Backhoe, a packet trace and log browser

Sergey Bratus[1], Axel Hansen, Fabio Pellacini, and Anna Shubina

Dartmouth College, NH 03755, USA

Abstract. We present *Backhoe*, a tool for browsing packet trace or other event logs that makes it easy to spot "statistical novelties" in the traffic, i.e. changes in the character of frequency distributions of feature values and in mutual relationships between pairs of features. Our visualization uses feature entropy and mutual information displays as either the top-level summary of the dataset or alongside the data. Our tool makes it easy to switch between absolute and conditional metrics, and observe their variations at a glance. We successfully used *Backhoe* for analysis of proprietary protocols.

1 Introduction

Analysis of packet traces and event logs derived from packet traces by aggregation or by matching for certain types of events is likely the most frequent task performed by analysts. In particular, *browsing for anomalies*, for points where the traffic changes in character, is a frequent fallback of the analyst or administrator trying to form a hypothesis of what might have happened.

More precisely, users look for groups of records (selected according to some restriction, for example by time intervals) where the distribution of some feature substantially changes (e.g., goes from an almost random uniform distribution to a highly skewed one, or vice versa), or the nature of the mutual relationship between two features substantially changes (e.g., one feature stops being a good predictor for the value of another, or vice versa). Such changes very often have clear security implications; related heuristics are popular in anomaly-detecting intrusion detection systems (IDSes).

Considering that such "novelties" may occur in any one of the usually many features of interest (and in any one of the even more numerous pairwise relationships between these), the users need a tool that lets them examine such relationships at a glance, pivot it easily on selected features, and quickly switch between selections. In this paper, we present *Backhoe*, a proof-of-concept visualization tool for exploring the features' variability and mutual relationships across slices of data, which builds on the *prefuse* toolkit [5].

Although many tools exist to help with the task of anomaly detection on either static (captured) or dynamically flowing packet traces or event streams, most of these tools use an IDS-style model, trained on "normal" input, for judging records as anomalous and highlight the detected anomalous records in the display. We, on the other hand, build our display itself around information-theoretic metrics and let the user make judgements about the overall character of feature interdependence; thus we are not bound by any particular IDS model.

Our approach is based on information-theoretic metrics (explained later in 3) and inspired by the multi-strata visuals of *NameVoyager* ([9]). Unlike in *NameVoyager*, however, the mutual positioning of strata in our display of a packet trace or log is determined from the current dataset, and the relative thickness of some layers is strongly determined by others (in fact, understanding such dependencies is likely what the user is after). Accordingly, after the initial layout, the users can manually re-order and otherwise manipulate the strata, as well as switch the entire view into a mode relative to ("conditioned on") a chosen layer and back.

2 Backhoe in operation

Backhoe's[1] goal is to help the user make sense of a sequence of packets, or, generally speaking, a sequence of log entries, each multi-field record representing an event. Backhoe relies on some other tool to parse these packets or records into rows of a table, the columns of which correspond to protocol fields or other relevant features. For packets we use *tshark*'s PDML output, for other kinds of records – the output of our *Kerf* log browsing tools.[2] From now on, we will refer to either parsed packets or other parsed log entries as "records" or "sequence elements". Also, we use "fields" and "features" interchangeably, because at the point when the record (packet) is parsed and represented as a set of key–value pairs, it does not matter much whether the value is extracted directly from the record or computed.

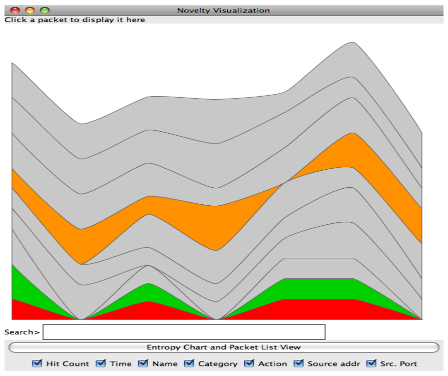
Besides this input sequence, Backhoe needs to know how to partition this record sequence into groups, across which features' distributions and their statistical relationships could be compared and visualized. In most analysis scenarios where sequence elements are time-stamped, it is natural to divide them by time intervals; a group is made up by all records with the timestamp falling within the interval, $G_i = \{r : t(r) \in [t_i, t_{i+1}]\}$. Alternatively, when the user is interested in seeing how much adding new records changes the character of the features' overall distribution, the groups can be defined as comprised of all records to date, $G_i = \{r : t(r) \leq t_i\}$.

However, the splitting of records into groups need not necessarily be done by a timestamp: any features, the values of which could be meaningfully ordered, would do. For example, the user might choose to group packets by size, port numbers, and so forth, or by ranges thereof. In fact, our *Treeview* tool from the above-mentioned Kerf suite attempts to establish the series of features that, when used to recursively group a log dataset, results in groupings that are most convenient for spotting anomalies and classifying "normal" traffic [1].

Strata mode. Regardless of how the grouping is accomplished, Backhoe computes a set of features $\{F_k\}$ for each group and graphs the thickness of stratum k at point i, i.e., over the group i to be $F_k(G_i)$. We call the visualization

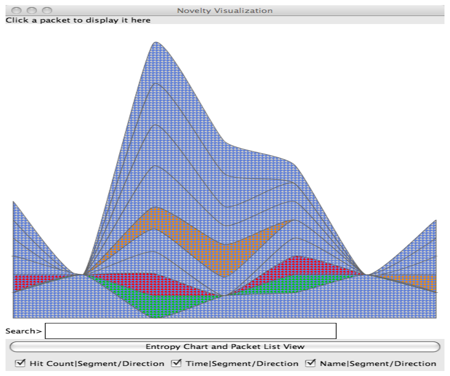
[1] The resulting strata configurations reminded us of cross-sections of exposed rock strata (in the proverbial "mountains of data"), hence the name for the tool that "exposes" these strata.

[2] For a description of our Kerf project, see `http://kerf.cs.dartmouth.edu/`.


Novelty Visualization
Click a packet to display it here
Search>
Entropy Chart and Packet List View

☑ Hit Count ☑ Time ☑ Name ☑ Category ☑ Action ☑ Source addr ☑ Src. Port

Fig. 1. Strata mode, absolute entropies


Novelty Visualization
Click a packet to display it here
Search>
Entropy Chart and Packet List View

☑ Hit Count|Segment/Direction ☑ Time|Segment/Direction ☑ Name|Segment/Direction

Fig. 2. Strata, conditional entropies

mode in which only this data is displayed *strata mode* (Fig. 1). This strata mode helps the user to zero in on the interesting interval where the relative changes in the thickness of the strata attract the user's attention.

Packetfall mode. The strata mode helps the user choose an initial interval for further exploration. Once the interval is chosen, the user can switch into the mode we call the *Packetfall mode*, inspired by the *Rainfall* mode of the Rumint tool by Greg Conti.[3] In this mode, the packet summaries (or record group summaries for non-packet data) are displayed to the right of the strata diagram, which in turn is rotated (Fig. 3). To find and examine packet summaries of interest, the user can take advantage of the fisheye and search functions, explained below.

Switching features sets. The default set F_k of strata features is $H(F_k) = \{H(F_k(G_i))\}_{k,i}$, the entropies of the respective features over the record groups. The rationale for this choice is provided in Section 3. However, the user can switch to other sets with a single keystroke. For example:

'v' switches the thicknesses of the k-th stratum to the count of distinct values of the feature F_k across each group, $\{\#\{\text{distinct}_\text{v}\text{alues}(F_k|_{G_i})\}\}$, and

'C' pivots every layer except the selected one k_0 to be the conditional entropy $H(F_k|F_{k_0}) = \{H(F_k|F_{k_0})|_{G_i}\}_i$.

We call these "conditional" (Fig. 9, left side) and "distinct values" sets. Strata conditioned on another stratum are painted with a distinctive brush.

We also use custom features calculated from the underlying packets, such as the length of the packet's payload when compressed as a byte string by a Lempel–Ziv type algorithm (see 4).

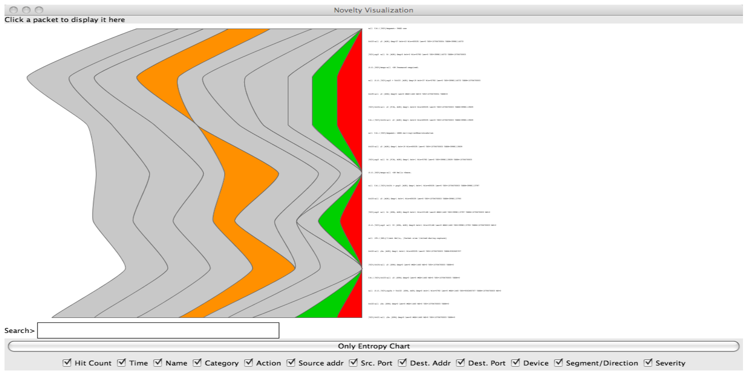
Mixing in conditional strata. When convenient, the user can mix-and-match conditional layers with absolute ones, choosing the conditioning per layer. This is useful when we know that some feature X is strongly dependent on Y , and W is strongly dependent on Z, but Z and Y are virtually independent, and we want to show this manner of mutual dependencies at a glance and also save screen space. The left side of Fig. 7 provides an example.

[3] Available at `http://www.rumint.org/`

With additional Chow–Liu style precomputation (see [2]), the user can choose to see the top m pairwise correlations (more precisely, the m feature pairs with the highest mutual information) drawn from the start as conditional layers, showing, at a glance, where these (on average) strongest dependencies are weakened, as an exception to the general trend.[4]

Strata (re-)ordering. In the simplest display, the strata are sorted according to their average thickness. The user is given the option of choosing the order and the colors of strata via simple keystroke commands (e.g., 'd' lowers the stratum, 'u' raises it, 'f' toggles colors, 'h' hides it altogether). Since reordering operations change the overall layout, their results are animated with fade-in color actions to help the user spot the new position of the just-moved layer.

Fisheye and search. In packetfall mode, the records in the right ("packet") pane are necessarily rendered in tiny fonts or degenerate to pixel lines. Individual records can be viewed by using the fisheye effect (Fig. 4, 6–7). The base (non-fisheye) view allows text query searches on its elements: all matching characters (or pixels, when such squashing is necessary) are highlighted in it, showing the occurrences of the sought substrings (Fig. 5). This highlighting can then be used to zero in on individual matches with fisheye (Fig. 6).



✓ Hit Count ✓ Time ✓ Name ✓ Category ✓ Action ✓ Source addr ✓ Src. Port ✓ Dest. Addr ✓ Dest. Port ✓ Device ✓ Segment/Direction ✓ Severity

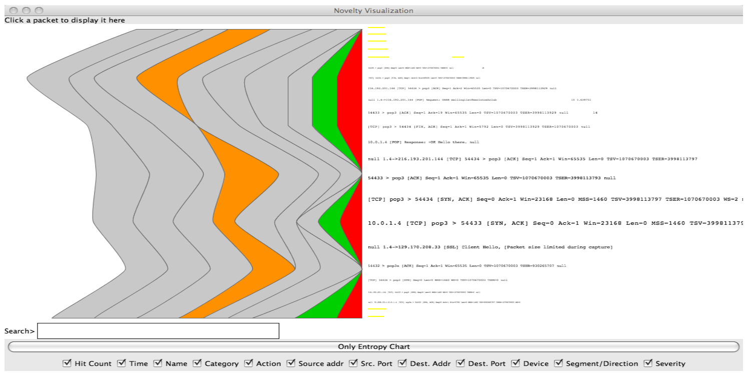
Fig. 3. Packetfall mode, base view

3 Why use information-theoretic metrics?

Entropy of a random variable X that takes distinct values $\{x_i, i = 1, \ldots, n\}$ with probabilities $\{p_i, i = 1, \ldots, n\}$ is defined as $H(X) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$. $H(X)$ is interpreted as the "information content" of X, or a *measure of uncertainty* about X. We are interested in the latter interpretation.

Consider a set of N log records with just one field of interest per record. The values occurring in that field form a discrete probability distribution X, each

[4] For reasons of space, we omit the details of our layout algorithm and refer the reader to our upcoming technical report.


Novelty Visualization
Click a packet to display it here
Search>
Only Entropy Chart
☑ Hit Count ☑ Time ☑ Name ☑ Category ☑ Action ☑ Source addr ☑ Src. Port ☑ Dest. Addr ☑ Dest. Port ☑ Device ☑ Segment/Direction ☑ Severity

```
[illegible]
[illegible]
216.193.201.144 [TCP] 54434 > pop3 [ACK] Seq=1 Ack=2 Win=65535 Len=0 TSV=1070670003 TSER=3998113929 null
null 1.4->216.193.201.144 [POP] Request: USER [illegible]                    15 [illegible]
54433 > pop3 [ACK] Seq=1 Ack=19 Win=65535 Len=0 TSV=1070670003 TSER=3998113929 null          14
[TCP] pop3 > 54434 [FIN, ACK] Seq=1 Ack=1 Win=5792 Len=0 TSV=3998113929 TSER=1070670003 null
10.0.1.4 [POP] Response: +OK Hello there. null
null 1.4->216.193.201.144 [TCP] 54434 > pop3 [ACK] Seq=1 Ack=1 Win=65535 Len=0 TSV=1070670003 TSER=3998113797
54433 > pop3 [ACK] Seq=1 Ack=1 Win=65535 Len=0 TSV=1070670003 TSER=3998113793 null
[TCP] pop3 > 54434 [SYN, ACK] Seq=0 Ack=1 Win=23168 Len=0 MSS=1460 TSV=3998113797 TSER=1070670003 WS=2
10.0.1.4 [TCP] pop3 > 54433 [SYN, ACK] Seq=0 Ack=1 Win=23168 Len=0 MSS=1460 TSV=3998113
null 1.4->129.170.208.33 [SSL] Client Hello, [Packet size limited during capture]
54432 > pop3s [ACK] Seq=1 Ack=1 Win=65535 Len=0 TSV=1070670003 TSER=930265707 null
[TCP] 54434 > pop3 [SYN] Seq=0 Len=0 MSS=1460 WS=0 TSV=1070670003 TSER=0 null
[illegible]
[illegible]
```

Fig. 4. Packetfall mode, fisheye view

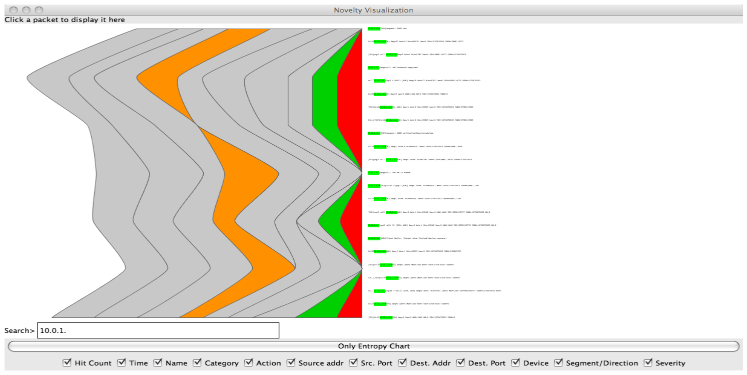

Novelty Visualization
Click a packet to display it here
Search> 10.0.1.
Only Entropy Chart
☑ Hit Count ☑ Time ☑ Name ☑ Category ☑ Action ☑ Source addr ☑ Src. Port ☑ Dest. Addr ☑ Dest. Port ☑ Device ☑ Segment/Direction ☑ Severity

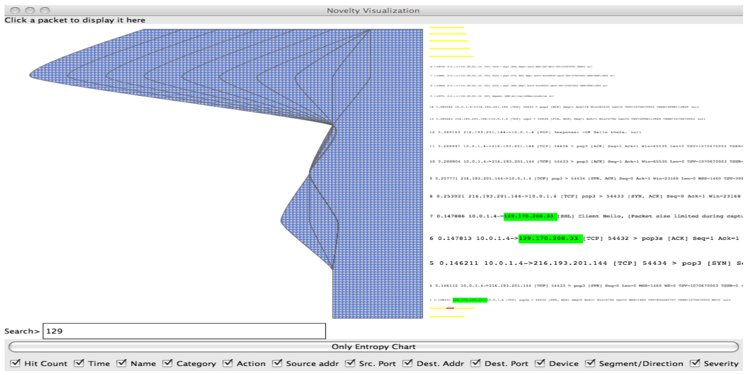
Fig. 5. Packetfall mode, search activated

distinct value x_i having the probability $p_i = n_i/N$, where n_i is the number of times x_i occurs in the set. The less certain we are about which of these values you encounter in a record, the higher is the entropy $H(X)$.

Even for one-value records entropy can provide important security clues, especially when the expected variation of that value in normal operating conditions is small. A raise in entropy would then signal a change of conditions (see, e.g., Lee et al. [8], [7]).

Entropy of a joint distribution of two variables X, Y is defined as $H(X,Y) = \sum_{ij} p_{ij} \log \frac{1}{p_{ij}}$, where p_{ij} is the probability that the distinct values x_i of X and y_j of Y are taken simultaneously (in the simplest case, consider log records with just two fields of interest and the distribution of pairs of values seen in these fields together in a record). For independent X and Y we have $H(X,Y) = H(X) + H(Y)$; in this case knowing the value taken by X tells us nothing about the likelihood of any particular value of Y co-occurring with it. Whenever dependence exists, knowing the value of X does help us predict the value of Y, through our knowledge of the likelihood of their observed co-occurrence.

Statistics that compare the joint entropy $H(X,Y)$ with the single feature entropies $H(X)$ and $H(Y)$ are very useful for describing dependence between

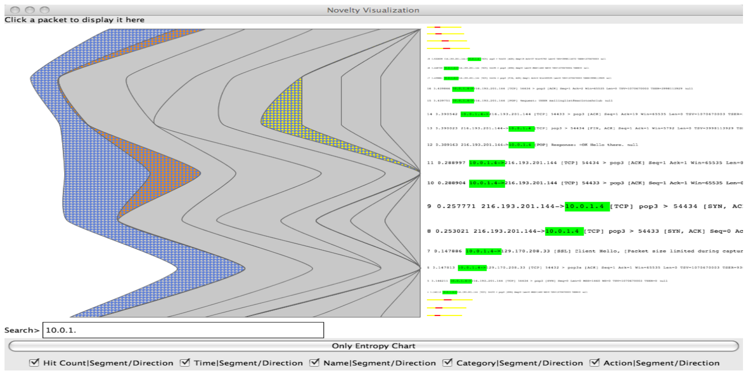


```
[illegible]
[illegible]
[illegible]
[illegible]
14 0.390542 10.0.1.4->216.193.201.144 [TCP] 54433 > pop3 [ACK] Seq=1 Ack=19 Win=65535 Len=0 TSV=1070670003 TSER=3998113929 null
13 0.390023 216.193.201.144->10.0.1.4 [TCP] pop3 > 54434 [FIN, ACK] Seq=1 Ack=1 Win=5792 Len=0 TSV=3998113929 TSER=1070670003 null
12 0.389163 216.193.201.144->10.0.1.4 [POP] Response: +OK Hello there. null
11 0.288997 10.0.1.4->216.193.201.144 [TCP] 54434 > pop3 [ACK] Seq=1 Ack=1 Win=65535 Len=0 TSV=1070670003 TSER=
10 0.288904 10.0.1.4->216.193.201.144 [TCP] 54433 > pop3 [ACK] Seq=1 Ack=1 Win=65535 Len=0 TSV=1070670003 TSER=
9 0.257771 216.193.201.144->10.0.1.4 [TCP] pop3 > 54434 [SYN, ACK] Seq=0 Ack=1 Win=23168 Len=0 MSS=1460 TSV=399
8 0.253021 216.193.201.144->10.0.1.4 [TCP] pop3 > 54433 [SYN, ACK] Seq=0 Ack=1 Win=23168
7 0.147886 10.0.1.4->129.170.208.33 [SSL] Client Hello, [Packet size limited during captu
6 0.147813 10.0.1.4->129.170.208.33 [TCP] 54432 > pop3s [ACK] Seq=1 Ack=1
5 0.146211 10.0.1.4->216.193.201.144 [TCP] 54434 > pop3 [SYN] S
4 0.146112 10.0.1.4->216.193.201.144 [TCP] 54433 > pop3 [SYN] Seq=0 Len=0 MSS=1460 WS=0 TSV=1070670003 TSER=0
3 0.138231 129.170.208.33->10.0.1.4 [TCP] pop3s > 54432 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=930265707 TSER=1070670003 WS=0 null
```

Search> 129

☑ Hit Count ☑ Time ☑ Name ☑ Category ☑ Action ☑ Source addr ☑ Src. Port ☑ Dest. Addr ☑ Dest. Port ☑ Device ☑ Segment/Direction ☑ Severity

Fig. 6. Packetfall mode, all strata conditioned on `destAddr`, search + fisheye



```
[illegible]
[illegible]
[illegible]
16 0.439846 10.0.1.4->216.193.201.144 [TCP] 54434 > pop3 [ACK] Seq=1 Ack=2 Win=65535 Len=0 TSV=1070670003 TSER=3998113929 null
15 0.439751 10.0.1.4->216.193.201.144 [POP] Request: USER mailinglist@macintoshclub null
14 0.390542 10.0.1.4->216.193.201.144 [TCP] 54433 > pop3 [ACK] Seq=1 Ack=19 Win=65535 Len=0 TSV=1070670003 TSER=
13 0.390023 216.193.201.144->10.0.1.4 [TCP] pop3 > 54434 [FIN, ACK] Seq=1 Ack=1 Win=5792 Len=0 TSV=3998113929 TS
12 0.389163 216.193.201.144->10.0.1.4 [POP] Response: +OK Hello there. null
11 0.288997 10.0.1.4->216.193.201.144 [TCP] 54434 > pop3 [ACK] Seq=1 Ack=1 Win=65535 Len=0
10 0.288904 10.0.1.4->216.193.201.144 [TCP] 54433 > pop3 [ACK] Seq=1 Ack=1 Win=65535 Len=0
9 0.257771 216.193.201.144->10.0.1.4 [TCP] pop3 > 54434 [SYN, AC
8 0.253021 216.193.201.144->10.0.1.4 [TCP] pop3 > 54433 [SYN, ACK] Seq=0 Ac
7 0.147886 10.0.1.4->129.170.208.33 [SSL] Client Hello, [Packet size limited during captur
6 0.147813 10.0.1.4->129.170.208.33 [TCP] 54432 > pop3s [ACK] Seq=1 Ack=1 Win=65535 Len=0 TSV=1070670003 TSER=93
5 0.146211 10.0.1.4->216.193.201.144 [TCP] 54434 > pop3 [SYN] Seq=0 Len=0 MSS=1460 WS=0 TSV=1070670003 TSER=0 null
[illegible]
```

Search> 10.0.1.

☑ Hit Count|Segment/Direction ☑ Time|Segment/Direction ☑ Name|Segment/Direction ☑ Category|Segment/Direction ☑ Action|Segment/Direction

Fig. 7. Mixed mode conditional, search + fisheye

variables. In particular, the *conditional entropy* $H(X|Y) = H(X,Y) - H(Y)$ is a measure of uncertainty about the value X when that of Y is already known.

A big increase in $H(X|Y)$ means that Y is no longer a good predictor for X, which has strong security implications for many mutually predictive field pairs of many protocols designed with flexibility in mind but no longer exercising that variability in normal operating environments, such as the use of diverse IP options or fragmentation, overly long variable length headers, and so on.

Unusual use of protocol fields is characteristic of many exploits, but sophisticated attackers take pains to disguise it, as IDSes might be watched for it[5] It is much harder, however, to disguise unusual payloads in such a way that does not introduce unusual statistical effects in any pair of protocol features. It is these effects that our visualization aims to make conspicuous.

Scalability is a significant concern, since computing information-theoretic metrics has high CPU and RAM costs. We are exploring lowering these costs through use of streaming entropy estimation algorithms.

[5] The "Dissembler" by Jon Erikson provides an interesting example among many.

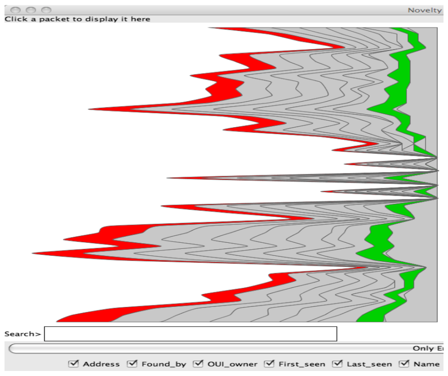


Fig. 8. The BlueSnarfing scenario, entropies

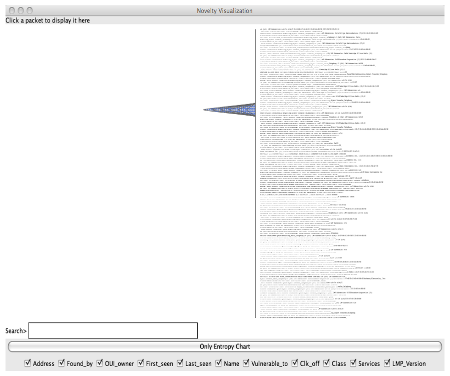


Fig. 9. BlueSnarfing, conditional entropies

4 Evaluation

Backhoe was developed to assist with specific traffic and log analysis tasks. Here we briefly cover two scenarios in which it proved itself.

Proprietary protocol analysis. In the course of vulnerability testing a proprietary product, we needed to discover some facts about a proprietary client–server protocol. Luckily, the protocol's packets were not encrypted, and we could obtain a packet trace. Furthermore, the trace contained many repetitive transactions, and we were interested in spotting new kinds among these repetitions.

We configured one of the features computed on the packets to be *the length of the byte string resulting from Lempel–Ziv compression of the packet*, using the string table accumulated so far from compressing previous packets. This feature served as a rough measure of the packet's "novelty": packets that consisted mostly of previously seen substrings compressed really well, unlike "novel" packets that started some new types of transactions.

The running stratum of the compression-length feature allowed us to quickly locate different kinds of command packets in the *packetfall* mode, whereas highlighting of search byte strings helped us see the degree of repetition of function codes and their approximate offsets in the packet traces at a glance.

The BlueSnarfing story. In another study, a group of Ohio-based security researchers made available to us the logs of proximity scanning of Bluetooth devices (the so-called BlueSnarfing). These logs were collected in the public spaces of security conferences such as Defcon and Notacon. One of the objectives of the analysis was to find out whether any spoofing of Bluetooth device MACs was actually occurring. The "snarfed" log consisted of records that showed MAC, manufacturer ID and capabilities strings, and other device-specific information.

Having loaded the logs into Backhoe (Fig. 8) and taking *conditional* views, we noticed that the other features were perfectly predictable conditioned on the device MAC address, expect for one interval. In that interval, capability and manufacturer strings strata showed non-zero thickness, i.e., non-zero conditional entropy $H(\cdot|\text{MAC})$, making it obvious that the relationship between their values was many-to-one (Fig. 9 shows that conditional view with the MAC stratum hidden away for further clarity). This suggested that within that particular time interval some device responses were inconsistent and probably spoofed. Backhoe enabled us to zero in on this property of the log dataset right away.

5 Related work

Most log visualization tools currently available are concerned with ways to represent frequency distributions of the data according to predefined rules. Despite the complexity of selecting the optimal representation from the sheer graphical point of view (as summarized, for example, in [6]), these tools frequently fall short in their support for anomaly detection. Another broad summary of security visualization tools is given in [3].

Lee et al. in [8] argued for using information-theoretic measures to build models for anomaly detection in datasets, and Lakhina et al. in [7] demonstrated their usefulness for practical traffic monitoring. Although sophisticated detection methods have been proposed since then (e.g., [4]),we are not aware of uses of entropic metrics for visualization proper, although Kaminsky's *Sequitur* visualizations[6] show the potential of related ideas.

References

1. Javed Aslam, Sergey Bratus, and Virgil Pavlu. Semi-supervised data organization for interactive anomaly analysis. In *ICMLA '06: Proceedings of the 5th International Conference on Machine Learning and Applications*, pages 55–62, 2006.
2. C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. In *IEEE Trans. Information Theory*, volume 14, pages 462–467, 1968.
3. Gergory Conti. *Security Data Visualization: Graphical Techniques for Network Analysis.* NO STARCH PRESS, 2007.
4. Yu Gu, Andrew McCallum, and Don Towsley. Detecting anomalies in network traffic using maximum entropy estimation. In *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet measurement*, pages 1–6, 2005.
5. Jeffrey Heer, Stuart K. Card, and James A. Landay. prefuse: a toolkit for interactive information visualization. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, 2005.
6. Daniel A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, 2000.
7. Anukool Lakhina, Mark Crovella, and Christiphe Diot. Characterization of network-wide anomalies in traffic flows. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 201–206, 2004.
8. Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *Proc. of the 2001 IEEE Symposium on Security and Privacy*, pages 130–143, 2001.
9. Martin Wattenberg. Baby names, visualization, and social data analysis. In *INFO-VIS '05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, page 1, 2005.

[6] See `http://www.doxpara.com/slides/dmk_shmoo2007.ppt`, `http://seattle.toorcon.org/2007/talks/dankaminsky.ppt`.