RUMINT Imagery Analysis

23 May 2006

Greg Conti

Interpreting security visualization images is a bit like interpreting Rorschach inkblots. Each viewer may see different things or nothing at all. This paper will help you understand the primary techniques found in RUMINT and how to interpret them. Rather than jump off into an esoteric case study, we'll look at images from the ping (Packet INternet Groper) command as a straightforward example. Despite its simplicity ping generates very distinctive visual fingerprints.

ping is a network utility that tests whether hosts and routers are reachable via an IP network. It works by first sending out an Internet Control Message Protocol (ICMP) echo request (type 0) and waits for the destination to respond with an ICMP echo response (type 8). If the target fails respond within a given period of time ping reports that the attempt timed out. ICMP echo requests typically use an optional data field that must be echoed back by the target's echo response.

In Windows XP the command:

```
ping www.google.com
```

will generate output similar to the following:

```
Pinging www.l.google.com [66.102.7.147] with 32 bytes of data:

Reply from 66.102.7.147: bytes=32 time=63ms TTL=235
Reply from 66.102.7.147: bytes=32 time=52ms TTL=235
Reply from 66.102.7.147: bytes=32 time=53ms TTL=235
Reply from 72.14.207.104: bytes=32 time=53ms TTL=235

Ping statistics for 66.102.7.147:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

```
Approximate round trip times in milli-seconds:
    Minimum = 52ms, Maximum = 63ms, Average = 55ms
```

By capturing the ICMP packets and displaying the results in ASCII they should look like the following. Note that bytes not in the printable ASCII range are replaced with the period character.

```
.......0......E..<......$....dBf.....X..N.abcdefghijklmnopqrstuvwabcdefghi
.......0......E..<......$....dBf.....X..O.abcdefghijklmnopqrstuvwabcdefghi.
.......0......E..<......$....dBf.....X..P.abcdefghijklmnopqrstuvwabcdefghi.
.0............E..<........Bf.....d...Y..P.abcdefghijklmnopqrstuvwabcdefghi.
.......0......E..<......$....dBf.....X..Q.abcdefghijklmnopqrstuvwabcdefghi.
.0............E..<........Bf.....d...Y..Q.abcdefghijklmnopqrstuvwabcdefghi.
.......0......E..<......$....dBf.....X..R.abcdefghijklmnopqrstuvwabcdefghi.
.......0......E..<......$....dBf.....X..S.abcdefghijklmnopqrstuvwabcdefghi.
```

From these captured packets we see that Windows XP uses portions of the lowercase alphabet for the optional data field. Most existing tools provide textual ASCII and hexadecimal views of packets and our example system provides similar functionality, as seen in the following image.

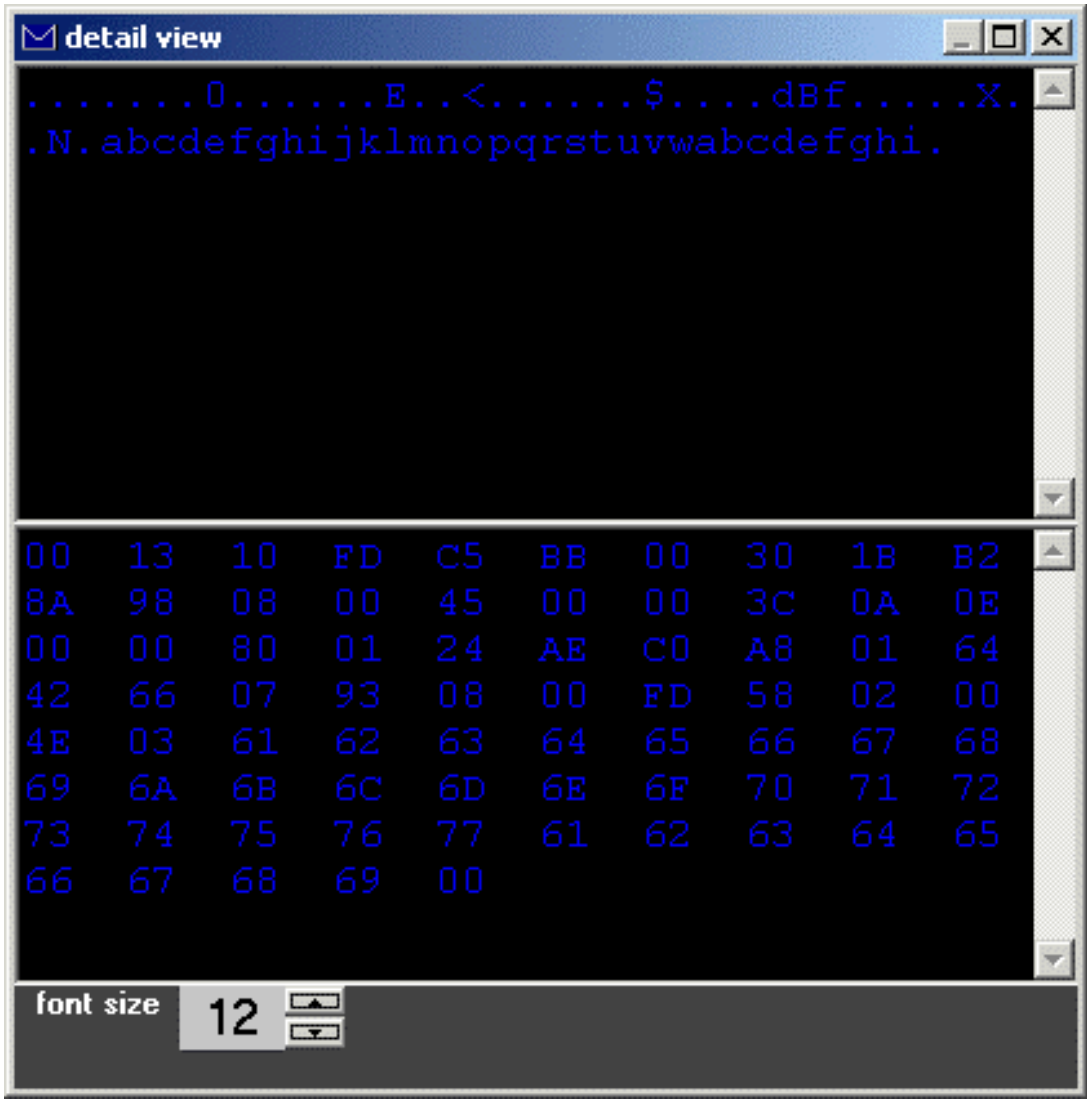

Figure 1: Traditional view of an ICMP packet in ASCII and hexadecimal.

Figure 2: ASCII representation of ICMP network traffic generated by the ping command. Windows XP's use of the lowercase alphabet for the payload can be clearly seen.

While the canonical hexadecimal and ASCII view is useful for very detailed analysis, it lacks the ability to compare larger numbers of packets. To this end, RUMINT provides a text rainfall view that allows 24 to 40 packets to be displayed at the same time, as seen in Figure 2. From this image we can see that many packets are behaving in the same manner. While this is an improvement, the text rainfall view is still limited to approximately 40 packets at one time.

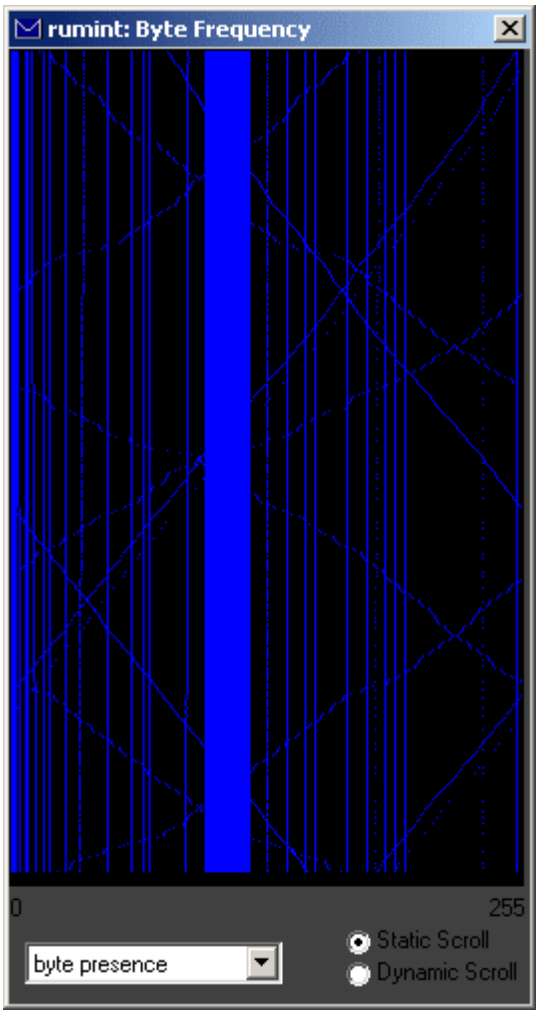

Figure 3: A byte presence representation of several hundred ICMP packets generated by ping. One packet per line. The solid vertical bar shows that each packet contained the entire lower case alphabet (ASCII 97-122). The diagonal lines represent consistently changing values between packets such as the IP identification header field.

The byte presence view compresses this information by plotting one packet per horizontal line. Each horizontal line is 256 pixels long and pixels along this line are illuminated based on the presence of bytes within the packet. For example if packet one contains a byte of value 97 (an ASCII 'a') then the pixel at position 97,1 will be illuminated. If packet 37 contains a byte of value 122 (an ASCII 'z') then the pixel at position 122,37 will be illuminated. When this technique is used on a set of several hundred ICMP packets the result is dramatic. Figure 3 allows us to verify that every packet's payload was likely constructed in the same manner as indicated by the thick

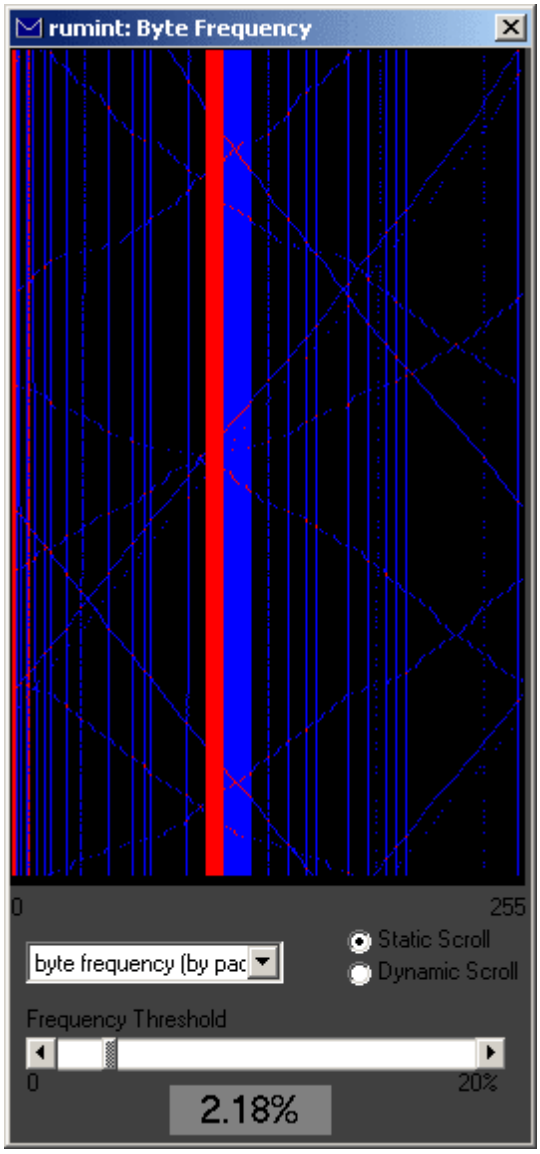

Figure 4: A byte frequency representation of the same packets. More frequently occurring bytes are encoded in red and less frequently occurring bytes are encoded in blue. The payload's repetition of the letters a-i (ASCII 97-105) is clearly visible as the red vertical bar.

vertical bar in the printable ASCII range (97-122). Vertical lines show constant values between packets, in many instances these are header fields such as the IP Version (typically 4 for IPv4). Diagonal lines indicate values that change at a constant rate between packets. The slope of the line indicates the direction and rate of change. The primary advantage of this view is the ability to compare up to about 1,000 packets at one time as we are only limited by the horizontal resolution of the monitor. This technique is largely independent of packet length as larger packets, even up to the

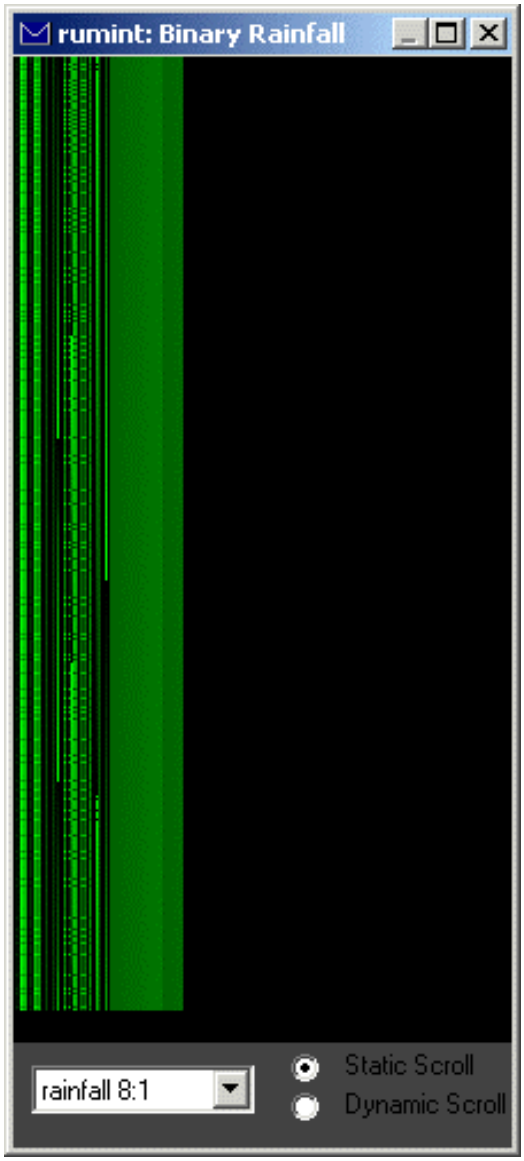

Figure 5: A binary rainfall visualization of the same ping packets. One packet per line and one byte per pixel. The packet header regions appear on the left half of the image as intermittent pixels. The packet payload appears as the gradient on the right side of the image. Note how the gradient restarts as the alphabet is repeated.

traditional Ethernet limit of 1518 bytes, can contain only 256 possible bytes. The primary disadvantages are the loss of human readable packet contents and the inability to see which byte values occur more frequently. We can address this second shortcoming by color-coding the view by byte frequency instead of simply the presence or absence of bytes.

Figure 4 depicts the same ICMP packets but includes a threshold adjustment slider. Any bytes (relative to a given packet) are color coded red (hot) for higher frequency and those that fall below the threshold are color coded blue (cold) for their lower frequency. By adjusting the slider it becomes immediately obvious that the first portion of the lower case alphabet occurs more frequently. You can confirm this by taking another look at the textual ICMP packets in Figure 2. Both the byte presence and byte frequency views raise a question. What exactly is causing the diagonal lines that can be seen in Figures 3 and 4? To answer this question we will explore another view of the same packets that provides greater insight into the actual locations (offsets) of bytes within packets: the binary rainfall.

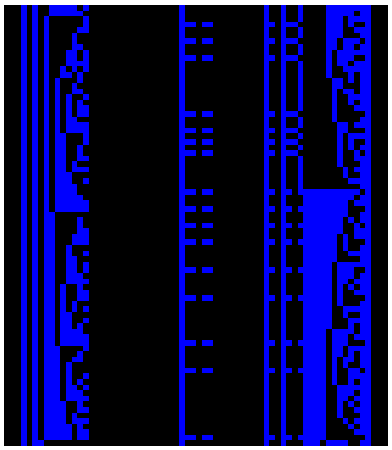

Figure 6: Zoom View of Binary Rainfall. Constant bit values appear as vertical lines. Consistently changing values appear roughly as triangles. A triangle with its tip facing upwards indicates a value that is increasing. A triangle with its tip facing downwards is a decreasing value.

The binary rainfall directly maps values in packets to pixels on the screen. Again there is one packet per horizontal line, but each pixel is illuminated based on the value of the byte at that offset in the packet. For example if packet number 50 has a value of 42 at the 100th position in the packet, the pixel at position 42, 50 will be illuminated. The actual value of the byte (42) determines the color of the pixel. Bytes have 256 possible values so the binary rainfall view uses a 256 level

A B C D

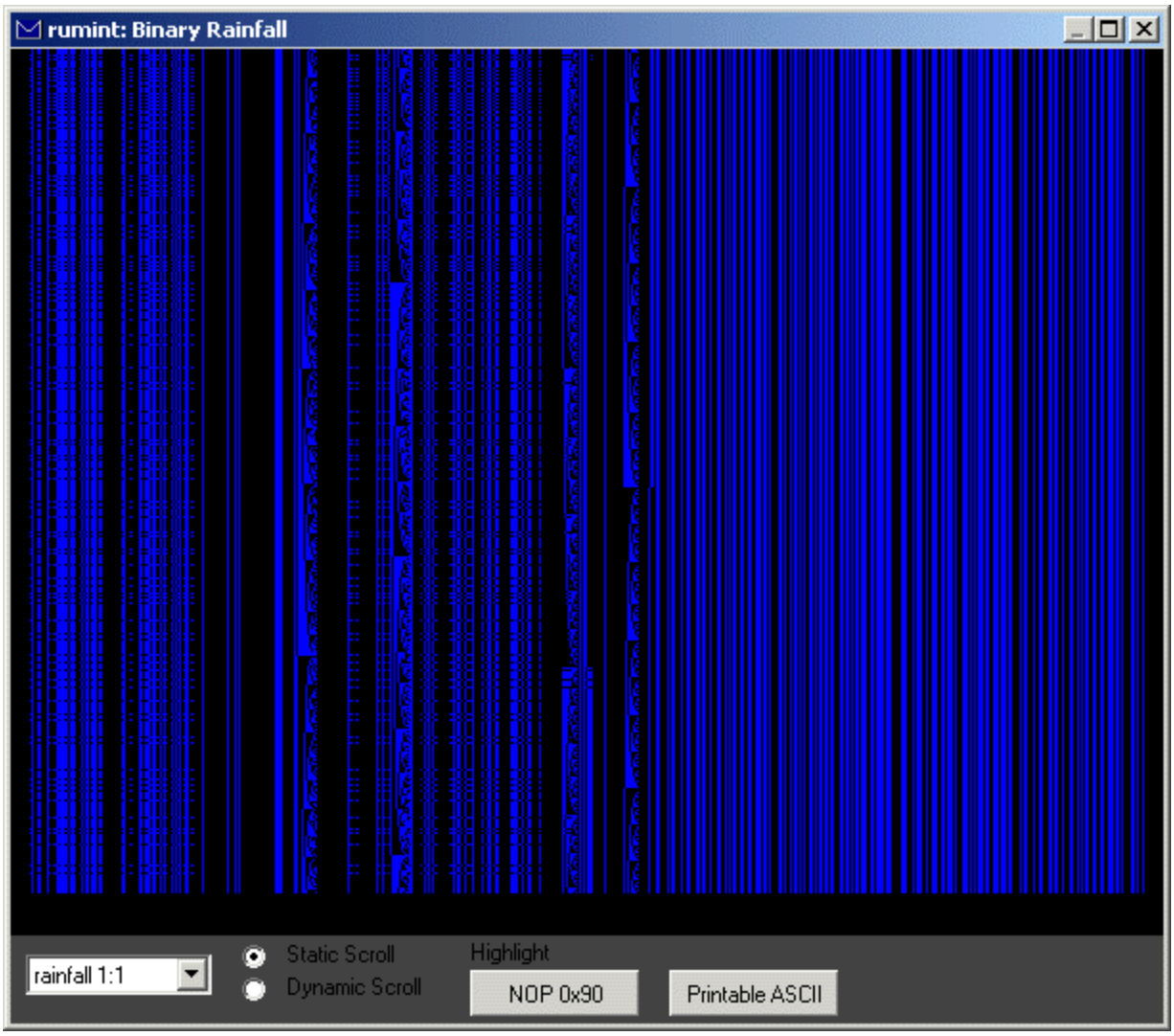

Figure 7: A binary rainfall visualization with one packet per line and one bit per pixel. The packet header regions appear on the left half of the image as intermittent pixels. The packet payload appears as vertical bands on the right side of the image. Consistently changing values occur at the columns marked A (IP Identification Field), B (IP Header Checksum), C (ICMP Checksum), D (ICMP Sequence Number).

color scale, to depict packet contents. By using this technique the actual structure of the packet can be seen. If you examine Figure 5 you will see the header fields at the left of the image, appearing as intermittent pixels. The alphabetic payload can be seen on the right. The smooth gradient is caused by the one up progression of alphabetic characters, and hence byte values. Note how the gradient starts over as the alphabetic payload wraps around and begins again.

This visualization again allows you to compare about 1,000 packets at one time and view almost the entire packets. You would need a monitor with greater than 1518 bit horizontal resolution to view the largest packets. Subtle differences between packets are harder to detect and to identify the cause of the diagonal lines we use a more detailed view.

To create a more detailed view we use the identical technique, but instead map each *bit* in the packet to a monochrome pixel. This truncates packets such that we see only the first 1000 bits of each packet (~125 bytes), but allows us to see packet headers in far greater detail. If you take a look at Figure 6 you will see that both constant values and consistently changing values are easy to detect. Constant bit values appear as vertical bands and consistently changing values look roughly like triangles. Figure 7 shows several hundred ICMP packets. By using the tool's mouseover to determine the position of the columns containing triangular regions we see that they occur at positions 20, 26, 37 and 41. By using the protocol analyzer Ethereal we confirm that they are the IP Identification, IP Header Checksum, ICMP Checksum and ICMP Sequence Number fields respectively.

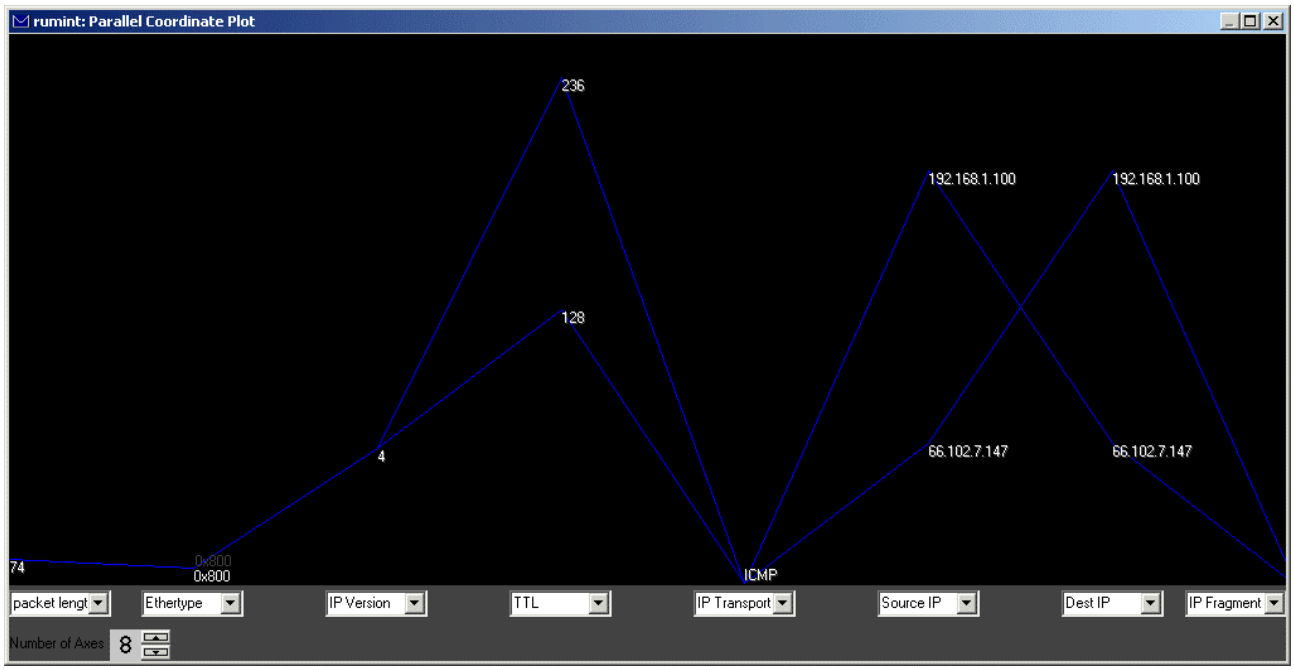

Figure 8: The parallel coordinate plot allows from 2 to about 30 variables to be compared at one time. This image shows that all 500 packets in the ping dataset had identical lengths, ethertypes and transport protocol. We also see that only two IP addresses and two TTL values were present.

While a text-based protocol analyzer is useful for analysis of individual packets, visualization excels at providing big picture context. I like to use a parallel coordinate plot when I'm first given a new packet capture file to analyze. The parallel coordinate plot allows from 2 to about 30 variables to be compared at one time. The concept is straightforward. Consider Figure 8. You will notice that there are eight vertical axes: packet length, ethertype, IP version, TTL, IP transport protocol, source IP address, destination IP address and IP fragmentation fields. These are taken from the header field of each packet and are plotted on each vertical axis. For example, if a packet had a length of 74, it would be plotted on the first axis (packet length) . The vertical position is scaled to fit on the screen. In this example 74 is a small fraction of the maximum legal packet size and thus appears near the bottom of the screen. There are 500 packets represented in the figure. At a glance we can tell that they all share the same packet length (74 bytes), ethertype (0x800) and use IP version (IPv4) with no fragmentation. All 500 packets used only two IP addresses and two TTL values (236 and 128). The parallel coordinate plot is useful for rapidly characterizing large sets of

packets and determining which fields are constant, near-constant, random and sequential, but it does suffer from one significant shortcoming. One or many packets may take the same path and it is difficult to tell the difference. While there are approaches that can handle occlusion reasonably well, such as fading older packets or changing brightness to indicate increased activity, our current system lacks those capabilities. As an alternative we will use the combined visualization technique [1].

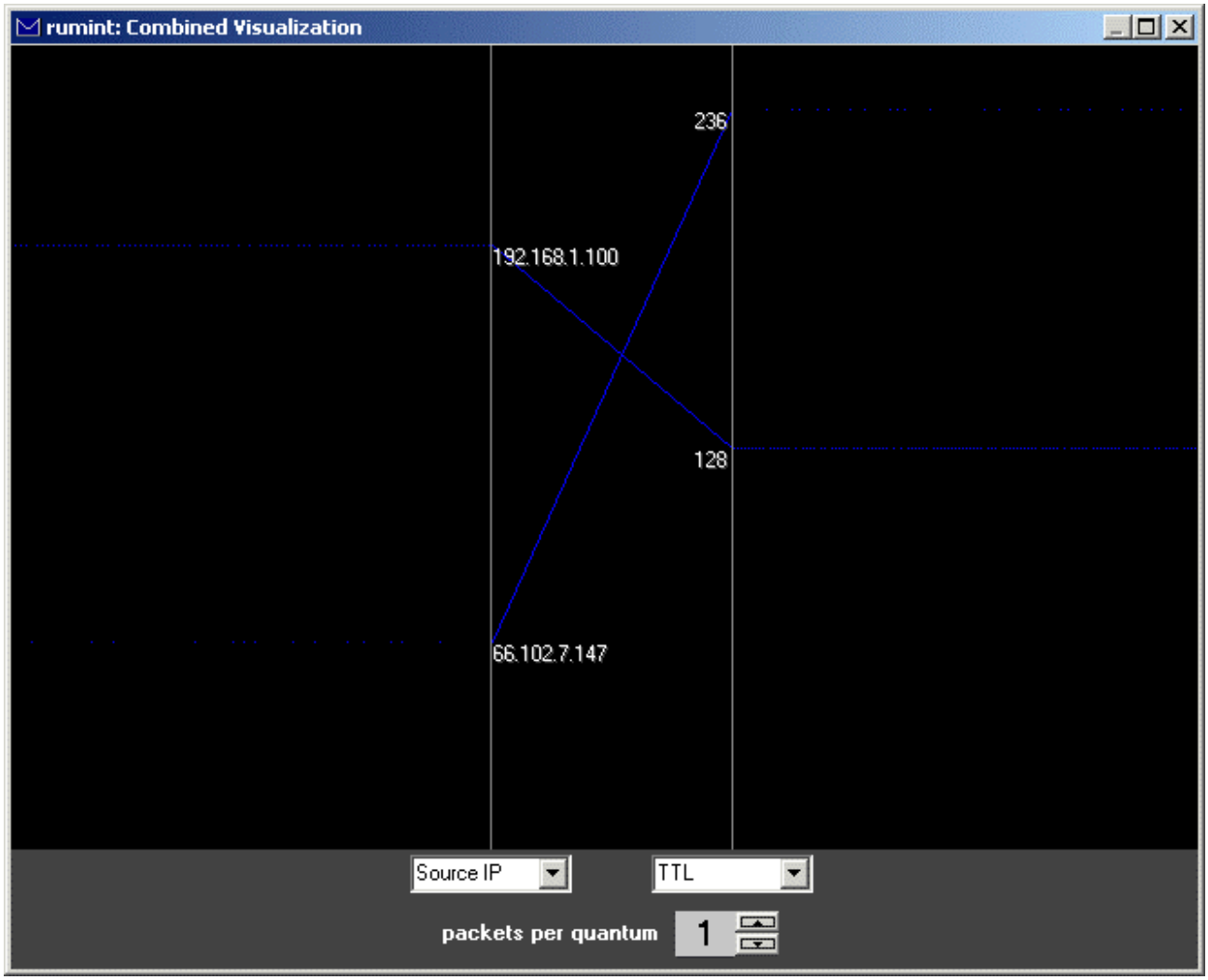

Figure 9: The combined visualization combines a two axis parallel coordinate plot with two animated panels. Each packet generates two glyphs that slide outward off the screen. In this image the source of the ping (192.168.1.100) generates a steady stream of ICMP packets all with a TTL of 128, the typical Windows XP default. The target (66.102.7.147) returns intermittent responses all with a TTL of 236.

The combined visualization combines a two axis parallel coordinate plot with animated glyphs for each packet. It overcomes many occlusion problems because packets are animated and move outward off the screen, like the spaceship missiles in Asteroids. While we chose to look at the source IP address and compare it to the TTL, the dropdown boxes in both the traditional parallel coordinate plot (Figure 8) and the combined visualization (Figure 9) allow you to flexibly choose any set of header fields based on your needs.

[1] S. Krasser, G. Conti, J. Grizzard, J. Gribschaw and H. Owen; "Real-Time and Forensic Network Data Analysis Using Animated and Coordinated Visualization;" IEEE Information Assurance Workshop (IAW); June 2005.