about what qualifies as the same message type, typically using the result of Message Type Identification. Equally, the identified message types are used in a heuristic to detect loops of message exchanges within state sequences. Additionally, states can be merged if their subsequent trees are isomorph. This kind of PTA generalizations are employed by PEXT and AutoFuzz, while ReverX again also uses it for its specific method of message format inference (see Section VI-E).

State Splitting: A special method of simplification of a state machine was contributed by Trifilo et al. Instead of constructing a large state machine with one state for each observed message transition and then afterwards merging the states, Trifilo et al.'s strategy is the opposite. Its initial state machine, created directly from the trace is tightly collapsed, representing different sequences by the same state. Therefore, Trifilo et al. contains a State Splitting algorithm[8] to reconstruct the minimal valid behavior-model state machine. It creates a generic protocol state machine without transition probabilities or structural constraints. To do so, sequences of states in the initial, collapsed state machine are split if these states actually represent not one but multiple portions of protocol behavior in the trace.

Hidden Markov model: Specific adaptions of hidden Markov models have been proposed as alternatives to PTAs for the representation of a protocol's behavior. A Markov model is a statistic graph model that exhibits the Markov property: Subsequent states only depend on the current state. In hidden Markov models [87], observations are related to a state of a Markov model, however, the current states are not always known, so they are called hidden. PRISMA and Veritas use hidden Markov models to retrace the observed messages to the underlying behavior. Veritas calls its adaptation of a hidden Markov model "Probabilistic Protocol State Machine" (P-PSM). Compared to the application of Markov models in Section VI-E, the model does not contain the sequence of bytes or fields of one message type, but the sequence of message types as they were exchanged between entities. In a hidden Markov model of a protocol's behavior, each state has a probability of producing a specific message. A sequence of messages, therefore, can provide information about the sequence of states. Among other properties, a hidden Markov model of the protocol behavior allows for the likelihood of the type of the next message to be calculated.

2) Natural Language Processing-based

Regarding the reconstruction method of the behavior, two approaches originating from natural language processing have been proposed.

Apriori: AutoReEngine employs the keyword frequency analysis by the Apriori algorithm [64], not only to identify message types (see VI-D), but also to reconstruct the protocol behavior. The algorithm is used to identify the most frequent sessions in the traffic trace. This approach will, however, only yield the partial solution of a list of the most prevalent examples of message type sequences and does not provide a general behavior model.

Angluin L:* In contrast to all other presented algorithms, Angluin L* [89] is a *dynamic traffic analysis* method. As an active grammar inference algorithm, it learns a protocol

[8]Own contribution of Trifilo et al. [31], not to be confused with the State-splitting algorithm by Shalizi and Shalizi [79] described in Section VI-E.

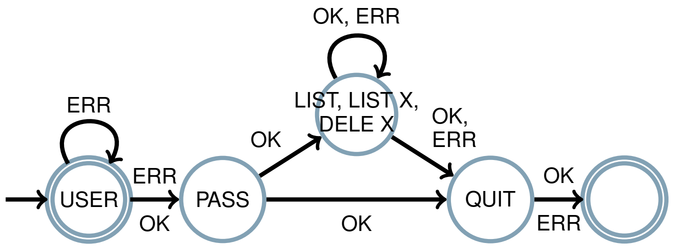


Fig. 10: A representation of a global protocol state machine.

behavior by probing a genuine entity communicating with the protocol under test. Netzob proposes to formulate a hypothetical model from the observed behavior and then refine the behavior model by querying an entity according to Angluin L*'s membership and equivalence queries. Via membership queries, the algorithm determines whether or not a specific variation of a message is also a valid member of the type at the current position in the behavior model. By equivalence queries, the completeness of the model, compared to the input trace, is estimated. The entity's responses are interpreted to provide a Teacher for membership queries and an Oracle for equivalence queries of the behavior model.

3) Conclusions

The reconstruction of a behavioral model requires the recognition of the message type of each observed message. Then, a given message type is interpreted as one state, so an **abstracted representation of each message type** is needed, as provided by the Message Format Inference.

In contrast to the behavioral model reconstruction during PRE, it is common for protocol specifications to separately define the states of each entity during the protocol run. This is well suited for implementation guidelines of the server and the client. Thus, nodes represent the internal state of the entity while waiting for or preparing the next message, and edges are transitions caused by receiving or sending a message. This is not the kind of desired result for PRE, where a **global protocol state machine**, as illustrated in Figure 10, is to be derived. A global protocol state machine is not specific to one entity but contains messages of both communication directions. The depicted example shows the request as states of the state machine, while the replies are annotated at the transitions.

Prefix Tree Acceptors (PTAs) are the **prevalent** application of state machines to represent protocol state relations. PTAs do not allow for alternative branches to be represented by the automaton, thereby preventing the introduction of ambiguities in the state machine. PREUGI promises an additional way to reconstruct the behavior of a protocol that is already partly known. Several details of the approach remain undisclosed, for example, how to determine the required set of protocol keywords and how to obtain a similar but unrelated protocol as negative sample. **Hidden Markov models** (HMM) enable natural inference of a protocol-state-machine representation, since the actual state machine cannot be observed directly. Current approaches use hidden Markov models with considerable **simplifications** to handle the complexity of the model. Nevertheless, these tools show **promise**. In addition, to overcome the challenges in deducing a hidden Markov model, sophisticated generation algorithms have been proposed by Veritas and by PRISMA. A direct comparison between the **suitability of PTA and HMM** to represent protocol behavior is **not available** in current research.

Another level of inter-message dependencies can be analyzed if the interrelations between state transitions can be specifically monitored over multiple states. PRISMA and the approach by Trifilo et al. are the first tools to offer this kind of functionality to track the history of message exchanges.

Natural language processing offers interesting alternative methods for the Behavior Model Reconstruction but pose several drawbacks. In particular, Apriori is used only to determine examples of valid behavior and it is unable to deduce a complete model. Netzob's application of Angluin L* is not a static traffic analysis method but the only known dynamic traffic analysis. Therefore, there are several exceptional requirements: The analyst needs access to a runnable entity and it must be controllable to ensure its deterministic execution and to reinitialize on demand the protocol to an initial state. Furthermore, entities need to fail in some obvious way, by crashing, for example, if the submitted sequence of symbols is not valid. If the entity is available and these requirements can be satisfied, it is a promising approach that should gain more attention in the future.

To summarize, the behavior reconstruction needs a suitable model and a method of simplification and generalization to abstract from the single analyzed trace. To represent the full understanding of the model, it needs to support transition probabilities and a horizon of state transition dependencies that can be higher than just the previous state. However, more research is necessary to improve the handling of the complexity in reconstructing such a potent model. For any use case of STA, it is vital to remember that the generalization of the model will always be incomplete because the traces can only contain positive samples. Therefore, dynamic traffic analysis is interesting but yet unexplored, aside from the proposition in Netzob.

H. *Processing of Results*

The representation of analysis results needs to contain message type, format, or behavior model in a way suitable for the specific PRE use case. The few solutions discussed in the presented frameworks are outlined by Table IX.

A basic approach for presenting the inferred message format is to *visualize* each message type as a matrix of messages against their aligned fields. Netzob offers this straightforward representation of aligned messages.

1) Formal Syntax Notations

The following syntax notations represent the analysis outcome regarding the format.

Approach	Visualization	Regular Expressions [90]	Consensus Sequences [91]	Substitution Rules	dot	Fuzzer configuration [92]	honeyd script [93]
PI			●				
RolePlayer				●			
Discoverer							
Whalen et al.							
Biprominer							
ProDecoder							
Li et al.							
FieldHunter							
Cai et al.							
PRE-Bin							
Xiao et al.							
NEMESYS							
ScriptGen							●
PEXT					●		
Trifilo et al.							
Veritas							
PREUGI							
AutoFuzz			●			●	
ReverX		●			●		
PRISMA				●			
AutoReEngine							
Netzob	●	●					

TABLE IX: Solutions for Processing of Results

Regular Expressions: Instead of domain-specific notations, the representation of aligned messages is achieved by Regular Expressions [90] in ReverX and Netzob. Thus, the common content of each field throughout all messages of one type is typically expressed.

Consensus Sequences: An alternative representation of aligned messages is Consensus Sequences [91]. Consensus Sequences originate in amino-acid sequence representations of molecular biology use cases and are employed by PI. AutoFuzz calls the same concept generic message sequences (GMS). Consensus Sequences show the most generalized sequence that results from the alignment of a set of messages.

Substitution Rules: Instead of representing common portions of messages of a whole message type, substitution rules denote individual sub-sequences of single messages that are variable throughout the type. This is useful for validly replaying traffic by replacing ephemeral and variable message content. RolePlayer and PRISMA use this kind of result representation, since it fits their use case of generating traffic from the inferred protocol.

2) Graph Notations

Only one solution to represent a state machine of the behavior was mentioned by any PRE tool.

dot: A typical result of Behavior Model Reconstruction is a state machine describing valid sequences of message formats. For the representation of state machines in visualized state graphs, ReverX and PEXT use the GraphViz format dot.[9]

[9]www.graphviz.org/doc/info/lang.html

3) Tool Input

The specific use case of the following PRE approaches is to provide input for another tool to make use of the knowledge gained about the protocol. Since the transition from the PRE tool to the subsequent application should be effortless, it needs to be automated.

Fuzzer configuration: One application of the gained protocol knowledge is the configuration of a smart fuzzer [92]. By varying the values of message fields during the replay of captured sessions, vulnerabilities in the entity implementation can be discovered. AutoFuzz provides an internal fuzzer that gets its configuration directly from traffic analysis.

honeyd script: A similar usage of the inferred protocol model is to use it as rules for a honeypot. A honeypot simulates a service for the assessment of attacks performed against this service. The integration of the protocol knowledge allows the honeypot to reply to a message sent by an attacker with a valid response, although the service and protocol were originally unknown. ScriptGen is designed to generate a Python representation of the protocol, compatible with the format that the honeyd-honeypot [93] expects as the rule-set of the service which it should thus emulate.

4) Conclusions

Formal syntax notations for message formats, such as BNF, are employed for protocol design [94], but have **not been regarded for reverse engineering**. Examples of output formats that may be suitable in different PRE use-cases are BNF/EBNF [95], finite state machines [96], or ASN.1 [97], which all currently are not found in PRE tools.

Tool support for the **output data type may be a relevant decision criterion** for a tool or framework in practical application. Specific import formats for other tools, like Wireshark and Scapy dissectors, would achieve improvement in efficiency [46]. Another usage scenario is that the result may be intended as configuration for the widespread Peach-fuzzer.[10] Since no such exporter exists in any tool, an individual implementation would be necessary. Both scenarios require a description file to be generated that represents field lengths, value domains, and other gathered information from the respective tool's internal protocol representation.

In summary, it may be desirable to gain a formal textual report as the output of the analysis. Alternatively, the analysis output may be required as input for a specific, different tool. No consensus has emerged from the existing approaches about how to represent the resulting protocol model.

The few propositions in the surveyed literature for results processing are limited in applicability. Therefore, **practical applications of PRE require additional manual effort to refine the result format**. This is due to the fact that all proposed protocol representations remove relevant information when abstracting result data. For example, abstraction of message formats to regular expressions leads to a long list of ORed values as they were observed in a field. Complete generalization, on the other hand, may remove information about the valid content or length of a field. The representation

[10]www.peach.tech

therefore needs to be augmented by additional human-readable annotations, which cannot, however, be interpreted automatically afterwards.

VII. Conclusion

We conclude by setting our survey into its practical and research context. Therefore, we validate our process model by applying it to existing approaches. We discuss different levels of utility of the static traffic analysis (STA) approach and protocol reverse engineering (PRE) in general. Finally, we identify open research questions and summarize conclusions to be drawn from our survey.

A. Validation of the PRE Process

In Section V we distilled the PRE process from an analysis of all the surveyed literature (Section IV). In Section V-I, we show how the varying tools map well onto this process. Additionally, the Appendix provides graphical representations of all the tools' and frameworks' building blocks to further illustrate the process model's validity. The figures show the sequence of tasks performed by each tool and our mapping to the aligned process-model step of that task. Most tools and frameworks follow the same sequence, although every one does not cover every step (see Table II). There are some tools that propose, however, to proceed differently. In the current section, we identify each of the distinct deviations and discuss the reasons for them. Therefore, it is important to recall here the steps of the process model: A. Preparation; B. Data Collection and Preprocessing; C. Feature Extraction; D. Message Type Identification; E. Message Format Inference; F. Semantic Deduction; G. Behavior Model Reconstruction; and, H. Processing of Results.

Of the approaches considered, three use methods that combine two steps into one but otherwise retain the order of the process. ScriptGen combines Behavior Model Reconstruction and Message Type Identification in its method for generalizing the Prefix Tree Acceptor (PTA), while AutoReEngine combines Message Type Identification and Message Format Inference during the mapping of messages to keyword sequences. Finally, ReverX uses a PTA as model for the message formats and thereby combines Message Format Inference during generalization with Message Type Identification by following the paths through the format graph.

Five approaches swap single process steps. One is Biprominer, which, like ReverX, performs both Message Type Identification and Message Format Inference in one step. Therefore, Biprominer prepares the Message Format Inference by creating graph states from high-frequency byte patterns. This step takes place before the Message Type Identification, which is performed during the generation of the graph model. Therefore, Message Format Inference and Message Type Identification are exchanged in Biprominer. The same two steps are swapped in FieldHunter and Cai et al., which base their Message Type Identification on the message formats inferred beforehand. Another approach that swaps steps is PRISMA. It places Behavior Model Reconstruction before Message Format Inference and Semantic Deduction, since the knowledge from the construction of the Markov model during the Behavior Model Reconstruction is used by the methods of the latter two steps. Discoverer also switches two steps of its process in comparison to our common model. Semantic Deduction and Message Format Inference are exchanged since the type-based alignment derives the syntax using the deduced semantic dependencies. Discoverer also has two further Feature Extraction-refinement-steps within an iterative cycle of the steps Message Type Identification, Feature Extraction, and Semantic Deduction. Besides the process steps recounted here, all other steps of the mentioned deviating tools are performed exactly as stated in our model. Furthermore, all other eleven tools fit perfectly in the process model.

Our process model can be a guideline for analysts and researchers of static traffic analysis for protocol reverse engineering. Cooperative work on one analysis assignment and identifying research directions for future work need not only evaluations of isolated approaches, but a comparison of the applied algorithms. Future discussion will also depend on integrating improved approaches at the corresponding positions of the process. By organizing cooperation between analysts during the practical application, a clearly defined process will render analysis results more dependable and more comparable and will ultimately improve the effectiveness of the procedure. While applying the process, individual and intuitive deviations must be possible. Our unified process model is the most common way to automate static traffic analysis.

B. Levels of Utility of PRE for Different Use Cases

Depending on the target use case of the analysis, differing requirements exist for PRE results, their completeness, and the desired level of detail. Each use case presented in Section III exhibits a specific set of expectations towards the result. In the current section, we examine prevalent examples of the relation between the use case and the level of utility of specific PRE results; as a result, this section also contains what an analyst needs to consider when selecting a tool for a concrete analysis task.

Partial knowledge, gained only about a subset of the protocol model, already satisfies the use case of valid traffic replay by a smart fuzzer or by a honeypot. In particular, details about message fields that will not change, according to the analyzed trace, may be sufficient for the fuzzer or the honeypot. Other use cases include intrusion and misbehavior detection, two examples that benefit the most from a concise protocol model. It should contain only relevant aspects for the detection of malicious traffic, like message fields specific to an attack or statistical distribution of keywords of benign traffic. From a penetration tester's point of view, for example, when assessing the risk for spoofing messages, it may be satisfactory to confirm a state-of-the-art message authentication code. In contrast, malware-, implementation-compliance-, or network-topology-analysis need arbitrarily detailed and correct, concise results [15, 29, 98] with full coverage of the inferred specification. An even more demanding use case, regarding the

PRE result quality, is entity re-implementation. Since a full protocol inference is difficult, sometimes impossible or at the very least time-consuming, some approaches have focused on use-case-specific results [2, 98]. For example, when only the formats of small message headers are of interest, the analysis can be constrained to the corresponding message part. These approaches reach beyond pure PRE to increase efficiency in their respective use case.

Preparation and *Data Collection and Preprocessing:* It is not only the analysis output that determines the usability of a tool, but also the available input data: not all kinds of analyses are suited for any set of traces, and some tools are very specialized in this regard. Representing one of those cases, Trifilo et al. simply regard bytes at the same position within each message, which makes their tool applicable only to fixed-length protocols. RolePlayer, on the other hand, requires ample context-knowledge on the part of the analyst and expertise in selecting exactly two messages to be RolePlayer's input. Whalen et al. only regard the format of messages, while message types need to be discerned beforehand. Likewise, the applicability of many tools depends on the analyst selecting whether a protocol is binary or textual. We also determined from the investigation of the solutions for the Preparation step (Section VI-A) that some context information is gathered from TCP/IP or UDP/IP-encapsulation as a requirement of many tools, so other protocols' stacks, especially unknown ones, cannot be handled.

Message Type Identification: The distinction between textual and binary protocol designs is the only characteristic of an input protocol that is evaluated significantly enough in the surveyed literature to use it as a criterion for the tool's applicability. In particular, statistical and language processing methods are well understood and work efficiently on textual protocols; however, these methods' applicability is limited for binary protocols. These limitations comprise a severe performance impact, therefore posing restrictions on expected patterns inside of messages, such as fixed length fields (Trifilo et al.), key-value patterns (AutoReEngine), or regarding only the beginning of the message (Veritas). The use of sequence alignment as a similarity metric especially prevents analysis of input traces larger than a few hundred messages.

Message Format Inference: Despite this limitation, the subsequent task of Message Format Inference is also dominated by sequence alignment methods, which perform unsatisfactorily for all but small input sets when aligning byte by byte throughout multiple messages. Besides variations of sequence alignment, NEMESYS is the only available universal solution for format inference of binary protocols. Whalen et al., Cai et al., Biprominer, and ReverX use multiple kinds of state machines to infer and represent a message format. While Whalen et al. and ReverX specialize in textual protocols, Biprominer and Cai et al. also address binary ones. This demonstrates that the approach is applicable for both types of protocols, however, the assumptions Biprominer and Cai et al. must make about the protocol format does limit their usage.

Almost any of the Message Format Inference analysis methods are suitable for a use case requesting the format, while the result quality will depend on the characteristics of the protocol. Exceptions are PRISMA, which derives templates for messages, and AutoReEngine, which discerns only between static and variable parts of messages.

Semantic Deduction: The term semantics represents a wide range of meanings according to the different PRE approaches surveyed. While some already regard the value domain of a field as semantics, others mean addressing information from the encapsulation; dynamic runtime information about the entity is sometimes also called semantics. Besides information which is contained in encapsulation, RolePlayer and Netzob are the only tools to collect environmental data and try to correlate it with observed patterns in messages. In contrast, Non-negative Matrix Factorization as applied by PRISMA is suitable for deducing the purposes of messages in whole, rather not for single fields within. The limitations of all other tools in terms of the definition of "semantic" leaves only Netzob and PRISMA as options. The availability and the dependency on environmental relations, or whether the analysis goal is to identify the message purposes, determines which of the two are applicable.

Behavior Model Reconstruction: The analysis by a typical tool for Behavior Model Reconstruction yields a behavior model for an inferred protocol that can be used to recognize or predict valid sequences of messages. In contrast, AutoReEngine only discovers the most frequent message sequences of the trace. Trifilo et al. and PRISMA regard not only isolated states but also chains of states during the state machine optimization; this way, a protocol run can be more realistically simulated at the cost of a more complex model. Additionally, it is specific to all tools for Behavior Model Reconstruction that they do not provide a fine-grained format inference. If this is desired, a separate analysis of the message format is required.

To summarize, the use case of a protocol analysis is a selection criterion for the suitable tool. The specialization of methods for a specific kind of input trace further narrows the suitable tools. The lack of implementations makes it necessary to decide definitively which approach is best for investing re-implementation effort.

C. Challenges for STA-Tool Designers

We believe one way to improve the application of STA would be to synthesize and publish a new tool, which **combines the most promising methods** for each step from the surveyed tools. To improve the full coverage of the protocol reverse engineering process, one fully integrated solution for every single step is missing. The "Conclusions" sections of each step in Section VI provide detailed information to select candidates for algorithms to incorporate in such a tool. In the current section we discuss challenges inherent to the selection for each step. It is out of the scope of this survey to discuss possible design decisions of a new tool.

Data Collection and Preprocessing: This step should contain a solution for filtering of messages that matches the requirements of the subsequently used algorithms. The goal should be to optimize the performance and overall usability of the tool. Moreover, the collection of context information should be considered. Given that multiple analysis methods are offered alternatively, e. g., different ones for binary or textual protocols, how to automatically decide at this point which kind of analysis method to utilize for a specific trace should be solved.

Feature Extraction: Plenty of working methods are available to handle tokenization during Feature Extraction of textual protocols. Solutions for binary protocols are underrepresented in this respect, especially considering their importance in Industrial Control Systems or the Internet of Things. In those or similar cases, the protocol design is governed by limited entity and network resources, often leading to proprietary binary protocols. A new solution should, therefore, focus on selecting an appropriate method applicable to binary protocols, since this is not trivial. For textual protocols, selecting one of the existing keyword analysis methods for Feature Extraction is most promising.

Message Type Identification: The used clustering method, taken by itself, is not decisively important. The only requirement is that the number of expected clusters must not be a parameter of the algorithm preventing the utilization of partitioning clustering methods. The type of feature extraction is more crucial, and a suitable clustering algorithm for the provided input may then be fitted accordingly. The requirements for the type identification method are to have reasonable performance while proving able to cope with the unknown properties and parameters on which the result depends.

Message Format Inference: A byte-wise comparison of messages for format inference has considerable performance impact, however, no universal alternative is known for STA. To select an appropriate method of Message Format Inference requires an investigation of the subtle differences between multiple byte-wise comparison methods and the implications of protocol properties. This is especially required to address binary protocols.

Behavior Model Reconstruction: It is a common notion to utilize a probabilistic state machine. Prefix Tree Acceptors are mostly applied for textual protocols, while hidden Markov models are also used for binary protocols. Either kind of state machine needs to contain probabilistic information about transitions. More important than finding a suitable representation is how to generate any kind of state machine; this is the decisive difference between the known approaches and should be considered carefully for the implementation of a new tool.

Processing of Results: Many tool authors have used HTTP as sample input for their evaluations. While this is illustrating the approaches well, HTTP is an unsuitable choice in regard to most target use cases of STA. Most proprietary protocols that need to be reverse engineered by static traffic analysis differ considerably from HTTP in design. Such main target protocols are used between, for example, Malware and their Command-and-Control-Servers, or are from embedded devices in Industrial Control Systems or Internet of Things applications.

D. Open Research Questions for Future Work

Automated approaches currently cannot replace manual analysis. While we agree with Duchêne et al. [10] that preprocessing of traces is one major effort to still undertake manually, we argue that the inference itself also poses several challenges to overcome. Although the analysis of the network protocols' type, format, and behavior is based on methods from pattern recognition, natural language processing, statistical tests, data mining, and bioinformatics, for which widespread fully automated solutions are known, PRE approaches only apply algorithms in a semi-automated way. Therefore, automated approaches currently cannot fully replace manual analysis. To improve this situation, we propose in this section future research directions for each individual PRE process step.

Data Collection and Preprocessing: Filtering for the unknown target protocol is still unsolved. Other fields of research may help solve the challenge of noisy input traces. One example is Protocol Identification (Securitas [99], Iustitia [100], KISS [101], or SPID [102]) for filtering for the protocol under consideration. Moreover, the surveyed literature does not discuss, how message traces of high diversity can be compiled into a high-quality input set of messages. This is, however, necessary to reduce the performance impact of the subsequent steps that rely on algorithms with high computational complexity.

Feature Extraction: The applied statistical natural-language and data-mining approaches LDA, KS, t-test, Apriori, and Viterbi all promise good results for determining the similarity of textual messages. A comparative evaluation of these concurring methods is necessary to reveal their strengths and weaknesses for specific use cases. Since in general the performance of the feature extraction needs improvement, especially for binary protocols, additional methods should be adapted to static traffic analysis and evaluated.

First, it is necessary to empirically determine characteristics of messages in transfer, in particular, field data types. To consider patterns of protocol design in future work, which are observable in messages, would allow for domain-specific features to be extracted. All steps dependent on the extraction of features would benefit from the domain specifics of network traces that the determined characteristics could reveal.

Message Format Inference: It should be explored whether instead of optimal alignment methods, more efficient heuristic alignments or alignment-free methods, e. g., Leimeister et al. [103], suffice to infer message format. The applied methods could thus handle larger test-sets of messages, gaining more variance in field values and thereby improving the analysis result.

Semantic Deduction: Currently, for a successful Semantic Deduction, human interpretation is a fundamental requirement. However, manually observing correlation is almost impossible due to the variety of employed encodings and conversions or chronological implications of value changes in network protocols. Statistical correlation for subsequent human interpretation is much more promising; therefore, more study of possible improvements of correlation is necessary.

Behavior Model Reconstruction: In addition to the conventional behavior-modeling by state machine generalization, Netzob has proposed a completely different approach. It uses dynamic traffic analysis to probe an entity with the reconstructed, presumably valid, message sequence. The idea looks highly promising but requires significant setup effort. On one hand, it needs more evaluation, and on the other, it could also be extended to other aspects of the protocol, beyond the behavior alone, such as to format inference.

Processing of Results: Current tools provide insufficient support to interpret and use the analysis results. The application of visual analytics needs to be explored to enable human introspection for a most beneficial utilization of the result representation. Finally, we identify as necessary future work the automatic calculation of a result metric, and the tackling of the Oracle Problem of deciding whether a result for an unknown protocol is correct.

Beyond the single steps of the process, we also identify directions that future work could explore to increase the utility of static traffic analysis in general.

As seen in Section VI, the steps *Preparation*, *Data Collection and Preprocessing*, and *Processing of Results* needed considerably more explanation than the core steps of the process and still appear somewhat blurry. This is due to the sparse coverage of the requirements of these steps by established methods. These steps undoubtedly are necessary parts of the process since all tools expect tasks to be performed in their place to some extent. However, methods for automated discovery of analysis parameters are not well established and need to be investigated. To that end, methods for automatically quantifying the inference result quality and thus re-adjusting the analysis parameters need to be devised.

To the best of our knowledge, no general investigation of properties of message-in-transfer encodings is available, resulting in a lack of knowledge regarding the statistical features of individual message parts. Having such features available would enable analysts to recognize them more reliably in unknown protocols. Since protocols are designed by humans, a general characterization of properties of protocol parts should be possible.

Ongoing research focuses on enhancing Netzob as framework and integrating new inference methods. Our ongoing work seeks to integrate the most promising methods into one tool and to evaluate new methods for the analysis of binary protocols. Another recent proposal is the prevention of protocol reverse engineering by obfuscation to hinder attacks on network protocols. This is the goal of work currently being performed by Duchêne et al. [41].

E. Summary

In this paper, we presented an explicit, structured process model for protocol reverse engineering by static traffic trace analysis (Section V). Based on a collection of classification schemes (Section III), this process model reveals the common structure of tools and frameworks from previous research (Section IV). Our model provides the necessary overview and structure to protocol reverse engineering for research and practice. To the best of our knowledge, no similar previous work exists. By discussing the tools and frameworks from the new perspective of our model, we showed differences and similarities in these approaches. Embedding into the process model detailed discussions about the methods applied by the tools, reveals fundamental insights about how tools and methods are intertwined (Section VI). We validated our model in the light of the analysis of the solutions and algorithms utilized in the considered tools and frameworks (Section VII-A), and we outlined the utility of protocol reverse engineering for typical use cases (Section VII-B).

For an analyst, our survey provides structure for practical protocol reverse engineering by providing a defined purpose, available methods, and considerations to make for each step of the process. The mapping of tools to steps (Section V-I), in which these can be applied, guides the selection of the appropriate tools for a particular protocol reverse engineering project.

To point out future relevant research opportunities, we dissected the frameworks into specific mechanisms and single algorithms (Section VI), making it possible to re-combine these into new tools with either more specific applicability or with improved results quality (Section VII-C). Combined with the process description, we initiate a discussion of what current tools may accomplish and which limitations exist. This leads to an overview of which solutions and algorithms have already been investigated and where challenges remain so that novel solutions may be searched for in the future. In particular, we highlight that the Preparation, Semantic Deduction, and Processing of Results steps are not well covered by the current methods.

Other high-level insights are:

- HTTP and similar protocols are unsuitable specimens for tool evaluation.
- The analysis solutions for binary protocols are underrepresented, considering the importance of protocols designed for low bandwidth links like industrial control protocols and Internet of Things.
- The domain-specific characterization of types and formats is more relevant than the clustering algorithm.
- The performance of the solutions is not considered sufficiently, so that the analysis of large input traces is impractical.

Except for ReverX [14], PRISMA [15], and Netzob [16], the lack of protocol-reverse-engineering-tool implementations thwarts comparable statements about the performance and quality of methods, while the surveyed papers are not always specific enough about implementation details to adequately recreate their approach. For the whole field of protocol reverse engineering, it is a prevalent problem that only very few implementations of tools and frameworks exist which can be evaluated, refined, and developed further to keep a constant momentum of innovation. Although a growing interest in static traffic trace analysis exists, a sufficient, universal solution is currently not available.

References

[1] Eduard Marin, Dave Singelée, Bohan Yang, Ingrid Verbauwhede, and Bart Preneel. "On the Feasibility of Cryptography for a Wireless Insulin Pump System". In: *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*. CODASPY. ACM, 2016.

[2] Chia Y. Cho, Domagoj Babić, Eui C. R. Shin, and Dawn 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. ACM, 2010.

[3] CAPEC Content Team. *CAPEC - CAPEC-192: Protocol Reverse Engineering (Version 2.6)*. June 2014. URL: https://web.archive.org/web/20140725160124/http://capec.mitre.org:80/data/definitions/192.html (visited on 08/22/2018).

[4] Juan Caballero and Dawn Song. "Automatic Protocol Reverse-Engineering: Message Format Extraction and Field Semantics Inference". In: *Computer Networks* 57.2 (Feb. 2012). Elsevier.

[5] Weiming Li, Meirong Ai, and Bo Jin. "A Network Protocol Reverse Engineering Method Based on Dynamic Taint Propagation Similarity". In: *12th International Conference on Intelligent Computing*. ICIC. Springer, 2016.

[6] Daniel Halperin, Thomas S. Heydt-Benjamin, Benjamin Ransford, Shane S. Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. "Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses". In: *IEEE Symposium on Security and Privacy*. SP. IEEE, 2008.

[7] Edward J. Schwartz, Thanassis Avgerinos, and David Brumley. "All You Ever Wanted to Know about Dynamic Taint Analysis and Forward Symbolic Execution (but Might Have Been Afraid to Ask)". In: *IEEE Symposium on Security and Privacy*. SP. IEEE, 2010.

[8] Xiangdong Li and Li Chen. "A Survey on Methods of Automatic Protocol Reverse Engineering". In: *Seventh International Conference on Computational Intelligence and Security*. IEEE, 2011.

[9] John Narayan, Sandeep K. Shukla, and T. Charles Clancy. "A Survey of Automatic Protocol Reverse Engineering Tools". In: *ACM Computing Surveys* 48.3 (Dec. 2015). ACM.

[10] Julien Duchêne, Colas Le Guernic, Eric Alata, Vincent Nicomette, and Mohamed Kaâniche. "State of the Art of Network Protocol Reverse Engineering Tools". In: *Journal of Computer Virology and Hacking Techniques* 14.1 (Feb. 2018). Springer.

[11] Marshall A. Beddoe. *Network Protocol Analysis using Bioinformatics Algorithms*. Tech. rep. McAfee Inc., 2004. URL: http://www.4tphi.net/~awalters/PI/pi.pdf (visited on 08/22/2018).

[12] Jeremy Rauch. *Protocol Debug (PDB)*. Black Hat USA. 2006. URL: http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Rauch.pdf (visited on 08/22/2018).

[13] Weidong Cui, Jayanthkumar Kannan, and Helen J. Wang. "Discoverer: Automatic Protocol Reverse Engineering from Network Traces". In: *Proceedings of 16th USENIX Security Symposium*. USENIX Association, 2007.

[14] João Antunes, Nuno Neves, and Paulo Verissimo. "Reverse Engineering of Protocols from Network Traces". In: *18th Working Conference on Reverse Engineering*. WCRE. IEEE, 2011.

[15] Tammo Krueger, Hugo Gascon, Nicole Krämer, and Konrad Rieck. "Learning Stateful Models for Network Honeypots". In: *Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence*. AISec. ACM, 2012.

[16] Georges Bossert, Frédéric Guihéry, and Guillaume Hiet. "Towards Automated Protocol Reverse Engineering Using Semantic Information". In: *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*. AsiaCCS. ACM, 2014.

[17] Jian-Zhen Luo and Shun-Zheng Yu. "Position-Based Automatic Reverse Engineering of Network Protocols". In: *Journal of Network and Computer Applications* 36.3 (May 2013). Elsevier.

[18] Weidong Cui, Vern Paxson, Nicholas Weaver, and Randy H. Katz. "Protocol-Independent Adaptive Replay of Application Dialog". In: *Proceedings of the Network and Distributed System Security Symposium*. NDSS. Internet Society, 2006.

[19] Sean Whalen, Matt Bishop, and James P. Crutchfield. "Hidden Markov Models for Automated Protocol Learning". In: *6th International Conference on Security and Privacy in Communication Systems*. SecureComm. Springer, 2010.

[20] Yipeng Wang, Xingjian Li, Jiao Meng, Yong Zhao, Zhibin Zhang, and Li Guo. "Biprominer: Automatic Mining of Binary Protocol Features". In: *12th International Conference on Parallel and Distributed Computing, Applications and Technologies*. PDCAT. IEEE, 2011.

[21] Yipeng Wang, Xiao-Chun Yun, Muhammad Zubair Shafiq, Liyan Wang, Alex X. Liu, Zhibin Zhang, Danfeng Yao, Yongzheng Zhang, and Li Guo. "A Semantics Aware Approach to Automated Reverse Engineering Unknown Protocols". In: *20th IEEE International Conference on Network Protocols*. ICNP. IEEE, 2012.

[22] Haifeng Li, Bo Shuai, Jian Wang, and Chaojing Tang. "Protocol Reverse Engineering Using LDA and Association Analysis". In: *11th International Conference on Computational Intelligence and Security*. CIS. IEEE, 2015.

[23] Ignacio Bermudez, Alok Tongaonkar, Marios Iliofotou, Marco Mellia, and Maurizio M. Munafò. "Automatic Protocol Field Inference for Deeper Protocol Understanding". In: *IFIP Networking Conference*. IFIP Networking. IEEE, 2015.

[24] Ignacio Bermudez, Alok Tongaonkar, Marios Iliofotou, Marco Mellia, and Maurizio M. Munafò. "Towards Automatic Protocol Field Inference". In: *Computer Communications* 84 (June 2016). Elsevier.

[25] Jun Cai, Jian-Zhen Luo, and Fangyuan Lei. "Analyzing Network Protocols of Application Layer Using Hidden Semi-Markov Model". In: *Mathematical Problems in Engineering* 2016 (Mar. 2016). Hindawi.

[26] Siyu Tao, Hongyi Yu, and Qing Li. "Bit-Oriented Format Extraction Approach for Automatic Binary Protocol Reverse Engineering". In: *IET Communications* 10.6 (Apr. 2016). IET.

[27] Ming-Ming Xiao, Shi-Long Zhang, and Yu-Ping Luo. "Automatic Network Protocol Message Format Analysis". In: *Journal of Intelligent and Fuzzy Systems* 31.4 (Sept. 2016). IOS Press.

[28] Stephan Kleber, Henning Kopp, and Frank Kargl. "NEMESYS: Network Message Syntax Reverse Engineering by Analysis of the Intrinsic Structure of Individual Messages". In: *12th USENIX Workshop on Offensive Technologies*. WOOT. USENIX Association, 2018.

[29] Corrado Leita, Ken Mermoud, and Marc Dacier. "ScriptGen: An Automated Script Generation Tool for Honeyd". In: *Proceedings of the 21st Annual Computer Security Applications Conference*. ACSAC. IEEE, 2005.

[30] Maxim Shevertalov and Spiros Mancoridis. "A Reverse Engineering Tool for Extracting Protocols of Networked Applications". In: *14th Working Conference on Reverse Engineering*. WCRE. IEEE, 2007.

[31] Antonio Trifilo, Stefan Burschka, and Ernst Biersack. "Traffic to Protocol Reverse Engineering". In: *IEEE Symposium on Computational Intelligence for Security and Defense Applications*. CISDA. IEEE, 2009.

[32] Yipeng Wang, Zhibin Zhang, Danfeng Daphne Yao, Buyun Qu, and Li Guo. "Inferring Protocol State Machine from Network Traces: A Probabilistic Approach". In: *Applied Cryptography and Network Security*. ACNS. Springer, 2011.

[33] Ming-Ming Xiao and Yu-Ping Luo. "Automatic Protocol Reverse Engineering Using Grammatical Inference". In: *Journal of Intelligent and Fuzzy Systems* 32.5 (Apr. 2017). IOS Press.

[34] Serge Gorbunov and Arnold Rosenbloom. "AutoFuzz: Automated Network Protocol Fuzzing Framework". In: *International Journal of Computer Science and Network Security* 10.8 (Aug. 2010).

[35] Tammo Krueger, Nicole Krämer, and Konrad Rieck. "ASAP: Automatic Semantics-Aware Analysis of Network Payloads". In: *Privacy and Security Issues in Data Mining and Machine Learning*. PSDML. Springer, 2010.

[36] Juan Caballero, Heng Yin, Zhenkai Liang, and Dawn 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. ACM, 2007.

[37] Zhiqiang Lin, Xuxian Jiang, Dongyan Xu, and Xiangyu Zhang. "Automatic Protocol Format Reverse Engineering through Context-Aware Monitored Execution." In: *Proceedings of the Network and Distributed System Security Symposium*. NDSS. Internet Society, 2008.

[38] Paolo Milani Comparetti, Gilbert Wondracek, Christopher Kruegel, and Engin Kirda. "Prospex: Protocol Specification Extraction". In: *30th IEEE Symposium on Security and Privacy*. SP. IEEE, 2009.

[39] Zhi Wang, Xuxian Jiang, Weidong Cui, Xinyuan Wang, and Mike Grace. "Reformat: Automatic Reverse Engineering of Encrypted Messages". In: *14th European Symposium on Research in Computer Security*. ESORICS. Springer, 2009.

[40] Zhiqiang Lin, Xiangyu Zhang, and Dongyan Xu. "Automatic Reverse Engineering of Data Structures from Binary Execution". In: *Proceedings of the Network and Distributed System Security Symposium*. NDSS. Internet Society, 2010.

[41] Julien Duchêne, Colas Le Guernic, Eric Alata, Vincent Nicomette, and Mohamed Kaâniche. "Protocol Reverse Engineering: Challenges and Obfuscation". In: *11th International Conference on Risks and*

Security of Internet and Systems, Revised Selected Papers. CRiSIS. Springer, 2016.

[42] Andrew Tridgell. *How Samba Was Written*. Aug. 2003. URL: https://www.samba.org/ftp/tridge/misc/french_cafe.txt (visited on 08/23/2018).

[43] Joseph Tartaro and Matthew Halchyshak. *Cyber Necromancy*. Dec. 2014. URL: https://media.ccc.de/v/31c3_-_5956_-_en_-_saal_2_-_201412281400_-_cyber_necromancy_-_joseph_tartaro_-_matthew_halchyshak (visited on 08/23/2018).

[44] Yating Hsu, Guoqiang Shu, and D. Lee. "A Model-Based Approach to Security Flaw Detection of Network Protocol Implementations". In: *International Conference on Network Protocols*. ICNP. IEEE, 2008.

[45] Ran Ji, Jian Wang, Chaojing Tang, and Ruilin Li. "Automatic Reverse Engineering of Private Flight Control Protocols of UAVs". In: *Security and Communication Networks* 2017 (July 2017). Hindawi.

[46] Hossein Fereidooni, Jiska Classen, Tom Spink, Paul Patras, Markus Miettinen, Ahmad-Reza Sadeghi, Matthias Hollick, and Mauro Conti. "Breaking Fitness Records Without Moving: Reverse Engineering and Spoofing Fitbit". In: *20th International Symposium Research in Attacks, Intrusions, and Defenses*. RAID. Springer, 2017.

[47] Shameng Wen, Qingkun Meng, Chao Feng, and Chaojing Tang. "Protocol Vulnerability Detection Based on Network Traffic Analysis and Binary Reverse Engineering". In: *PLOS ONE* 12.10 (Oct. 2017). Public Library of Science.

[48] Jędrzej Bieniasz, Piotr Sapiecha, Milosz Smolarczyk, and Krzysztof Szczypiorski. "Towards Model-Based Anomaly Detection in Network Communication Protocols". In: *International Conference on Frontiers of Signal Processing*. ICFSP. IEEE, 2016.

[49] Juan Caballero, Pongsin Poosankam, Christian Kreibich, and Dawn Song. "Dispatcher: Enabling Active Botnet Infiltration Using Automatic Protocol Reverse-Engineering". In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*. CCS. ACM, 2009.

[50] James Newsome, David Brumley, Jason Franklin, and Dawn Song. "Replayer: Automatic Protocol Replay by Binary Analysis". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS. ACM, 2006.

[51] Corrado Leita. "SGNET: Automated Protocol Learning for the Observation of Malicious Threats". PhD Thesis. Sophia Antipolis: University of Nice, Dec. 2008.

[52] Florence Corpet. "Multiple Sequence Alignment with Hierarchical Clustering". In: *Nucleic Acids Research* 16.22 (Nov. 1988). Oxford University Press, pp. 10881–10890.

[53] Francesco Gringoli, Luca Salgarelli, Maurizio Dusi, Niccolo Cascarano, Fulvio Risso, and Kimberly C. Claffy. "GT: Picking Up the Truth from the Ground for Internet Traffic". In: *ACM SIGCOMM Computer Communication Review* 39.5 (Oct. 2009). ACM.

[54] James M. Joyce. "Kullback-Leibler Divergence". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovrić. Springer, 2011, pp. 720–722.

[55] Ullas Gargi. *Consumer Media Capture: Time-Based Analysis and Event Clustering*. Tech. rep. HP Laboratories, Palo Alto, Aug. 2003. URL: http://hpl.hpl.hp.com/techreports/2003/HPL-2003-165.pdf (visited on 08/24/2018).

[56] Konrad Rieck, Christian Wressnegger, and Alexander Bikadorov. "Sally: A Tool for Embedding Strings in Vector Spaces". In: *Journal of Machine Learning Research* 13.Nov (Nov. 2012).

[57] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research* 3.Jan (Jan. 2003).

[58] Thomas L. Griffiths and Mark Steyvers. "Finding Scientific Topics". In: *Proceedings of the National Academy of Sciences* 101.suppl 1 (2004). National Academy of Sciences of the USA.

[59] Raul H. C. Lopes. "Kolmogorov-Smirnov Test". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovrić. Springer, 2011, pp. 718–720.

[60] Paul Jaccard. "The Distribution of the Flora in the Alpine Zone". In: *New Phytologist* 11.2 (1912). Wiley, pp. 37–50.

[61] Damir Kalpić, Nikica Hlupić, and Miodrag Lovrić. "Student's t-Tests". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovrić. Springer, 2011, pp. 1559–1563.

[62] Sture Holm. "A Simple Sequentially Rejective Multiple Test Procedure". In: *Scandinavian Journal of Statistics* 6.2 (1979). Wiley, pp. 65–70.

[63] Nitis Mukhopadhyay. "Correlation Coefficient". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovrić. Springer, 2011, pp. 315–318.

[64] Rakesh Agrawal, Ramakrishnan Srikant, et al. "Fast Algorithms for Mining Association Rules". In: *Proceedings of 20th International Conference on Very Large Data Bases*. VLDB. Morgan Kaufmann, 1994.

[65] Jiawei Han, Jian Pei, and Yiwen Yin. "Mining Frequent Patterns Without Candidate Generation". In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD. ACM, 2000.

[66] Robert R. Sokal and Charles D. Michener. "A Statistical Method for Evaluating Systematic Relationships". In: *University of Kansas Scientific Bulletin* XXXVIII-2.22 (Mar. 1958). University of Kansas, pp. 1409–1438.

[67] G. David Forney. "The Viterbi Algorithm". In: *Proceedings of the IEEE* 61.3 (Mar. 1973). IEEE, pp. 268–278.

[68] Li Wenchao, Zhou Yong, and Xia Shixiong. "A Novel Clustering Algorithm Based on Hierarchical and k-Means Clustering". In: *Chinese Control Conference*. IEEE, 2007.

[69] Saul B. Needleman and Christian D. Wunsch. "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins". In: *Journal of Molecular Biology* 48.3 (Mar. 1970). Elsevier, pp. 443–453.

[70] Temple F. Smith and Michael S. Waterman. "Identification of Common Molecular Subsequences". In: *Journal of Molecular Biology* 147.1 (Mar. 1981). Elsevier, pp. 195–197.

[71] Robert C. Edgar. "MUSCLE: Multiple Sequence Alignment with High Accuracy and High Throughput". In: *Nucleic Acids Research* 32.5 (Mar. 2004). Oxford University Press.

[72] Fionn Murtagh and Pedro Contreras. "Algorithms for Hierarchical Clustering: An Overview". In: *Data Mining and Knowledge Discovery* 2.1 (Jan. 2012). Wiley.

[73] Leonard Kaufman and Peter J. Rousseeuw. "Partitioning Around Medoids (Program PAM)". In: *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Inc, 2005, pp. 68–125.

[74] Brendan J. Frey and Delbert Dueck. "Clustering by Passing Messages Between Data Points". In: *Science* 315.5814 (Feb. 2007). American Association for the Advancement of Science, pp. 972–976.

[75] Daniel D. Lee and H. Sebastian Seung. "Algorithms for Non-Negative Matrix Factorization". In: *Advances in Neural Information Processing Systems 13*. NIPS Conference. The MIT Press, 2000.

[76] Wei Xu, Xin Liu, and Yihong Gong. "Document Clustering Based on Non-Negative Matrix Factorization". In: *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR. ACM, 2003.

[77] Noam Slonim and Naftali Tishby. "Agglomerative Information Bottleneck". In: *Advances in Neural Information Processing Systems 12*. NIPS Conference. The MIT Press, 1999.

[78] Naftali Tishby, Fernando C. N. Pereira, and William Bialek. "The Information Bottleneck Method". In: *arXiv:physics/0004057* [physics.data-an] (2000).

[79] Cosma R. Shalizi and Kristina L. Shalizi. "Blind Construction of Optimal Nonlinear Recursive Predictors for Discrete Sequences". In: *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*. UAI. AUAI Press, 2004.

[80] Shun-Zheng Yu. "Hidden Semi-Markov Models". In: *Artificial Intelligence*. Special Review Issue 174.2 (Feb. 2010). Elsevier.

[81] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains". In: *The Annals of Mathematical Statistics* 41.1 (Feb. 1970). Institute of Mathematical Statistics, pp. 164–171.

[82] Colin De la Higuera. "The Prefix Tree Acceptor (PTA)". In: *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010, pp. 238–242.

[83] Daniel D. Lee and H. Sebastian Seung. "Learning the Parts of Objects by Non-Negative Matrix Factorization". In: *Nature* 401 (Oct. 1999). Macmillan Magazines Ltd., pp. 788–791.

[84] Colin De la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.

[85] Lusheng Wang and Tao Jiang. "On the Complexity of Multiple Sequence Alignment". In: *Journal of Computational Biology* 1.4 (Jan. 1994). Mary Ann Liebert, Inc., pp. 337–348.

[86] David N. Reshef, Yakir A. Reshef, Hilary K. Finucane, Sharon R. Grossman, Gilean McVean, Peter J. Turnbaugh, Eric S. Lander, Michael Mitzenmacher, and Pardis C. Sabeti. "Detecting Novel Associations in Large Data Sets". In: *Science* 334.6062 (Dec. 2011). American Association for the Advancement of Science, pp. 1518–1524.

[87] Lawrence R. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". In: *Proceedings of the IEEE* 77.2 (Feb. 1989). IEEE, pp. 257–286.
[88] Jean Berstel, Luc Boasson, Olivier Carton, and Isabelle Fagnot. "Minimization of Automata". In: *arXiv:1010.5318* [cs.FL] (Oct. 2010). arXiv: 1010.5318.
[89] Dana Angluin. "Learning Regular Sets from Queries and Counterexamples". In: *Information and Computation* 75.2 (Nov. 1987). Elsevier, pp. 87–106.
[90] "Regular Expressions". In: *The Open Group Base Specifications*. Ed. by IEEE and Open Group. IEEE Std 1003.1-2008 Issue 7, 2016 Edition. 2016.
[91] Thomas D. Schneider. "Consensus Sequence Zen". In: *Applied Bioinformatics* 1.3 (2002). Adis Press, pp. 111–119.
[92] Michael Sutton, Adam Greene, and Pedram Amini. *Fuzzing: Brute Force Vulnerability Discovery*. Addison-Wesley, June 2007.
[93] Niels Provos. "A Virtual Honeypot Framework". In: *Proceedings of the 13th USENIX Security Symposium*. USENIX Association, 2004.
[94] Hartmut König. *Protocol Engineering*. Springer, 2012.
[95] Paul Overell and Dave Crocker. *Augmented BNF for Syntax Specifications: ABNF*. The Internet Engineering Task Force. Jan. 2008. URL: https://tools.ietf.org/html/rfc5234 (visited on 08/24/2018).
[96] Richard Jerry Linn and M. Ümit Uyar, eds. *Conformance Testing Methodologies and Architectures for OSI Protocols*. IEEE Computer Society Press, 1995.
[97] ITU-T Study Group 17. "ITU-T Recommendation X.680: Information Technology - Abstract Syntax Notation One". In: *ITU-T Recommendations*. Ed. 5. International Telecommunication Union, 2015.
[98] Hugo Gascon, Christian Wressnegger, Fabian Yamaguchi, Daniel Arp, and Konrad Rieck. "PULSAR: Stateful Black-Box Fuzzing of Proprietary Network Protocols". In: *11th International Conference of Security and Privacy in Communication Networks, Revised Selected Papers*. SecureComm. Springer, 2015.
[99] Xiao-Chun Yun, Yipeng Wang, Yongzheng Zhang, and Yu Zhou. "A Semantics-Aware Approach to the Automated Network Protocol Identification". In: *IEEE/ACM Transactions on Networking* 24.1 (Feb. 2016). IEEE.
[100] Amir R. Khakpour and Alex X. Liu. "An Information-Theoretical Approach to High-speed Flow Nature Identification". In: *IEEE/ACM Transactions on Networking* 21.4 (Aug. 2013). IEEE.
[101] Alessandro Finamore, Marco Mellia, Michela Meo, and Dario Rossi. "KISS: Stochastic Packet Inspection Classifier for UDP Traffic". In: *IEEE/ACM Transactions on Networking* 18.5 (2010). IEEE.
[102] Erik Hjelmvik and Wolfgang John. "Statistical Protocol IDentification with SPID: Preliminary Results". In: *Swedish National Computer Networking Workshop*. 2009.
[103] Chris-Andre Leimeister, Marcus Boden, Sebastian Horwege, Sebastian Lindner, and Burkhard Morgenstern. "Fast Alignment-Free Sequence Comparison Using Spaced-Word Frequencies". In: *Bioinformatics* 30.14 (July 2014). Oxford University Press.

Appendix

For reference, we provide the tasks performed by each of the surveyed tools of Section IV in a simplified graphical representation. The graphics contain the mapping to our PRE process model as annotation overlay using the following colors and label notations:

🟫 DP Data Collection and Preprocessing	🟥 MFI Message Format Inference	🟧 PR Processing of Results
🟪 FE Feature Extraction	🟩 SD Semantic Deduction	↔ steps exchanged in the tool compared to the PRE process model
⬜ MTI Message Type Identification	🟦 BMR Behavior Model Reconstruction	

A. Inference of Message Formats

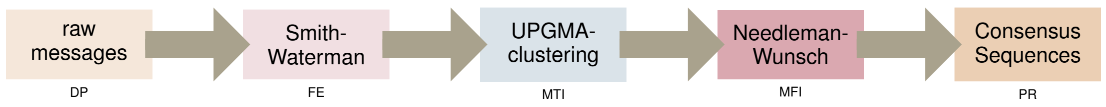


Fig. 11: Protocol Informatics (Beddoe [11], 2004)

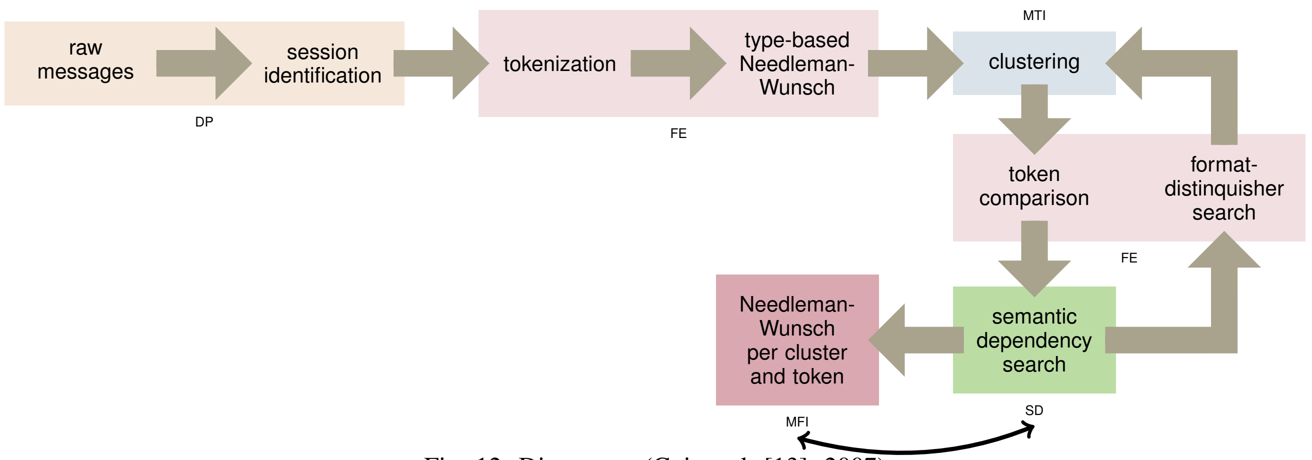

raw messages
session identification
DP
tokenization
type-based Needleman-Wunsch
FE
MTI
clustering
token comparison
format-distinquisher search
FE
Needleman-Wunsch per cluster and token
semantic dependency search
MFI
SD

Fig. 12: Discoverer (Cui et al. [13], 2007)

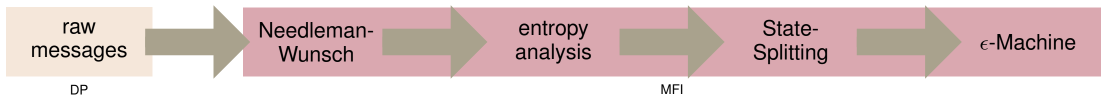


Fig. 13: Whalen et al. [19] (2010)

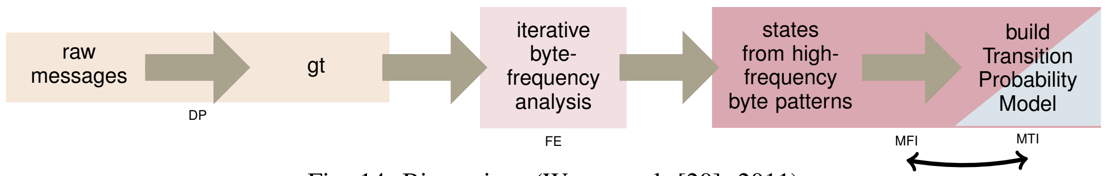

raw messages
gt
DP
iterative byte-frequency analysis
FE
states from high-frequency byte patterns
build Transition Probability Model
MFI
MTI

Fig. 14: Biprominer (Wang et al. [20], 2011)

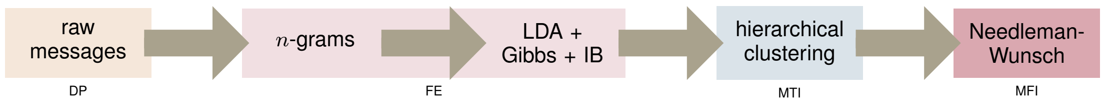


Fig. 15: ProDecoder (Wang et al. [21], 2012)

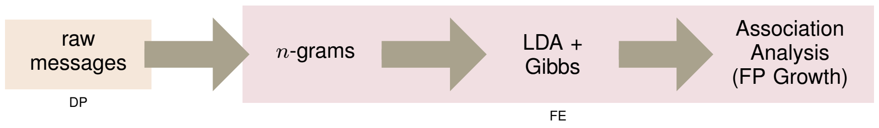


Fig. 16: Li et al. [22] (2015)

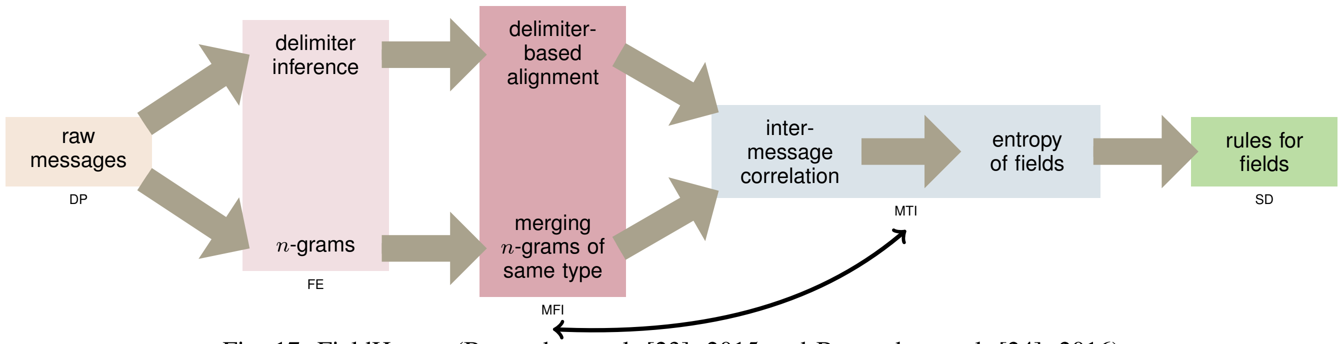


Fig. 17: FieldHunter (Bermudez et al. [23], 2015 and Bermudez et al. [24], 2016)

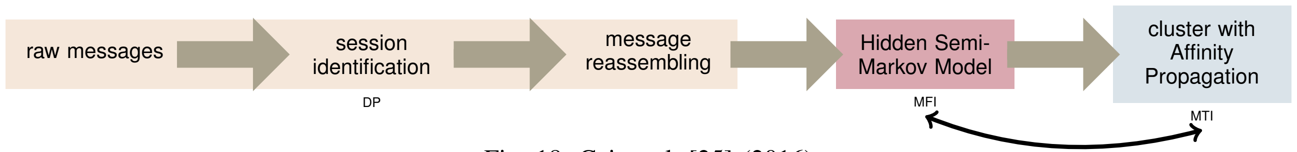


Fig. 18: Cai et al. [25] (2016)

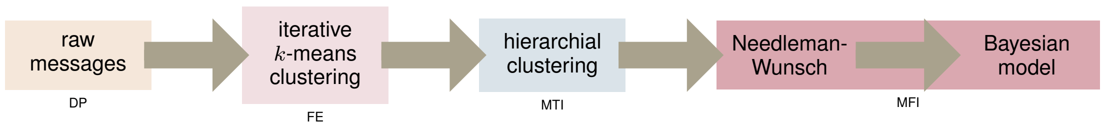


Fig. 19: PRE-Bin (Tao et al. [26], 2016)

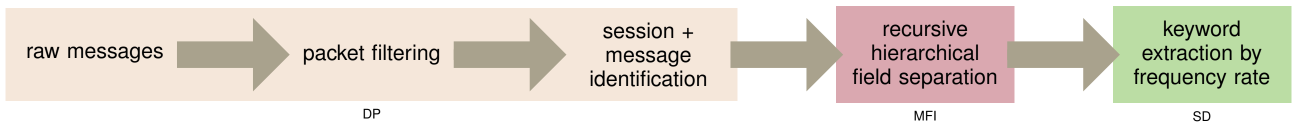

raw messages
packet filtering
session + message identification
DP
recursive hierarchical field separation
MFI
keyword extraction by frequency rate
SD

Fig. 20: Xiao et al. [27] (2016)

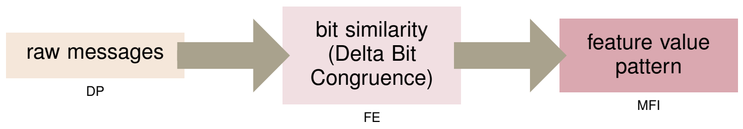


Fig. 21: NEMESYS (Kleber et al. [28], 2018)

B. Reconstruction of the Behavior Model

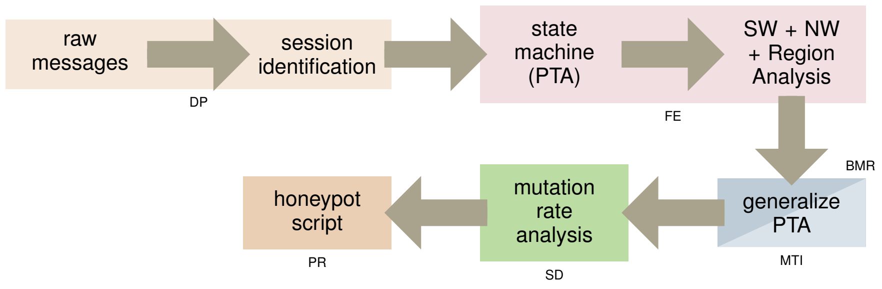


Fig. 22: ScriptGen (Leita et al. [29], 2005)

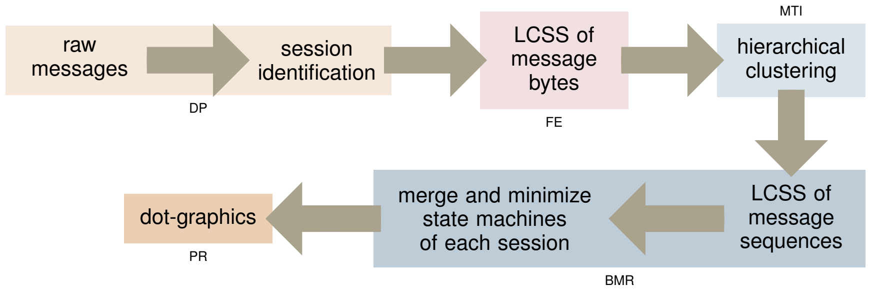


Fig. 23: PEXT (Shevertalov and Mancoridis [30], 2007)

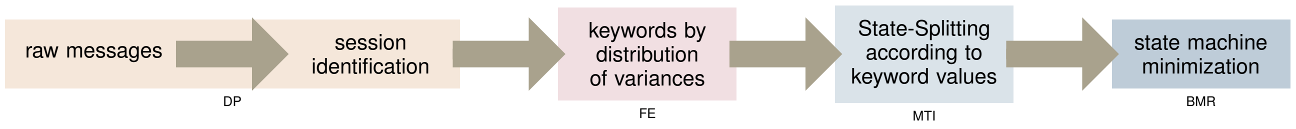

raw messages
session identification
DP
keywords by distribution of variances
FE
State-Splitting according to keyword values
MTI
state machine minimization
BMR

Fig. 24: Trifilo et al. [31] (2009)

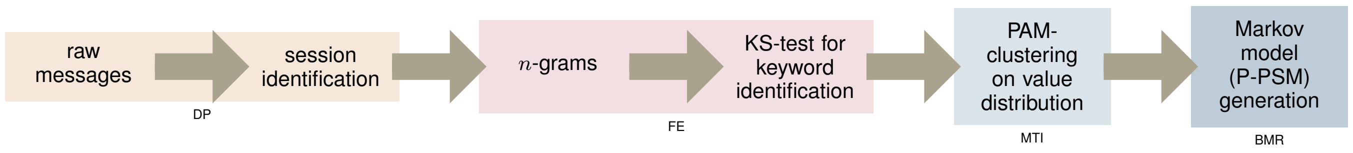

raw messages
session identification
DP
n-grams
KS-test for keyword identification
FE
PAM-clustering on value distribution
MTI
Markov model (P-PSM) generation
BMR

Fig. 25: Veritas (Wang et al. [32], 2011)

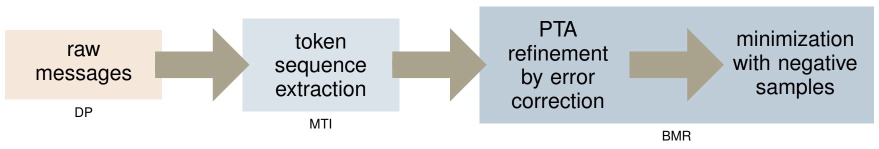

raw messages
DP
token sequence extraction
MTI
PTA refinement by error correction
minimization with negative samples
BMR

Fig. 26: PREUGI (Xiao and Luo [33], 2017)

C. Deduction of the Complete Specification

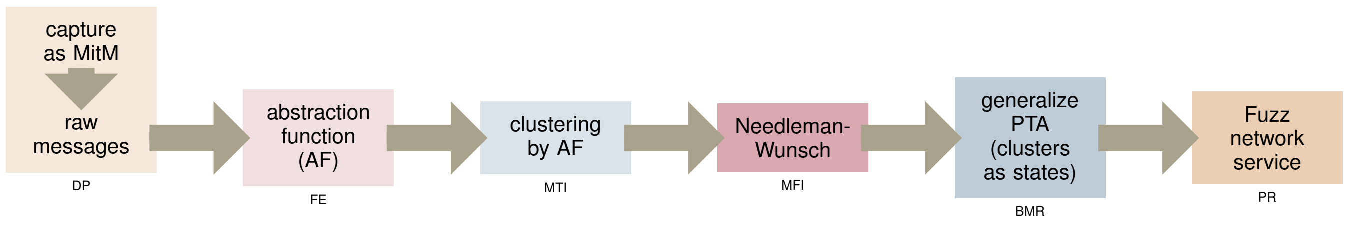


Fig. 27: AutoFuzz (Gorbunov and Rosenbloom [34], 2010)

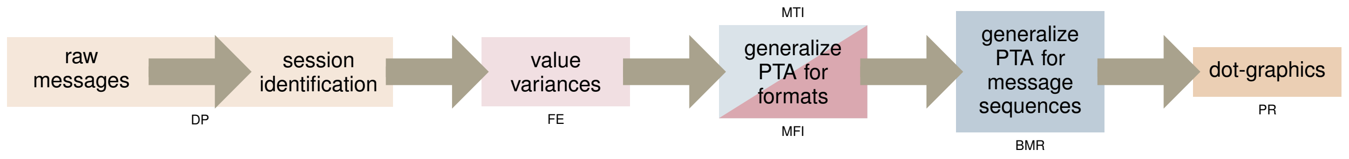


Fig. 28: ReverX (Antunes et al. [14], 2011)

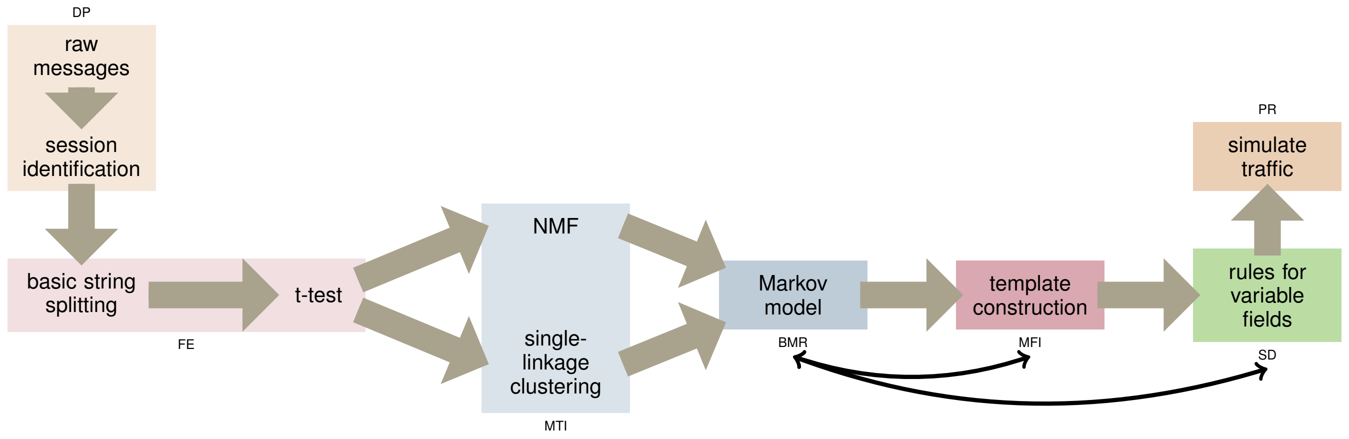


Fig. 29: PRISMA (Krueger et al. [15], 2012)

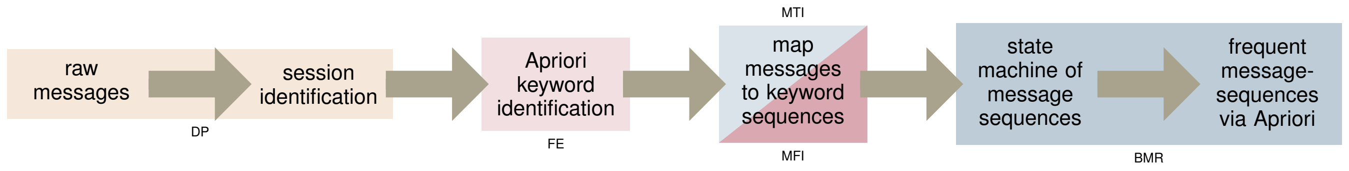

raw messages
session identification
DP
Apriori keyword identification
FE
MTI
map messages to keyword sequences
MFI
state machine of message sequences
frequent message-sequences via Apriori
BMR

Fig. 30: AutoReEngine (Luo and Yu [17], 2013)

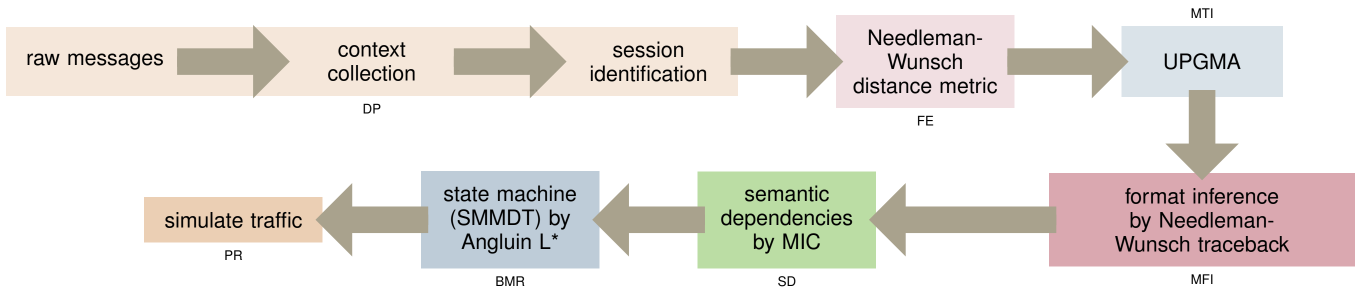

raw messages
context collection
session identification
DP
Needleman-Wunsch distance metric
FE
MTI
UPGMA
format inference by Needleman-Wunsch traceback
MFI
semantic dependencies by MIC
SD
state machine (SMMDT) by Angluin L*
BMR
simulate traffic
PR

Fig. 31: Netzob (Bossert et al. [16], 2014)