Network Message Field Type Clustering for Reverse Engineering of Unknown Binary Protocols

Stephan Kleber and Frank Kargl
Institute of Distributed Systems,
Ulm University, Germany
{stephan.kleber,frank.kargl}@uni-ulm.de

Milan Stute and Matthias Hollick
Secure Mobile Networking Lab,
Technical University of Darmstadt, Germany
{mstute,mhollick}@seemoo.de

Abstract—**Reverse engineering of unknown network protocols based on recorded traffic traces enables security analyses and debugging of undocumented network services. One important step in protocol reverse engineering is to determine data types of message fields. Existing approaches for binary protocols (1) lack comprehensive methods to interpret message content and determine the data types of discovered segments in a message and (2) assume the availability of context, which prevents the analysis of complex and lower-layer protocols. Overcoming these limitations, we propose the first *generic* method to analyze message field data types in unknown binary protocols by clustering of segments with the same data type. Our extensive evaluation shows that our method in most cases provides clustering of up to 100 % precision at reasonable recall. Particularly relevant for use in fuzzing and misbehavior detection, we increase the coverage of message bytes over the state-of-the-art to 87 % by almost a factor of 30. We provide an open-source implementation to allow follow-up works.**

Index Terms—**field data type clustering, protocol reverse engineering, vulnerability research, network security**

I. INTRODUCTION

Protocol reverse engineering (PRE) based on traffic traces aims to infer the specification of unknown network protocols by analyzing traces of network messages typically gained from observing communication of devices implementing this protocol. PRE is often applied to understand malware communication and uncover data exfiltration [4], to configure smart fuzzers [8], or to validate the correct and secure design and implementation of undocumented network services [23]. As a recent example, PRE was necessary to discover a severe vulnerability in the proprietary Apple Wireless Direct Link (AWDL) protocol stack [20], enabling a zero-click exploit [1] affecting all of Apple's iOS-based product lines. Thus, PRE helps in identifying security implications that result from the intended or unintended use of a specific unknown protocol.

PRE based on traffic traces encompasses the uncovering of message types, message formats, semantics, and behavior of the protocol. During this kind of analysis, semantic deduction is one of the most tedious and scarcely automated tasks [13, 20]. One step in semantic analysis is inferring the data type or value domain of fields which can help, for example, to more efficiently configure smart fuzzers or to identify exfiltration [2, 4, 24]. While some methods are available that recognize single field data types and correlations of values, no approach determines relations between fields by their value similarity that can be used to interpret the message contents.

Contribution. This paper proposes a novel method to automatically cluster field data types. We base this inference on the analysis of segments, i.e., subsequences of network messages. We propose to distinguish segments into *clusters of the same field data type* according to their similarity to each other without actually identifying the data type. The resulting knowledge of segments with identical type simplifies follow-up analyses as value domains can be inferred and spoofing or fuzzing require this knowledge. As opposed to previous approaches [2, 3, 5] and particularly important for security assessments of custom and proprietary protocols, we make very few assumptions about the format and sequence of messages. We summarize our main contributions as follows:

- We design the first method to cluster field types of network messages and do so without a limiting set of individual rules per type, making our approach applicable to a wide range of protocols with diverse and unanticipated data representations.
- Based on empirical observations of typical network protocols, we devise a fully-automated parameter selection method that is use-case-specific to the clustering of field values.
- We implement our method as well as FieldHunter [2] and CSP [9] and make all three publicly available.[1]
- Through extensive evaluation of both well-known and proprietary protocols, we show that our method on average achieves an F-score of 0.92 for field type clustering. At the same time, coverage of 87 % message bytes exceeds the state-of-the-art by almost a factor of 30.

II. RELATED WORK

Surveys have proposed to structure the overall PRE process into multiple phases [6, 12]. Typical phases are data collection into traces, feature extraction, message type identification, message format inference, semantic deduction, and behavior model reconstruction. Existing PRE approaches differ substantially for textual and binary protocols, where analysis of textual protocols is often considered the easier task [2, 5, 6, 12]. Thus,

---

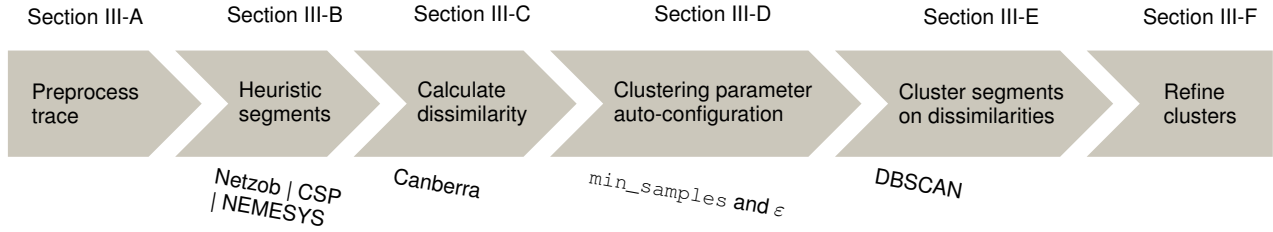[1]https://github.com/vs-uulm/nemesys, fieldhunter, and goo-csp

Fig. 1. Clustering of common kinds of message content data.

we focus on providing a solution for binary network protocols. Previous work, like Discoverer [5], PRISMA [14], Netzob [3], Goo et al. [9], NEMETYL [10], and many others, focused either on the message type and format or on the behavior model of unknown protocols. While most phases are well covered in literature, approaches specifically addressing the interpretation of the message contents, i. e., semantic deduction of fields, are rare. All existing methods are rule-based [2, 3, 5, 9], i. e., they consist of a finite set of individual heuristic rules that explicitly deduce the semantics of a predefined small number of single specific field types, like number, identifier, or network address. We consider FieldHunter [2] the state-of-the-art approach as it has also been re-applied in recent work [9]. If either, the protocol uses a representation of data types that was not anticipated in any of the heuristic rules, or the encapsulation is unknown so that context like addresses is not available, FieldHunter fails to work. As there is no public implementation, we re-implemented FieldHunter and evaluate its results in comparison to our approach in this paper.

As opposed to FieldHunter and all previous rudimentary field type inference approaches mentioned in this section, we aim for a more generic goal than using only a limited number of individual heuristics for field types: Our work is the first to propose clustering of arbitrary data types of message *fields*. We stress that we do not attempt to identify particular field types so that we are not limited to a predefined set of data types. In this work, we do not consider clustering whole messages into different message types since previous work like , e. g., Discoverer [5], PRISMA [14], Netzob [3], or NEMETYL [10], already achieve this goal.

Our approach relies on message segmentation and different methods might achieve different quality. To consider segmentation accuracy in our evaluation, we compare three existing segmenters that work with variable-length fields: Netzob [3] is based on sequence alignment, CSP [9] applies frequency analysis of byte-strings, and NEMESYS [11] uses statistical properties of the message contents to discern one approximated field candidate from the other, forming heuristic segmentations of unknown binary messages. Furthermore, to compare segments to each other, we use the Canberra dissimilarity [10], which we originally proposed for message type identification. We now apply it directly for clustering of segments while, in contrast, its original usage was to be input for sequence alignment of messages.

III. Clustering Data Types

Our approach provides the means to cluster independent message segments into what we call *pseudo data types* without any further knowledge about the protocol. Individual steps of this process are outlined in Figure 1. It is a heuristical method to cluster the same types of data into groups of similar field contents. Having such clusters of segments throughout different messages of a trace reveals relationships of values between messages regardless of the byte positions of the segments within each message. We call the resulting clusters *pseudo* data types because, at this point, we do not know which data type or field semantic the cluster represents. An analyst can still use this knowledge as basis to analyze the properties of the clustered segments and infer their semantic meaning. We now discuss the individual steps.

A. Preprocessing

We first **preprocess** each raw trace. This step includes filtering for the desired protocol and de-duplicating payloads. Our analysis method exploits variances in the contents of messages, so duplicates carry no additional information.

B. Segmentation

We define a **field** in a binary protocol specification as a sequence of bytes at a specific position in a message, with a specific data type such as an integer, a sequence of chars, or a timestamp, and a value domain. In contrast, we define a **segment** to be a field candidate determined from the inference that—in an optimal case—matches the true field from the unknown protocol specification. Segmentation is an important prerequisite for *characterizing the contents of messages*, which is needed to determine the segments' data types, infer their semantics, and ultimately deduce an accurate field definition.

To obtain **segments** from the messages in the traces, we split individual messages into subsequences. Messages of known protocols can be segmented by using dissectors, like those provided by Wireshark.[2] While dissectors are unavailable for unknown protocols to reliably determine message fields, heuristic approximations can be used to find probable field boundaries and obtain segments that are field candidates. Thus, we require a segmenter that can identify segments in unknown protocols. Available solutions include Netzob [3], Goo et al. [9], and NEMESYS [11]. We evaluate these three

---

[2]https://www.wireshark.org

segmenters that have the advantage that they work equally well for protocols of fixed structure and such with dynamic field lengths differing between messages. The idenfied segments are now treated as candidates for protocols fields.

C. Dissimilarity

To calculate a similarity measure for segments, we interpret each of these as a **vector** of byte values. We then calculate a normalized **dissimilarity** value for each pair of segments using the so-called Canberra dissimilarity [10], which extends the better-known Canberra distance [15] to vectors of different dimensions. We store the pairwise dissimilarities between all segments in a dissimilarity matrix $\mathbf{D}$.

We exclude segments from the analysis that are only one byte long as coincidental similarity of arbitrary single bytes throughout messages prevents meaningful analysis of such short segments. Using alternative analysis methods, like frequency analysis, these one-byte segments can later be reincorporated in the analysis. Furthermore, we consider duplicate segment values only once since they increase the computational load without adding new information for the subsequent clustering.

The dissimilarity values for each pair of remaining unique segments serve as affinity values to guide clustering by Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [7] in the next steps.

D. Auto-Configuration

Before clustering, we need to configure two parameters of DBSCAN: the minimum number of elements to form a density core `min_samples` and a measure ε of the least density to be considered part of a cluster. Normally, these parameters need to be configured and tuned manually. For unsupervised and fully automated, configuration-less clustering, we present a new method to automatically determine the parameters for DBSCAN from the properties of the segments identified in the previous step.

The ε **auto-configuration** searches for the knee point in the empirical cumulative distribution function (ECDF) [22] $\widehat{E}_k(d)$ of the dissimilarities between the k-nearest-neighbors (k-NN) of unique segments. For each trace, a set of functions $\widehat{E}_k$ exists, one ECDF for each k. An ECDF is an evenly-spaced step function, jumping by $\frac{1}{n}$ for each of the n samples with a measured value d. In our case, the samples are the segments s_i and s_j in a trace, and their measured value is the dissimilarity $d(s_i, s_j)$. Applied to the k-NN function, the ECDF's value thereby is the fraction of all segments in a trace that have a Canberra dissimilarity less or equal to their respective kth nearest-neighbor. The ECDF plots the changes in distances between neighbors. A clear drop d_κ, located at the knee point $\widehat{E}_k(\kappa)$, is then considered a suitable choice for ε that allows DBSCAN to reliably detect cluster boundaries.

Of all possible $\widehat{E}_k$, we want to dynamically select k in such a way that its ECDF has the most distinct drop in the density of the segment similarity. The function that contains the most distinct change in distances between neighbors has the sharpest knee point. Consequently, we search for the $\widehat{E}_k$ with the sharpest knee, with sharpness measured as the value of the δd at the maximum of $\delta \widehat{E}_k$. Algorithm 1 describes the process to select the desired k. We iterate k only between 2 and $\mathrm{round}(\ln n)$ to limit the number of unnecessary calculations. This is sufficient since the sharpest relevant knees always are in the distance distributions of neighboring segments.

To determine the rightmost knee point in $\widehat{E}_k$ with the selected k, we apply the Kneedle algorithm [19]. Kneedle requires smoothing of the ECDF, for which we use a spline, to remove local statistical fluctuations before accepting it as input. Figure 2 illustrates the ECDF, the $\max(\delta \widehat{E}_k)$, the effect of smoothing, and the detected knee, used as ε with segments generated from a trace of 1,000 NTP messages. For the 2nd parameter `min_samples`, we note that DBSCAN is not very sensitive and setting it to $\ln n$ simply prevents scattering large traces into too many small clusters.

E. Clustering

Next, we **cluster segments** with DBSCAN using the determined parameters ε and `min_samples`. DBSCAN is a popular and efficient clustering algorithm that makes no assumptions about the shape of clusters, does not require the target number of clusters as input, and treats outliers as noise. These properties set it apart from traditional clustering methods, e. g., k-means or spectral clustering, which are unsuitable for our purpose since we do not know the shape and number of clusters. For other clustering methods, like agglomerative clustering, affinity propagation, or support vector machines, automating the tuning of the parameters for previously unseen traces is challenging. In comparison, DBSCAN's main advantage is that we can design a method to directly derive its parameters from the dissimilarity distribution of a trace, as described in the previous section. Thus,

---

Algorithm 1: ε auto-configuration

input : Set of dissimilarities D;
Sensitivity parameter of Kneedle S;
Smoothness parameter of B-Spline interpolation s;
output: ε

function kNN(D, k)
   Determine the k-NN of all segments represented in
    D;
   return Dissimilarities of all segments' kth-NN;
end

foreach $2 \leq k \leq \mathrm{round}(\ln n)$ **do**
   $\widehat{E}_k \leftarrow$ ecdf(kNN(D, k));
   $\widehat{B}_k \leftarrow$ bSpline($\widehat{E}_k$, s);
end
$k' \leftarrow \underset{k}{\mathrm{argmax}}\ \delta \widehat{B}_k$; /* Value of the maximum
  increase in distance */
$d_\kappa \leftarrow$ Kneedle($\widehat{B}_{k'}, s$);
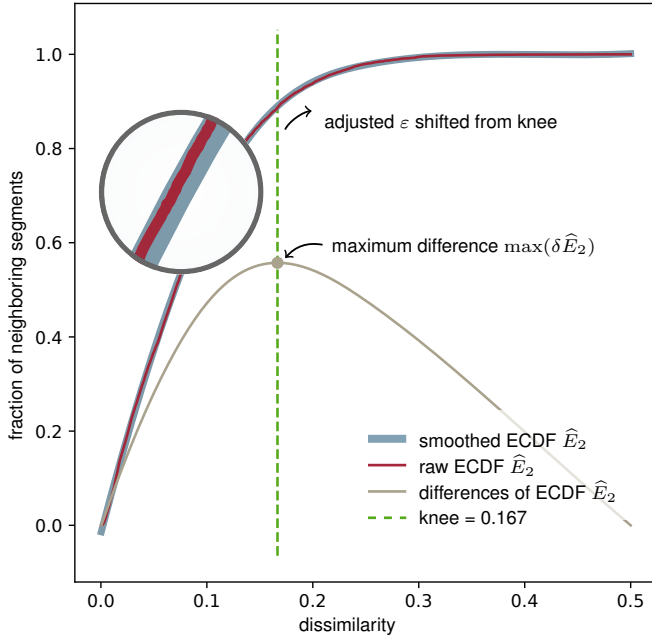$\varepsilon \leftarrow d_\kappa$;

Fig. 2. ECDF $\widehat{E}_2$ and its knee, detected by Kneedle, at a dissimilarity of 0.167 used as ε.

the proposed configuration procedure is completely automated and requires no re-training or iterative tuning for new traces as may be the case for other clusterers.

The algorithm identifies high-density cores within noisy data and determines them to be clusters of similar segments. The segment density is high in areas where segments have a low dissimilarity to each other. Each cluster groups similar segments and thus comprises fields of a common data type. We validate the underlying assumption that clusters regularly coincide with data types in the first part of our evaluation (Section IV-B).

In rare cases where the dissimilarity distribution leads to multiple knees in the ECDF, the so-determined ε does not denote suitable densities to cluster field data types and is too large. In this situation, a single large cluster contains more than $60\,\%$ of the segments that are not considered noise. To prevent this and instead select the next smaller knee for an ε, we consider only a subset of the original $\widehat{E}_k$. More specifically, we repeat the whole ε auto-configuration process for a $\widehat{E}_k'$ that is trimmed to the last detected knee κ, which becomes the rightmost value. Thus, $\widehat{E}_k' = \widehat{E}_k(\{d < d_\kappa : d \in \mathrm{D}\})$. We then cluster with the new ε value.

F. Cluster Refinement

In situations where the variability of the field values in the input trace is not uniformly distributed, multiple clusters may result for the same data type. This overclassification is not only a limitation of DBSCAN and we noticed that similar alternatives, e. g., HDBSCAN and OPTICS, suffer from the same effect. We favor DBSCAN since it provides more possibilities to fine-tune the cluster boundaries for our use case. During pilot analyses of known protocols, we observed that overclassified clusters are often linked via sparsely populated but detectable areas. To mitigate the overclassification, we introduce an additional step: **cluster refinement**.

In cluster refinement, we aim to automate the detection and merging of clusters that are nearby and have a similar density. For any two clusters c_i and c_j, we define the *link segment* $s_{i,j}^{\mathrm{link}}$ as the segment in c_i that is closest to c_j, i. e.,

$$s_{i,j}^{\mathrm{link}} = \underset{s_i \in c_i}{\mathrm{argmin}}\, d(s_i, s_j), \quad \forall s_j \in c_j$$

$d_{i,j}^{\mathrm{link}}$ is the distance between the link segments $s_{i,j}^{\mathrm{link}}$ and $s_{j,i}^{\mathrm{link}}$.

Using this definition, we propose two heuristic cluster merge conditions: (1) The clusters are very close-by, and the densities within an ε around the link segments are similar. (2) The clusters are somewhat close-by, and the whole clusters have similar densities. We quantify closeness and density differently in both conditions since we intend to capture different notions of closeness and density, i.e., local ε-density at elements linking clusters and density of clusters as a whole.

In **Condition 1**, *clusters are very close by* if the link-dissimilarity is less than the mean $\overline{D(c_i)}_{\mathrm{arithm}}$ of the set of pairwise dissimilarities in c_i or respectively c_j:

$$d_{i,j}^{\mathrm{link}} < \max(\overline{\mathrm{D}(c_i)}_{\mathrm{arithm}}, \overline{\mathrm{D}(c_j)}_{\mathrm{arithm}})$$

Further, we use a density definition for an ε-neighborhood around the nearest points between similar clusters. W. l. o. g., we define $s_l = s_{i,j}^{\mathrm{link}}$. The *density ρ within an ε around the link segment* in c_i with the set of dissimilarities $\mathrm{D}(c_i)$,

$$\mathrm{D}(\varepsilon, s_l) = \{d(s_l, s_c)\ :\ d(s_l, s_c) \le \varepsilon,\ s_l \neq s_c,\ s_c \in c_i\},$$

with $\mathrm{D}(\varepsilon, s_l) \subseteq \mathrm{D}(c_i)$, is thus defined by $\rho_\varepsilon(s_l) = \overline{\mathrm{D}(\varepsilon, s_l)}_{\mathrm{median}}$. We observed that a suitable ε is half of the maximum extent $d_{\max}$ of the cluster with the fewer segments: $\varepsilon = \frac{d_{\max}}{2}$. The density around the link segment in c_j is defined accordingly. Finally, ε-densities around link segments are considered similar if their difference is less than $\varepsilon\rho Threshold$:

$$\left| \rho_\varepsilon(s_{i,j}^{\mathrm{link}}) - \rho_\varepsilon(s_{j,i}^{\mathrm{link}}) \right| < \varepsilon\rho Threshold$$

Condition 2 allows a larger cluster distance but has a stronger density requirement. Close-by here means closer than the mean between both cluster's "neighbor densities" normalized to the extent of the cluster. To formalize this, we need $\mathrm{minmed}(i)$, the median values for the 1st-nearest neighbors that are the minimum distances for each segment to any other in c_i, respectively c_j:

$$\mathrm{minmed}(i) = \overline{\left\{ \underset{s_b \in c_i}{\min}(\{d(s_a, s_b) : s_a \neq s_b\}) : s_a \in c_i \right\}}_{\mathrm{median}}$$

In terms of Condition 2, *clusters are somewhat-close-by* if

$$d_{i,j}^{\mathrm{link}} < \frac{\dfrac{\mathrm{minmed}(i)}{\overline{\mathrm{D}(c_i)}_{\mathrm{arithm}}} + \dfrac{\mathrm{minmed}(j)}{\overline{\mathrm{D}(c_j)}_{\mathrm{arithm}}}}{2}$$

As expression of the overall density in the cluster, we use minmed. In contrast to the ε density defined above, whole *clusters have similar density* if

$$|\mathrm{minmed}(i) - \mathrm{minmed}(j)| < neighborDensityThreshold.$$

The selection of the values $\varepsilon\rho Threshold = 0.01$ and *neighborDensityThreshold* $= 0.002$ results from empirical observation of real-world protocols.

Unlike overclassification, occasional underclassification combines different field data types in one cluster. This may be the case if a single value is similar to a group of others but has a distinct function, like an enumeration value. To compensate for this, we split clusters if they have extremely polarized value occurrences, e.g., they exhibit many unique values, together with very few, very high occurring ones. For this purpose, occurrences are defined as the count $|b|$ of segments $s \in c$ with a value b. We count all different values b_i and calculate the standard deviation $\sigma(\{|b_i| : 0 \le i < |c'|\})$ for this cluster c', with b_i being the value of one or multiple segments s. To interpret the counts of values of one cluster, we use the percent rank PR [18] as a combined measure of the value occurrence frequency and value diversity. A $\text{PR}(c', F) = 95$ means that 95 % of the value counts in cluster c' are below the given occurrence frequency of interest F. We select F depending on the cluster size to be $\ln |c'|$. Thus, the same value is the pivot to split the cluster into two subclusters containing all segments with value count $|b_i| \le F$ and another for $|b_i| > F$ for i enumerating all distinct values of segments in the cluster. Consequently, if $\text{PR}(c', F) > 95 \ \wedge \ \sigma(\{|b_i| : 0 \le i < |c'|\}) > F$, we split c' at the pivot $F = \ln |c'|$.

G. Summary

After completing this fully automated procedure, we now have generated vectors for each segment, calculated their pairwise Canberra dissimilarity, determined a suitable ε value, clustered the segments using DBSCAN, and refined the clusters. This completes the clustering of segments into pseudo data types. Next, our evaluation will show how accurately this can be done.

IV. EVALUATION

Using our proof-of-concept implementation, we evaluate two different aspects of our approach. First, we validate that data types can be clustered accurately by segment similarity. Second, we evaluate the accuracy achievable by using heuristic segmentation in the absence of ground truth. We illustrate the validity of these two aspects by clustering statistics.

A. Metrics and Setup

For a quantitative representation of the clustering quality, we calculate **precision** P and **recall** R of the clusters compared to the true data types by the number of true positives (TP), false positives (FP), and false negatives (FN) as:

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{and} \quad R = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

For clustering into more than two clusters, TP, FP, true negatives (TN), and FN are defined combinatorically via the correct and incorrect pairwise assignments of unique segments,

as described by Manning et al. [16]. Hence, the number of positives and negatives for m clusters c_i are given as:

$$\text{TP+FP} = \sum_i \binom{|c_i|}{2} \quad \text{and} \quad \text{TN+FN} = \sum_{i,j} \left(|c_i| \cdot |c_j|\right),$$

where $j = \{0 \dots (m-1)\} \setminus i$. The true positives are:

$$\text{TP} = \sum_i \sum_l \binom{|t_{i,l}|}{2},$$

where $t_{i,l}$ denotes the segments of data type l in cluster i. The false negatives are defined through the missed true pairs by false assignments to different clusters and to the noise. Thus, the count of false negatives is given by the sum of both kinds of false negatives through:

$$\text{FN} = \sum_i \sum_l \frac{(|t_l| - |t_{i,l}|) \cdot |t_{i,l}|}{2}$$
$$+ \sum_l \binom{|t_{n,l}|}{2} + \sum_l \frac{(|t_l| - |t_{n,l}|) \cdot |t_{n,l}|}{2},$$

where $t_{n,l}$ are the segments of data type l assigned to the noise.

To compare the quality between different protocols and input segments, we require an overall quality measure. Therefore, we calculate the $F_{\frac{1}{4}}$ **score** from precision and recall. The F_β score is a common measure for the clustering accuracy and defined by the harmonic mean of precision and recall [17]. Parameter β adjusts the weight of precision and recall in the mean. With $\beta = \frac{1}{4}$, we place four times more emphasis on precision than recall. We decided on this weighting since precise clusters are crucial for a meaningful data type analysis in protocols. At the same time, low recall diminishes the coverage but does not reduce the validity of the overall analysis result. As coverage we define the ratio between the number of inferred bytes and all bytes of all messages in a trace. Since coverage refers to the number of bytes and precision and recall to segment pairs the result statistics are not directly correlated.

The messages we use for developing our approach are collected from **traces** of the binary network protocols DHCP, DNS, NBNS, NTP, and SMB.[3] All traces are publicly available.[4] In addition, we also use traces of two proprietary protocols, namely Apple Wireless Direct Link (AWDL) and Auto Unlock (AU). AWDL is a Wi-Fi-based link-layer protocol for peer-to-peer communication. AU is a proprietary distance bounding protocol.[5] Both protocols were not publicly documented until they recently were reverse engineered manually. The reverse-engineered specification of AWDL, including a dissector, is publicly available [20], and we had access to a private Wireshark dissector of the AU protocol. Thus, both

---

[3]Dynamic Host Configuration Protocol (RFC 2131), Domain Name System (RFC 1035), NetBIOS Name Service (RFC 1002), Network Time Protocol (RFC 958), and Server Message Block

[4]DHCP, NBNS, NTP, and SMB extracted from http://download.netresec.com/pcap/smia-2011/; DNS extracted from https://ictf.cs.ucsb.edu/archive/2010/dumps/ictf2010pcap.tar.gz

[5]https://support.apple.com/en-us/HT206995

TABLE I
CLUSTERING STATISTICS FOR DATA TYPE
CLUSTERING FROM GROUND TRUTH.

proto.	msg.s	fields	ε	P	R	$F_{\frac{1}{4}}$
DHCP	1000	1017	0.172	0.96	0.93	0.95
DNS	1000	839	0.063	1.00	0.95	1.00
NBNS	1000	734	0.049	1.00	0.91	0.99
NTP	1000	3822	0.121	1.00	0.96	1.00
SMB	1000	1175	0.218	**0.59**	**0.70**	**0.60**
AWDL	768	2190	0.096	1.00	0.77	0.98
DHCP	100	229	0.212	0.76	**0.66**	**0.75**
DNS	100	114	0.143	1.00	0.89	0.99
NBNS	100	131	0.121	1.00	0.56	0.96
NTP	100	470	0.198	1.00	1.00	1.00
SMB	100	171	0.169	0.92	**0.48**	0.87
AWDL	100	396	0.101	0.99	0.59	0.95
AU	123	316	0.366	1.00	0.44	0.93

Worst cases are printed in bold.

TABLE II
COMBINATORIAL CLUSTERING STATISTICS AND COVERAGE (COV.)
FOR PSEUDO DATA TYPES OF HEURISTIC SEGMENTS.

proto.	msg.s	Netzob				NEMESYS				CSP			
		P	R	$F_{\frac{1}{4}}$	cov.	P	R	$F_{\frac{1}{4}}$	cov.	P	R	$F_{\frac{1}{4}}$	cov.
DHCP	1000		fails			**0.88**	**0.33**	**0.80**	99 %	0.85	0.35	0.79	99 %
DNS	1000	**0.99**	**0.96**	**0.99**	100 %	1.00	0.85	0.99	99 %	0.95	0.76	0.93	99 %
NBNS	1000	0.99	0.74	0.97	100 %	**1.00**	**0.95**	**1.00**	100 %	0.90	0.30	0.80	99 %
NTP	1000	**0.94**	**0.90**	**0.94**	88 %	0.65	0.61	0.64	95 %	0.68	0.53	0.67	73 %
SMB	1000		fails			**0.57**	**0.02**	**0.24**	81 %	0.38	0.01	0.11	79 %
AWDL	768	**1.00**	**0.93**	**0.99**	99 %	0.80	0.16	0.64	98 %		fails		
DHCP	100	0.44	0.11	0.38	83 %	**0.83**	**0.52**	**0.80**	87 %	0.24	0.07	0.21	87 %
DNS	100	**0.98**	**0.86**	**0.97**	100 %	0.98	0.75	0.96	95 %	0.46	0.13	0.40	87 %
NBNS	100	0.91	0.85	0.91	93 %	**0.98**	**0.56**	**0.94**	99 %	0.93	0.32	0.84	82 %
NTP	100	**0.98**	**0.23**	**0.82**	65 %	0.87	0.01	0.19	39 %	0.71	0.00	0.05	65 %
SMB	100	0.59	0.20	0.53	81 %	**0.84**	**0.12**	**0.63**	67 %	0.42	0.11	0.36	74 %
AWDL	100	**0.99**	**0.51**	**0.94**	90 %	0.59	0.05	0.35	92 %	0.99	0.43	0.92	92 %
AU	123		fails			1.00	0.05	0.49	84 %	**1.00**	**0.14**	**0.74**	81 %

Best (green) and worst (red) cases are printed in bold and colored.

protocols constitute realistic use cases where ground truth is available to verify our results. We use only protocols with ground truth to compare our results to, which is not available for truly unknown protocols. Otherwise, statistical analysis of the quality of our approach would not be possible.

As the source of the ground truth, we parse the Wireshark dissectors' output for each message. All evaluated protocols are binary, while DNS, DHCP, SMB, and AWDL also contain embedded char sequences. The binary fields of DNS, NBNS, and NTP have fixed length, while DHCP, SMB, AWDL, and AU use a mix of fixed and variable-length fields. DHCP, DNS, NBNS, SMB, AWDL, and AU support varying numbers of fields in different messages while NTP has a fixed structure. Thus, our set of traces represents a wide variety of different protocol properties. For the evaluation of clustering and recognition, we truncate the traces to achieve comparable results. We truncate to 100 and 1 000 messages per protocol to show the impact of the trace size on the inference quality. Fewer messages were available for AWDL and AU, which we consider in the discussion of the results.

B. Pseudo Data Type Clustering Validation

First, we validate our base assumption that data types of segments can be clustered using the Canberra dissimilarity. Section III describes the process to cluster for pseudo data types. For validation, we compare the clustering results to the true field data types from the Wireshark dissectors. This provides a baseline to validate that different data types can correctly be distinguished by our dissimilarity measure.

Cluster statistics quantify the accuracy of the match between data types and clusters. As overall quality metrics, we provide P, R, and F-score for our test protocols in Table I. For reference, we include the number of messages in the trace, the number of *unique* fields in the trace, and the auto-configured ε. The amount of noise identified by DBSCAN is always zero.

The F-score values in Table I are near the optimum of 1 which shows that data types can be clustered with high utility. However, the SMB trace with 1 000 messages stays behind the other results due to its low precision. Inspection of the individual clusters shows that timestamps and signatures have

erroneously been placed together in one cluster. Ignoring this single cluster for the sake of the argument, we gain a precision of 0.96 while the recall drops to 0.37. This is the only instance in all our test runs where a parameter selection fails with such a significant impact, hinting towards great robustness of the method. Protocols with complex message formats, like DHCP and SMB, require a large amount of variability in the trace to allow for a decent analysis result. Table I shows this by the lower F-scores and specifically the lower recall for these complex protocols with smaller traces of 100 messages compared to the results for 1 000 messages of the same protocols. This is due to multiple clusters representing a disjointed group of similar segments, reducing the recall.

Based on the high precision of almost all clustering results, we conclude that most field types can accurately be clustered by means of dissimilarity. Overall, this aspect of our evaluation shows that clusters match with true field types and thus validate our approach of data type clustering.

C. Clustering with Imperfect Segmentation

Next, we present our evaluation of clustering similar segments of real-world protocols without relying on perfect segmentation from Wireshark dissectors. Instead, we use the existing heuristic segmenters Netzob [3], NEMESYS [11], and CSP [9] on our set of known test protocols. This way, we emulate the lack of ground truth during clustering while retaining the possibility to measure the inference quality.

We compare three existing heuristics segmenters that are available for unknown binary protocols as a basis for our field data type clustering. According to our results, no single segmenter is clearly superior to the others and each has its strengths and weaknesses with regard to the kind of analyzed protocol. Table II contains the clustering statistics P, R, and the F-score per test protocol. We mark the best-performing segmenter for each protocol trace by bold printed values in the table. Four analysis runs fail due to exceeding runtime or memory constraints.

A significant number of segments cannot be clustered correctly and concisely as their boundaries are shifted relative to the true position they should optimally mark. These fragments

NTP timestamp A	`d2 3d1903b3 fcdab1`
NTP timestamp B	`d2 3d197a 01581062`
NTP timestamp C	`d2 3d191cd025 d074`

Fig. 3. Typical errors in heuristically inferred segment boundaries (vertical lines) that should approximate timestamps. The shaded area marks static bytes.

blur some segment clusters to the extent that we cannot clearly separate the affected data types. Figure 3 illustrates how this affects the dissimilarity measure and, thus, the clustering result with an example of three timestamps that have incorrect additional boundaries splitting the true field. These least significant bytes of the timestamps, regarded by themselves, seem random and thus cannot be clustered based on their value. This error is not an effect of the dissimilarities used as segment features or the clustering algorithm, but stems from incorrect partitioning of the message by the segmenters.

This error in the approximated boundaries of high-entropy fields is the reason for SMB's low recall as it contains a signature that is randomly split by all of the segmenters, since its contents look random across different messages. AU's segments suffer from a slightly different but related issue: long sequences of 32-bit integers, representing measurement results, look static in some instances and random in others so that the dissimilarity is not successfully exploitable for clustering. Since for AU we only have 123 messages available to evaluate, we hypothesize that the variance incurred by larger traces would have a positive impact if available. For the other traces of different sizes of all protocols, the precision stays high compared to the true-fields baseline (Section IV-B).

Considering the best case per protocol, only the larger trace of SMB exhibits an unsatisfactory precision of of 0.57, which is still remarkable, since knowing the true segments leads to only a very small improvement ($R = 0.59$). The smaller SMB trace and the AU trace are unsatisfying due to their low recall while precision in both cases remains high. We marked the three unsatisfying cases by red-colored F-scores and in contrast colored all F-scores of at least 0.8 green, which we consider successful analyzes. Most of the results even score above 0.9 with a precision of also better than 0.9. In the face of the identified problems that are realistic for working with unknown protocols, we argue that our method can cope with the inaccurate segmentation to a large degree.

The remaining challenge is to select the most suited segmenter for a protocol trace. We see that Netzob is most suited for protocols with distinct patterns of repeating value sequences, e. g., NTP having fixed structure and AWDL with a type-length-value (TLV) record structure. Large messages cause Netzob to fail due to the exponential increase in runtime, which is the case for larger traces of DHCP and SMB, and for the AU trace. NEMESYS deals well with large and complex messages, especially since they contain a mixture of number values and chars, which fits the heuristic of NEMESYS best. CSP performs minimally worse for larger traces than NEMEYS, but it lags behind for smaller traces. As CSP is more dependent an the variance in the trace, it is best applied to large traces where it poses an alternative to NEMESYS.

D. Evaluation Summary

This evaluation provides two insights about our approach: (1) field data type clustering works as intended with very little requirements towards and assumptions about the protocols, but (2) field data type clustering highly depends on the segmentation result, where we rely on existing approaches that provide results of only limited quality. The higher the correctness of the heuristic segmentation, the better the message field type clustering can perform.

In comparison, FieldHunter is able to discern the concrete data type of typically one or two fields per message, leading to a coverage of 3 % on average across all protocols. While, in contrast, our clustering method per se cannot determine the field type, it achieves an average coverage of 87 % (see Table II), which means that we can provide information about the structure of messages in terms of field similarity and field's value domains for almost the complete content of all messages.

V. CONCLUSION

In this paper, we propose a novel method to cluster field data types in messages of unknown binary protocols. It requires recorded network traces and leverages the similarity of segments to group them into clusters representing a common data type. Our efficient clustering of message segments facilitates subsequent analyses to identify their likely semantic function. We envision that identified data types and visual analytics will improve the analysis efficiency of unknown network messages by providing the means to determine the most security relevant message parts to investigate further in a given trace.

In PRE, a typical high-effort task is to understand the large-scale structure of messages. Knowing such structure is often the basis to analyze, e. g., data exfiltration by malware, privacy violations, targets for spoofing and fuzzing for vulnerability testing as we illustrated, e. g., in Kröll et al. [13] and Stute et al. [21]. Automating this process saves effort and time and our work contributes to such automation. Opposed to previous work that uses a set of heuristics to recognize a fixed number of field types, clustering of segments is also applicable if the protocol contains unanticipated data representations, e. g., encodings, since it only relies on the segments' similarity and occurrence. Thus, we can cover large parts of the messages in the trace, while previous work—with a coverage of only 3 % on average—leaves most of the message content completely unintelligible. While clustering per se does not reveal data types, it simplifies an analyst's interpretation of the message content. Our method increases the coverage of the interpretable message content to 87 % on average, outperforming the state-of-the-art by almost factor 30 and enabling comprehension of the large-scale message structure.

We first evaluated our approach for both publicly documented as well as undocumented protocols relying on ground truth message fields derived from Wireshark dissectors. We find that most field data types can be clustered with high

precision when knowing correct field boundaries. In realistic situations, where field boundaries are not known and heuristic segmenters like Netzob, NEMESYS, or CSP are applied, the recall is lower, but data types can still be distinguished with a precision close to 100 % in most cases. Our approach works also for protocols without IP encapsulation, like AWDL and AU, where previous work could not be applied due to the field type heuristics' reliance on context information.

We see two main areas for future work. Firstly, we propose to combine our data type clustering with the deduction of intra- and inter-message semantics similar to FieldHunter [2]. This would enable the interpretation of, e. g., length fields and message counter fields. Moreover, we intend to automatically learn value generation rules from the cluster contents using LSTM or similar machine learning methods to predict probable field values for fuzzing and misbehavior detection.

REFERENCES

[1] I. Beer. *Google Project Zero: An iOS Zero-Click Radio Proximity Exploit Odyssey*. 2020. URL: https://googleprojectzero.blogspot.com/2020/12/an-ios-zero-click-radio-proximity.html.

[2] I. Bermudez, A. Tongaonkar, M. Iliofotou, M. Mellia, and M. M. Munafò. "Towards Automatic Protocol Field Inference". In: *Computer Communications* 84 (June 2016). Elsevier.

[3] G. Bossert, F. Guihéry, and G. Hiet. "Towards Automated Protocol Reverse Engineering Using Semantic Information". In: AsiaCCS. 2014.

[4] C. Y. Cho, D. Babić, E. C. R. Shin, and D. Song. "Inference and Analysis of Formal Models of Botnet Command and Control Protocols". In: CCS. 2010.

[5] W. Cui, J. Kannan, and H. J. Wang. "Discoverer: Automatic Protocol Reverse Engineering from Network Traces". In: USENIX Security. 2007.

[6] J. Duchêne, C. L. Guernic, E. Alata, V. Nicomette, and M. Kaâniche. "State of the Art of Network Protocol Reverse Engineering Tools". In: *Journal of Computer Virology and Hacking Techniques* 14.1 (Feb. 2018). Springer.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: KDD. 1996.

[8] H. Gascon, C. Wressnegger, F. Yamaguchi, D. Arp, and K. Rieck. "PULSAR: Stateful Black-Box Fuzzing of Proprietary Network Protocols". In: SecureComm. 2015.

[9] Y.-H. Goo, K.-S. Shim, M.-S. Lee, and M.-S. Kim. "Protocol Specification Extraction Based on Contiguous Sequential Pattern Algorithm". In: *IEEE Access* 7 (2019).

[10] S. Kleber, R. W. v. d. Heijden, and F. Kargl. "Message Type Identification of Binary Network Protocols using Continuous Segment Similarity". In: INFOCOM. 2020.

[11] S. Kleber, H. Kopp, and F. Kargl. "NEMESYS: Network Message Syntax Reverse Engineering by Analysis of the Intrinsic Structure of Individual Messages". In: WOOT. 2018.

[12] S. Kleber, L. Maile, and F. Kargl. "Survey of Protocol Reverse Engineering Algorithms: Decomposition of Tools for Static Traffic Analysis". In: *IEEE Communications Surveys and Tutorials* 21.1 (Feb. 2019). Firstquarter.

[13] T. Kröll, S. Kleber, F. Kargl, M. Hollick, and J. Classen. "ARIstoteles - Dissecting Apple's Baseband Interface". In: ESORICS. 2021.

[14] T. Krueger, H. Gascon, N. Krämer, and K. Rieck. "Learning Stateful Models for Network Honeypots". In: AISec. 2012.

[15] G. N. Lance and W. T. Williams. "Computer Programs for Hierarchical Polythetic Classification ("Similarity Analyses")". In: *The Computer Journal* 9.1 (May 1966).

[16] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Online edition. Cambridge, England: Cambridge University Press, 2009.

[17] C. J. van Rijsbergen. *Information Retrieval*. 2nd Revised edition. London, Boston: Butterworth-Heinemann Ltd, Mar. 1979.

[18] J. T. Roscoe. *Fundamental Research Statistics for the Behavioral Sciences*. 2nd edition. International Series in Decision Processes. New York: Holt, Rinehart and Winston, 1975.

[19] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan. "Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior". In: ICDCSW. 2011.

[20] M. Stute, D. Kreitschmann, and M. Hollick. "One Billion Apples' Secret Sauce: Recipe for the Apple Wireless Direct Link Ad hoc Protocol". In: MobiCom. 2018.

[21] M. Stute, S. Narain, A. Mariotto, A. Heinrich, D. Kreitschmann, G. Noubir, and M. Hollick. "A Billion Open Interfaces for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link". In: USENIX Security. 2019.

[22] A. W. van der Vaart. "Empirical Processes". In: *Asymptotic Statistics*. Cambridge, England: Cambridge University Press, 1998.

[23] S. Wen, Q. Meng, C. Feng, and C. Tang. "Protocol Vulnerability Detection Based on Network Traffic Analysis and Binary Reverse Engineering". In: *PLOS ONE* 12.10 (Oct. 2017). Public Library of Science.

[24] C. Wressnegger, A. Kellner, and K. Rieck. "ZOE: Content-based Anomaly Detection for Industrial Control Systems". In: DSN. 2018.