Amplifying Limited Expert Input to Sanitize Large Network Traces

Xin Huang, Fabian Monrose, Michael K. Reiter
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
{huangxin,fabian,reiter}@cs.unc.edu

Abstract—We present a methodology for identifying sensitive data in packet payloads, motivated by the need to sanitize packets before releasing them (e.g., for network security/dependability analysis). Our methodology accommodates packets recorded from an incompletely documented protocol, in which case it will be necessary to consult a human expert to determine what packet data is sensitive. Since expert availability for such tasks is limited, however, our methodology adopts a hierarchical approach in which most packet inspection is done by less-trained workers whose designations of sensitive data in selected packets best match the expert's. At the core of our methodology is a data reduction and presentation algorithm that selects candidate workers based on their evaluations of a small number of packets; that solicits these workers' designations of sensitive data in a larger (but still minuscule) subset of packets; and then applies these designations to mark sensitive data in the entire data set. We detail our algorithms and evaluate them in a realistic user study.

Keywords-sanitization; packet payloads; sensitive data

I. Introduction

Visibility into packet payloads supports numerous network security defenses and dependability analyses. For example, technologies ranging from simple signature-based intrusion detection systems (e.g., Snort, www.snort.org) to advanced techniques for developing exploit signatures (e.g., [32]) require access to network packet payloads. To evaluate the efficacy of such proposed defenses, therefore, it is necessary to have access to payload-bearing network traffic traces on which to test them.

While there has been significant progress in the release of other types of network traffic traces for research purposes in the last decade (e.g., see www.caida.org, www.predict.org), the release of packet payloads remains severely limited due to privacy concerns, and this continues to hamper research progress in numerous types of network defense and performance tests. Packet payloads can contain sensitive information ranging from personal user information to security-relevant data about network topology and service configuration. Thus, data publishers that release packet traces must first sanitize the traces by removing the sensitive information — and in virtually every case, this sanitization includes deleting the payloads in their entirety. This obviously destroys the utility of the trace for research that requires packet payloads.

The extreme rarity with which payload data is released is due to numerous challenges that data publishers face in trying to sanitize packet payloads. First, almost any interesting trace contains too many packets for an administrator to examine exhaustively. Second, even if the trace contains packets of only one protocol (e.g., selected by filtering on ports), the packet formats within that protocol may be numerous or, even worse, undocumented. For example, the Samba project required many researchers' efforts over several years to reverse engineer the file-sharing protocol in Microsoft Windows networks, for which the protocol specification was not released to the public. Third, packet payloads may contain many types of information that may be deemed sensitive, e.g., user names, IP addresses, passwords, host names, and a range of user-generated content. Indeed, the sheer diversity of content that one might deem as sensitive in a free-form protocol like HTTP is overwhelming.

As a step forward in this space, in this paper we propose a framework and tool to support packet trace sanitization. To accommodate incompletely documented protocols, our framework is built around a human *expert* who can explore selected protocol packets well enough to accurately identify the sensitive information they contain. However, since dataset release is rarely a business priority, we presume that this expert has very little time to devote to this effort. For this reason, we structure our framework hierarchically, using the expert's input to select others from a set of candidate *workers*, based on their abilities to mark sensitive data in packets similarly to the expert. These selected workers then mark sensitive data in a small group of packets that can best represent the characteristics of the overall dataset, which our technique then applies to automatically identify sensitive data of the remaining packets in that dataset. We stress that the expert is involved only in marking sensitive data for a very small number of packets — generally far fewer than the total number of packet formats available in the protocol. As such, we cannot impose upon the expert to analyze even one packet of each format, and of course, we may not even know how many formats there are due to the unavailability of the protocol specification.

At the core of this technique is an algorithm that selects and presents packet data to the workers in a fashion that best enables them to identify fields that they deem sensitive and

then to extrapolate from those inputs on that selected data to sanitize the entire dataset. (The expert examines only a small subset of the representative packets selected for the workers.) Doing so in a way that achieves good precision and recall in sanitizing the whole dataset requires that our technique (i) judiciously select the data that the workers will examine; (ii) organize the presentation of the selected data to maximize each one's competence in identifying sensitive fields; and (iii) effectively draw inferences from their inputs to sanitize fields in packet data not presented to them directly.

At a high level, our technique accomplishes these goals through a multistep process. Packets in the trace are first divided into contiguous tokens, each with a type. Stratified sampling is applied to these typed token sequences in order to select a fraction of the packets for further analysis, while minimizing the likelihood of excluding any particular packet type. These selected packets are clustered into groups with similar structure; intuitively (and ideally), these clusters correspond to packet formats. We then select representatives from each cluster, which we present to a worker in an aligned fashion so as to best reveal their common structures, a technique known to accelerate visual recognition of homogeneous structures [9]. The best workers are selected from a group of workers by comparing each one's performance to that of the expert in marking a small subset of representatives; the selected workers then mark sensitive fields in all representatives. After the workers identify sensitive tokens in the representatives for each cluster, these identified tokens are mapped onto the entire dataset, in order to identify sensitive tokens in packets that the workers never examined.

We describe our design and implementation of a tool that implements this approach and present an evaluation based on a user study involving professional network administrators as workers. Our results show that both clustering and alignment have a statistically significant, positive effect on their abilities to identify sensitive data and that the effects of the two are additive. The study also reveals that even network administrators show substantial variability in their abilities to locate sensitive information in network data, underscoring the difficulty of the task at hand and the need to down-select workers on the basis of the similarity of their markings to an expert's. We show that combining the sensitivity determinations of the two best workers and using these to mark the entire dataset identifies sensitive data in the original dataset with at least 0.9 recall and precision.

II. RELATED WORK

Over the past decade, there has been a marked increase in the number of proposals for anonymizing network data (e.g., [16, 12, 31, 26, 25, 10]). For the most part, these works attempt to sanitize network data by applying various transformations to fields within packet headers (e.g., using prefix-preserving anonymization [25]), by using domain knowledge to search for specific patterns (such as URLs or bytecode) using regular expressions [17, 11, 26], by shuffling payloads while preserving the ability to search for short substrings [27], or by deleting the payloads altogether. Unfortunately, many, if not all, of these proposals require specific parsers for each protocol of interest.

In what follows, we attempt to move the field forward by taking advantage of techniques for inferring packet formats, without relying on having a protocol specification at hand. In particular, we extend prior work from the protocol reverse engineering community (e.g. [3, 19]) where byte-based sequence alignment has been applied to raw network traces for uncovering protocol message formats.[1] However, as Cui et al. [7] discovered, byte-based sequence alignment is not particularly well suited for this task when messages of the same format can have high variance in the bytes of certain fields. To address this, we further exploit sequence alignment to derive a new measure of similarity for packet payloads, and then generate compact clusters suitable for human inspection afterwards.

Our work is also inspired by the rich history of research that takes advantage of human cognition to explore the spatio-temporal multivariate patterns in high dimensional, large datasets [4, 30]. These approaches combine computational techniques and human capacities to discover novel and useful information, in ways that may be difficult to do otherwise. As Duncan and Humphreys [9] have shown, highlighting the similarity between objects can be an effective way to accelerate visual recognition, e.g, quickly rejecting homogeneous non-targets. Indeed, Avraham and Lindenbaum [1] extend the work of Duncan and Humphreys to show that dynamic visual search can be enhanced with the usage of inner-scene similarity. Their intuitive hypothesis was that the more visually similar objects are, the more likely they are to share the same identity. Using these studies as a guide, we propose an approach for presenting streams of network data to a user in a visually aligned form.

Lastly, interactive tools have been recently proposed as ways to help anonymize microdata [33, 8]. Barros et al. [2] even suggest ideas for involving an expert to validate the correctness of methods for sanitizing personally identifiable information (PII) in microdata. Unlike our work, however, these approaches are not used to help data publishers better identify sensitive information in the trace, but only allow them to choose how such data should be anonymized. Furthermore, because of the complexity of network data compared to microdata [6], these tools cannot be directly applied to network traces.

---

[1]Other protocol reverse-engineering works infer message types by analyzing process execution traces (e.g., [20]). In this work, we do not have the luxury of taking advantage of such information because in the vast majority of cases the network traces that publishers are willing to make available have already been collected. Moreover, it is unrealistic to presume that data publishers would collect process execution traces (or even know how to) for all protocols appearing in their traces.
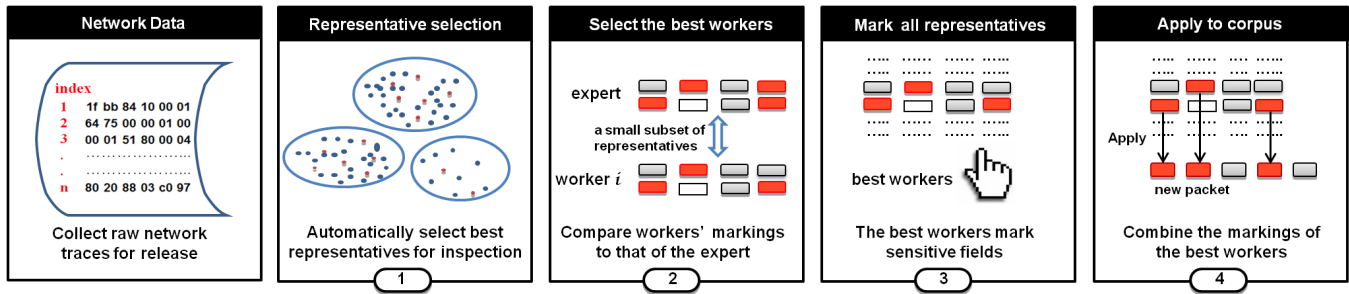
Figure 1. Framework of network trace sanitization by leveraging best worker input

III. OUR APPROACH

At the core of our approach to identifying sensitive information in packet payloads is an algorithm that selects a subset of packet payloads to present to workers (a subset of which is also presented to the expert). At a high level, this algorithm can be viewed as a natural application of clustering and sequence alignment techniques for assisting workers in more readily identifying sensitive information in network data. Our proposed solution requires that we first tokenize the payloads of the packets by labeling them in a more compact representation composed of generic types of fields. Next, we cluster the tokenized packets in order to organize them roughly according to their formats.

Obviously, presenting a large corpus of data—even in an aligned form—to a human being, and expecting her to effectively sift through such data would not be a fruitful task, to say the least. In order to ease the arduous job of finding potentially sensitive information in a large corpus of data, we present to the workers only representatives for each cluster that capture the most mutual information (and that have low redundancy). As the worker marks tokens in the view displayed to her (e.g., by highlighting regions within the visually-aligned representatives), these annotations are recorded. Once the interactive session has completed, her selections made during the process are then used to automatically infer other sensitive tokens in the remainder of the corpus, without further participation from the worker. The overall process is depicted in Figure 1. We discuss the specifics of how we select representatives from the corpus in more detail in Sections III-A–III-E and then discuss how we use the tokens marked sensitive by a worker to identify sensitive fields in the larger corpus in Section III-F.

A. Tokenization

We remind the reader that in order to be protocol agnostic, we deliberately assume no prior knowledge of the protocol format, and so before we can select the best candidates to present to workers, we must first tokenize the data. To do so, we abstract each packet by grouping its bytes into tokens, each of a certain type. The token types we use take advantage of a growing body of work on protocol reverse engineering (e.g., [7, 20]) that suggests suitable token types for text protocols (e.g., HTTP, FTP), binary protocols(e.g., DNS, DHCP) and so-called hybrid protocols (e.g., SMB). Specifically, our three token types are: (1) **Length** fields: consecutive printable characters proceeded by a byte value indicating the number of characters to follow. Both the printable characters and the byte are combined into a single length field. (2) **Text** fields: several consecutive printable characters, the length of which is greater than some threshold.[2] And (3) **Binary** fields: any single byte except those defined as a length field or text field. Each packet is tokenized by scanning the payload from beginning to end. Figure 2 shows an example of the tokenized representation of two packets. For the remainder of the paper, all subsequent operations are on tokenized sequences, and we denote the tokenized sequence of a packet pkt by tokenize(pkt).
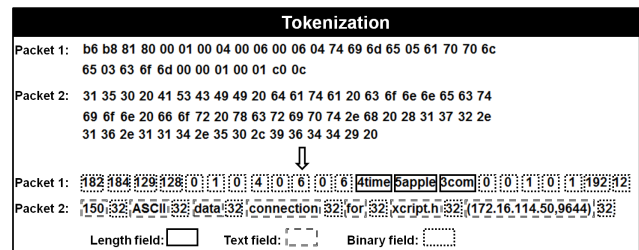

Figure 2. Example tokenization of two packet payloads.

B. Sampling the Data

For improved performance, we next sample packets from the entire dataset and use only the selected packets in subsequent stages. However, sampling packets uniformly at random risks omitting packet formats present in the full dataset, especially if those formats are rare. To overcome this obstacle, we use *stratified sampling* [23, 5] in order to preserve the diversity of the entire dataset. Specifically, we partition the tokenized sequences into homogeneous subgroups; lacking information about packet semantics or formats, we simply partition sequences according to their

---

[2]Similar to [7], we choose a threshold of 3 printable characters.

496

lengths (i.e., the number of tokens in each). We then draw a random sample from each stratum of size proportional to the stratum size, i.e., the number of sequences it contains.

C. Grouping Similar Packet Formats

Given a selection of sequences (i.e., tokenized packets), the next task is to effectively cluster these sequences into groups representing different packet formats. To do so, we first define a distance between sequences, and then provide an algorithm for performing clustering using those distances.

Our chosen distance is based on sequence alignment [14]. That is, to define a distance between two packets, we first find the optimal *token-based sequence alignment* of the packets, which is an alignment of the pair of sequences of tokens corresponding to those packets. Specifically, for an alignment of two token sequences, each aligned pair of tokens is assigned a positive score (an "award") if they match, and a negative score (a "penalty") if they are a mismatch or if one of them is a gap inserted by the alignment process. We use $\mathsf{Penalty_{mis}}$ and $\mathsf{Penalty_{gap}}$ to represent the mismatch and gap penalties, respectively. In assigning awards for any two matching tokens, we not only consider their types, but also consider their values. In particular, if two tokens have the same type and value, we assign a larger matching score $\mathsf{Award_{val}}$. If they are of the same type but different values, we assign a smaller matching score $\mathsf{Award_{typ}}$. The overall alignment score between packets $\mathsf{pkt}, \mathsf{pkt'}$, denoted $\mathsf{Score_{aln}}(\mathsf{pkt}, \mathsf{pkt'})$, is computed using the matching scores, mismatch penalties and gap penalties. Our scoring function is formulated as $\mathsf{Score_{aln}}(\mathsf{pkt}, \mathsf{pkt'}) = N_{val} \times \mathsf{Award_{val}} + N_{typ} \times \mathsf{Award_{typ}} + N_{mis} \times \mathsf{Penalty_{mis}} + N_{gap} \times \mathsf{Penalty_{gap}}$, where N_{val}, N_{typ}, N_{mis} and N_{gap} correspond to the number of tokens with matched values, tokens with matched types, mismatched tokens and inserted gaps, respectively, for an alignment of $\mathsf{tokenize(pkt)}$ and $\mathsf{tokenize(pkt')}$ that maximizes $\mathsf{Score_{aln}}(\mathsf{pkt}, \mathsf{pkt'})$. We then define the distance as

$$\mathsf{dist}(\mathsf{pkt}, \mathsf{pkt'}) = 1 - \frac{\mathsf{Score_{aln}}(\mathsf{pkt}, \mathsf{pkt'})}{\mathsf{Score_{max}}(\mathsf{pkt}, \mathsf{pkt'})} \quad (1)$$

where $\mathsf{Score_{max}}(\mathsf{pkt}, \mathsf{pkt'}) = \max\{\mathsf{Score_{aln}}(\mathsf{pkt}, \mathsf{pkt}), \mathsf{Score_{aln}}(\mathsf{pkt'}, \mathsf{pkt'})\}$ denotes that largest possible value of $\mathsf{Score_{aln}}(\mathsf{pkt}, \mathsf{pkt'})$. The optimal sequence alignment for two token sequences, and hence the distance (1), can be computed efficiently using the well-known Needleman-Wunsch algorithm [22].

Clustering method: Given this distance calculation, we apply iterative K-medoids clustering [15] to partition the packet sequences into different clusters. Unlike K-means, which takes the arithmetic mean of each cluster's points as its centroid, the K-medoids algorithm chooses a member of the cluster as its centroid, namely that which minimizes the average distance to all cluster members. Instead of deciding the number of clusters in advance, we employ the following

algorithm to iteratively grow the number of clusters, which is parametrized by a value r, $0 < r < 1$:

1) Assign all packets into one cluster, and find the medoid of the cluster.
2) For each cluster, find the packet furthest from its medoid as a candidate for a new medoid. Choose the furthest one among these candidates as the medoid of a new cluster.
3) Re-cluster all packets, assigning each packet to the closest existing medoid. After that, re-compute the medoid of each cluster.
4) Repeat steps 2 and 3 until no packet is further than r times the average medoid-to-medoid distance.

Therefore, the number of generated clusters depends on the input parameter r.

D. Alignment of Packets

The clusters output from the procedure described in Section III-C are the clusters from which our algorithm selects representatives (with at least one representative being selected from each cluster). In preparation for performing this selection, we first align all of the packets in each cluster collectively. We now detail how the alignment is done, and discuss the selection of representatives later.

A key challenge in performing multiple sequence alignment in our setting is the fact that we may need to operate over clusters with thousands of packets. For efficiency, we use a progressive method, which generates an alignment by first aligning the most similar sequences and then successively adding less similar sequences to the growing alignment until all packets in the cluster have been incorporated.

With progressive alignment, the quality of the final alignment generally depends on the order with which the sequences are incorporated. To determine this order, we treat the cluster as a graph with vertices being the packets and an edge between each pair of packets weighted by their distance (1). We then use Prim's algorithm [28] to create a minimum spanning tree, and integrate (i.e., align) the vertices together in the order in which they are included in the tree. Since Prim's algorithm adds vertices in increasing order of their distance to their nearest vertex already in the tree, aligning vertices in this order should intuitively delay the insertion of gaps in the overall alignment as long as possible (and hopefully render most gaps unnecessary).

The detailed algorithm is shown as Algorithm 1. The nodes of the minimum spanning tree are denoted mst, which is initialized to the medoid of the cluster (line 2). The use of Prim's algorithm is evident in the **while** loop on lines 4–15, which selects the next closest packet to incorporate (line 5) and adds it (line 15). The construction of a mutual alignment for the packets is done using the align function in line 6. This call aligns the chosen packet and a reference alignment denoted consensusSeq[ ], which is updated by align. In addition, align outputs a boolean array isNewGap[ ] of length

497

equal to the updated consensusSeq[ ], such that isNewGap[j] is true iff consensusSeq[j] is a gap inserted in this call to align. As its last step, align outputs the aligned form of the chosen packet.

Algorithm 1 ClusterAlign (cluster)

1: medoid $\leftarrow \arg\min\limits_{\mathsf{pkt}\in\mathsf{cluster}} \operatorname*{avg}\limits_{\mathsf{pkt}'\in\mathsf{cluster}} \mathsf{dist}(\mathsf{pkt}, \mathsf{pkt}')$

2: mst $\leftarrow$ {medoid}; mstSize $\leftarrow$ |mst|

3: consensusSeq[ ] $\leftarrow$ tokenize(medoid)

4: **while** mstSize $\leq$ |cluster| **do**

5:    pkt* $\leftarrow \arg\min\limits_{\mathsf{pkt}\in\mathsf{cluster}\backslash\mathsf{mst}} \min\limits_{\mathsf{pkt}'\in\mathsf{cluster}} \mathsf{dist}(\mathsf{pkt}, \mathsf{pkt}')$

6:    (consensusSeq[ ], isNewGap[ ], seq[mstSize + 1][ ])
        $\leftarrow$ align(consensusSeq[ ], tokenize(pkt*))

7:    **for** $k = 1 \ldots$ mstSize **do**

8:       $i \leftarrow 1$; temp[ ] $\leftarrow$ [ ]

9:       **for** $j = 1 \ldots$ |isNewGap[ ]| **do**

10:          **if** isNewGap[j] **then**

11:             temp[j] $\leftarrow$ gap

12:          **else**

13:             temp[j] $\leftarrow$ seq[k][i]; $i \leftarrow i + 1$

14:       seq[k][ ] $\leftarrow$ temp[ ]

15:    mst $\leftarrow$ mst $\cup$ {pkt*}; mstSize $\leftarrow$ |mst|

16: **return** seq[1 . . . |cluster|][ ]

The gaps inserted by align, in locations indicated by isNewGap[ ], are then propagated to the aligned forms of the packets already integrated into the tree, i.e., by inserting the gaps into the same positions in those sequences. This is shown in lines 7–14, where the new version of the aligned sequence for the k-th integrated packet is assembled in an array temp[ ] and then copied back into seq[k][ ] (line 14).

The behavior of align differs in an important way from sequence alignment as described in Section III-C. To avoid the introduction of gaps into the aligned sequences seq[k][ ] to the extent possible, we alter align in line 6 to (i) output in consensusSeq[j] the *disjunction* of the tokens from inputs consensusSeq[ ] and tokenize(pkt*) that it aligns to position j (under an optimal alignment as defined in Section III-C); and (ii) assign a matching award (in value or type) to tokens if the token of tokenize(pkt*) matches *any disjunct* of the token of input consensusSeq[ ] (and where a gap in consensusSeq[ ] type-matches nothing). Consequently, after propagating gaps to the other sequences already integrated into the tree (lines 7–14), an invariant of the loop 4–15 is that consensusSeq[j] $= \bigvee_{k=1}^{\mathsf{mstSize}}$ seq[k][j]. Figure 3 shows the result of aligning a cluster of ten packets using this algorithm. Notice that fields with similar type or value have been correctly aligned.

E. Selecting Representatives for Inspection

At this stage in our overall process, the elements of each cluster are aligned in equal-length sequences seq[1 . . . |cluster|][ ]. To select representatives and present them to the worker for inspection, we borrow a technique from Pan et al. [24]. The input required by this technique is a collection of equal-length feature vectors. To generate this input, we simply define feature vectors seqFV[1 . . . |cluster|][ ] so that seqFV[k][i] $= 0$ if seq[k][i] $=$ gap and seqFV[k][i] $= 1$ otherwise. Due to the alignment of seq[1 . . . |cluster|][ ], the tokens of all sequences at position i typically have the same type, and so a binary gap/no gap representation for these feature vectors suffices. The technique of Pan et al. then uses these feature vectors to select representatives that maximize their mutual information and minimize their redundancy.

Figure 3 shows representative sequences generated for a particular cluster. The most closely similar sequences, grouped via the method of [24], are labeled with the same shape (e.g., star). One representative is picked per group of similar sequences, based on maximizing the mutual information and minimizing redundancy. In this way, the approach for selecting representatives improves the efficiency of worker inspection by dramatically decreasing the number of sequences presented for inspection.

Index	Network packets with aligned fields										
1★	74		3www	14was	3com	6aka	3net	4	12	129	147 65
2★	8		3www	14was	3com	6aka	3net	4	12	129	147 65
3★	5		3www	5appl	3com	6aka	3net	4	17	112	152 32
4★	206		3www	5appl	3com	6aka	3net	4	17	112	152 32
5★	17		2us	2rd	6yaho	6aka	3net	4	216	109	118 82
6●	78		5anrl	4gslb	6taco		3net	4	69	7	234 203
7●	232		5ycs-	6yaho		6aka	3net	4	209	73	188 78
8●	3		5ytim	1l	6goog	3com		4	72	14	207 176
9■	64	5farm	6stat	6flic	6yaho	6aka	3net	4	69	147	123 56
10●	67		3www	14kri	2de			4	85	190	2 45
	Selected representatives										
8●	3		5ytim	1l	6goog	3com		4	72	14	207 176
9■	64	5farm	6stat	6flic	6yaho	6aka	3net	4	69	147	123 56
4★	67		3www	14kri	2de			4	85	190	2 45

Figure 3. Example of representative selection; each representative and its most similar packets are denoted by the same shape (e.g., star).

F. Applying Worker Feedback

The representatives selected as described in Section III-E are presented to a group of workers, via an interface such as that described in the appendix. Our technique does not require a specific interface, though it should present the representatives to the worker in a way that promotes the identification of sensitive fields and that provides the worker an ability to mark which fields she believes to be sensitive. In Section IV-B, we evaluate two features of such a user interface that we believe, based on previous findings about user perception [9, 1], can ease the worker's task, namely presenting similar representatives from one cluster at a time and presenting tokenized representatives in their aligned form.

498

The distinct capacities of the workers in identifying sensitive fields make it challenging to apply their markings to the full dataset — recruited workers may have diverse skill levels and training. This motivates our attempt to achieve better accuracy by leveraging inputs from only the most skilled in the worker pool. To identify these most skilled workers, we compare the fields indicated as sensitive by each worker in a small subset of the representatives that worker examined, to ground truth for those representatives as determined by the expert. We make this comparison on the basis of standard measures, namely precision and recall:

$$\text{Precision} = \frac{|\{\text{identified fields}\} \bigcap \{\text{sensitive fields}\}|}{|\{\text{identified fields}\}|}$$

$$\text{Recall} = \frac{|\{\text{identified fields}\} \bigcap \{\text{sensitive fields}\}|}{|\{\text{sensitive fields}\}|}$$

To select the *best* workers, however, it is necessary to reduce these two measures to one, on which worker performance can be ordered. For this, we use the F-score [29] statistic, which computes a weighted average of recall and precision:

$$\mathsf{F}_\alpha = (1 + \alpha^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\alpha^2 \cdot \text{Precision} + \text{Recall}}$$

Essentially, F_α measures the effectiveness of identification with respect to the participant who places α times as much importance to recall as precision [29]. (We will comment on the values of α we employ in Section IV.)

Those workers with the highest F-scores on these representatives are selected for applying their inputs to the entire dataset, in a manner described below. For the remainder of our discussion, we presume that the best two workers are used. Once these best workers are selected, the goal is to utilize their identification of sensitive fields in the representatives they examined to identify sensitive fields in the rest of the dataset. To do so, we process each new packet not directly examined by the workers by first finding the examined representative that is closest to this packet (i.e., for which the distance (1) is the smallest). Pairwise sequence alignment is performed between the new packet and each representative of the cluster that contains this closest representative. We then adopt the most liberal strategy in marking tokens; that is, we mark a token in the new packet as sensitive if it aligns to a field in any of these representatives that either worker marked as sensitive. We do so because in the domain of packet sanitization, higher recall is typically favored over precision.

IV. EVALUATION

In this section, we evaluate the effectiveness of our approach when it is used to identify the sensitive fields contained in packet payloads. For the purpose of our evaluation, we deemed several types of fields as sensitive. These fields were domain names, IP addresses, file names (and directories), user names, passwords, host (server) names, and email addresses. These seven types of fields were used as ground truth for what in the data is "sensitive". We emphasize that these specific fields were chosen only to measure the recall and precision achieved by subjects using our approach. The datasets we used are:

The `UNV-DNS` dataset. This dataset consists of 20,000 network packets recorded at a university campus. The trace contains bidirectional traffic to a DNS server. Of the seven specified sensitive fields, DNS packets contain domain names and IP addresses.

The `KDDCup-FTP` dataset. This dataset was selected from the International Knowledge Discovery and Data Mining Tools Competition.[3] We prepared the dataset by specifically choosing the raw FTP Control packets, which contain 31,020 FTP queries and responses. The specified sensitive fields contained in FTP Control traffic are domain names, IP addresses, file names (directories), user names, passwords, host (server) names, and email addresses.

The `Wireshark-SMB` dataset. This dataset is from the Wireshark[TM] trace repository. It contains 22,807 SMB (server message block) requests and responses. The specified sensitive fields it contains are domain names, file names (directories), user names, passwords, and host (server) names.

The motivation for selecting these three datasets is that they contain packets with diverse types of sensitive fields and complex message formats. For example, the DNS response packets in the `UNV-DNS` dataset are very diverse and can be quite complex (e.g., with IP addresses appearing in many different places in the response packets). The `KDDCup-FTP` dataset has packets with all the sensitive fields specified above, and also has many different types of message formats in FTP reply packets. Similar reasons justify our choice of the `Wireshark-SMB` dataset. For these datasets, we wrote a parser that read the XML packet detail exported by Wireshark to automatically locate all instances of the seven specified fields contained in the payloads. The number and locations of these fields are used only as ground truth.

For the remainder of this paper, we apply standard measures of effectiveness when evaluating our approach, specifically the F-score F_α (see Section III-F) achieved for identifying all sensitive fields either in the entire dataset or simply in the representatives for each cluster. (We will clarify which is used in each case.) Because in the context of packet trace sanitization, recall is often more important than precision — after all, the most common practice when releasing network traces is simply to remove all payload

---

[3]While this dataset has been criticized as being too unrealistic as a basis for evaluating intrusion detection systems (e.g., [21]), we use it here for a completely different purpose, namely as a source of payload-bearing packets that contain some of the sensitive field types listed above.
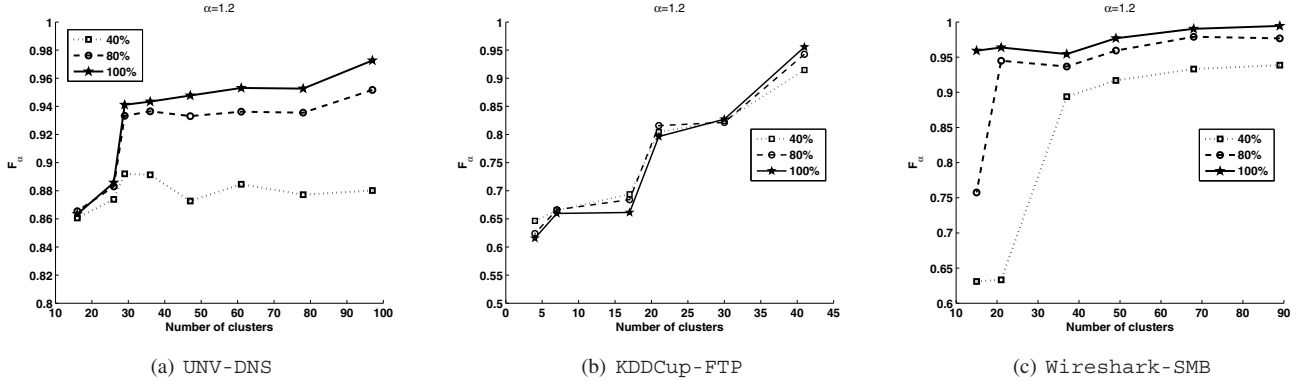
Figure 4. Average F_α when 40%, 80% or 100% of the sensitive tokens in representatives are marked (at random) by a simulated worker.

information, yielding a recall of 1.0 but potentially very low precision — we will generally set $\alpha \geq 1$ in our analysis.

In the analysis that follows, we present results from a user study in which professional administrators were recruited to participate, in order to gain a better understanding of the effectiveness of our approach in enabling them to identify sensitive fields. To estimate parameter settings for this study, we first conducted a simulation-based analysis (with no human interaction) to evaluate the effectiveness of propagating marked tokens in the representatives (i.e., tokens identified as sensitive) to the remainder of the dataset.

A. Exploring the Parameter Space

One advantage of our technique is in generating a limited number of representative packets that capture the characteristics of the packets in the dataset. That said, the manner in which we do so could impact our identification accuracy. Therefore, to choose the most appropriate parameters for our user study (in particular, the number of clusters to use), we performed an analysis in which we simulated a single worker who marked (identified) each instance of a sensitive field independently and with a fixed probability. We reiterate that the sole purpose of the simulation-based analysis was to provide guidance on parameter choice for the field study that followed. With that in mind, we made certain assumptions (about independence) for the simulated user to simplify the task of exploring the parameter space. We then measured the F-score when mapping these random markings of sensitive fields to the full dataset, as described in Section III-F (though using the inputs of only a single simulated worker, not two in combination). In this evaluation, the simulated worker did not mark non-sensitive fields as sensitive, leading to higher precision than might occur in practice (though the precision on the full dataset was nevertheless always less than 1.0). This was done to focus on the effects of recall or, more specifically, F-score with $\alpha \geq 1$.

Below, we evaluate F_α of our technique across different numbers of clusters and for $\alpha = 1.2$. We selected 2000 samples from each original dataset using the sampling

described in Section III-B. We also controlled the number of representative packets by fixing it irrespective of the number of clusters. Specifically, the number of representative packets was chosen to be 140 in the UNV-DNS dataset, 108 in the KDDCup-FTP dataset and 120 in the Wireshark-SMB dataset; these numbers constituted only 0.70%, 0.34% and 0.54% of the total number of packets in each dataset, respectively. These numbers of representatives resulted from using the technique described in Section III at the finest clustering (i.e., yielding the most clusters). This number was then fixed as the target number of representatives in Algorithm 1 when fewer clusters were allowed.

The average F-score for each simulated case and dataset is shown in Figure 4. Each point in this figure is the average of five runs. The standard deviation is 0.006, 0.020 and 0.025 across all datapoints for the UNV-DNS, KDDCup-FTP and Wireshark-SMB datasets, respectively. While our primary use for these simulations is parameter selection (see below), we pause to make three observations from these figures. First, the number of clusters has a large impact on how well the process works, even when identification of sensitive tokens in the representatives is perfect. For example, in Figures 4(b)–4(c), a clustering with too few clusters decays the F-score to roughly only 60% of its optimal. We presume this occurs because with enough clusters, clusters better separate the packets of different message types, yielding higher quality representatives. Second, once an adequate number of clusters is attained, the F-scores are robust to imperfect identification of sensitive tokens in the representatives. Third, when there are sufficiently many clusters, the F-scores that can be realized indicate that the final outcome can be quite successful (e.g., F-scores near 1.0 in all cases).

Based on these tests, we selected 40 clusters for the tests described in the rest of the paper. For our datasets, this provides a good balance between minimizing the number of clusters that workers are asked to inspect and providing the opportunity for good accuracy in identifying sensitive data, once worker markings are applied to the full dataset.

When selecting the number of clusters in practice, we expect that an expert would be helpful here, as well; that is, the expert can be used to judge the quality of the clustering based on the visual similarity of the packets in each cluster. Thereafter, these clusters can be refined iteratively using the approach presented in Section III-C.

B. User Study with Professional Network Administrators

Our techniques for selecting representatives and incorporating user feedback about those representatives to sanitize the full dataset (Section III) are not dependent on any particular method for soliciting that feedback from users. However, we expect that the method of presenting representative packets to users will have a large impact on their abilities to identify and mark sensitive tokens. Two design decisions we made—based on what is known about visual pattern recognition by humans—were to present representative packets as groups based on the clusters to which they belonged (Section III-C) and to present the representatives in their aligned forms (Section III-D).

To determine the impact of these design decisions, and more generally, to evaluate the utility of our overall approach, we conducted an IRB-approved user study with participants recruited from our department's Technical Support Center and the university's Information Technology Service group. All participants were professional administrators with good networking background and familiarity with inspecting packet traces as part of routine network monitoring or diagnostic duties. We targeted professional administrators as they are the natural audience for our tool; after all, they are likely the people who would be tasked with the job of sanitizing network data before its release. This stringent criterion for selecting study participants, however, severely limited the available pool of participants at our university, resulting in our study population of size 15. We note that we obtained consent from these 15 participants only after significant efforts to recruit them. These participants are considered our "workers" in the remaining discussion.

1) Study Design: Recall that our primary goal was to assess the impact of both clustering and alignment in helping workers uncover potentially sensitive fields in packet payloads. To that end, each worker was tasked with identifying the seven specified fields of interest within the packets displayed via a graphical user interface (see appendix). The study itself comprised four trials in which the payloads of packets were presented to the subjects in different ways. Each trial employed a set $\mathcal{R}$ of representative packets. However, the payloads of these representatives were displayed in different forms in the four trials as follows:

Trial I (Clustering+Alignment). The representative packets $\mathcal{R}$ were partitioned according to the clusters from which they were selected. The representative packets were displayed to the worker, one cluster per page, in their aligned forms produced during their selection (Section III-D).

Trial II (Alignment+NoClustering). The representative packets $\mathcal{R}$ were partitioned randomly into blocks. The total number of blocks was the same as the number of clusters in Trial I, but the representatives were evenly distributed across all blocks. The representatives were displayed to the worker, one block per page. Since the packets in each block were randomly selected and not aligned with each other, we aligned these packets using the method described in Section III-D and presented them in that aligned form to the worker.

Trial III (Clustering+NoAlignment). The representative packets $\mathcal{R}$ were partitioned according to the clusters from which they were selected. The representatives were displayed one cluster per page, in their original form (unaligned).

Trial IV (NoClustering+NoAlignment). The representative packets $\mathcal{R}$ were partitioned randomly into blocks. The total number of blocks was the same as the number of clusters in Trial I, but the representative packets were evenly distributed across all the blocks. The representative packets were displayed to the worker, one block per page, with each packet displayed in its original form (unaligned).

For the user study, we chose to use the UNV-DNS and KDDCup-FTP datasets because they contain diverse types of potentially sensitive information. Two groups of representative packets, one for each dataset, were generated by applying the techniques in Section III to the two datasets separately. In each trial for a given subject, only one set of representative packets were used, that is, either $\mathcal{R}(\text{UNV-DNS})$ or $\mathcal{R}(\text{KDDCup-FTP})$.

Each worker undertook all four trials in individual meetings over a period of several weeks, with at least three days between trials. To avoid any learning effects across trials, we incorporated several additional design elements into our study. First, for the trials taken by each worker, we ensured that the datasets used were evenly split across the trials. Second, to prevent displaying the same representatives on the same page in any two trials for a particular user, we ensured that Trials I and III displayed different representative packets. This constraint was also applied to the two non-clustering trials (i.e., Trials II and IV). Third, the order of the trials was randomly chosen (per subject), and we ensured that no two trials that used the same data (e.g., $\mathcal{R}(\text{UNV-DNS})$) were undertaken back-to-back.

Moreover, to limit any factors due to fatigue, the worker was restricted to only one trial per meeting. Meetings were limited to roughly 30 minutes in length, with the exception of the first meeting where the worker was given a brief introduction (with ample time for questions and answers), and time to familiarize herself with the GUI using an artificial dataset. All trials were administered on a dedicated laptop, in a location of the subject's preference. At each

meeting, the worker was asked to mark any occurrence of the specified field types, with timeliness as a secondary goal. Care was taken to ensure that the subject was *not* asked to mark content she *thought* could be sensitive, as doing so would be subjective and would inevitably lead to uncertainty about what should, or should not, be marked.[4] That is, her job was to simply mark any tokens in the displayed sequences that she believed to be a domain name, an IP address, a file (or directory) name, a user name, a password, a host name, or an email address.
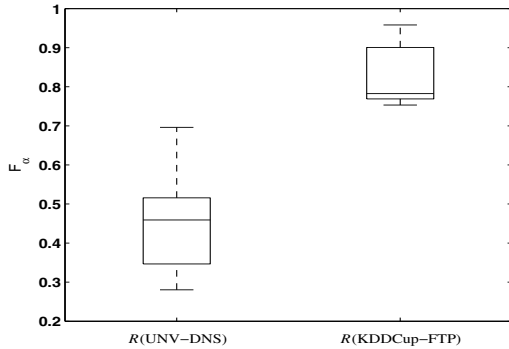
Figure 5. F-scores ($\alpha = 1.2$) per worker in Trial I

2) Results: Figure 5 shows box-and-whisker plots of F_α of Trial I for all workers, with $\alpha = 1.2$. Note that these F-scores were computed using the worker's precision and recall on the representatives only, rather than after applying their markings to the full dataset. Each box represents the first, second, and third quartiles; whiskers cover the remaining points. Figure 5 illustrates that the workers were generally much more successful in identifying sensitive fields in FTP packets, in some cases reaching an F-score exceeding 0.95. The results on $\mathcal{R}(\text{UNV-DNS})$ were not as encouraging; no F-score greater than 0.70 was achieved by any worker. We believe this reflects the substantial challenge represented by DNS payloads, where the variety of locations in which IP addresses can appear makes identification of such fields a real challenge.

This motivates the need to select only the best workers for identifying sensitive data, and then to employ multiple workers; see Section III-F. For the remainder of our study, we chose the two best workers as determined by their F-scores on a randomly selected 20% of the representatives that each marked. This choice simulates a scenario in which the expert marked 20% of the representatives, and then workers were tasked with marking the remaining 80%. The chosen workers were selected based on their F-scores using the expert-marked data as ground truth. In our tests, we

[4]Consider, for example, the username "anonymous" which is not uncommon in FTP; is it sensitive, or not?

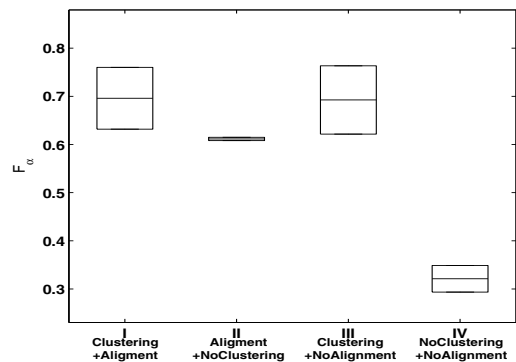possessed ground truth and so did not need to involve an expert directly.

(a) `UNV-DNS`

Best worker	Recall	Precision	F-score
1	0.504	0.991	0.631
2	0.833	0.674	0.760
Combined	0.900	0.930	0.912

(b) `KDDCup-FTP`

Best worker	Recall	Precision	F-score
1	1.000	0.974	0.989
2	0.958	0.974	0.964
Combined	1.000	0.974	0.989

Table I
RESULTS OF APPLYING MARKINGS TO THE FULL DATASET FOR A SINGLE WORKER AND THE COMBINED WORKERS
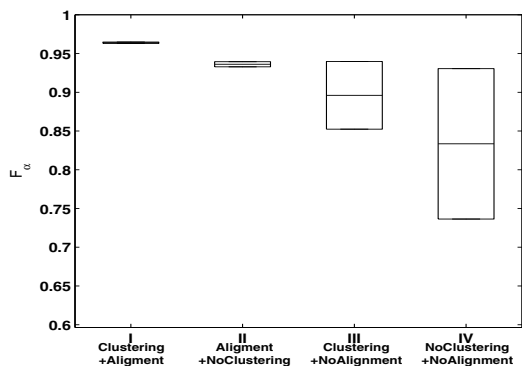
Once the two best workers were selected in this way, their markings were applied to the full dataset as described in Section III-F. Table I provides the F-scores for the full datasets when applying each of these workers' markings individually and then in combination. As these results show, in the case of the `UNV-DNS` dataset (Table I(a)), the recall of the combined case increases up to 0.9 with a small loss of precision when we incorporate the opinions of the two best workers. For the `KDDCup-FTP` dataset, the measurement of the single worker versus the combined result remains very close (nearly 1.0) because each worker already had high recall and precision on that dataset.

3) On Understanding Mixed Effects: While the previous results show that substantial improvement in accuracy can be achieved by picking the best workers and combining their input, it is yet to be shown that the clustering and/or alignment aspects of our approach are indeed factors in boosting the workers' performance. To explore the extent to which these two components influence a worker's performance, we first show (in Figure 6) box-and-whisker plots of F_α across the four trials for the selected best workers. Notice that Trials I–III generally outperform Trial IV. Notice as well that in Figure 6(b), Trial I performs the best, and offers a substantial improvement over Trial IV.

To gain a deeper understanding of the statistical significance of these trends, we apply a mixed-effect regression model to analyze the four trials of the selected best workers shown in Figure 6. A mixed-effect model is an extension of the general linear regression model that allows for correlations within observations [18]. For instance, in our context, this would mean that we consider the performance (say, in terms of efficiency) of a particular worker across different datasets to be correlated, but consider that of different workers to be independent. Conceptually, the mixed effect regression model can be formulated as: $y = $ fixed effects $+$ random effects $+$ error, where the random effects control for variables that are not of particular interest (i.e., the datasets used or different skill levels of our workers), while the fixed effects incorporate the variables that are of interest (i.e.,

502

(a) UNV-DNS

(b) KDDCup-FTP

Figure 6.   F-scores ($\alpha = 1.2$) of the best workers

(a) F_α, $\alpha = 1.2$			(b) Efficiency		
Effect	Estimate	p-value	Effect	Estimate	p-value
β_c	0.217	0.011	β_c	3.311	0.007
β_a	0.197	0.017	β_a	1.092	0.271
β_{ca}	−0.160	0.123	β_{ca}	−2.097	0.147

Table II
RESULTS OF MIXED-MODEL TESTS (SECTION IV-B3)

Table II(a) is smaller (in absolute value) than both β_c and β_a, suggesting that there is an additive effect of these two factors in improving F_α.

Efficiency: Yet another important consideration is how clustering and alignment influence efficiency; that is, do they impede or advance a worker's ability to complete the task at hand? Let "efficiency" be defined as $\frac{|\{\text{identified fields}\}|}{t}$, where t represents the total time to completion in each trial. Table II(b) shows the results for a similar hypothesis test for efficiency. The estimate for the clustering term, 3.311, shows that there is a strong, statistically significant, correlation between efficiency and clustering. Although no statistically significant effect of the alignment or the interaction term is found (p-values of 0.271 for β_a and 0.147 for β_{ca}), our tests still indicate that a user's efficiency benefits from both clustering and alignment together, due to the strong positive influence of the clustering.

V. DISCUSSION

The fact that our methodology yielded recall of 0.9 and even better precision for the UNV-DNS dataset (see Table I) is, we believe, a very encouraging result, particularly considering the complexity of the DNS protocol. We note that it might be tempting to argue that the results herein could be improved by permitting workers more time to mark packets; recall that each trial lasted roughly 30 minutes. We believe, however, that permitting more time would yield diminishing returns. Our perception was that packet inspection was a tiresome process for the workers, an observation supported by the fact that none of our participants chose to continue a trial past 30 minutes, though they were given the opportunity to do so. We thus expect that additional innovations will be required to assist workers in identifying sensitive fields more accurately; it is not clear that more time will help.

We note that our methodology provides a framework only for identifying sensitive fields in packets. Once applied, additional steps must be taken to anonymize those fields using whatever policies the data publisher desires, e.g., consistently mapping sensitive values to others in a one-way fashion, and perhaps in ways that preserve certain structures (such as a prefix-preserving mapping of IP addresses).

Lastly, a natural question is whether the workers' F-scores were "good enough" to provide a basis for sanitizing the full datasets. Answering this question depends ultimately on the data owner's goals for sanitization, which are notoriously difficult to specify or measure [6]. Nevertheless, to provide

clustering, alignment, and the interaction between the two). The model can be formulated as:

$$y = \beta_c \cdot x_c + \beta_a \cdot x_a + \beta_{ca} \cdot x_c \cdot x_a + \epsilon \qquad (2)$$

where x_c, x_a are booleans indicating whether clustering or alignment is used, and y is the performance measure under consideration (i.e., F_α or efficiency). The interaction effect of clustering and alignment, i.e., the effect of clustering after alignment is used, or vice versa, is expressed by the product of x_c and x_a ($x_c \cdot x_a$). The random effects derived from workers and datasets are included in term ϵ.

F-Score: In the analysis that follows, we first test the null hypotheses $\beta_c = \beta_a = \beta_{ca} = 0$, by fitting all observations of F_α of the best workers using (2), the results of which are presented in Table II(a) with α set to 1.2. We consider p-value < 0.05 as the requirement for rejecting a null hypothesis. As Table II(a) shows, F_α is positively related to the clustering, since the hypothesis $\beta_c = 0$ is rejected and the estimate of coefficient β_c is positive (0.217). Similarly, the alignment significantly increases F_α by 0.197. There is little evidence of an interaction effect $x_c \cdot x_a$ since p-value $= 0.123$, i.e., the hypothesis $\beta_{ca} = 0$ stands. Even if the hypothesis had been rejected, the estimate of β_{ca} in

503

a degree of insight, we performed a cursory evaluation of the ability of an adversary to infer the contents of sanitized fields by comparing packets after sanitization to the packets in the original dataset, before sanitization. The details can be found in our accompanying technical report [13] that suggests, in short, that the aforementioned approach works well when applied to trace sanitization.

VI. Conclusion

In this paper we presented a methodology for supporting the daunting task of sanitizing network packet payloads. Our approach is inspired by studies in cognitive science that suggests perceptual grouping of similar patterns can accelerate visual detection. The need for involving humans during the sanitization process is motivated by the complexity of many of today's protocols, and is compounded by the possibility that such protocols may be incompletely documented. However, due to the sheer number of diverse packet formats and the size of a typical network trace, it is unrealistic to expect that an expert will have the time available to accurately sanitize them all. For this reason, our methodology adopts a hierarchical approach in which an expert's input on a small subset of packets is used to select additional workers to examine a larger subset. These selected workers' markings are then used to mark sensitive fields in the (typically much) larger dataset in an automated fashion. At the heart of our methodology is an approach for selecting from the dataset relatively few representative packets for workers to inspect, and presenting these representatives to workers in a way that helps them identify sensitive fields. Our evaluation demonstrated the factors that influence the effectiveness of our methodology, and showed through a user study that our methodology can be effective in supporting sanitization of large network datasets.

References

[1] T. Avraham and M. Lindenbaum. Dynamic visual search using inner-scene similarity: Algorithms and inherent limitations. In *European Conf. on Computer Vision*, 2004.

[2] M. Barros, J. Shiau, C. Shang, K. Gidewall, H. Shi, and J. Forsmann. Web services wind tunnel: On performance testing large-scale stateful web services. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2007.

[3] M. Beddoe. The protocol informatics project. In *Toorcon*, 2004.

[4] J. Chen. Visual inquiry of spatio-temporal multivariate patterns. In *IEEE Sym. on Visual Analytics Science and Technology*, 2006.

[5] W. G. Cochran. *Sampling Techniques*. John Wiley Sons, Inc., New York, NY, 1977.

[6] S. Coull, F. Monrose, M. K. Reiter, and M. Bailey. The challenges of effectively anonymizing network data. In *Cybersecurity Applications and Technology Conference for Homeland Security*, pages 230 – 236, 2009.

[7] W. Cui, J. Kannan, and H. Wang. Discoverer: Automatic protocol reverse engineering from network traces. In *USENIX Security*, pages 199–212, 2007.

[8] C. Dai, G. Ghinita, E. Bertino, J.-W. Byun, and N. Li. TIAMAT: a tool for interactive analysis of microdata anonymization techniques. In *International Conference on Very Large Data Bases (demo track)*, 2009.

[9] J. Duncan and G. Humphreys. Visual search and stimulus similarity. *Psych. Review*, 96:433–458, 1989.

[10] J. Fan, J. Xu, M. H. Ammar, and S. Moon. Prefix-preserving IP address anonymization. *Computer. Networks*, 46(2):253 – 272, 2004.

[11] M. Foukarakis, D. Antoniades, and M. Polychronakis. Deep packet anonymization. In *European Workshop on System Security*, pages 16–21, Mar. 2009.

[12] T. Gamer, C. Mayer, M. Scholler, T. Gamer, and C. Sciences. PktAnon: A Generic Framework for Profile-based Traffic Anonymization. *PIK Praxis der Informationsverarbeitung und Kommunikation*, 2:76–81, 2008.

[13] X. Huang, F. Monrose, and M. K. Reiter. Amplifying limited expert input to sanitize large network traces. Technical Report 11-001, Dept. of Computer Science, University of North Carolina at Chapel Hill, 2011.

[14] N. Jones and P. Pevzner. *An Introduction to Bioinformatics Algorithms*. MIT Press, 2004.

[15] L. Kaufman and P. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley, Chichester, 1990.

[16] D. Koukis, S. Antonatos, D. Antoniades, E. P. Markatos, and P. Trimintzios. A generic anonymization framework for network traffic. In *IEEE International Conference on Communications*, 2006.

[17] G. Kuenning and E. L. Miller. Anonymization techniques for URLs and filenames. TR UCSC-CRL-03-05, University of California at Santa Cruz, Sept. 2003.

[18] N. Laird and J. Ware. Random-effects models for longitudinal data. *Biometrics*, 38:963–974, 1982.

[19] C. Leita, K. Mermoud, and M. Dacier. ScriptGen: An automated script generation tool for Honeyd. In *Computer Security Applications Conf.*, Dec. 2005.

[20] Z. Lin, X. Jiang, D. Xu, and X. Zhang. Automatic protocol format reverse engineering through context-aware monitored execution. In *Network & Distributed System Security Symposium*, Feb. 2008.

[21] J. McHugh. Testing intrusion detection systems: A critique of the 1998 and 1998 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and Systems Security*, 3(4), 2000.

[22] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[23] J. Neyman. On two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *J. Royal Stat. Soc. B.*, 97:558–606, 1934.

[24] F. Pan, W. Wang, A. Tung, and J. Yang. Finding representative set from massive data. In *IEEE International Conference on Data Mining*, 2005.

[25] R. Pang, M. Allman, V. Paxson, and J. Lee. The devil and packet trace anonymization. *SIGCOMM Comput. Commun. Rev.*, 36(1):29–38, 2006.

[26] R. Pang and V. Paxson. A high-level programming environment for packet trace anonymization and transformation. In *ACM SIGCOMM*, pages 339–351, 2003.

[27] J. Ponce, M. Loebl, and L. Kencl. Packet content anonymization by hiding words. In *IEEE Infocom (Demo)*, 2006.

[28] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[29] C. J. V. Rijsbergen. *Information Retrieval(2nd ed.)*. Butterworth, London, UK, 1979.

[30] B. Shneiderman. The eyes have it: A task by data type taxonomy of information visualizations. In *IEEE Symp. on Visual Languages*, pages 336 – 343, 1996.

[31] A. Slagell, K. Lakkaraju, and K. Luo. FLAIM: A multi-level anonymization framework for computer and network logs. In *USENIX Large Installation System Administration Conference*, pages 63–77, 2006.

[32] K. Wang, G. Cretu, and S. J. Stolfo. Anomalous payload-based worm detection and signature generation. In *International Symposium on Recent Advances in Intrusion Detection*, 2005.

[33] X. Xiao, G. Wang, and J. Gehrke. Interactive anonymization of sensitive data. In *SIGMOD Conference (Demo track)*, pages 1051–1054, 2009.

Appendix

In what follows, we briefly describe our prototype graphical user interface (GUI) used in the user study described in Section IV-B. The GUI (shown in Figure 7) is composed of one main panel and two sub-panels. The main panel is used to display the representative packets from one cluster. Each row corresponds to one representative and each column corresponds to one token in the tokenization of the packet, after alignment with the other packets in its cluster.
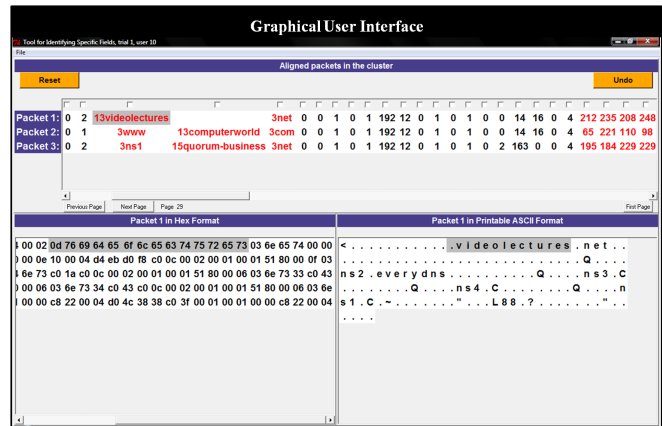

Figure 7. An example GUI for identifying potentially sensitive data

The two sub-panels display the raw bytes of the packet currently in focus (i.e., that the worker last clicked); one sub-panel shows hexadecimal format, and the other shows printable ASCII. The raw bytes for the token on which the worker actually clicked are highlighted in each of these sub-panels. The two sub-panels allow the worker to focus on one particular token and to view it in different formats.

Through the flexible marking facilities provided by the tool, a worker can interactively mark the tokens of the representatives she considers sensitive. Figure 7 shows an example of the sensitive fields identified (in red) by a particular user while using our graphical interface.

505