Data Cleansing & Preparation at the Gates: A Data-Streaming Perspective

Don Faxon, George Mason University; R. Duane King, George Mason University; John T. Rigsby, Naval Surface Warfare Center Dahlgren Division; Steve Bernard, George Mason University, Edward J. Wegman, George Mason University.

Abstract
Collection of Internet traffic data at the gates of a large enterprise necessarily involves data cleaning, integration, selection and transformation, especially if data streaming strategies are employed. The huge quantities of packets that typically cross the enterprise gateway – some 1½ to 2½ gigabytes an hour of IP traffic here at George Mason University – makes multiple passes through the data cost prohibitive. Data cleansing, customarily perceived as the removal of noise and inconsistent data, is instead seen as a flagging and tagging procedure to facilitate detection of malformed or corrupted IP packets associated with malicious intrusion, or subtle reconnaissance activity as a precursor to a massive attack on the enterprise computing infrastructure. Since real-time or near real-time implementation of data analysis comprising such innovative concepts as data streaming or evolutionary graphics, fast in-line data cleansing and preparation is required. This paper discusses and illustrates the strategies we have incorporated into our data collection and analysis effort to identify and correlate anomalous activities that point to the discovery of underlying attack strategies or the theft of its knowledge base.

1. Introduction
The focus of this paper is how we go about collecting Internet traffic at the University gateway with a data-streaming perspective in mind; more specifically, how we go about capturing the so-called IP/TCP (Internet Protocol/Transport Control Protocol) packet headers within that traffic.

The vision underlying our program is to achieve a three-fold objective: to detect malicious intrusion activity within our enterprise computing infrastructure, and in near real-time; to discover the underlying strategic methodologies and techniques being employed against us; and to anticipate future malicious intrusion events.

Inherent to the task of achieving this vision are certain presumptions for the collection of the relevant data. First, we must capture the right data for our purpose: data must be sufficient with respect to quality and complete with respect to range of coverage. Secondly, we must capture the data in a manner that seamlessly permits in-line visual analysis at the source in real- or near real-time. And finally, we must implement a data representation scheme that is immediately interpretable and yet easily adaptable to the techniques of visual statistical analysis.

The approach we have chosen to explore at George Mason University is to capture all IP “header” traffic just outside the University gateway, parse the component elements of each packet for their specified values, and then display the packets as a data stream over time. This methodology presumes no a priori knowledge of the data nor its source, and is subsetted (for practical or operational reasons) only at the time of rendering vis-à-vis at the time of collection. Hence, the entire data stream can be “reconstructed” for post-online analyses if deemed important without the need to retrieve the archived raw data.

For the remainder of this paper, we begin by describing the computing environment in which this research was conducted. Next, we describe the data capture strategy employed, the terminology of data packet protocols, and the typical traffic flows experienced at the Mason gateway. Lastly, we detail both the traditional, and then our generic approach to pre-analysis data preparation, and give an example using a parallel coordinate display.

2. The GMU Computing Environment
The following schematic depicts the design layout of the gateway to the University computing infrastructure. Conceptually, the Mason computing infrastructure connects to the outside Internet world through two portals: a primary router which we refer to as Internet Edge A; and a secondary router referred to as Internet Edge B.

The primary router services the Enterprise core and the Backbone core and distribution routers, and can function as a backup for the secondary router in the event of failure.

The Gateway at GMU:
Its input/output fiber is OC-3 specified with a 55 Mbps ATM permanent virtual circuit (PVC) for the Internet and a 100 Mbps ATM PVC for Internet II. Since the Internet II channel is presently significantly underutilized, the channel for the Internet is over subscribed. This router also has a special purpose 2 Mbps ATM PVC wireless connection to the WAN (Wide Area Network).

The secondary router provides dedicated service to the Dorm Core, and can also function as a backup for primary router in the event of failure. Its channel comprises a partial 100 Mb GigE Ethernet line.

The Dorm Core comprises that part of the computing infrastructure directly supporting the student Resident halls, and depends primarily on host-based security measures.

Dorm Core Topology

Internet traffic to and from the Dorm Core routers and servers is supported via the Edge B gateway router to channel most of the non-academic usage away from the Enterprise and Academic Departments computing infrastructure. Additionally, a bandwidth management device is deployed between the Dorm Core router and the Edge B router to prevent abusive activities sometimes associated with the Resident hall environment.

The Enterprise Core comprises that part of the computing infrastructure that provides critical services to the University academic communities – namely the Banner System and “MESA” testbed segment – along with the typical array of mail and control servers. Protection of critical components is provided through firewalls by both network- and host-based security measures deployed in depth.

Enterprise Core Topology

The Banner System consists of the University financial and enrollment management systems. This system provides an interactive user interface using a WEB-deployed service network operating within a so-called “DMZ” (from the term Demilitarized Zone, implying a no-man’s land), while protecting critical internal systems (database servers, etc.) behind a substantial firewall and switching configuration.

The “MESA” testbed segment, or Mason Enterprise Security Architecture, deploys developing prototype software and hardware solutions based on new technologies and methodologies currently being researched. MESA is said to be “… as much a concept as a system”, with the goal of providing better up-to-date cutting-edge security for the overall University computing infrastructure.

The Enterprise Core also encompasses typical mail and control servers, and in particular the University enterprise mail servers, etc., each configured with its own security policy.

The “Backbone Core and Distribution Routers”, for the lack of a better term, is that part of the computing infrastructure charged with servicing the remaining – i.e., academic – communities of the University. This diversity of computing subsystems is supported through an extended array of core and distribution routers, and is provided protection by a plethora of security measures, both network-based and host-based, implemented to differing degrees of effectiveness by technicians with varying degrees of expertise.
The compressed and zipped i-files/o-files are stored in daily directories containing 4 file pairs per hour each x 24 hours or 192 total, and are pulled down once daily to an analysis station on the Galaxy subnet for processing.

The data traffic itself is captured in the form of so-called “IP packets”, nominally consisting of a hierarchy of headers prepended to some sort of payload through a process known as encapsulation.

### Encapsulation of “layers”

**Encapsulation** is the process by which payloads destined for travel across the Internet or local intranet are packaged through a sequence of processing “layers”, starting with the application that provides the original payload data to be transmitted, and continuing down through successive layers responsible for providing the necessary addressing and delivery information. At each successive layer, an appropriate administrative header combines with the “payload” provided to it, which then becomes the next layer’s payload.

**Application Layer:** (HTTP, FTP, telnet) breaks up payloads such that the resulting IP datagram can be sent without fragmenting; this data packet is then passed to the Transport Layer as its payload.

**Transport Layer:** (TCP, UDP) prepends a Transport header specifying the servicing ports and control data, if appropriate; this combined header-data packet is then passed to the Network Layer as its payload.

**Network Layer:** (IP, ICMP, etc.) prepends an IP header specifying the IP addresses and control data, if appropriate; this combined “dual-header” data packet is then passed to the Link Layer as its payload, which is referred to as the “IP Datagram”.

**Link Layer:** (ARP, etc.) for Ethernets, prepends a header specifying the next-hop device’s MAC address (6 bytes), its own device’s MAC address (6 bytes), and the frame type (2 bytes). The so-called “Cyclic redundancy check” or CRC was used...

---

### 3. Data Capture and Terminology

The following schematic depicts the data capturing strategy employed at the University gateway.

**Data Capture Strategy**

A tap is used to replicate both **inbound** and **outbound** Internet traffic data streams for a pair packet capture sensor boxes, colloquially referred to as “sniffers”, operating in tandem.

**Sniffer Box A** captures both replicated inbound and outbound traffic streams on separate network interface cards (NIC) running in promiscuous mode (meaning without interfering with the traffic flow) using a software package known as CoralReef, to an **i-file** and an **o-file**, respectively, both specified by the industry standard native **libpcap** format.

After approximately ten minutes, Box A goes silent, then commences to compress and zip both the **i-file** and **o-file**, and finally moves the completed processed file pair to a protected **file server farm** for storage. Concomitantly, **Sniffer Box B** presumes the capturing tasks for next capture period commencing at the next quarter-hour.

Thusly, Sniffer Boxes A and B alternate as necessary, each producing an **i-file** and **o-file** pair for every 10 minute window on the quarter-hour.
Next, we have the **TCP header** structure, which is part of the “payload” passed to the network layer, and is thus considered as an “embedded protocol” with respect to the IP datagram above. The TCP protocol is used when package delivery has to be reliable. As before, the highlighted elements are those of analytical interest.

### TCP/IP Header Specifications

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>16</td>
</tr>
<tr>
<td>Destination Port</td>
<td>16</td>
</tr>
<tr>
<td>Sequence Number</td>
<td>32</td>
</tr>
<tr>
<td>Acknowledgement Number</td>
<td>32</td>
</tr>
<tr>
<td>Offset</td>
<td>16</td>
</tr>
<tr>
<td>Rsvd</td>
<td>4</td>
</tr>
<tr>
<td>U</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>Window Size</td>
<td>16</td>
</tr>
<tr>
<td>Urgent Pointer</td>
<td>16</td>
</tr>
<tr>
<td>Options (optional)</td>
<td>128</td>
</tr>
</tbody>
</table>

Note that this header includes the source and destination ports associated with the “function” of the traffic, sequence numbers, and the TCP flags, all of which may be indicative of a given type or category of attack. Unlike the IP flags, TCP flags typically have one or more of several settings depending on the traffic and the state of the connection, some combinations of which are obviously “crafted” for some dubious purpose – the setting of the “SYN” flag requesting a connection with the “RESET” flag also set comes to mind.

The following **UDP header** is a shortened version of the TCP header, and is used when reliability of delivery is of secondary importance; this form of delivery is commonly referred to as “send and pray”.

### TCP/IP Header Specifications

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>16</td>
</tr>
<tr>
<td>Destination Port</td>
<td>16</td>
</tr>
<tr>
<td>Length of UDP-datagram</td>
<td>16</td>
</tr>
<tr>
<td>Checksum</td>
<td>16</td>
</tr>
</tbody>
</table>

### UDP Header Specifications

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>16</td>
</tr>
<tr>
<td>Destination Port</td>
<td>16</td>
</tr>
</tbody>
</table>

### ICMP Header Specifications

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>8</td>
</tr>
<tr>
<td>Code</td>
<td>8</td>
</tr>
<tr>
<td>Checksum</td>
<td>16</td>
</tr>
</tbody>
</table>

The **ICMP header**, shown with the UDP header above, is like the UDP header in one respect: it is short. Theoretically, ICMP is considered as an experimentally, called trailer encapsulation, but is now deprecated. **It is this frame, then, that is what hits the wire and sent to the next device upstream.**

Since all data captured at the University gateway is IP traffic and traverses between the same two devices at the gateway, this information has been truncated. Obviously, adding 14 bytes of useless information to each of the gigabytes of header packets would be an enormous waste of both time and storage.

At the designated receiving device somewhere upstream, the reverse process takes place. In a process called “**demultiplexing**”, these packets are processed layer-by-layer and stripped of the appropriate header(s), with checks being made, error messages being sent and/or packets dropped, control responses formulated if appropriate, etc. Eventually the original payload from the source host is passed to the appropriate application in the destination host.

The following schematic shows the structure of the IP header which is prepended to the payload passed to the network layer by the transport layer. The numbers at the tick marks at the top are the bit offset values of the byte boundaries, starting with bit offset 00 for each 4-byte or 32-bit “word” (shown as a row).

### TCP/IP Header Specifications

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source IP Address</td>
<td>32</td>
</tr>
<tr>
<td>Destination IP Address</td>
<td>32</td>
</tr>
<tr>
<td>Options (optional)</td>
<td>128</td>
</tr>
</tbody>
</table>

This header and its payload constitute what is called the IP datagram. Highlighted elements of this header indicate information of interest to a traffic analyst; check sums are nominally for integrity checks, and fragmentation is nominally avoided at the application level in modern architectures due to the inherent overhead cost involved. Note that this header includes the IP address of the both the source and destination machines. Also note in particular that the IP FLAG fields usually only have the middle flag set to DF (don’t fragment), meaning the three IP bits are set to “010”. Thus, for example, different combinations of flags are easily recognized as a possible indication of **malicious traffic**.
adjunct of the IP or network layer; nonetheless, it IS encapsulated in an IP header just as are the transport protocols (TCP, UDP). However, it is really a network layer protocol, and is primarily used to indicate an error condition with some type of failed or blocked delivery – real or “otherwise” (i.e., related to some sort of mischievous activity). Note that ICMP type/code combinations can easily identify potential malicious probing. For example, these conventions specify echos and echo replies (or pings), destination unreachable (net, host, port, host/communication administratively prohibited), redirects, and time exceeded.

The following is an example of a `tcpdump` output of the first two steps of the 3-way handshake used to establish a TCP connection – the external client’s “SYN” message requesting the TCP connection, and the GMU host’s “SYN-ACK” message acknowledging its acceptance.

```
An external client attempting a “handshake” with a host on GMU subnet 172:

16:00:17.048895 168.11.1.123.15991 > 129.174.172.14.1433: [8 492545241:492545241(0) win 65535 <msm 1440,nop,nop,sackOK> (DF)]
 4500 0030 653b 4000 606 cf49 a80b 017b
 81ae ac0e 3e77 0599 1d5b a4d9 0000 0000
7012 ffff a597 0000 0204 05b4 0101 0402

The host on GMU subnet 172 acknowledges the request to the external client:

16:00:17.057973 129.174.172.14.1433 > 168.11.1.123.15991: [8 2365307122:2365307122(0) ack 492545242 win 65535 <msm 1440,nop,nop,sackOK> (DF)]
 4500 0030 e1bb 4000 7e06 43c9 81ae ac0e
 a80b 017b 0599 3e77 8ca0 41b2 1d5b a4d9
7012 ffff df20 0000 0204 05b4 0101 0402
```

This next graphic shows how these same two steps of this 3-way handshake looks in a parallel coordinate display, where the “data point” representing external client’s packet is translated into the blue segmented line, and the “point” representing the local GMU host’s responding packet into the red segmented line.

“SYN” and “SYN-ACK” Packets of 3-Way Handshake

Note that while most axes plot values between zero and 255, such as the octets of the source and destination IP addresses, others plot “more categorical” values, such as the IP and TCP flags, which reflect ranges of zero to 63.

### 4. Gateway Traffic

The Internet traffic inbound and outbound to and from the Internet Edge A across the OC-3 fiber is replicated by the tap. Both inbound and outbound fiber data streams are recorded separately and reflect only IP traffic, and are devoid of any ARP or “address resolution” traffic. Furthermore, the link-level header has been truncated, as all traffic captured at the gateway would only yield the MAC numbers of the devices adjacent to the sniffer, and thus be of no interest in packet analysis. Indeed, inclusion of such information would add a 20% frivolous overhead to packet storage requirements.

Every quarter-hour, a ten-minute window of packet traffic is captured and recorded on each wire, resulting in four `i`-files (inbound packets) and four `o`-files (outbound packets) or eight files per hour. For a 24-hour period, this sums to 96 `i`-files and 96 `o`-files for a total of 192 files daily. Since recent experience of Internet traffic at the gateway shows a nominal traffic flow of approximately 200 MB per wire per 10 minute capture window over a typical 24-hour period, this equates to some 40 GB per day. (This value is for packet headers only, without the 14-byte link layer headers – they alone would require another eight GB of storage daily!)

Below is an extract of a typical data collection summary for the first two hours of the captures recorded in the directory for December 8, 2003, starting at midnight, just as the traffic flow is starting to slow and approach its daily low, typically occurring around 6:00 a.m. in the morning.

<table>
<thead>
<tr>
<th>Time</th>
<th>Flow</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>in</td>
<td>157M</td>
</tr>
<tr>
<td>15</td>
<td>in</td>
<td>106M</td>
</tr>
<tr>
<td>30</td>
<td>in</td>
<td>103M</td>
</tr>
<tr>
<td>45</td>
<td>in</td>
<td>108M</td>
</tr>
<tr>
<td>100</td>
<td>in</td>
<td>162M</td>
</tr>
<tr>
<td>115</td>
<td>in</td>
<td>119M</td>
</tr>
<tr>
<td>130</td>
<td>in</td>
<td>140M</td>
</tr>
<tr>
<td>145</td>
<td>in</td>
<td>152M</td>
</tr>
</tbody>
</table>

```
```

Extract of Data Collection Summary

Traffic Flows for Directory "\data\gm120803": --
The following six graphics depict 24-hour IP traffic flows observed at the end of the Fall 2003 Semester at George Mason, during the period from Monday, December 8th to Saturday, December 13th, inclusive. The first two days were designated “reading days”, at the end of which Final Examinations were conducted.

Legend for 24-Hour Traffic Flows:
+ inbound fiber
* outbound fiber

5. Traditional Data Handling

Now that we have a general idea of the computing environment we must operate in, how we capture the data within this environment, and what its component parts are, let us now take a look at the traditional approach to pre-analysis data preparation typically taken by intrusion detection system (IDS) analysts using a priori knowledge. Then we will compare such an approach to a generic approach which makes use of no such a priori knowledge, and thereby allows us to eventually evolve into a data-capturing and visualization strategy with a data-streaming perspective.
Using the traditional approach, we read the data stream using the Unix utility `tcpdump` with certain options set (such as whether or not we want address and/or port name-resolution performed) and filters applied (such as whether or not we want traffic to or from only certain addresses or ports). As the data stream passes through our “network sensors”, certain characteristics of the captured packets are periodically summarized and displayed to give us an initial near real-time “look see” of our traffic. Each set of options and/or filters will result in a separate display.

In our particular case, we are dealing with stored, zipped/compressed files recorded separately for inbound and outbound traffic. Nevertheless, unzipping and uncompressing such files using a utility such as `zcat` and “piping” its output through the `tcpdump` utility “on the fly” conceptually achieves the same effect. (Piping is the fundamental Unix concept by which each piece of data processed by the first utility in the pipe is immediately passed to the second utility in the pipe, piece by piece, not waiting until the entire output is ready – the delay is negligible. That is, each process in the pipe – here, `zcat` and `tcpdump` – is said to be running in pseudo-simultaneous execution.)

To collect and display the session-oriented character of the network traffic, both inbound and outbound data streams obviously have to first be merged into a single data stream for tracking query-response activity between the relevant communicating hosts and clients, especially if duration of connection is being measured. Using the Unix utility `mergecap` that accomplishes this for `pcap` formatted data, we replace the i-file/o-file pair for the given ten-minute time window with a mixed file. Unfortunately, this requires the building of a complete mixed file before passing the merged data to the `tcpdump` utility, and thus even conceptually we are now performing our initial data analysis off-line.

## Pre-Analysis Data Preparation: traditional approach

**For user “session” analyses:**

### mergecap:

```
mergecap: i-file | tcpdump [options] [filters] ➔ display
```

### zcat i-file | tcpdump [options] [filters] ➔ display

```
“zcat” ➔ uncompress/unzip utility
```

```
“tcpdump” ➔ packet analysis utility
```

```
“piping” (|): unix feature that allows zcat to pass uncompress & unzipped packets to tcpdump “on the fly”
```

There is, however, a potential problem associated with doing session analyses on merged capture i-file/o-file pairs where the inbound and outbound data streams were originally captured and stored separately. Within each sniffer box, the initiation of the capturing process for the two data streams coming in through different NIC – here, one for inbound traffic and another for outbound traffic – may be slightly out of time synchronization. This may be significant because packet sorting is done using time stamps embedded at the time of capture. Just consider the three-way handshake example shown above: the client request SYN packet and the host (server) response SYN-ACK packet were only a hundredth of a second apart. That would require processes for the inbound and outbound data streams be synchronized within the better part of a millisecond, or our session packets could well be mismatched.
So the question arises, “Why not capture the data as a single stream?” In the case of our gateway monitoring system, the reason is the physical medium we are working with. Fiber optic patch cables use one physical fiber for transmission, and one for reception. On a fiber optic NIC, one port of the interface can only receive and the other can only transmit. Thus, two NIC are required. One might think a more accommodating physical medium such as copper would be found further “down stream” into the enterprise infrastructure, thereby allowing us to avoid this problem. But at Mason, fiber optics are in abundant use throughout the campus, not only for larger bandwidth, but also for the ability to maintain signal strength over long distances and a relative impunity to electromagnetic interference. There are solutions to this problem, but they are not cheap.

A well known IDS example that follows this type of traditional approach is SHADOW (SANS’s Heuristic Analysis System for Defensive Online Warfare), a network-based intrusion detection system, originally called the Cooperative Intrusion Detection Evaluation and Response (CIDER) project. This system collects and processes the captured packets on an hourly basis, displays items of interest and/or concern, and then amalgamates and stores daily summaries for further analysis if required. It should be noted that such production systems are designed to be able to capture as much of the IP datagrams as is necessary to achieve the requisite monitoring objectives. In any case, the original data streams are usually stored in their entirety in the event forensic analysis is required. The GMU captures used for this research project are of packet headers only, and for the duration of this project were nominally constrained to 68 bytes, even if the packet headers contained optional elements necessitating a longer total length. The SHADOW IDS is currently maintained and developed by NSWC (Naval Surface Warfare Center) in Dahlgren, VA.

**Now here comes the glitch.** A first major deficiency with the traditional approach is that it is essentially based on “anecdotal evidence.” That is to say, the approach traditionally taken by intrusion detection analysts is to sequentially apply filters and option sets to either weed out undesirable traffic, and/or to focus (or subset) on traffic that reveals certain characteristics of interest. This approach emphasizes building a portrayal of the data on “evidence” of a partial if not transient or even deceptive nature. Consequently, we may be led to a myopic and unrepresentative assessment of malicious activity – or the lack of it – without the benefit of incorporating evolving changes taking place within the data flow.

A second major deficiency with the traditional approach has to do with the question: “What exactly constitutes a session?” The answer is important if average session duration is used as a detection metric. Computer security specialists are not only required to react and neutralize an intrusion event, but often are called upon to describe what has happened or is about to happen to potentially everyone, from system administrators to enterprise management, and some of those caught in-between. And this translates into having to deal with a myriad of opinions of what actually comprises a session.

By a session, an analyst typically takes to mean a TCP session, a well-defined IP/TCP notion that denotes the time between the establishment of a connection to the mutual acknowledgement of termination – emphasis on *mutual*. On the other hand, most of us regard a session to mean the time spent by a user connected to a particular IP site. This latter notion is fraught with ambiguities.

First, take the so-called “well-defined” IP/TCP notion of a session. What if the acknowledgement of session termination is NOT mutual? One machine may “think” the session is over, but if this half-open link was intentionally created by a malformed packet, the session is likely anything but over!

Then there is the connection that is established and legitimately remains open for a substantial period of time, yet all the while the contained traffic activity is sparse and interspersed with long durations of inactivity, as is typical of web traffic. This latter situation goes to the heart of what constitutes an attack session, especially with regard to attack evaluation and threat assessment. Typically, malicious reconnaissance activity often starts with the use of relatively benign utilities for assessment of network connectivity, and later turns to exploiting malformed UDP and ICMP packets to probe a targeted network space (i.e., our computing infrastructure).

From the hacker-victim perspective, for example, a ping-sweep of a network space can be considered a “session” in the sense that it constitutes a well-defined duration of hacker’s connectivity with the victim’s network – though technically it is not. Similarly, SYN-ACK attacks against a network space (which do use TCP) are also not technically sessions, but rather a sequence of attempted sessions, together which can be considered a session in this sense. And SMTP probing, in which one simply sends emails to non-existent addresses to obtain useful network information by analyzing the content of the failure-
of-delivery responses, might even be construed as a session of sessions.

6. Generic Approach and Visualization

Finally, we come to an approach that collects data without a priori assumptions, and renders them in a manner that robustly preserves data characteristics, even through large changes in the nature of the traffic flow. Such an approach is mandated for the proper use of statistical visualization techniques that exploit human mind and eye coordination and pattern recognition capabilities – it is mandated that all relevant data for a given time period be displayed without a priori filtering.

In this approach, each packet in the data stream is first parsed for the values in each of the IP fields, especially noting the type of embedded protocol (specified in byte-offset 0x09). Next, based on the relevant specifications, the embedded protocol fields are parsed for their values.

Pre-Analysis Data Preparation:
“generic approach” *(NO a priori assumptions)*

For each packet:
- parse the relevant fields to form a packet vector
- portray sequence of vectors as they appear over time

The resulting IP and embedded protocol field values are then constituted into a packet vector, which can then be portrayed in an array of 2x2 graphic displays, or alternatively, in parallel coordinates. The magnitude of the network traffic and the graphic resolution and manner of presentation will dictate the duration of time that may be displayed at any given moment. The greatest advantage of such an approach is the inherent ability to observe all the attributes of the data stream packets together so that as experienced is gained, even relatively benign patterns associated to the more obvious activities may be noted. We have in effect replaced pattern recognition through machine learning with pattern recognition using the human mind. We have also obviated the need to overly concern ourselves with how we specify what constitutes a session, as the duration of an activity in visualization pretty much defines itself, at least relative to its environment.

As an example of using visualization techniques given data captures represented by generic packet vectors, we present what a ping sweep of a local address space would look like, and why it is easily distinguishable in a parallel coordinate display. But first we discuss some algorithmic issues that arise in using a parallel coordinate representation.

Two issues dominate computational considerations. First, the data from the inbound and outbound fibers may simply be processed and plotted separately, side by side, letting the two be correlated cognitively via stereoscopic viewing. Thus, there is no need to merge the two data streams at the gateway, which in turn allows us to approach achieving real-time data visualization given a sufficient funding level for its development.

Secondly, the field data are typically scale-transformed and in some instances mapped to proxy values. For example, the values displayed for most axes will be that of the relevant byte, i.e. between zero and 255. IP addresses will need to be displayed by octet, requiring four separate axes for each address (source and destination). TCP flag settings, although specified by 0 or 1 (for on/off) values in six different bit-fields, nevertheless may be uniquely represented by a proxy combined value between zero and 63, or 1 less than 2^6.

Similarly, three algorithmic issues dominate rendering considerations. First, packet “signatures”, which may be thought of as bands within which packet line segments meeting certain relevant criteria would fall, may vary dramatically between inbound and outbound packets of the same session traffic. So allowances must be made to prevent obfuscation of which traffic belongs to which.

Secondly, color, intensity and persistence of packet displays will need to be regulated to differentiate between types and duration of traffic sessions, and possibly other nuisances as they become evident with experience. It should be noted that in most instances, TCP sessions will tend to be port related, albeit differently for inbound and outbound packets. It is especially noteworthy to mention that in this sense, persistent malicious UDP/ICMP packets, or traffic that elicits such packets, will also tend to be port related, in the sense that network scans present a pattern of ports over a short time.

Thirdly, rendering of “background noise” from large data flows will in practice require density rendering, as parallel coordinate displays typically become massively over plotted fairly easily. Even though this background noise will present its own structure, it will generally tend to be portrayed as static, where
scanning activity should be fairly obvious due to the dynamic nature of changes taking place within its packet header values.

One can get a good idea of these effects by downloading and running the compressed video presentation of the *Spinning Cube of Potential Doom* developed by Stephen Lau and his NERSC group at the Lawrence Berkeley National Labs, found at the following Internet address:

```
http://www.nersc.gov/nusers/security/TheSpinningCube.php
```

The NERSC video shows a three-dimensional view of connection information derived from their Bro IDS in terms of the *Scinet’s* IP address space (the test site network), the all possible address space, and all possible port numbers. When you view this video, you will notice a lot of background activity, especially in the well-known ports, numbered in the range 1-1023 (before you ask, zero is not allowed). Against this background, port scans, “barber poles” (a smart combination of IP address and ports scans), and “lawnmower” scan patterns immediately stand out due to their relatively dynamic nature. The most important idea to take away from this video is that it provides a means to demonstrate network and intrusion detection activity in a graphical format understandable by those unfamiliar with networking and computer security techniques – i.e., to your enterprise stakeholders and the more senior members of your incident handling teams.

We see in the following two figures how the above ping sweep is visualized in parallel coordinates. The first figure is a very small portion of the tcpdump output. Note the very short time intervals and the apparent randomness of the values in the 4th octet of the destination address (highlighted in yellow).

---

**Example of external mapping of GMU subnet 17:**

```
16:00:33.876165 207.112.129.2 > 129.174.17.34: icmp: echo request
4500 0025 e1e0 0000 7201 838c cf70 8102
Blas 114a 0800 bff1 0200 73dc 6865 6c6c 6f20 3f3f 3f

16:00:33.944984 207.112.129.2 > 129.174.17.99: icmp: echo request
4500 0025 e1e3 0000 7201 8370 cf70 8102
Blas 1163 0800 bcf1 0200 76dc 6865 6c6c 6f20 3f3f 3f

16:00:33.947569 207.112.129.2 > 129.174.17.130: icmp: echo request
4500 0025 e1e4 0000 7201 8350 cf70 8102
Blas 1182 0800 bbf1 0200 77dc 6865 6c6c 6f20 3f3f 3f

16:00:33.950712 207.112.129.2 > 129.174.17.100: icmp: echo request
4500 0025 e1e5 0000 7201 836d cf70 8102
Blas 11d0 0800 baf1 0200 78dc 6865 6c6c 6f20 3f3f 3f

16:00:33.959719 207.112.129.2 > 129.174.17.75: icmp: echo request
4500 0025 e1e8 0000 7201 8383 cf70 8102
Blas 11e6 0800 b7f1 0200 7bdc 6865 6c6c 6f20 3f3f 3f
```

*In this 10min window alone, subnets 17 thru 31 were scanned!*

As time increases, the pattern moves to the right along the “DstIP4” axis as subnet 129.174.17.xxx is being scanned. The triple band pattern arises from the algorithm being used which attempts to hide its activity by apparently randomly jumping around – yet it does so in a cyclic pattern reminiscent of those exhibited by the early random number generators. Once the 129.174.17.xxx subnet addresses have been scanned for active devices, the pattern resumes for the 129.174.18.xxx subnet until all of its addresses have been scanned, and so on.

The second graphic shows the first second of the sweep as above, but now superimposed with the a similar portion of ping sweep at the end of this ten-minute window. Note that during this time, the ping sweep has progressed from subnet 129.174.17.xxx to subnet 129.174.31.xxx, as indicated by the positions of the “points” on the axis of the 3rd octet of the destination (GMU) address. Background noise, not shown in either of these two figures, would form a back drop similar to the noise in the NERSC video.

---

**Scan of Subnets 129.174.17 -- 31.xxx**

The first graphic of this ping scan shows a triple band pattern of just over one second, which is what would be observed with an image persistence of one second.
Summary.

We have discussed how we go about collecting Internet traffic at the University gateway with a data-streaming perspective in mind. Our objective is to detect malicious intrusion activity within our enterprise computing infrastructure; to discover the underlying strategic methodologies and techniques being employed against us; and to anticipate future malicious intrusion events. This objective dictates that our data captures be sufficient with respect to quality and complete with respect to range of coverage, that we achieve this in a manner that permits in-line visual analysis at the source in real- or near real-time, and that we implement a data representation scheme that is at once interpretative but yet easily adaptable to the techniques of visual statistical analysis.

The approach we have chosen to explore at George Mason is to capture all IP header traffic just outside the University gateway, parse the component elements of each packet for their specified values, and then display the packets as a data stream over time. This methodology presumes no a priori knowledge of the data nor its source, and is subsetted only as a practical matter of display design, and then interactively and only at the time of rendering.

Finally, we described both the traditional approach of data cleansing and preparation used by IDS analysts, and then the generic approach using packet vectors that allows seamless input to visualization techniques that are understandable by all with a vested interest in what intrusion activity is taking place, or about to take place.

References:


Acknowledgements:

The parallel coordinate graphics are provided using in-house *Crystal Vision* software, available at the Center for Computational Statistics FTP server at “ftp://www.galaxy.gmu.edu/pub/”.

Don Faxon, Associate Professor affiliated with the GMU Center for Computational Statistics, presented this paper at the JSM 2004 in Toronto in August. The work of Dr. Faxon was supported by Contract No. F49620-01-1-0274 from the Air Force Office of Scientific Research.

R. Duane King, Systems Engineer with GMU SCS, continues to be the mainstay of providing guidance in infrastructure security development and deployment.

J. T. Rigsby, Systems Engineer with NSWC Dahlgren Lab, has provided procurement and configuration support throughout.

Steve Bernard, Network Security Engineer with GMU ITU, has been instrumental in establishing and maintaining the sensor boxes at the University gateway.

Edward J. Wegman, Professor and Director, GMU Center for Computational Statistics, has provided invaluable guidance and overall direction to every aspect of this research project, and has substantially facilitated the acquisition and development of the gateway data collection capability. Dr. Wegman was also the Principal Investigator for this Air Force supported contract.