Large-Scale Packet Visualization

Max Goldstein

Abstract

What's in a network capture? We present a visualization of 13-minutes worth of packets captured at DEFCON 2013. Visible are the distribution of packet sizes over time and the geographical locations of the connections. Uploads and downloads are visualized in parallel. The work demonstrates the tension between the specificity of visualization techniques for networks and the generality required by the massive amount of data. With packet captures and other online data collection becoming increasingly common, novel methods are needed to analyze it for trends and anomalies.

1 Introduction

This work presents a visualization of network packet data on a large scale. The visualization consists of two pre-rendered png images, a world map, and an IP grid. The images are heatmaps of uploads and download packet sizes, plotted against time. Using the mouse, the user can interactively select a subset of time and see only that data reflected on the map and grid. The world map displays geoIP information, with brighter dots indicating more data transfered to that location. The 16x16 IP grid displays via brightness the amount of data sent to that block. Brighter squares saw more activity. The visualization has been optimized for a 1080p display.

The work was implemented in `d3.js` and JavaScript, with preprocessing done in Ruby. It is available online.

The implications for privacy, security, and ethics are discussed, as is the role of visualizations, in Section 2. Section 3 briefly discusses related work. Section 4 describes the techniques used to implement the tool, in the hopes that they generalize to other tools. Section 5 concludes with a discussion of what was learned.

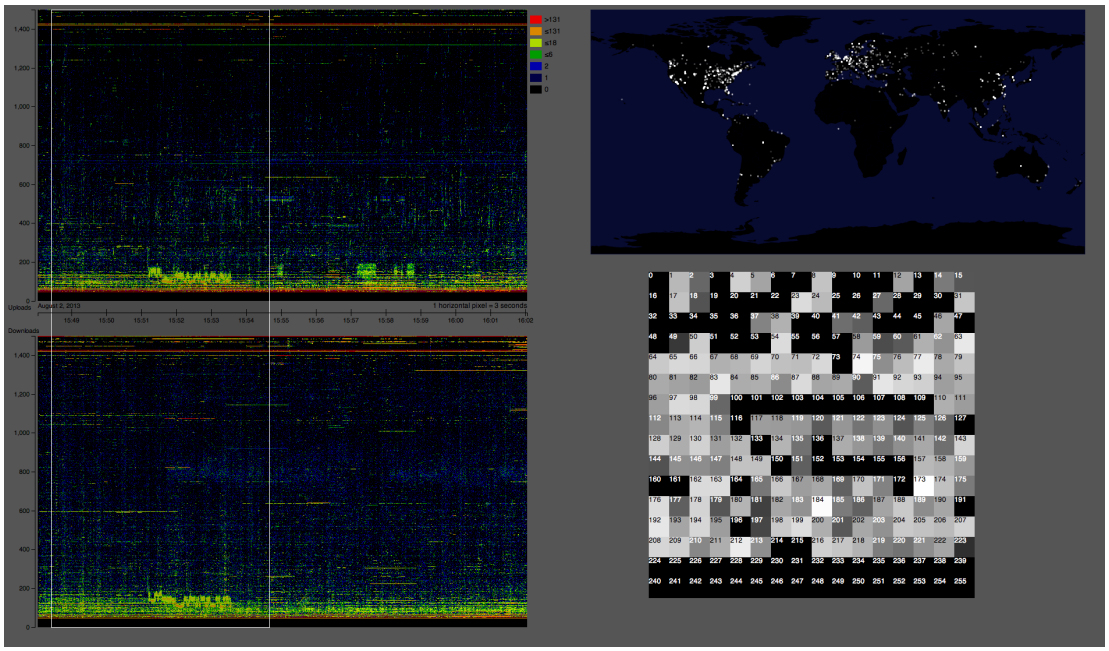

Figure 1: The visualization, with the pre-rendered images on the left, the map on the upper-right, and the IP grid on the lower-right.

2 To the Community

It's no secret that the internet has given us far more data to analyze than was available 20 years ago. Unlike scientific or geopolitical data, this data is uniquely personal. Whether from packet captures, social media websites, or "data vendors" such as Acxiom, this data carries ethical weight. What private information does it contain? How could someone nerfarious use this data to blackmail the people it concerns? Was it collected illicitly, or in a way whose legal precedent is outdated, disputed, or even nonexistant?

At one point are features of the data best left unfound? An encrypted channel could carry the words of dissidents of an oppressive regime, financial transactions legitimate or fraudulent, criminal activity, or innocuous chatter. This is true on any online communications forum, and large social media websites have access to a tremendous amount of unencrypted data, in the semantic context of friend graphs or search histories.

The sorts of tools that allow facts to be gleaned from data will become more powerful, but ultimately the best telescope is worthless if pointed at the ground. Increasingly important are tools that tell us where to look. Here the technical restrictions dovetail with ethical restraint. By *identifying* indicators of unusual

behaviour, we can both focus our limited tools and time on *inspecting* the parts of the data that are likely suspect, while maintaining privacy of all other data.

The primary identification tools are machine learning algorithms. It is far cheaper, scalable, and reliable to allow a computer to guess that a credit card charge in the Cayman Islands is fraudulent than to ask a person to study dots on a map. However, no such algorithm will ever be perfect, and visualizations can play a secondary role in identifying patterns that slip through the gaps. Machine learning algorithms are ill-equipped to handle truly novel data.

However, once the alarm has been sounded, inspecting a particular piece of traffic will fall almost exclusively to visualizations. A single stream can be visualized as to encode most information in some way (visualizing large amounts of data involve discarding most of it). By contrast, traditional command-line or GUI network tools keep the data almost or entirely textual, so not all of it is visible at the same time, and numeric patterns are inaccessible. Armed with a powerful visualization, a human can verify that the activity is suspicious, prevent, undo, or mitigate the damage done, and refine the dataset of the identification algorithms (e.g. by marking the perpetrator for closer following). In this way, the search for anomalies in large datasets is a self-reinforcing cycle of working at multiple zoom levels.

The work presented is an attempt at large-scale analysis, identification. A scalpel-like visualization intended for inspection is an area for future work — and past work.

3 Prior Work

Existing security visualizations primarily focus on individual packets or other binary data files, such as detecting anomalously long ASCII strings in mp3 files. Greg Conti[1] created rumint,[2] a package that allows for these sorts of visualizations to be created easily. Rumint's binary rainfall allows the user to see (literally) the similarities and differences among packets. Conti also deploys standard multivariate plots to chart arbitrary numeric data.

Many other packet visualization software exists, such as PacketViz[3] and TNV [4], but they all analyze very small amounts of data, either a single file or at most a single connection.

The visualization to be created for this project, though less general and comprehensive than rumint, improves upon Conti's work in the following ways:

[1] *Security Data Visualization*, 2012
[2] rumint.org
[3] `http://packetviz.sourceforge.net/`
[4] `http://tnv.sourceforge.net/`

1. By tolerating orders of magnitude more packets than rumint by the use of binning in 3-second intervals and extensive preprocessing.
2. By focusing on heterogenous packet data, searching for statistical trends and anomalies, rather than bit changes across mere dozens of packets from a single stream.[5]
3. By using visualizations tailored to the data, e.g. a world map for GeoIP information, rather than a scale of longitude values.
4. By allowing the interactive selection and query of packets.
5. By being more portable than rumint, achieved by running in the browser rather than on Microsoft Windows.

4 Applications

4.1 Compression & Preprocessing

The scale of data compression is more than one-thousandfold: from 12.5GB of raw pcap data down to 7.1MB of json, plus two mid-size png images. The following discussion of techniques used can inform general techniques for working with large datasets. The guiding principle was to do as much work in advance so that the visualization could be interactive.

The Ruby library Packetfu was used to preprocess the pcap file metadata. Although the packet data was discarded, PacketFu needs to load all of a pcap file in order to operate on it. Therefore, this was the stage at which large amounts of processing power were brought to bear. The script includes a logging mechanism and a way to run on a given range of pcap files, so that execution may be done in parallel.

The result was a 389MB csv file with one entry per packet, consisting of UNIX timestamp, capture number, IP, "up" or "dn", and the packet size in bytes. A small except follows; note the tight spacing of timestamps indicative of the frequency of packet transmission.

```
1375472882.542238,0,141.0.11.253,dn,1426
1375472882.542364,0,141.0.11.253,dn,1426
1375472882.542508,0,173.194.26.87,dn,1500
1375472882.542609,0,173.194.26.87,dn,1500
1375472882.542616,0,87.98.229.84,up,48
1375472882.542688,0,184.85.98.217,dn,52
1375472882.543438,0,63.236.103.241,up,46
```

[5] As previously stated, one focus is superior than the other, merely different.

The second phase was responsible for "binning" the data and generating pre-rendered png images. These were combined into one ruby script as they share some parameters and assumptions. "Binning" is the process of discretizing a continuous interval into a finite number of pieces, usually of even size. Packet sizes were binned 3-to-1: The largest packets were 1500 bytes (a restriction of the Ethernet protocol), so 500 bins were made, so that packets of size 1498, 1499, and 1500. Time was binned three seconds to a bin, as it created a reasonably-sized image for the size of the data.

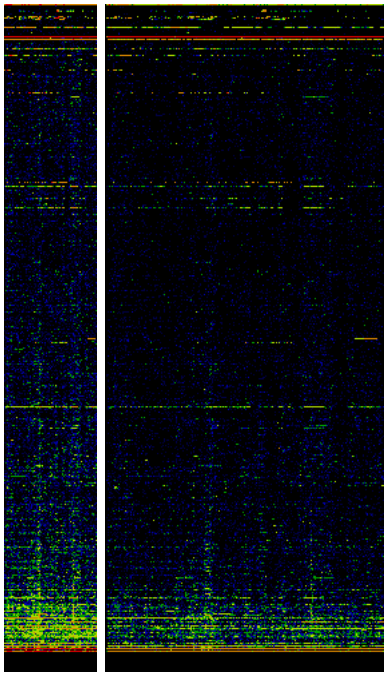

Figure 2: The same data binned in three-second (left) and one-second bins.

The binning script also performed GeoIP lookups on the data. Done in preprocessing, they would be done per-packet and not per-host; done in the browser they would take too much time.

The script must determine the bin sizes of the heatmap - what is the cutoff between a red pixel and an organge one? As the number packets in a particular time bin and size bin exhibited a "long tail" distribution, cutoffs were made at 0, 1, and 2 packets, and at the 50th, 80th and 95th percentiles. The percentiles were for both uploads and downloads, so both heatmaps use the same scale. For generality, the scale and color scheme is included in the json data along with all the binned values - this is why the binning and png functionalities were combined. The pre-rendered png images actually start as svg images, as the format is XML and easily written; it then invokes (from the command line)

a svg-to-png conversion program. The bin script has three outputs: the two images, and a json file of the binned data, along with the parameters of the binning, and the timestamp of the first packet.

4.2 Bespoke visualizations

Rumint used very general visualizations capable of plotting any numeric data. Specifically, parallel coordinate plots are used, which introduce artifacts as data must be scaled and ordered somewhat arbitrarily. This approach falls short where a customized display can help target specific questions.

The most obvious bespoke visualization is using a world map to plot geoIP information. Numeric latitudes and longitudes have little intuition attached. More fundamentally, a general plot will separate these two related dimensions. Looking at only latitude, for example, Europe and the United States blur together.

The IP grid is another example, but it becomes far more tailored for IPv4 addresses specifically when the layout of the grid is altered from the standard "reading order". Given that IP subnets are based on bitmasks, there is an inherent binary halving/doubling nature to how they are allocated. The solution was first proposed, it seems, by *xkcd* author Randall Munroe: a space-filling Hilbert curve to place IP blocks in clusters.[6] This way, subnets form squares and larger allocations are grouped.[7] Time did not permit the implementation of this layout.

The data scientist Edward Tufte proposed a metric for the information density of an infographic: numbers per inch.[8] There were 9593618 packets captured, and each contributes two numbers: size of payload and timestamp. Each image is 841 x 500 pixels, for a total of 841000 pixels. This gives us 2*9593618/841000 = 22.8 numbers per pixel. A 13" MacBookPro with Retina display has 227 pixels per inch. This means the images have on average 5179 numbers per inch.

5 Summary

Given the rate of data collection online, new tools are desperately needed to help us understand the new data. As much as possible, we would like to detect fraudulent, illegal, and destructive behaviour without inconveniencing and violating the privacy of law-abiding citizens. While machine learning algorithms are likely to bear most of this load, visualizations are also valuable.

[6] Randall Munroe, `http://xkcd.com/195/`, 2006
[7] For a command line version, see `http://maps.measurement-factory.com/software/`
[8] Edward Tufte, *Envisioning Information*, 1990.

Much of the data exhibits a very specific distribution. Firstly, there are several `0` values. Secondly, for discrete measures like packets, there are several small integer values. However, most of the data follows a "long tail" where the maximum value is quite large compared to the, for example 80th percentile datum. Therefore scaling by that maximum compresses the scale for the bulk of the data. The solution used to obtain a normalized value was to divide the log of the datum by the log of the max. This worked well in that it gave a large amount of visual variety of the results. A statistical analysis could show whether the data actually follow a log-normal distribution.

However, it also means that the visual mapping is not what the reader might expect. Certainly less/more comparisons will still be accurate, but numeric judgements will be skewed. Scales can partially counteract this effect, along with proper labeling.

Despite the high density of numbers in the images, the data fail to startle: there are some peculiarities of low-size packets lasting several seconds, but otherwise the profile shows what one might expect if familiar with the domain. The geoIP map is similar to any map of an indicator of technological progress: heavy in North America and Europe, sprinklings in East Asia, and not much of Africa. The question then becomes: is the data inherently unexciting, at least at this zoomed-out level? Or could a different approach yield something more useful? If the latter, we may see new visualizations innovations as tools capable of gleaning information emerge, much as happened with data mining tools. If the former, if the data is fundamentally boring, then those collecting it will need to grapple with mountains of useless bits.