Statistical Analysis of Network Data for Cybersecurity

Article *in* CHANCE · September 2012

DOI: 10.1080/09332480.2004.10554881

2 authors:

David Marchette
NSWC
161 PUBLICATIONS **2,176** CITATIONS
SEE PROFILE

Edward J. Wegman
George Mason University
216 PUBLICATIONS **4,069** CITATIONS
SEE PROFILE

How do you detect attacks on a network computer that were sent from a remote host?

Statistical Analysis of Network Data for Cybersecurity

David J. Marchette and Edward J. Wegman

Every day or so a new virus is released on the internet. Every week companies report server break-ins. Several times a year major attacks on the internet are announced. These attacks are reported to cost billions of dollars in lost productivity, lost data, and lost business. Detecting and modeling these attacks has only recently begun to attract the attention of statisticians.

Transaction data in cybersecurity takes many forms. There are the basic protocol of the internet, that governs everything from email to web browsing, remote logins, chat sessions, and instant messaging. For banking purposes, authentication and encryption are essential to ensure that the person accessing the account is who they say they are, and that no unauthorized person can observe the transaction.

On the host, there is the interaction with the operating system, and the requirement that programs and users are acting in their proper manner, and are who they say they are.

The field of cybersecurity is large, and we will only touch on a very small part of it. We will discuss primarily network security, in which we are interested in detecting attacks on a network computer from a remote host. This will require a brief introduction to the network protocols and the data that we observe. We will provide some examples of attacks from the past and discuss some areas for statistical analysis. Then we will consider the problem of visualizing network data. For a good discussion of computer security, see Bishop [2003]. Statistical issues of computer security are covered in Marchette [2001]. A good discussion on the issues of user profiling, fraud detection, and the detection of masqueraders can be found in Schonlau et al. [2001] and Bolton and Hand[2002].

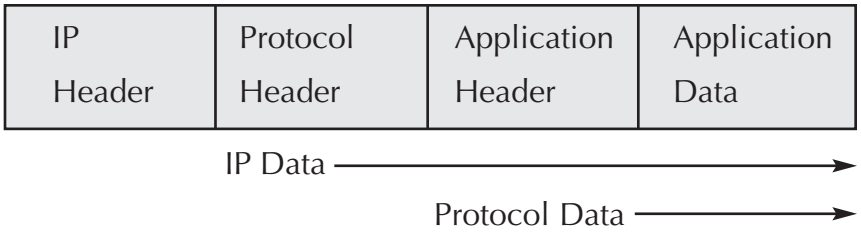

Figure 1. A typical IP packet. Each layer prepends a header onto the packet as it is passed down the IP stack.

Network Data

The basic data structure on a network is the packet. All communications—email, web, chat or a remote login---are encapsulated in a series of packets. Each packet consists of a header of routing information (basically, the to and from address of the packet), plus the data. Transactions are broken into a series of relatively small packets and sent out on the network. It is important to realize that packets are dynamically routed. Each router determines the best next hop for the packet and sends it along to the next router closer to the destination. This means that different packets of the same communication can take different routes and thus do not necessarily arrive in the order in which they are sent.

The basic protocol of the internet is IP (Internet Protocol). This is a nonreliable protocol (meaning that there are

no reliability guarantees made by the protocol). Other protocols are available to provide these guarantees, and these are implemented within the data portion of the IP packet (see Figure 1). An IP packet contains the basic address information, source and destination IP address, a unique identifier for the packet, the protocol of the data portion of the packet, fragmentation information, and various options for routing.

Let's illustrate this with an example of an email session (this is a simplification, but will serve to illustrate the basic ideas). The user types the email into her email application and hits "send." The email client sends the data to the application layer, which breaks the email into manageable pieces and places an application header onto each packet. This header is specific to the email protocol. These packets are then sent to the protocol layer, where a header specific to the protocol is prepended (in this case the protocol will be TCP, which is designed to provide guarantees of delivery). Finally, the packets are given to the IP layer, which prepends the IP header (consisting of the source and destination IP addresses) and the packet is sent out on the wire. At the receiving end, the process is reversed, and the application data is then put together into an email message and provided to the appropriate user.

The IP header is illustrated in Figure 2. Each row represents four bytes of the header, and the header should be thought of as contiguous bytes, arranged in a table merely for ease of presentation. A field that spans a whole row is a four-byte field, while a field that only spans half the row is two-bytes. Most fields are either four-bits, 1-, 2-, or 4-bytes, as indicated by the size of the corresponding boxes. We will not discuss the fields in detail [see Stevens (1994)] or any standard book on networking for more detail], but we will only point out those fields that are necessary to understand the rest of the paper. More details can also be found in Marchette [2001].

The most important fields in the IP packet are the source and destination IP addresses. These are 32 bits, usually represented in "dotted" notation, such as 10.10.125.37. These are used to identify the sending and receiving hosts.

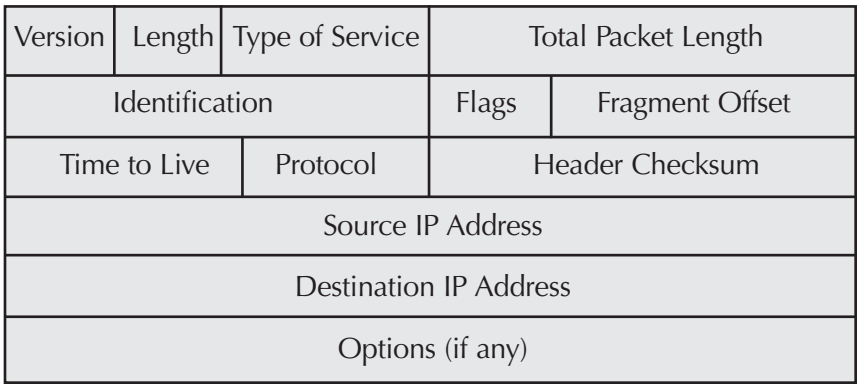

Figure 2. The IP header.

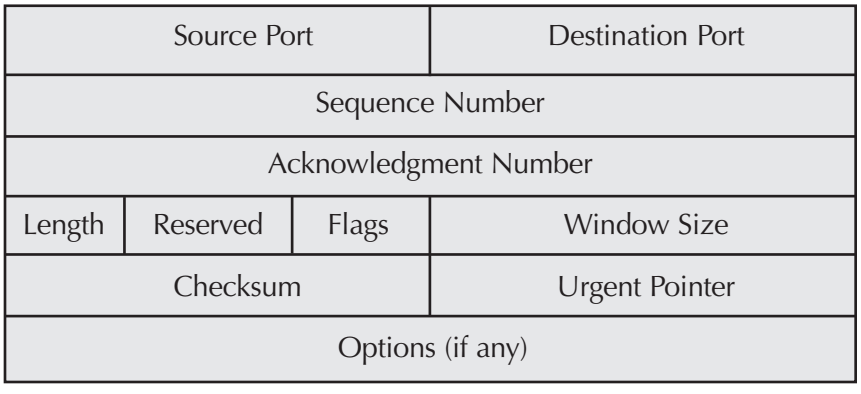

Figure 3. The TCP header.

The other important fields are the flags and fragment offset fields. As mentioned previously, packets that are too big may be broken up into smaller packets, called fragments. These fields are used to ensure that the packets can be reassembled at the destination. If a packet is fragmented, the first fragment consists of a fragment offset of zero and the bit corresponding to the "more fragments" (MF) flag is set. Subsequent fragments set the fragment offset field to the value corresponding to their place in the packet. All but the last packet have the MF flag set. By assembling the packet as indicated by the fragment offset field, the original packet is reassembled. All fragments for a given packet have the same identification field as the original, so that the receiver can determine which fragments belong with which packet.

The main protocols that we are concerned with, beyond IP, are the Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). ICMP is used for error messages and diagnostics, while UDP and TCP provide the basic one- and two-way communication channels used by most applications.

The IP protocol does not provide any guarantees of delivery or reliability. A packet is sent off, and if it is received there is no necessity for the receipt to be acknowledged to the sender. If the packet is lost, it is lost, with no mechanism for resending. There are error mechanisms built in, using the ICMP protocol, in which packets that cannot be delivered result in an error message sent to the sender, but beyond this, any desired reliability is implemented in the other protocols.

The vast majority of traffic on the internet is either UDP or TCP. UDP

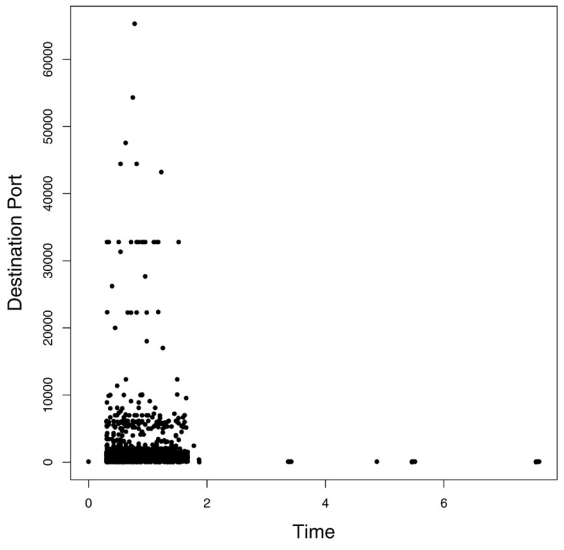

Figure 4. An nmap scan against a single host. Only the TCP packets are shown.

provides a one-way connection with no guarantee of service, while TCP provides two-way connections with assurance: packets which are not delivered are re-sent. Both protocols implement the concept of connection ports, which can be thought of as an extra two bytes of addressing. This provides a unique communication path between the two computers.

TCP implements the two-way connection via an initiating handshake with sequencing controlled by sequence numbers. The TCP packet header is shown in Figure 3. The main fields of interest are the port numbers, sequence numbers, and flags. The port numbers determine the applications associated with the connection, and allow the computers to have several sessions between the applications without mixing them up.

To illustrate, consider a single web session. Web is on port 80, so the initiating (or client) computer, A, will send a packet to port 80 on the web server, B. This first packet has only a single bit set in the flag field, the bit associated with the SYN (or synchronize) flag. Such a packet is called a "SYN packet." No other data is sent in this packet. The packet has a sequence number, and all subsequent packets will increment this sequence number. The source port is set to an unused port, but is otherwise arbitrary (there are some restrictions, but these are not important for our discussion). The web server, upon receipt of the SYN packet, agrees to make the connection by acknowledging it (ACK): it sends a packet with the SYN and ACK flags set (a "SYN/ACK" packet). It sends its own sequence number and sets as acknowledgment number the original packet's sequence number incremented by one. Finally, the client computer responds with an ACK packet, acknowledging the completion of the "three-way" handshake. Packets are now free to flow back and forth, along the communication channel set up between the two IP/port pairs. When one side wishes to stop sending, it sends a FIN (finish) packet, which is acknowledged with a FIN/ACK packet. Once both sides are closed off, the communication ceases. If at any time something goes seriously wrong, or if the server decides it does not wish to communicate, a reset packet (flag: RST) can be sent to kill the session.

To give an indication of the magnitude of network data, consider the following statistics, taken from a relatively small network (on the order of 3,000 computers) on Jan 15, 2002. The headers from this day were 2.4G, compressed. At midday, there were over 200M per hour, compressed, 600M uncompressed. This corresponds to around 8 million packets (7.5 million TCP packets) in one hour. As these data are over a year old, and the trend is always toward faster/bigger/larger, this provides a lower bound on the magnitude of the data that must be analyzed. We will use these data in the discussion and figures in the rest of the paper.

As a contrast, the network at George Mason University typically sees between 1.5G and 2.5G per hour, compressed. This illustrates the need for methodologies to handle large, complex, streaming data.

Network Attacks

We class an attack as a "network attack" if it takes place on a network between two distinct network devices (hosts), utilizes the network protocols or information specific to network applications, and is detectable by a sufficiently knowledgeable analyst looking at network headers only, without looking at the data in the packet.

Network attacks fall into three broad categories. There are information gathering probes, denial of service attacks, and attempts to obtain unauthorized access. We will provide some examples of attacks from each of these categories, but will focus primarily on denial of service attacks in this paper.

Some Illustrative Attacks

Next we discuss a few attacks that have been popular in the past. These attacks are no longer viable, due to improvements in security and the fixing of bugs in the operating systems, but they illustrate some of the main approaches that attackers take.

Probes and scans are used to gain information about the network or hosts that are potential targets. These usually take the form of a series of packets sent to various IP addresses, or a collection of ports on a specific machine. An example of this type of activity is provided by the nmap program (see www.insecure.org/nmap/).

An example of an nmap scan is given in Figure 4. The first few seconds involve scanning for applications (open ports). Note that the ports are not checked sequentially, and that it does not bother to check all possible ports (in fact, this scan chose to check only 1,538 of the possible 65,536 ports). After the first few seconds, a few packets are sent to a single port. These packets are attempting to do active fingerprinting of the operating system. By sending carefully crafted packets that have some unusual characteristics, it is possible to tell the operating system of the host from the

response to these packets.

A scan of this type is easy to detect. So called stealthy scans, where the packets are spread out in time and an effort is made to use packets that are ignored by most intrusion detection systems, are harder to detect, and are an interesting challenge for intrusion detection systems.

Once the victim has been identified through reconnaissance, such as a scan or other means, the attack can proceed. For the most part, cyber attacks rely on one or more of the following:

- Bugs in an application or operating system.
- The kindness of strangers.
- Misconfigured hosts.

We will illustrate each of these in turn.

Many attacks take advantage of bugs in an operating system. One such is the "ping of death." A ping packet is an ICMP echo request packet. The usual intent is that the receiving machine will echo the packet back, thereby verifying that the machine is up and receiving and that there is a route between the hosts. There is a maximum allowable size for such packets, and unfortunately some operating systems failed to check for packets that were too big and crashed if an illegally large packet was received. Thus all the attacker needed to do to crash such a machine was to send a very large ping packet, either by writing custom code, or by using one of the "ping" programs that allowed such illegal packets to be sent.

The "teardrop" attack is similar. In this, a packet is fragmented by the attacker in such a way that the fragments overlap. Since overlapping fragments cannot happen naturally, some operating systems did not handle this case properly and crashed.

Some attacks rely on the kindness of strangers. The classic example of this is the "smurf" attack. Recall that when a packet is sent out, the IP address of the sender is set as the source IP for the packet. Unfortunately, there is nothing to stop a malicious coder from putting any IP address he wishes in this field. Placing some other IP address as the source is referred to as "**spoofing**" the source address. The smurf attack works by selecting a large network as an intermediary. The attacker sends pings to the network with the source address replaced by the address of the victim. The machines on the network respond to the pings. If the network is sufficiently large, the victim is unable to process all the incoming packets, and is effectively removed from the network.

The key to this attack is the implementation of a "broadcast" mechanism, whereby a single packet can elicit a response from many machines. For example, if a packet is sent to the IP address 10.10.255.255, then all the machines with an IP address whose first two octets are 10.10 will respond (potentially over 65,000 machines). In practice, most networks do not allow such broadcast packets, and so this attack is no longer effective. However, it does illustrate one of the basic ideas: get someone else to do your dirty work for you.

An attack that was popular in 1999 is the "UDP storm" attack. There are two utilities that come with most operating systems: an echo port (port 7), which, if enabled, will echo packets sent to it back to the sending computer, and a character-generating port (port 19), which, upon receipt of a packet, sends a series of packets back, each one with the next letter of the alphabet. These are meant to be used to diagnose network problems, and should not be enabled on any system, except when such diagnosis is necessary. Many early systems were shipped with these ports enabled by default, however. The attacker selects two such machines, and sends a UDP packet with the source IP address set to one of the victims, Victim 1, and the destination set to the other, Victim 2. The source port is set to the echo port, and the destination is set to the character generator port. Upon receipt, Victim 2 sends back a bunch of packets, each with a different character in it. Victim 1 then echos these back. Each of these packets produce a new flood of characters. The result is that the two victim machines end up attacking each other, and both go down.

The final attack we will describe is TCP hijacking. This relies on two victim machines being configured to trust each other. To be specific, machine A "trusts" machine B in the sense that a login from B to A is allowed without requiring a password. This can be implemented in a variety of ways, for example by allowing remote login (rlogin) connections that are set up in the appropriate manner. Recall that the TCP session is sequenced by way of the TCP sequence number. When a SYN packet is received by A, it sends a SYN/ACK packet with a new sequence number. The responding ACK refers to this sequence number, ensuring that the two machines understand the sequencing for the session.

The purpose of the hijacker is to obtain a login on machine A by pretending to be machine B. That is, he will spoof the address of machine B. Since he will not be able to see any responses (they will be sent to machine B) he needs to be able to guess the sequence number that A sent in response to his initiating SYN packet.

The first thing he does is find out the algorithm used by machine A to choose sequence numbers. He sends a few SYN packets to A and observes the pattern of sequence numbers returned. Some machines simply choose the sequence number to be an increment of the last sequence number; for example, s+10,000. Once the pattern is determined, he sends resets to close off the connections. Next, he kills machine B, say, by a SYN flood, making B unable to respond to any new connections. He sends a SYN packet to A to determine the state of the sequence number generator, then sends a SYN packet to A with B's IP address spoofed as the source. The SYN/ACK packet goes to B, but B does not respond, due to the SYN flood. Even though he has not seen the SYN/ACK packet, the attacker knows what it must have looked like (because he can now guess the sequence number), and so sends an ACK packet, still spoofing B, that matches. The session is complete and the attacker is logged in!

Unfortunately, the attacker still cannot see any respons-

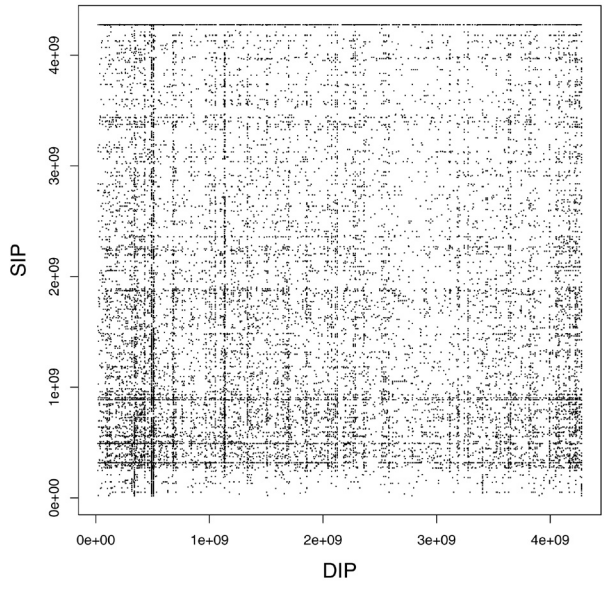

Figure 6. Destination IP address plotted against source IP address.

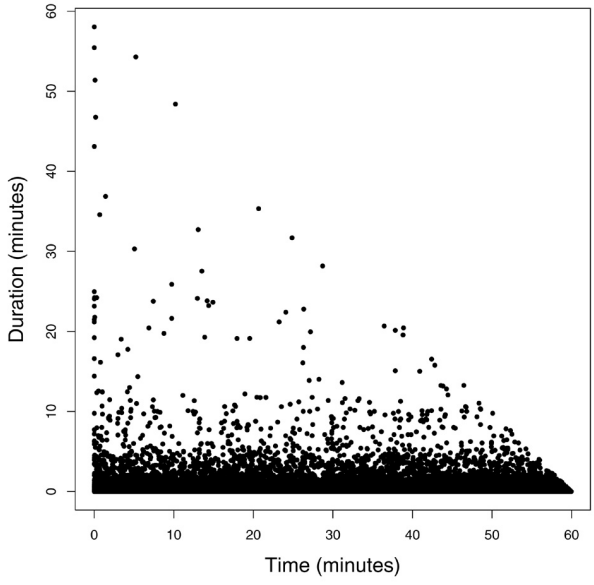

Figure 7. Duration plotted against session start time.

es from A, since he must continue spoofing B. However, by sending a few characters, he can change the trust relationship so that A trusts everyone! By doing this, he sets himself up to have a real, two-way connection. He can now log in as if he were a trusted computer, with no password required.

Note that all these attacks share a common theme: they all are "abnormal" and could, in principle, be detected by looking for abnormal activity. This is an area in which statistics can naturally play a part. By modeling the normal behavior on the network, one can detect novel attacks by comparison to the model and detection of outliers. Unfortunately (or fortunately, depending on your point of view), this is easier said than done, as we will see. In network data, outliers are the rule, rather than the exception, and so more thought needs to be put into the modeling.

Distributed Denial of Service

New attack tools use the philosophy that if one is good, many is much better. These tools compromise multiple systems, then use these as platforms to mount a distributed attack. Suddenly the victim is under attack not from a single attacker, but from hundreds. If the attackers also spoof their addresses, the victim will be hard-pressed to defend against the attack.

A typical distributed attack involves first compromising a set of computers, the masters, which will coordinate the attack. Each master then compromises a further set of computers, the slaves, which will actually mount the attack. This provides protection for the attacker, since once the masters are compromised, the attack can proceed without human interaction. It also provides a multiplying effect, since each master controls a large number of slaves, all of which can be used to attack a single victim.

One example of such an attack is a SYN flood. In this, each attacking machine sends many SYN packets with a spoofed source address, requesting many sessions that it has no intention of completing. While waiting for its SYN/ACK response to be answered, the victim computer holds a place in its connection queue for each connection. When the queue fills up, no legitimate clients can get through. By using hundreds of machines, each sending hundreds of requests, the queue can easily be kept full for as long as the attacker wishes.

There are some defenses against this attack. The simplest is a technique called a "SYN cookie" in which the connection information is encoded in the SYN/ACK packet, without actually allocating the session until the ACK comes back, at which time the session is initiated. Even with this approach, however, a sufficiently massive attack can keep most systems busy responding to bogus requests. In fact, some variations on the SYN flood attack use precisely the "flood them with garbage" approach, whereby the packets sent to the victim are all in error in some way, forcing the victim host to spend time handling all these errors.

Since the source addresses are spoofed in these attacks, the responses are sent to the spoofed address, not the attacker. As a result, machines on the internet receive responses to queries they never sent. These unsolicited packets are called "backscatter," and by analyzing these backscatter packets it is possible to estimate various quantities related to the attacks. In particular, one can estimate the number of denial-of-service attacks of these types on the internet. See Moore et al. [2001], Marchette [2002], and Marchette [To appear].

The basic idea of backscatter analysis is to note that (most) attack software selects the spoofed addresses at random, and so a network sensor watching traffic on the network will see a random sample of the responses to the attack packets. By making reasonable assumptions about

the attack, one can estimate the number of packets in the original attack.

For example, assume that the attacker is selecting spoofed IP addresses from N total addresses (usually $N=2^{32}$, or the entire address space). Further, assume that the sensor monitors n IP addresses. Note that n is (generally) much larger than the actual number of machines on the network: it corresponds to all IP addresses that the network has available to it. Finally, assume that the attacker sends k packets to the victim. Given these assumptions, the probability of detecting the attack (observing at least one packet) is

$$\text{P[detect attack]} = 1\text{-}(1\text{-}p)^k,$$

where $p = \frac{n}{N}$ is the probability of observing a single packet. The probability of seeing j packets is the standard binomial

$$P[\,j \text{ packets}] = \binom{k}{j} (p)^j \ (1-p)^{m-j}$$

The maximum likelihood estimate for k is

$$\hat{k} = \left[\frac{j}{p}\right].$$

So, given our assumptions, we can estimate the size of the attack from the packets that we observe. More difficult problems, beyond the scope of this paper, are estimating the number of attackers, and estimating the number of attacks that we did not observe. If one can make an assumption about the rate at which the attack software sends packets, one can get an estimate of the number of attackers by considering interpacket arrival times. This assumes that the victim can keep up with the attack packets, sending a response for each packet received. Extreme value theory is relevant to the problem of determining the number of attacks missed. A crude estimate can be obtained by computing the expected number of attacks observed, given M attacks, and solving for M:

$$\hat{M} = \frac{\#\text{attacks observed}}{1 - (1 - \text{p})^k} \ .$$

Unfortunately, this has poor properties when the denominator is small (which is generally true in our case). One might also ask for the number of attacks (or the probability that there is an ongoing attack) in the event that no packets have been detected. De Haan and Sinha [1999] may be relevant here.

Visualization

Visualizing network data is a challenge due to the size of the data and the complexity of the data structure. As an illustration, we will look at a few techniques using a single hour of traffic. First, we can illustrate the basic TCP handshake and standard flag combinations in Figure 5 (p.15). Looking at the left plot, we see the three-way handshake, then a series of PUSH and ACKs, followed by the FIN sequence. Note that instead of sending a FIN packet, this session sent a FIN/ACK, thus simultaneously acknowledging the received packets and closing off the session. This is not uncommon.

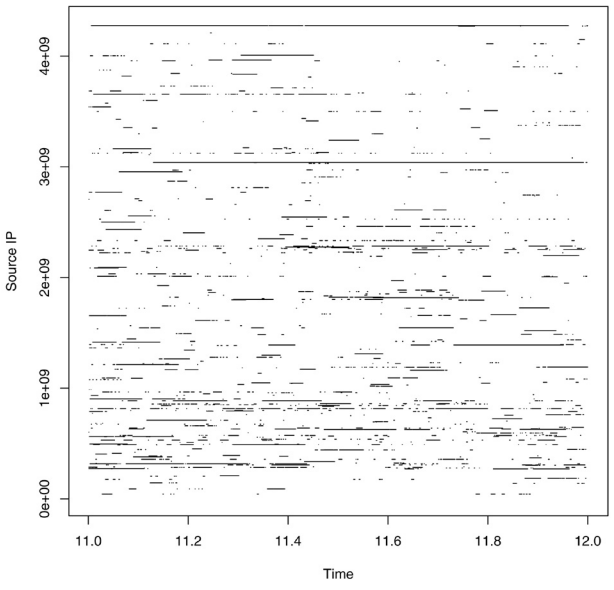

Figure 8. Some secure HTTP sessions. Each horizontal line corresponds to a single session, from the first packet to the last. The source IP addresses have been converted to 32-bit numbers using a standard network address mapping.

In Figure 5 (p.15), we see what happens when a user goes to a web site. Each image or page results in a separate session, and these are interleaved in time. Note that several sessions are started up at once, rather than waiting until the previous session finishes before the next one is started. This also shows the downside to this method of representing sessions: there is far too much "ink," and overlapping sessions are very difficult to interpret.

Simple exploratory data analysis and graphics can help one to better understand the data. In Figure 6 we have plotted destination IP address against source IP address for a single hour of data. The horizontal lines are indicative of scans, while the vertical lines are indicative of servers. By plotting the appropriate subsets of the data (where either SIP or DIP is constrained to be only those IPs in the protected network), one can quickly detect scans of the network, or machines which are unexpectedly acting as servers. In Figure 7, we see session duration plotted against session start time. The triangle is a result of censoring: since we are only looking at one hour of data, long sessions that start late in the hour are cut off at the end of the hour. Thus, we must take some care in our analysis to distinguish between completed and ongoing sessions.

Figure 8 shows another way of looking at sessions. Here we are only keeping track of the packets in time and the source IP address. The horizontal line segments show the session start and end time, plotted against time. What can

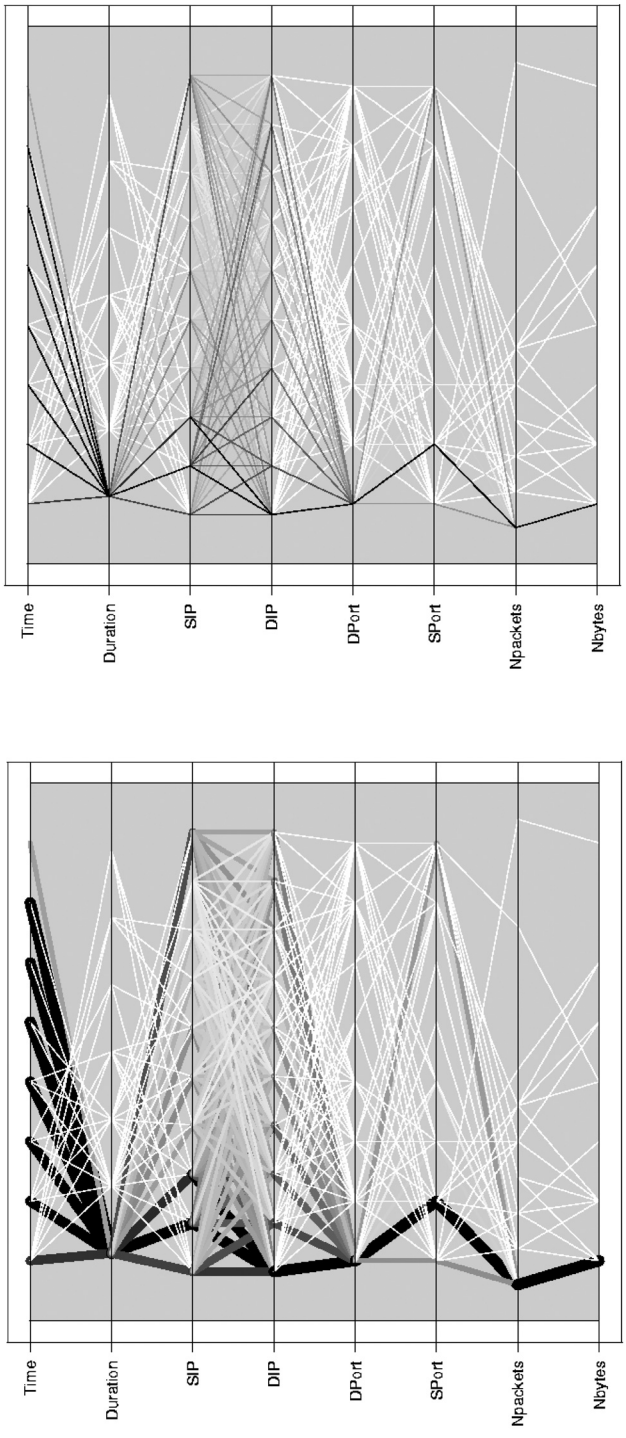

Figure 9. Two versions of parallel coordinates plots where the axes have been binned. The variables are start time, session duration, source IP and destination IP addresses, destination port, source port, number of packets in the session, and number of bytes transferred. For both the source and destination ports, the maximum bin corresponds to ports >10,000.

be seen in this plot is the large number of very small sessions, as well as the very large outliers. As with most network data, the duration of the sessions is known to be a heavy tailed distribution, since these data were taken over a single hour, and the sessions impinging on the boundaries are truncated, as shown in Figure 7. Note also that since we are not tracking destination IP addresses in this, we cannot tell if an IP has sessions with more than one destination. We might also be interested in the number of bytes transferred and the number of packets, making visualization of these data that much more difficult.

It is not easy to look at all the data at once, even when one reconstructs the sessions and plots only the session statistics. In the hour considered there were 135,605 sessions (where we are using the definition broadly, in the sense that any packet between two IP addresses is considered part of a session, even if it is not a legitimate packet). Figures 9 and 10 depict three versions of parallel coordinates plots of these data. In Figure 9, each variate is binned, and a line is drawn between the centers of two bins if there are observations whose corresponding variates fall in the bins. The darkness of the line indicates the number of such observations, with black indicating many and white indicating few. Darker lines are plotted last in this plot. The bottom plot is the same plot except that the line width also indicates the number of observations (wide is many, thin is few) and the darker lines are plotted last. Figure 10 shows the full dataset, unbinned, plotted using the saturation method of Wegman [Wegman and Luo (1997)], where regions of high overplotting receive a more saturated color. A cluster of sessions with similar destination ports has been brushed with a different color, allowing one to follow them throughout the plot. As can be seen, these correspond to (mostly) small duration sessions going to a subset of the destination IP addresses. A second view of these data is also provided in Figure 10, in which several clusters have been highlighted. Note that in regions of high overlap between the clusters the colors combine, resulting in different colors (yellow, purple, cyan, white) depending on which colors overlap. The two views (with or without binning) provide different information, and when the data volume is such that full-data plots are feasible, both should be considered. This illustrates the power of parallel coordinates for exploratory data analysis.

Several things are obvious in these plots. First, the sessions are almost all of short duration, yet there are some very long sessions (nearly an hour in length). Most of the destination ports are low-valued (corresponding to the common applications, most of which are on ports with value less than 1,024), and source ports tend to be larger than destination ports (due to being selected from values greater than 1,024). Most of the source ports are also low. The number of packets and number of bytes have quite long tails. One might guess that these would be correlated, and there is some evidence for this in the plots.

Source ports are typically chosen to have values greater than 1,024, but they are also usually selected sequentially. Figure 11 shows the source ports for sessions observed within the hour. Note that there is considerable linear struc-

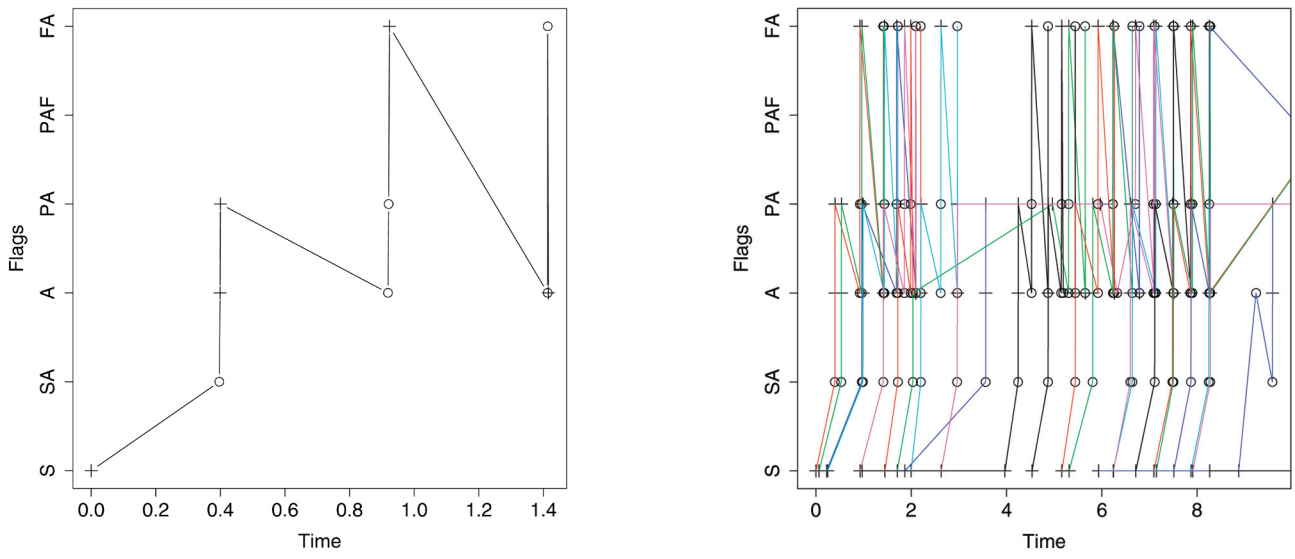

Figure 5. A "typical" TCP session (left) and several sessions in time order (right). Sessions are color-coded, with a repeating color scheme of seven colors. Different directions are plotted with "+" (client to server) and "o" (server to client).

Figure 10. (left) Parallel coordinates plot using color saturation to mitigate against over plotting. Black regions correspond to low density regions; brighter colors indicate high density regions. (right) Parallel coordinates plot using color saturation to mitigate against overplotting. Several clusters have been brushed with different colors.

ture in this plot, corresponding to a single source IP making many connections, each with a source port that is one larger than the last port chosen. Sessions or packet streams that deviate from this pattern are suspicious and likely to be scans or other attacks. (This advice must be taken with a grain of salt, since a machine with many sessions to other domains in between sessions to the protected network will not necessarily have sequential source ports.)

To investigate the issue of the relationship between the number of packets and the number of bytes transferred, consider Figure 12. In this figure we have plotted the total number of packets in a session against the total number of bytes transferred (in either direction), for a number of common applications. Obviously, there should be a correlation between the number of packets and the number of bytes. It is interesting that the slope of the line (even the number of lines) is different for different applications. Consider the plots for email and secure web in the figure. These have a fairly consistent slope, which is approximately the same, showing that they are both doing roughly the same thing: transferring data. The plot for telnet shows a much different picture.

In telnet, each character typed is sent as a single packet, and the responses are sent back in bulk. This results in a much different slope for the line than a straight data transfer program such as email. The web plot is very interesting.

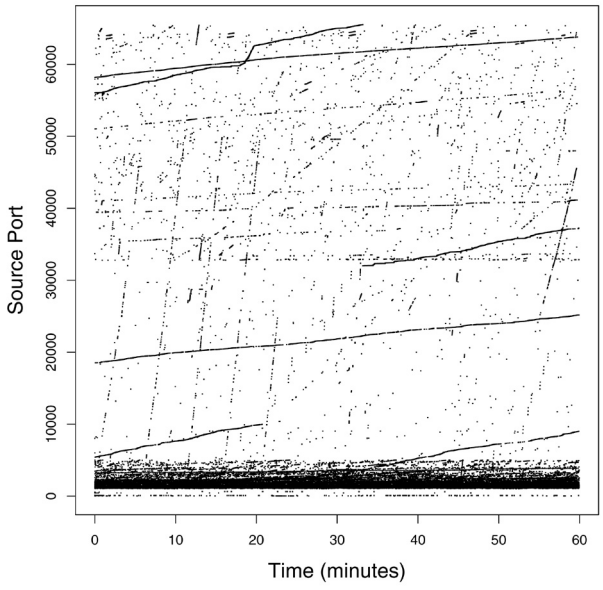

Figure 11. Source ports chosen for sessions within a one-hour period.

We see several distinct lines, indicating different average payloads. This is due in part to the semi-interactive nature of the application, and in part to attempts to optimize the throughput of the application. Subsequent analysis on larger datasets have shown that the other two "data transfer" applications also have this property; however, the number of sessions in this dataset was not large enough to show this for the email and secure web sessions.

This type of analysis is necessary for determining what "normal" sessions look like. An application might be to detect Trojan programs or worms that are using a legitimate port in a manner that is not typical for that port; for example, running a telnet session on the web port. In practice, this type of analysis must be done at a finer grain than the session level.

Another view of the data is shown in Figure 13. Here we have zoomed in to a section of the plot of duration against number of bytes. We see a "clump" of data indicating a group of sessions all with similar numbers of bytes and duration. These all correspond to sessions to a web server, in

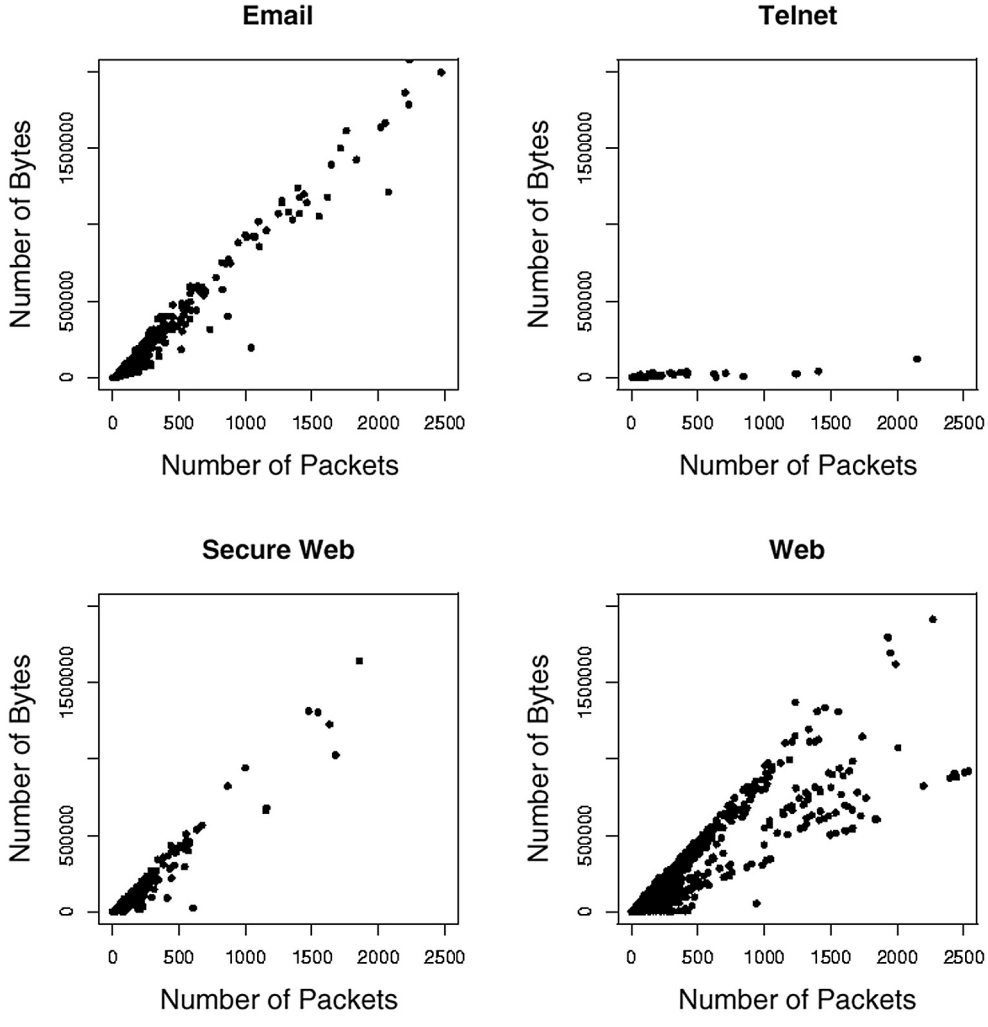

Figure 12. Number of packets plotted against number of bytes transferred per session for four different applications.

which several files (probably images) of approximately the same size were downloaded at once. This type of behavior could be an indication of a visit to a site with a large number of thumbnail images, which may or may not be an indication of potential misuse of computing resources.

Figure 14 shows the backscatter packets in the hour of data that we have been looking at. All unsolicited packets that are potential backscatter packets are displayed on the upper left, with source IP address plotted against destination IP address. Vertical lines correspond to potential scans of the protected network, or backscatter from very large attacks. We look at one such (plotted with triangles on the left) in the upper right plot. Here, we are plotting time against destination IP address to see if there is a pattern. An obvious pattern would be evidence that the packets are a scan of the protected network, instead of backscatter from a denial of service attack. As we can see, it is reasonable to assume that the destination IP addresses are random (a randomness test could also be applied to the data).

For attacks against a server, such as a web server, one expects that the backscatter packets should come from a single port, corresponding to the application being attacked. The attack in Figure 14 does not have this property: both the source and destination ports range from 1,024 to 1,279, typical ports chosen by the operating system when asked for a new unused port. The bottom plot in Figure 14 shows that these ports appear to be randomly chosen. Note that most operating systems choose these sequentially, so this is evidence that the packets have been crafted and are an attack of some type.

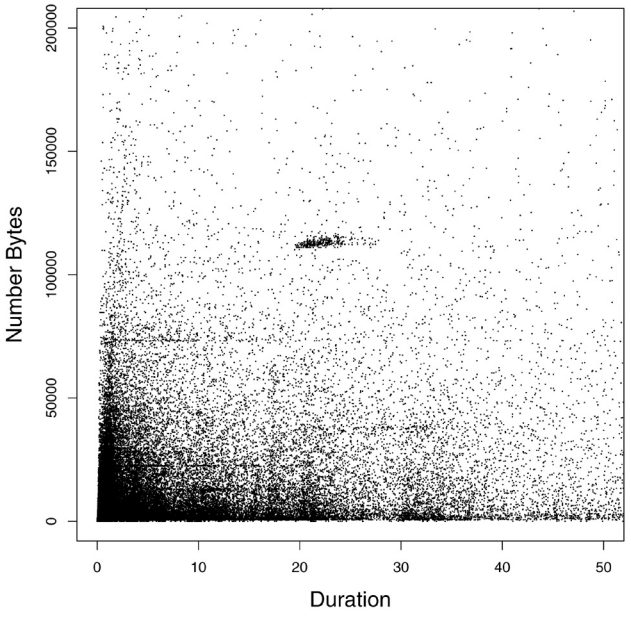

Figure 13. Duration against number of bytes, zoomed in.

Discussion

Network data is a rich area for statistical analysis, and we have only touched the surface in this paper. Many researchers are investigating network tomography, where information about the network is inferred from measurements taken at fixed sites [Vardi (1996)]. Another area of statistical interest is passive fingerprinting. In this, the operating system of the source is inferred by attributes taken from the packets sent by the computer. Certainly the people interested in network performance and routing make extensive use of statistical techniques and visualization. From a security perspective, the analysis of data on the host, such as log files, program execution, and user profiling, is another huge area for investigation.

We have only briefly mentioned the idea of profiling computers by their activity, an area that would benefit greatly from investigation by statisticians. Profiling activity is critical in detecting new attacks, Trojan programs, and covert channels. Most current intrusion detection systems rely heavily on signatures of known attacks, and perform poorly on detecting new attacks. Statistical analysis holds the promise of improving the performance of these systems by adding robust anomaly detection algorithms. This is a difficult task, as might be inferred from this paper, due to the nature of the data; in some sense, normal traffic is full of anomalies, and so domain knowledge must be incorporated to reduce the false alarms due to "normal anomalies."

Finally, there is a large body of knowledge about modeling epidemics that has only recently been used to model the spread of computer viruses and worms. These have slightly different properties than biological diseases, and yet much of the work done on these is relevant in the computer arena. Good models of propagation can be used to design defenses and to assess potential damage caused by these cyberattacks. Work needs to extend these models to distributed attacks, in which there may be several different methods and consequences of infection, depending on whether the infecting program is a master or slave.

References

Bishop, M. 2003. *Computer Security: Art and Science*. Addison Wesley, Boston.

Bolton, R. J., and Hand, D. J. 2002. "Statistical fraud detection: A review." *Statistical Science*, 17(3): 235-249.

De Haan, L., and Sinha, A. K. 1999. "Estimating the probability of a rare event." *Annals of Statistics*, 27(2): 732-759.

Marchette, D. J. 2001. *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint*. Springer, New York.

2002. "A study of denial of service attacks on the internet." In *Proceedings of the Army Conference on Applied Statistics.*

"Passive detection of denial of service attacks on the internet." In W. Chen, editor, *Statistical Methods in Computer Security*. Marcel Dekker.

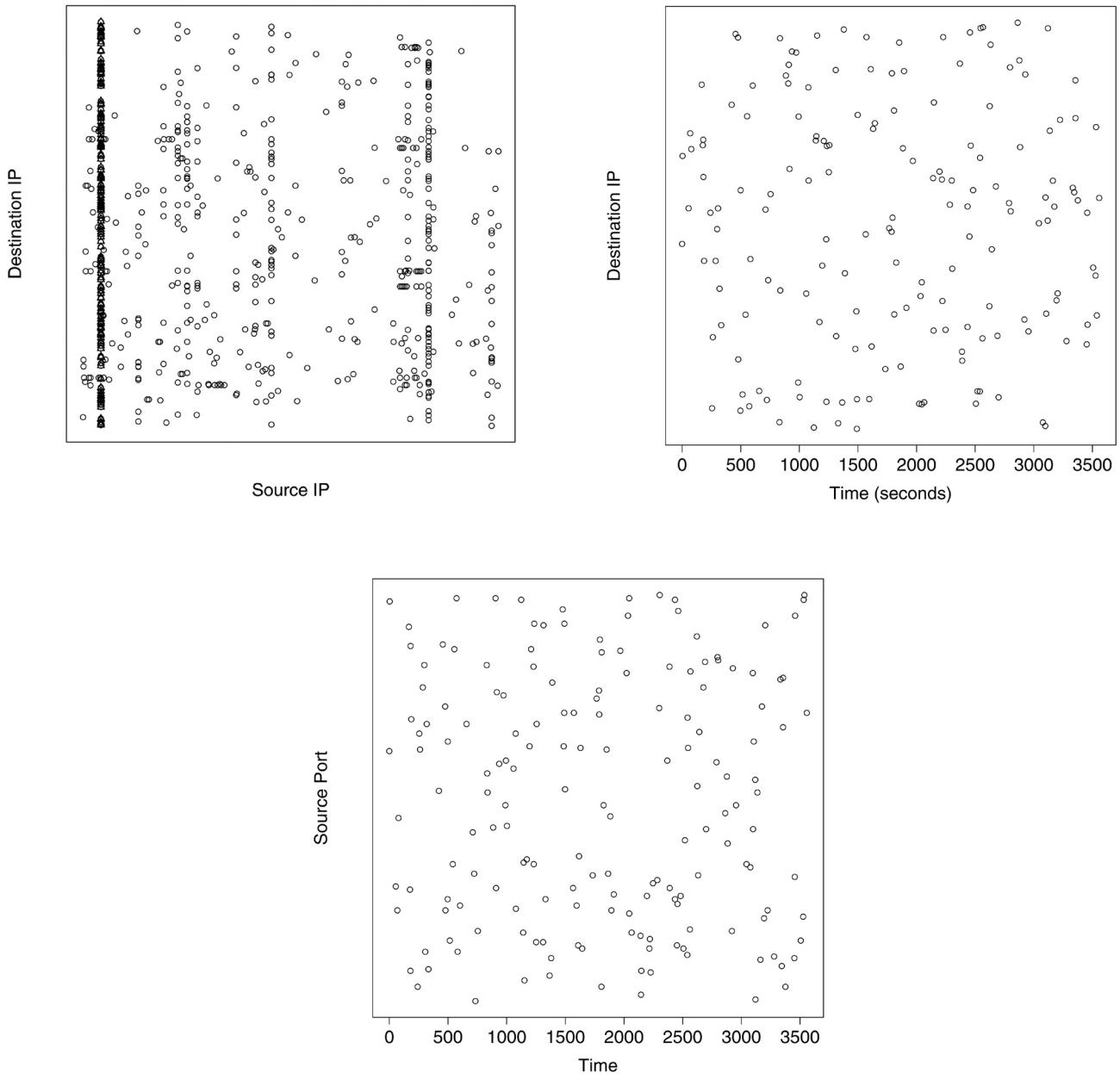

Figure 14. Potential backscatter packets. The upper left-hand plot depicts all the packets, with one source IP address represented with triangles. The upper right-hand plot shows the packets associated with this IP address. The bottom plot depicts source ports plotted against time.

Moore, D., Voelker, G. M., Savage, S. 2001. "Inferring internet denial of service activity." *www.usenix.org/publications/library/proceedings/sec01/moore.html*, USENIX Security '01.

Schonlau, M., DuMouchel, W., Ju, W., Karr, A. F., Theus, M., and Vardi, Y. 2001. "Computer intrusion: Detecting masquerades." *Statistical Science*, 16: 58-74.

Stevens, W. R. 1994. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, Reading.

Vardi, Y. 1996. "Network tomography: Estimating source-destination traffic intensities from link data." *Journal of the American Statistical Association*, 91(433): 365-377.

Wegman, E.J., Luo, Q. 1997. "High dimensional clustering using parallel coordinates and the grand tour." *Computing Science and Statistics*, 28:352-360. Republished in *Classification and Knowledge Organization*, R. Klar and O. Opitz editors, Springer-Verlag, Berlin, 93-101.