STATISTICAL ANALYSIS OF MALFORMED PACKETS AND THEIR ORIGINS IN THE MODERN INTERNET

A Thesis Presented To

The Faculty of the

Fritz J. and Dolores H. Russ
College of Engineering and Technology

Ohio University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Marina Bykova

March 2002

THIS THESIS ENTITLED

“STATISTICAL ANALYSIS OF MALFORMED PACKETS AND THEIR ORIGINS IN THE MODERN INTERNET”

by Marina Bykova

has been approved

for the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology

Shawn D. Ostermann
Associate Professor of Computer Science

Jerrel R. Mitchell, Interim Dean
Fritz J. and Dolores H. Russ
College of Engineering and Technology

BYKOVA, MARINA. M.S., March, 2002
Electrical Engineering and Computer Science

Statistical Analysis of Malformed Packets and Their Origins in the Modern Internet

Director of Thesis: Shawn D. Ostermann

With the tremendous growth of Internet resources, we observe a rapid increase in the number of network applications and protocol implementations, which are not always thoroughly evaluated and tested. A growing number of network attacks attempt to disrupt legitimate communication or deny access to network resources to legitimate users. Both poor implementations and intentional abuse of network resources "pollute" a network with malformed packets and can become a threat to sound communication. In this work, we collect and analyze all of the IP and TCP headers of packets seen on a network that either violate existing standards or should not appear in modern internets. Our goal is to determine the reason that these packets appear on the network and evaluate what proportion of such packets could cause actual damage. Thus, we examine and divide the unusual packets obtained during our experiments into several categories based on their possible cause, which ranges from errors in network implementations to carefully constructed attack packets, and show the results. The traces analyzed were gathered at two different data sources at Ohio University — the university's main Internet link connecting it to its ISP and a local network with student dormitory traffic — and provide a massive amount of statistical data.

Approved:

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

With the tremendous growth of Internet resources, we have observed a rapid increase in the number of network applications and protocol implementations that utilize these resources. These implementations are not always thoroughly evaluated and tested. Consequently they might not provide the best service possible, or may even inadvertently interrupt other communications underway.

Measurements performed in the networking area usually gather information about the functionality and performance of widely implemented protocols or their enhancements. Correctness of the implementations is left to protocol implementors and users. Poorly coded, some network programs might "pollute" a network with malformed packets and disrupt sound communication.

While implementors of network applications or lower level programs probably do not intend to introduce mistakes in their code and rather try to eliminate them, the problem of service disruption becomes more evident with intentional abuse of network resources. A growing number of network attacks attempt to disrupt legitimate communication or deny legitimate users access to network resources. The amount of efforts dedicated to intrusion detection and network security has grown dramatically in recent years. In particular, we see growth in the number of network intrusion detection systems (IDSs) that try to protect systems from unauthorized access and an increase in the number of ways they defend those systems.

The majority of IDSs available to date — commercial, open-source, or research projects — are signature-based misuse detection systems (see [1] for complete listing of IDSs). They detect only well-known attacks based on attack signatures that must be in place before these systems can detect them. New, unknown attacks are generally a weak point in system protection and are often difficult to recognize. In order to be able to catch novel attacks, a different approach termed “anomaly detection” can be taken. Anomaly detection, however, remains challenging even today.

Both poor implementations and intentional abuse of network resources have one thing in common — they can become a threat to sound communication not only for the communicating hosts themselves but also for other machines that rely on services provided by the global Internet. A logical place for such disruption to originate is the network protocols. By definition, each protocol implies a set of rules that every participant agrees to follow in order for communication to work. Should a participating host break the rules or provide misleading information, not only correct operation of the protocol can be guaranteed but the response behavior of the receiving end might also be hard to predict.

Control information that is necessary for the correct performance of network protocols is carried in their headers. Thus it becomes possible to detect abnormalities in packets seen on a network by checking the information carried in their headers. In this work, we collect and analyze all of the IP and TCP headers of packets seen on a network[1] that either violate existing standards or should not appear in modern internets. Such packets could be the result of an inaccurate implementation, misconfiguration, malicious activity (including new, unknown attacks), or could have another origin. The questions we try to answer here are as follows.

- What is the reason that these packets appear on the network?
- How often do we see them?
- What proportion of such packets could cause actual damage?
- What can be done in order to reduce the rate of malformed packets on a net-

[1] We also provide limited analysis of UDP packets.

work?

We examine and divide the unusual packets obtained during our experiments into several categories based on their origin and possible cause, which might range from errors in network implementations to carefully constructed attack packets, and show the results. We do not try to identify all network attacks, but rather try to determine how much information we might be able to obtain by looking at packet headers while ignoring their contents.

Chapter 2 provides a description of the experiment, which includes: the links monitored, tools used, the types and amount of data analyzed, and the analysis performed. Chapter 3 covers the results obtained from our experiments. We divide all of the errors that the system recorded into logical categories, provide detailed analysis of all of the categories, summarize statistical results, and give various distributions of error rates. Lastly, Chapter 4 summarizes our findings and also contains suggestions on improving the security of a site based on those findings.

2. DESCRIPTION OF THE EXPERIMENT

2.1 Link Description

In order to perform analysis for this research, we used data captured from two different sources. For the first source, we monitored Ohio University's main Internet link — the only link in and out of the university — including both incoming and outgoing packets. These traces were obtained from a 36Mbps Fast Ethernet connection between Ohio University (OU) and its ISP and carry packets for approximately 20,000 local hosts.

The second source of data was a 10Mbps Ethernet LAN carrying student dormitory traffic. There were approximately 2,500 computers directly connected to this network. Parts of this network were switched, but we could still see a large amount of TCP traffic and all broadcast packets traversing the link. The characteristics of the traffic on this link are different from traffic coming to and leaving the University and thus the two were analyzed separately. We were able to use computer hardware addresses to aid in the analysis of the data set from the local network since the packets were captured on the link of their origin.

2.2 Tools Used

We used `tcpdump` [10] to capture data from the monitored links and `tcptrace` [16] in real-time mode to analyze it. We modified `tcptrace` for the purpose of this analysis and wrote a special module for it. The module logged all packets that were considered abnormal along with other information retrieved from `tcptrace`. Further analysis and calculation of statistical results were performed with Perl and shell scripts.

2.3 Packet Analysis

Analysis of Ohio University's main link shows that almost all packets on the link are IP packets and the great majority of those are TCP packets. UDP traffic comprises approximately 2 percent of the monitored traffic, and IP packets that are neither TCP nor UDP make up an even smaller portion of the total number (less than one percent).

The ratio is different for the local monitoring point. UDP traffic makes up a much larger portion of packets, ranging from 5% to 90% in different trace files. Most of this UDP traffic belongs to NetBIOS services [14], [15] or communication on port 427 (Server Location, according to IANA assignment [24]). Packets that are neither TCP nor UDP comprise 1–2% of total traffic on average. Even though the portion of non-TCP traffic is rather high compared with the data on the global link, TCP traffic still comprises the greatest portion of traffic (about 60% of packets from all traces) on this link.

Our analysis is based on the IP and TCP headers of packets from the monitored traffic. We perform a detailed IP header analysis of all of the IP packets regardless of their transport layer protocol, analyze the headers of all TCP packets, and also conduct some statistical analysis of UDP packets. UDP, ICMP and other types of packets could also be used in security breaches but their analysis is out of scope for this work. We take into consideration only IP version 4 packets even though a small number of IP version 6 packets were present in the packet traces we captured. This section describes all of the packet header fields that we included in the analysis and illustrates what values we considered abnormal.

2.3.1 IP Header Analysis

With the wide spread of the Internet, the Internet Protocol (IP) has been used extensively — and not always in the way its designers intended. Setting some of the IP header fields to improper values might disrupt operation of the protocol making such packets undeliverable, while other fields can be modified to harm the destination host without any punishment to the sender. In this subsection we take into consideration those IP header fields that might violate the protocol requirements as they are seen at the receiver.

1. **Packet Size**. The IP header length should always be greater than or equal to the minimal Internet header length (20 octets), and a packet's total length should always be greater than its header length [20]. If either of these statements do not hold for a given packet, it is invalid and should be discarded at the destination host. IP packets that carry transport layer protocols known to the system (currently TCP and UDP) are also checked to confirm that they are large enough to hold the entire header of the next layer protocol.

2. **IP Checksum**. This field allows detection of corrupted packets and thus packets with bad checksums should be discarded. This checking can be particularly useful for packets that have other IP standard violations to help detect corrupted packets and adjust the results accordingly. For instance, if a packet with an invalid checksum has illegal values in other fields, the system should not trigger an error and the packet is to be excluded from analysis under assumption that it was corrupted.

3. **IP Address**. Values of the IP address field can violate the standard in several different ways. First, the IP address field is unprotected from spoofing (i.e. substituting it with an IP address that does not belong to the sender) and the source address extracted from a single packet can not be easily verified. Source address spoofing becomes harder with protocols that maintain a connection and have state, such as TCP, but it is still possible with, e.g., source routing [20].

   The problem of determining the validity of source addresses can not be easily solved when access to a network is unrestricted and a monitor sees both incoming and outgoing traffic. However, a number of addresses that are certainly invalid can still be identified. Prior literature contains examples of network attacks that use the same source and destination IP addresses, such as the so called "land attack" described in [25], [8]. Thus we verify that the source address of every packet is different from its destination address.

Another category of invalid addresses is private internet addresses [23]. Private addresses are invalid in public domains and should be filtered out by the routers connecting private networks to the larger Internet. Our experience, however, shows that a number of packets containing private addresses do exist in the public domain. Empirical results from the Lawrence Berkeley National Laboratory's (LBNL) network reported by Bro confirm this fact as well [17]. We check both IP source and destination addresses versus all types of private addresses.

Thirdly, there are certain special cases of IP addresses that can not be used as either source (broadcast), destination ("this network"), or either kind (loopback) of address on a public internet [24]. Many of them are based on the definition of "network number" and "subnet number." The difficulty in detecting these types of internet addresses arises from the variable length of network prefixes, and in general we do not know the network prefix length for any given IP address. In our analysis we do look for the special cases of IP addresses but record only those packets that clearly belong to one of these special cases. For example, we know that the destination network number cannot have a value of zero. We verify that at least the most significant byte of the destination network is not zero since this byte is always either the network number itself or a part of it, depending on the network prefix length.

Lastly, we verify that at least one of the source or destination IP addresses belongs to the Ohio University address space since all packets on the monitored links are expected to come to or from the university.

4. **'Time to Live' (TTL) Field**. We know that packets with small TTL values do

not violate any current standard. However, aside from being used for legitimate reasons, they can be a precursor to or a part of a network attack.

Low TTL values appear in legitimate packets (a good example is limited broadcast packets where standards recommend keeping TTL values small). A number of packets with small TTL values can also be caused by routing loops, although such cases are relatively easy to recognize. Other packets could be a result of the usage of `traceroute` [9].

On the other hand, the TTL field can be used by an attacker to explore the topology of a remote network [7]. When attempting to map a topology, a combination of `traceroute` attempts can provide a good picture of the network. In most cases, however, it is impossible to determine the reason for which `traceroute` was used.

Low TTL values can also be used in subtle attacks that try to subvert a monitor. Bro [17] has a detailed description of such attacks in which an attacker sends packets with small TTL values and retransmits the same packets with different data and a larger TTL so that only the retransmitted packets will reach the destination host. If the monitor does not check TTL values of the packets that traverse the link, it will not be aware of the fact that the original packets do not reach the destination host. In that case, the data the that monitor sees is different from the data that the destination host receives.

Our preliminary results showed that it is very difficult to determine the cause of packets with small TTL values, in particular the reason why `traceroute` may have been used. In order to make any conclusions, additional information is needed, and thus we exclude packets with small TTL values from the final analysis. This should be an area of future study.

5. **IP Options**. The IP Options field is difficult to abuse because implementations are supposed to discard unknown options if any are present in a packet [20].

We believe that there are still certain IP options that generally should not be present in IP packets. Truncated options also should not appear in valid packets, as they indicate that the entire IP header is not present in one packet, thus making the packet invalid. Checking for truncated options can be easily performed by comparing the IP header length of a packet with the packet's size.

From all available legitimate IP options we look only for the source routing option (both strict and loose) [20], because it is to be used only for debugging purposes and typically should not appear in modern internets. There are a number of attacks that use strict source routing together with a spoofed source IP address to enable an attacker to receive responses and establish forged communication with the target host (see [8] for a description of such attacks). Other IP options, at the time of this writing, are not known to have potential to harm the destination.

6. **Overlapping Data**. Overlapping fragments in which all fragments do not agree on the contents of the overlapped region and retransmitted packets that carry different data from the original transmission always violate protocol specifications and should generate alarms. Several IDS implementations ([17], [25]) report such cases. Even though our research does not take into account the content of packets but operates only on packet headers, it is possible in some cases to detect overlapping packets with differing content by using their checksums.

   In our implementation, we ran into a problem of heavy system load on one of the monitors. A number of the traces obtained from this monitor had a substantial number of missing packets (or "holes"), which could lead to invalid results for this particular part of the analysis. Thus, checking for overlapping packets with different content is not included in our study.

2.3.2 TCP Header Analysis

The values one can place in the TCP header fields are more restricted than in the IP header due to the connection-oriented nature of TCP. A single TCP packet is not, however, required to belong to an existing connection to be processed by the receiver in some way (this is widely used in denial-of-service (DoS) attacks). A malicious user might also intentionally try to establish a forged connection with the target machine. Our goal is to describe inappropriate values that might be placed in the TCP header fields and detect such violations as they appear on the monitored networks.

1. **Packet Size**. Unfragmented IP packets are required to be large enough to hold an entire TCP segment. According to the IP and TCP specifications ([20] and [21] respectively), the IP and TCP header lengths can not exceed 60 octets, while all implementations are required to accept and recommended to sent IP datagrams of at least 576 octets long. This guarantees that every unfragmented IP datagram contains control information for the network and transport layers in full.

   In the case of fragmentation, the data portion of the first IP packet of a fragment set containing a TCP segment should be large enough to hold an entire TCP header. If a TCP header includes many long options, then some of them may not be included entirely in the first IP packets and be continued in the next IP packet of the fragment set. However, the required part of the TCP header (20 octets) is normally present entirely in one IP datagram. Splitting TCP headers is sometimes used to pierce firewalls, so we check for fragmented headers.

2. **TCP Checksum**. Many packets with correct IP checksums have invalid transport layer checksums and should be discarded at that layer. Verification of TCP checksums can be useful in several ways.

   First of all, invalid TCP checksums might be used in subtle attacks where an attacker is aware of the presence of a monitor between them and the victim machine and tries to convey their activity undetected (the attack was previously described in the Bro paper [17]). In such attacks, an attacker sends a packet with an invalid checksum and resends it later with the correct value. If the monitor does not verify the checksums of packets traversing the link, it might see different data than the destination host.

Second, checksum verification can be useful for packets that already have other TCP violations (for example, invalid combinations of TCP flags) to determine whether the packets should be taken into consideration. If the checksum is invalid, they should be discounted on the assumption that they are corrupted and the results adjusted accordingly.

Finally, processing large amounts of data, we can use TCP checksum verification for purely statistical purposes and provide numerical conclusions about the rate of corrupted packets in the Internet.

We verify TCP checksums of the captured packets whenever possible. Not all of our packet traces include entire packets and a number of packets from the traces are truncated. Truncated packets were excluded from the statistical analysis of corrupted packets.

3. **Port Numbers**. Virtually any combination of source and destination port numbers can be valid. The only obvious exception to this rule is the reserved number zero. Neither the source nor destination TCP port number can be zero according to the current standard [21]. We record all packets where either one of these two port numbers is equal to zero after verifying that the packets themselves are valid whenever possible (i.e. their checksums are correct).

4. **TCP Flags**. The TCP flags occupy six bits in the TCP header, and only a few combinations of those six flags can be carried in a TCP packet. According to the TCP standard [21], URG and PSH flags can be used only when a packet carries

data. Thus, for instance, combinations of [SYN and URG] or [SYN and PSH] become invalid. Moreover, any combination of more than one of SYN, RST, and FIN flags is also invalid[1]. Illegitimate combinations of TCP flags are known to be used in so-called "Xmas Tree" scanning and operating system detection techniques [6]. We check whether a TCP packet has a valid combination of flags and any protocol violations are reported by the system.

5. **Reserved bits**. The original TCP specification reserves six bits in the TCP header for future use. More recent extensions to TCP [22], [11] utilize some of those bits. However, those extensions are not deployed yet and are mostly experimental documents at the time of this writing[2]. Setting the reserved bits to an arbitrary value might harm poor TCP implementations, therefore we check the reserved field in the TCP header and analyze all non-zero cases.

6. **Acknowledgments for never-sent data**. It would be useful to detect acknowledgments for packets that were never transmitted to defend against incorrect implementations or malicious users. Bro [17] is known to report such cases but we currently do not have statistics for such violations and do not include this in the analysis. Again, the reason is that a number of our packet traces had holes in them, which could substantially affect or even invalidate our results, decide we to perform such detection.

2.3.3 UDP Header Analysis

The main emphasis of this thesis is on TCP packets and thus we do not perform extensive analysis of transport protocols other than TCP. However, we collect information about invalid UDP checksums for statistical purposes.

[1] According to the T/TCP RFC [3], a packet that includes both SYN and FIN flags might be valid if it carries a CC or CC.NEW option. In our analysis, we take into account these options even though the implementation of T/TCP is experimental and is not a current standard.

[2] Explicit Congestion Notification (ECN) [22] has since become a proposed standard, but was not so at the time of capturing our packet traces.

2.4 Analyzed Data

During our experiments we analyzed traces gathered November 2000 through June 2001 at different times of the day on two links described above. Each trace file consisted of several million packets and the total number of analyzed packets totaled over 300,000,000. Traces gathered on different links are analyzed separately due to the different nature of the traffic traversing the links. The total number of reported warnings over all of the analyzed data was approximately 300,000, with over 75% of them coming from one trace file.

3. RESULTS

This chapter provides a detailed analysis of the results obtained from our experiments and a description of all of the types of errors generated by the system. We also provide statistical results and error rate distributions later in this chapter. All errors recorded after analyzing the packets on the global link are summarized in Table 3.1, and errors triggered by packets from the local network are shown in Table 3.2. The "Packets" column in these tables represents the number of unique packets that generated errors, and the "Warnings" column shows the exact number of errors that the system recorded, with possibly more than one error per packet. It can be seen that the system did not observe all known types of violations and did not generate all possible types of errors, which tells us either that the amount of data analyzed was not large enough to detect such packets and calculate their rate or that they do not exist in large numbers on the Internet.

The nature and content of the traffic from the two monitored links differs substantially, which directly influences the number and type of errors obtained from each link. We performed analysis of the packets from each link separately because of this. Every subsection in this chapter is divided into two parts — one for the global and local links accordingly — where we describe and categorize the errors on a particular link. Table 3.3 shows the total number of packets analyzed and the number of errors generated for both links for comparison. Note that the error rate on the global link reflects a more realistic number than the error rate on the local link. The latter number is greatly influenced by the number of packets where neither source or destination address belongs the OU address range — the number of such packets reaches 98% of

Table 3.1 Errors Detected on Global Link

Proto	Type	Packets	Error %	Warnings	Error %
IP	Private IP addresses	13,830	22.20%	16,199	24.99%
	Out of OU range IP addresses	283	0.45%	283	0.44%
	Other IP address violations	280	0.45%	280	0.43%
	Improper IP options	0	0.00%	0	0.00%
	Too short IP packets	0	0.00%	0	0.00%
	Same source and destination IPs	0	0.00%	0	0.00%
TCP	Invalid TCP flags	196	0.31%	196	0.30%
	Zero port number	136	0.22%	136	0.21%
	Non-zero reserved bits	1,047	1.68%	1,221	1.88%
	Too short TCP packets	0	0.00%	0	0.00%
	Invalid TCP checksums	46,466	74.60%	46,466	71.67%
UDP	Invalid UDP checksums	49	0.08%	49	0.08%
	Total number	62,287	100.00%	64,830	100.00%

all errors obtained from the link. More detailed analysis of these packets is provided later in this chapter.

We will start with a detailed analysis of the packets that triggered warnings during the checking of the IP header, then proceed with the analysis of TCP-based errors, and finally provide checksum statistics.

3.1 IP Analysis

Tables 3.1 and 3.2 show that we obtained quantitative results for only three types of IP abnormalities out of the six known to the system. In this section, we provide

Table 3.2 Errors Detected on Local Network

Proto	Type	Packets	Error %	Warnings	Error %
IP	Private IP addresses	2,703	1.08%	3,708	1.48%
	Out of OU range IP addresses	244,833	98.00%	244,833	97.59%
	Other IP address violations	0	0.00%	0	0.00%
	Improper IP options	0	0.00%	0	0.00%
	Too short IP packets	0	0.00%	0	0.00%
	Same source and destination IPs	0	0.00%	0	0.00%
TCP	Invalid TCP flags	51	0.02%	51	0.02%
	Zero port number	6	0.00%	6	0.00%
	Non-zero reserved bits	61	0.02%	108	0.04%
	Too short TCP packets	0	0.00%	0	0.00%
	Invalid TCP Checksums	2,178	0.87%	2,178	0.87%
UDP	Invalid UDP Checksums	6	0.00%	6	0.00%
	Total number	249,838	100.00%	250,890	100.00%

Table 3.3 Packets Analyzed on Both Links

Type	Total Packets	Errors	Error Rate
Global Link	247,873,366	62,291	0.025%
Local Network	54,696,049	249,838	0.457%

detailed analysis of only those types of errors that produced results.

3.1.1 Private IP Addresses

Currently Ohio University utilizes a few private networks for internal departmental use and private communication between certain types of hosts. These networks use the class A private IP addresses (10.0.0.0/8) and are typically protected with firewalls that perform IP address translation (NAT) [26]. Ohio University is not known to use the other ranges of private addresses and has not done so during recent years. Therefore packets destined to private IP addresses other than these class A addresses could not be caused by old configurations left from previous address schema. This allows us to assume that packets carrying private IP addresses other than 10.0.0.0/8 have some other origin.

3.1.1.1 Global Link

Results obtained during our experiments at the global link showed a large number of packets sent either to or from private IP addresses. Moreover, the logged packets contain IP addresses that belong to all classes of private networks. The distribution of these packets containing private IP addresses by address ranges and the type of address which is private — source, destination, or both — is shown in Table 3.4. Note that the total number of packets that fall into each address range represents the number of packets that triggered such warnings and is not necessarily equal to the sum of packets going to and from private addresses from that address range.

We found that packets destined for private IP addresses are sent by various Ohio University hosts that run different operating systems and have different configurations. Thus the presence of such packets can not be wholly explained either by errors in implementation or by improper default configuration of a certain operating system. These packets do not belong to the Ohio University IP address space and therefore tend to leave the domain. They are normally eventually blocked by router rules or discarded either due to absence of routes for such IP addresses or after the maximum

Table 3.4 Distribution of Packets Containing Private IP Addresses by Address Type on Global Link

Private IP Address Range	From	To	Both	Total
Class A private IP addresses (10.0.0.0/8)	2,489	4,661	2,263	4,887
Class B private IP addresses (172.16.0.0/12)	10	3,682	0	3,692
Class C private IP addresses (192.168.0.0/16)	523	4,834	106	5,251
Total number	3,022	13,175	2,369	13,830

number of hops is reached.

Packets with private source addresses can occur for several reasons. Some of these packets could have come from hosts with private IP addresses legitimately assigned to them and are seen by the monitor due to errors in router software or configuration. Another reason is that the hosts could have been unable to obtain legitimate IP addresses or used the private addresses by mistake. It is also possible that some packets could have come from hosts that spoofed source addresses. Unfortunately, it is impossible to obtain the real source addresses of those packets on the global link without applying special IP traceback techniques. All packets seen on the link come from the routers connected to the link and therefore lose their original hardware layer information. We determined, however, that the majority of the packets with private source addresses were sent to private destination addresses as well (normally to their corresponding directed broadcast address), which makes the probability of address spoofing smaller. The purpose of address spoofing is to eliminate the possibility of the attacker's machine to be disclosed, and if the destination IP address is private, i.e. there is no potential victim host, address spoofing is not likely to take place.

The distribution of packets with private IP addresses by protocol and packet type is shown in Table 3.5. One can see that the majority of all packets containing pri-

Table 3.5 Distribution of Packets Containing Private IP Addresses by Packet Type on Global Link – This table includes all types of addresses being private — source, destination, or both, — and all ranges of private IP addresses listed in Table 3.4.

Proto	Type	Packets	Percent	Total	Percent
ICMP	Echo Request	1,338	9.7%	1,512	10.9%
	Host Unreachable	92	0.7%		
	Time Exceeded	62	0.4%		
	Port Unreachable	16	0.1%		
	Other	4	0.0%		
UDP	NetBIOS Name Service	3,022	21.9%	8,142	58.9%
	DHCP	2,041	14.8%		
	DNS	407	2.9%		
	Other	2,672	19.3%		
TCP	No data	4,094	29.6%	4,175	30.2%
	With data	81	0.6%		
Other		1	0.0%	1	0.0%
Total		13,830	100.0%	13,830	100.0%

vate IP address are UDP packets. A very large number of them appear to belong to NetBIOS name service traffic and target private IP addresses. The machines sending these packets obviously do not get any response and could have been misconfigured or taken an incorrect default value. The next largest group of UDP packets are DHCP packets broadcast from private addresses to the directed broadcast address[1]. All of them appear to be unsuccessful attempts to locate a DHCP server. DNS traffic also comprises a large number of the erroneous packets that come from private IP ad-

[1]According to the DHCP specifications [4], DHCP messages broadcast by a client prior to obtaining an IP address from a DHCP server must have the source IP address set to 0.

dresses. These packets are failed attempts to connect to a DNS server. All of the remaining UDP packets are grouped as "other." This group is rather large and consists mostly of packets sent to or from private addresses on high port numbers. Some of them (those sent to routable addresses) triggered ICMP PORT UNREACHABLE messages [19] in response.

TCP traffic also comprises a large portion of the packets containing private IP addresses. The majority of them are attempts to establish connections with hosts having private IP addresses, but other types of packet are also not uncommon. The traces include a lot of single FIN, RST, and random data and ACK packets. Some of the packets sent from private IP addresses triggered RST packets in response. The most commonly used port numbers in the SYN packets are 80 and 139 (HTTP and NetBIOS, respectively). A large number of packets from this category (about 95% of all of the TCP packets) are generated locally and leave the domain. They are either attempts to connect to machines with private IP addresses from local IP addresses or try to reach global non-OU addresses and come from private addresses. The remaining 5% of these packets come to the university from private IP addresses. We consider packets coming to our network from unroutable addresses more suspicious than packets generated locally.

The third largest group of errors in this category belongs to ICMP traffic. The majority of these ICMP packets are ECHO REQUESTs sent to a host with a private IP address with no response (note that there are no ICMP ECHO REPLY packets corresponding to these request packets in the traces). Since the pings come from valid, routable IP addresses and target non-existent hosts, the probability that they were sent by a malicious user is slim. More likely, they could have been caused by misconfiguration. Other ICMP types present in the log files are as follows:

- HOST UNREACHABLE messages (sent from private IPs and reporting unavailability of regular, non-private addresses. We suspect they could have been sent by routers that use private IP addresses on one of their interfaces and were

Table 3.6 Distribution of Packets Containing Private IP Addresses Captured on Local Network

Physical Layer Address	Network Layer Address	Packets	Percent
Local to Local	Private to Broadcast	2,595	96.00%
Gateway to Local	Private to Local	69	2.55%
Local to Gateway	Local to Private	39	1.44%
Total		2,703	100.00%

not able to route the packets beyond that link);

- UDP PORT UNREACHABLE messages (reporting unreachable ports on machines with regular IPs and sent in response to packets coming from private IP addresses);
- TIME EXCEEDED messages (similar to HOST UNREACHABLE errors, sent from private IP addresses to Ohio University hosts).

3.1.1.2 Local Network

The distribution of packets containing private IP addresses that were captured on the local link is shown in Table 3.6. The first and largest category of these packets does not contain global IP addresses at all — the packets are sent from private to directed and undirected broadcast addresses. Since private IP addresses are valid on a local network, we leave this category and do not try to explore the cause of these packets. Note, however, that these broadcast packets did not trigger a response and thus are not likely to belong to valid connections. The two other categories are more valuable for our research, and they are further subdivided and shown in Table 3.7.

- **Case 1:** The first category includes UDP NetBIOS name service request packets sent from a private IP address and ICMP PORT UNREACHABLE messages

Table 3.7 Categories of Packets with Private IP Addresses Violations Captured on Local Network

Type	Number	Possible Cause
UDP NetBIOS NS packets and responses to them	6	Misconfiguration
ICMP packets	10	Misconfiguration
Strange TCP packets associated with port 80	78	Poor implementation
Unexpected TCP packets	14	Varies
Total	108	

sent in response. We believe that the packets in this category are not likely to have been sent by a malicious user since their number and rate are not high and their configuration looks typical. Also, the packets could not have been sent expecting a reply since the source IP address is not routable on the global Internet. Thus, we conclude that they were caused by misconfiguration on the sender.

- **Case 2:** All ICMP packets were included in one category. We captured several types of different ICMP packets with private IP addresses. Most of them are ICMP ECHO REQUEST packets sent from private addresses and replies to them. We also saw ICMP TIME EXCEEDED and SOURCE QUENCH packets.

  It is very difficult to determine the reason why we see such packets due to the stateless nature of ICMP. However, we believe that the sender did neither benefit from nor gain any information by sending these packets, and their number is not large enough to harm the destination[2]. Therefore we believe that the

[2] This group includes one ICMP SOURCE QUENCH packet sent form a private address. Even though SOURCE QUENCH packets may potentially be used in DoS attacks and substantially reduce

packets were caused by improper configuration or mistaken usage of private IP addresses.

- **Case 3:** This group includes TCP packets sent on port 80 from private addresses as well as packets sent in response to them. We analyzed the traffic associated with the hosts that received such packets and determined that all of them had connections with external servers on port 80 at the time of the above mentioned packets were captured. The packets originated from the private addresses would be expected to come from globally routable IP addresses (the destination IP address, both TCP port numbers, and the sequence number matched existing connections) but were sent from private addresses instead. There are two types of such packets — RST packets sent by the server after the client resets the connection and ACK packets coming from the server to acknowledge the second FIN and completely end the connection. The ACK packets triggered RST packets since there was no connection associated with the private IP address.

  The two types of these packets have one thing in common — they appear after either two FIN packets or a RST packet; i.e. when a connection can be considered closed. We found four different web servers that issued such packets. The web servers closed other connections correctly and transmitted packets with private IP addresses only in some rare cases. Therefore we suspect that the cause of the problem is either poorly written software running on these hosts or errors in software that performs address translation of corporate web servers behind firewalls.

- **Case 4:** The last group combines all of the remaining packets, which happened to be unsolicited TCP packets coming from private IP addresses. They are not

the rate at which the victim host sends packets, we believe that the only SOURCE QUENCH packet that our system recorded could not affect the rate at which the destination host sends packets dramatically even if the packet was formed correctly.

regular data packets but rather SYN, SYN ACK, or FIN packets.

One group of packets from this category contains a number of single FIN packets that we believe did not not belong to existing connections because there was no other communication between the endpoints specified in these packets in our trace file. The packets, therefore, could have had their IP addresses set to wrong values and could have been the result of an error in network software.

Some other packets, especially SYN ACKs, could easily have been backscatter packets issued by hosts under DoS attacks. Backscatter packets are transmitted by hosts being attacked when the attacker spoofs the source address and sets it to a random number. We see replies to the attack packets if the source address of the original packet happens to belong to our IP address range[3].

All other packets that fell in this category are SYN packets that do not belong to any standard service. Our analysis showed that there was rather a large number of unexpected TCP packets of various types (SYN, FIN, RST, regular data packets with ACK and/or PSH flags set), which were sent from random IP addresses and port numbers and targeted a single port on a single host. The private IP addresses appeared among other addresses. Even though the rate of such packets was not necessarily high enough to disrupt operation of the target hosts, we consider these packets to be a greater threat than all previous types of packets described in this subsection.

3.1.2 IP Addresses Out of the OU Address Range

Currently, Ohio University occupies a single class B network (132.235.0.0/16). All of the packets that we see on the global link are expected to come either to or from OU address space, while packets on the local link should have at least one IP address from that range and can also belong to communication between two OU hosts. In

[3]For more information about backscatter analysis see [13].

Table 3.8 Categories of Packets with Addresses Out of the OU Address Range Obtained on Local Link

Case	Category	Packets	Percent	Cause
1	Packets from private Microsoft IP addresses	7,260	2.97%	Microsoft OS specifics
2	DHCP packets from aol.com IP address space	8,490	3.48%	Erroneous software
3	IGMP packets from 2.0.0.x through 6.0.0.x	877	0.36%	Unknown
4	Limited broadcast packets from the router	228,054	93.43%	Distributed DoS attack
5	Other	152	0.06%	Unknown
	Total	244,833	100.00%	

both cases, at least one address in each IP packet should be from the OU address range. All packets that do not obey that simple requirement are worth examination.

First we proceed with the analysis of packets from the local network and then move on to analysis for the global link. One reason why analysis of the local link precedes the global link this time is that we obtained a larger number of packets where neither source nor destination IP address fell within the OU address range on the local link due to the network specifics. More importantly, we were able to use hardware addresses to aid in determining the cause of such packets and therefore consider the results from the local monitoring point more precise.

3.1.2.1 Local Network

The packets from the local link for which neither the source nor destination IP address were in the OU address space are categorized in Table 3.8. We describe every category of these packets and provide a possible cause for each of them.

- **Case 1:** This group of packets includes packets sent by local hosts using IP addresses in the range 169.254.0.0 – 169.254.255.255 as their source addresses. This range of IP addresses belongs to the Microsoft-reserved class B network 169.254.0.0/16, which is used by the Windows operating system. According to Microsoft documentation [12], Windows hosts use addresses from this range during so-called "automatic configuration" if they are configured to use DHCP [4] and cannot obtain an IP address from a DHCP server. The addresses are not routable in the global internet and cannot be used outside the network.

  Since we monitor the link to which these Windows hosts are attached directly, we can see all of the packets issued by them. Packets sent from the Microsoft private IP addresses consist of several types: IGMP packets sent to multicast addresses, NetBIOS name and datagram service packets sent to their directed broadcast address, and DHCP INFORM packets sent to the limited broadcast IP address. Packets from all of the above groups were sent to either multicast or broadcast addresses and none of them received responses.

  One type of packet sent by these computers with private Microsoft addresses is DHCP INFORM messages. DHCP INFORM messages were designed to be used by hosts that externally obtain their IP addresses (such as via manual configuration) and wish to get other network parameters by means of DHCP [4]. All hosts other than the router on our network use DHCP to get their IP addresses and other network parameters from the server. Therefore they should not be issuing DHCP INFORM messages to the server. Analysis of the traffic on the network shows that the hosts with private Microsoft addresses did not attempt to broadcast DHCP DISCOVER messages. Ironically, they sent only DHCP INFORM packets to retrieve network parameters from a DHCP server like machines that have statically configured and known IP address.

- **Case 2:** The next group of packets with both source and destination addresses outside of the range of our network are DHCP packets sent from 172.128.x.x – 172.186.x.x IP addresses. This range of IP addresses belongs to aol.com. The great majority of the packets are DHCP INFORM packets sent to the limited broadcast IP address, and a small portion of them are IGMP packets sent to a multicast address. There are also single TCP, ICMP, and other than DHCP UDP packets that comprise a negligibly small portion of the packets from this category.

  A web page on the AOL site [2] states that America On-line (AOL) client software uses IP addresses from the AOL address space even in cases when a client host has already been preconfigured with a globally valid IP address. The page says that packet encapsulation is used to route packets from AOL servers to their clients but no encapsulation is used in the other direction, from the clients to servers. This means that with the scheme described above we should see a large number of packets on the local network sent from the AOL IP addresses to their server machines. The majority of packets captured on our network that had an AOL address as the source IP address were, however, DHCP packets sent to the limited broadcast address, which is not conformant with the scheme presented on the AOL web page.

  The packets we captured on the local link were originated by a rather large number of hosts on the network that used AOL addresses instead of their Ohio University addresses. At the time they were using the source addresses from this range, these hosts did not use other IP addresses, i.e. their "real" addresses that appear in other packet traces. We suspect that local machines that were unable to obtain their global IP addresses from the DHCP server tried to use other IP addresses, i.e. assigned to them by AOL software, to obtain other network parameters. While the cause of the packets issued from the above mentioned IP addresses could be specifics in implementation of the AOL client software, hosts should not sent DHCP INFORM packets on networks where address assignment is performed dynamically.

- **Case 3:** Some other packets that fell into this category of erroneous packets are IGMP report packets sent to several multicast addresses from source addresses like 2.0.0.1, 3.0.0.2, 4.0.0.3, and 6.0.0.5. All hosts that sent such packets are known to run the Windows operating system. We were unable to find more information about these packets and discover the reason why they appear on the network.

- **Case 4:** The largest group of packets with IP addresses out of the OU range came to the monitored network from the router. They were sent from various IP addresses outside of the range of the local network and targeted the limited broadcast address. In order for such packets to enter the network, the router must allow forwarding of limited broadcast packets.

  Our first traces included very few packets of this type, usually sent from a single IP address. However, one trace included an enormous number of ICMP ECHO REQUEST packets that entered the network and triggered an even greater number of ICMP ECHO REPLIES in response. The ECHO REQUEST packets came from about 25 different IP addresses with domain names in Austria, the Czech Republic, Estonia, Finland, Germany, Greece, Iceland, Italy, the Netherlands, Sweden, the United Kingdom, the USA, and possibly others[4]. Even though our router allowed the limited broadcast packet to come through[5], forwarding of such packets should normally be disabled on routers. We believe that these packets could not be originated outside of the Ohio University

[4]We were unable to resolve the names of all IP addresses.

[5]After obtaining this trace, limited broadcast forwarding was disabled at the router and similar cases do not appear in the following traces.

network (they would have been discarded at the ISP before entering the university), but rather came from a local machine and had their IP addresses spoofed. Since the number of ECHO REQUESTs that we captured was very large and the number of responses was thousands of times larger, we — with some degree of certainty — can conclude that all machines on the local network participated in a distributed DoS (DDoS) attack against the above mentioned machines in Europe and the US. The attack targeted hosts running different services like HTTP, name service, database system, and IRC servers as the names of the machines suggest.

- **Case 5:** This category includes all packets that did not fall in any other cases listed above and consists of NetBIOS and IGMP packets sent from unusual IP addresses (for instance, 128.128.128.128 or 4.0.1.0) to the directed broadcast address. Similar to other packets coming from invalid IP addresses, these packets failed to receive responses. They were sent from only two hosts and we assume that the errors could have been caused by misconfiguration of the hosts.

3.1.2.2 Global Link

As mentioned above, we obtained a smaller number of packets with both source and destination addresses out of the OU address range on the global link than at the local monitoring point. The reason is that many of such packets on the local network are limited broadcast packets which were not forwarded up to the global link. The number of packet types is also less than the corresponding number from the local link. The results from the global link are shown in Table 3.9.

- **Case 1:** All packets from this group come from the private Microsoft addresses and are related to NetBIOS services. The majority of them are name and datagram service requests sent to the directed broadcast address. Some of the packets from this category are sent to IP addresses outside of the OU address

Table 3.9 Categories of Packet with Addresses Out of the OU Address Range Obtained on the Global Link

Case	Category	Packets	Percent	Cause
1	Packets from private Microsoft addresses	250	88.3%	Microsoft OS specifics
2	Packets from aol.com IP address space	33	11.7%	Unknown
	Total	283	100.0%	

range. It is evident that the equivalent category of packets from the local link (see subsection 3.1.2.1) included more diverse types of packets. This can be explained by the fact that not all packets from this category originating on local networks reach the global monitoring point. A large number of them are broadcast packets that are not intended to leave the physical link.

The private Microsoft IP addresses were designed to be used on the local network only when a global IP address is not available [12]. Unfortunately, it is impossible to either prevent packets sent from these addresses from leaving the local network or packets sent to these addresses from going outside[6] without adding special rules. We suggest that IP addresses from the range 169.254.0.0 – 169.254.255.255 are treated as private IP addresses and filtered out at the routers.

- **Case 2:** The second group of packets that do not carry IP addresses from the OU address space and consequently should not appear on the OU network come from IP addresses belonging to aol.com. The number of these packets

[6]More precisely, packets sent to the private Microsoft addresses tend to go to the default router and leave the domain since their location is undefined.

observed on the global link is much smaller than the number of packets from the equivalent category obtained from the local link (see section 3.1.2.1). Similar to Case 1, the majority of the packets sent from the aol.com addresses do not leave the network of their origin and do not reach the global monitoring point.

Almost all of the packets in this category are TCP RST packets sent to various IP addresses in the global internet. We verified that the packets are not related to any communication underway at the time of capturing these packets as best as possible — neither the IP of the other end nor port number matched existing, known connections. More detailed analysis of the hosts sending these packets could not be conducted due to the unavailability of the original physical layer information on the global link. Usage of IP traceback systems would be very beneficial in determining the real source of the packets.

Even though we know that America On-line software uses IP addresses different from the addresses assigned to the client machines [2], we can not conclude that the cause of these packets was improper software implementation. These packets target real-world hosts which will possibly trigger responses to aol.com machines. If such packets are not blocked by the end system routers, aol.com and other hosts on the internet get a lot of possibly unexpected packets.

An interesting observation is that the majority of the packets that had both the source and destination addresses out of the OU address range were sent during morning hours — 85% of all packets from this category were found in 9 a.m. packet traces. We do not possess enough information about the hosts sending these packets to explain this phenomenon, but found it noteworthy.

3.1.3 Other IP Address Violations

This section describes the remaining IP address violations that were detected during our experiments. These kinds of violations come from so-called "special" IP addresses that may not be legitimately used as either the source address, destination address, or possibly either one [24].

In general, IP addresses can be represented using the following notation:

$$\textit{IP-address} = \{<\textit{network\_number}>, <\textit{host\_number}>\}$$

or

$$\textit{IP-address} = \{<\textit{network\_number}>, <\textit{subnet\_number}>, <\textit{host\_number}>\}$$

The lengths of the network and subnet (if present) numbers are fixed and locally known for any given host. In the global internet, however, it is impossible to determine the network prefix length for any arbitrary host. With deployment of classless inter-domain routing (CIDR), network prefix lengths vary from subscriber to subscriber and range from 8 to 27 bits. Under these circumstances, not only can we not know the network or subnet number lengths, but we can not find out whether a particular network has subnets either. Therefore we use the first notation from the two given above and do not include subnets in our analysis. We also use the notation *"1...1"* to indicate that a field contains all 1 bits.

Using the assumptions above, some common special cases of IP addresses are as follows:

1. {*0, 0*} This host on this network. Can only be used as a source address.

2. {*0, <host_number>*} Specified host on this network. Can only be used as a source address.

3. {*1...1, 1...1*} Limited broadcast. Can only be used as a destination address, and a datagram with this address must never be forwarded outside the network of origin.

4. {*<network_number>, 1...1*} Directed broadcast to specified network. Can only be used as a destination address.

5. {*127, any*} Internal host loopback address. Should never appear outside a host.

Table 3.10 Detected IP Address Violations on Global Link

Case	Description	Used As	Packets	Percent
1	This host on this network	Destination	0	0.0%
2	Specified host on this network	Destination	153	54.6%
3	Limited broadcast	Source	127	45.4%
4	Directed broadcast to a network	Source	0	0.0%
5	Internal host loopback address	Either	0	0.0%
	Total number of packets		280	100.0%

It can be seen that Cases 2 and 4 require us to know network prefix lengths of all IP addresses that traverse our links. While it is impossible to determine the exact prefix lengths, we know that all of them are at least 8 bits long. We also believe that network prefix lengths greater than 24 bits are not common in the global internet. This gives us the host portion of an IP address being at least 8 bits long. Having these assumptions, we can check the most significant byte of an IP address for network number violations and the least significant byte for host number violations, if the network prefix length is not known in advance.

Our system verifies that all types of "special" IP addresses are used appropriately (for example, the limited broadcast address is used only as the destination address); otherwise errors are generated. None of these types of violations were detected on the local link and therefore in this subsection we provide analysis of the packets from the global link only.

3.1.3.1 Global Link

Table 3.10 summarizes all of the errors detected by the system on the global link and provides the total number and percentage of packets that fall into each category. As can be seen, none of the examined packets violated rules 1, 4 or 5. Other types

Table 3.11 Packets Coming from Limited Broadcast Address on Global Link

Case	Type	Packets
1	Outgoing ICMP packets (to private Microsoft addresses)	109
2	Outgoing packets (to global addresses)	16
3	Incoming packets (to local addresses)	2
	Total	127

were present in the trace files and analyzed below.

Case 2 shows packets that were sent to network 0. These packets could obviously not be routed to the destination because no host can have a zero network number in the global internet. The great majority of the packets with the destination network being 0 are SYN packets sent to well known TCP port numbers (we observed 25, 80, and 524). There are also several UDP packets that fell in this category, which appeared to belong to NetBIOS name service traffic. We believe that both types of erroneous packets were caused by misconfigured software. This observation becomes evident in certain cases. For example, we recorded packets from a host that periodically tried to connect to another machine with the network number 0 on port 25, day after day. Most likely, it had an incorrect IP address set in its mail server configuration, which could likely have caused this behavior. All of the packets included in this category belong to outgoing traffic and are not known to be either dangerous or useful. We recommend they be blocked by routers.

Case 3 provides statistics for packets sent from the IP address 255.255.255.255. These packets are further subdivided in Table 3.11.

The largest group of these packets are ICMP UDP PORT UNREACHABLE messages sent to private Microsoft addresses (169.254.0.0/16). The packets included error messages for different port numbers, but the majority of them were for port 2519.

We could not know the MAC addresses of the machines that sent these packets and thus could not determine what type of hosts generated the packets. However, we believe that these packets were sent in response to UDP packets broadcast from the private Microsoft addresses on local networks. The original broadcast packets sent from the private Microsoft addresses could have triggered responses from a computer or another device with a poorly implemented IP stack that used the limited broadcast address as the source address. We could not see the request packets themselves, because if this is the case they were originated on a local network and would not normally propagate to other networks. The responses, in turn, went to non-local IP addresses (the private Microsoft IP addresses are not known to the university's routers) and attempted to leave the domain. There could potentially be more packets on local networks sent from the limited broadcast address that do not reach the global monitoring point.

The second group includes TCP RST packets sent by Ohio University hosts. Some packets coming from the limited broadcast address inadvertently allowed us to detect a number of large network scans. During these scans, a SYN packet was sent to a particular port, usually 23 or 111 (telnet and SUN remote procedure call, respectively), on every host on a network including 0. We have discovered that some SYN packets sent to network addresses — those with the last octet equal to 0 — triggered replies back from the IP address 255.255.255.255. These replies could be either RST or SYN packets, but in either case they were sent from the same limited broadcast IP address. These replies are invalid and, similarly to the UDP PORT UNREACHABLE messages, could have been sent by a poorly implemented device that responded to packets intended for network IP addresses.

This group also includes packets not associated with port scans. We have determined that one machine outside of the university sent a number of unexpected RST packets to various IP addresses that belong to the OU address space. One of these addresses happened to be a network address (with the last octet in the IP address equal to 0) that triggered a response from the 255.255.255.255 IP address. These packets coming to the university are likely backscatter packets [13] that generated responses from the same machine or machines having similar IP implementations as in the previous cases.

The last group of packets coming from the limited broadcast address (Case 3 in Table 3.11) consists of UDP packets coming to the university. Note that this is in the opposite direction from all of the other packets with IP address violations listed in this subsection. These packets targeted a high port number and caused a UDP PORT UNREACHABLE message in response. Our analysis showed that along with the packets coming from the limited broadcast address a very large number of UDP packets from various global IP addresses targeted the same port number at that time. The distribution of the source IP addresses seemed random which makes the packets look like a DoS attack.

3.2 TCP Analysis

The results shown in Tables 3.1 and 3.2 provide the number of errors generated at both the global and local monitoring points. It can be seen that, similar to the errors in the IP header, not all of the types of errors checked for by the system were observed. In this case the system did not record any packets that did not carry the entire TCP header, but all other types of erroneous packets were present on both links. In this subsection we discuss each of the types of TCP errors that were logged by the system in more detail and present our conclusions about their origin.

3.2.1 TCP Packets with Zero Ports

The number of packets detected by the system where at least one of the port numbers (source or destination) was zero is not large. It can be seen that the rate of the packets from this category on our networks is less than one packet per several million. A reasonable explanation of these packets would be incorrect implementations which erroneously set the field to a value of zero (and as we will see further in

Table 3.12 Types of Packets with Zero Ports on Global Link

Case	Category	Packets	Percent	Possible Cause
1	Malformed packets	33	24.26%	Corruption
2	Unsolicited RST packets	2	1.47%	Unknown
3	SYN packets to port 0	3	2.21%	Misconfiguration
4	ACK packets to port 6	98	72.06%	Possible attack
	Total	136	100.00%	

this subsection, the percentage of the packets from this category that are corrupted or malformed due to incorrect implementations is not small). We are not aware of any work in the network security field that lists TCP port 0 as being used to aid network intrusion, but do not exclude such a possibility. In this section we analyze all of the packets with zero port numbers, divide them into several categories, and provide possible causes for each category.

3.2.1.1 Global Link

Table 3.12 summarizes all of the categories of packets with zero port numbers captured on the global link, and the percentage of these packets that fall into each category. The first three groups represent packets that we consider invalid and suspect are caused by incorrect implementations or configuration. Packets included in group 4 in our opinion are more likely to be harmful than the previous categories because there exists a possibility that they were sent by a malicious user. Below we provide a detailed description of each category.

- **Case 1:** The first group represents packets that were either corrupted in the network or at the sender. Even though not all of the packets from this category failed checksum verification, none of them looked like valid packets. The majority of these packets had an invalid TCP header length, incorrect combination of TCP flags, nonstandard options that could not be recognized, or otherwise improper values in other TCP header fields. A significant number of them were sent between hosts that had communications underway at the time these packets were issued, and the packets with zero port numbers look like they could have otherwise corresponded to open valid connections. In the majority of such cases, the source port number was set to zero and the real source port number (the source port number of an existing connection) appears to have been used as the destination port. This pattern is evident in the majority of those packets (31 out of 33 packets), which allows us to conclude that these packets were likely caused by similar errors in TCP implementations. The remaining packets (those that did not follow the scheme described above) were either responses to those packets or malformed packets without any other apparent communication between that pair of hosts. We believe that the latter packets were discarded at the destination.

- **Case 2:** The packets from this category are RST packets sent from the private Microsoft addresses (169.254.0.0/16)[7] to IP addresses in the Ohio University address range. We believe that those packets could either be actual responses to packets sent by machines that lied about their addresses or erroneous packets carrying incorrect endpoint information. We do not believe that the number of packets that fell into this category is sufficient to make more precise conclusions.

- **Case 3:** This group consists of valid SYN packets packets sent to port 0 on Ohio University hosts. There was no other communication between each pair of hosts, which makes these packets unlikely to be part of scanning or another malicious activity. We believe that misconfiguration or an incorrect usage of a network application is a reasonable assumption in this case. Packet retransmis-

[7]For more information on the private Microsoft addresses see section 3.1.2.1.

sions that fell into this group followed the standard time increments and did not look abnormal. We should notice, however, that only a poor TCP implementation would create these packets, regardless of software configuration at upper network layers.

- **Case 4:** The last group of packets with zero port numbers looks more suspicious than the packets described above, and all of the packets from this group followed a very specific pattern. The packets were simple ACK packets where the source and destination ports were 0 and 6 respectively, and all them advertised a TCP window of size 0. The majority of these packets were sent in groups logically increasing the sequence numbers they acknowledge. About 6% of these packets had private IP addresses as the source IP address. It is our opinion that they were sent on purpose using the same tool. The number of such packets was rather large — they comprise over 70% of all packets with zero ports — and could have been a part of a network attack or host detection.

3.2.1.2 Local Network

Our system recorded only six cases of packets with zero port numbers on the local network. We consider five of the six packets corrupted either in network or at the sender. The reasons range from failed checksum verification to invalid header length or incorrect values in other TCP header fields that invalidate the packets and their contents.

The last packet with a zero port number was a RST packet. Having a correct checksum, it could be either a backscatter packet or packet incorrectly constructed at the sender. In the latter case, not only is the port number portion of the header set to an invalid value, but the IP address also must be a result of a mistake because we did not observe any other communication between that pair of hosts.

Comparing the local link result with the packets from the same category obtained on the global link, we notice that the local packets have a lower rate of occurrence and are not very diverse. They are unlikely to correspond to malicious activity and thus are less likely to be dangerous for the destination machines. The packets that we consider more dangerous come from outside hosts (as seen on the global link), and could be blocked at the incoming router to improve the protection of internal hosts.

3.2.2 Invalid TCP Flags

The control bits SYN, FIN, and RST, when set in a TCP header, alter the state of a connection. The remaining flags — ACK, PSH, and URG — do not force a TCP connection to enter a different state, but still require additional processing. Atypical combinations of these flags are ambiguous and are not expected at the receiver. Therefore, responses to packets carrying incorrect flags might vary from implementation to implementation and are not always elegant.

In this subsection, we analyze and categorize all of the TCP packets obtained on the links monitored during our experiments which had improper combinations of TCP flags . Tables 3.1 and 3.2 at the beginning of this chapter show that the rate of these packets is low and, on each link respectively, it exceeds only the rate of packets with zero port numbers.

3.2.2.1 Global Link

Table 3.13 lists all of the groups of packets with invalid combinations of TCP flags as seen on the global link. All of the packets that failed checksum verification are included in the first category. We do not perform any further analysis on these packets, assuming that they were corrupted during transmission. Even if the packets were corrupted by a malicious user and transmitted with invalid checksums, they still should be excluded from the set of analyzed packets under the assumption that they will be discarded at the destination machine and never be processed.

The largest category of these packets is comprised by what we will call malformed packets from existing connections. This group consists of packets with various combinations of TCP flags where the source and destination IP addresses belonged to

Table 3.13 Types of Packets with Invalid TCP Flags on Global Link

Case	Type	Packets	Percent
1	Failed Checksum	74	37.76%
2	Malformed packets from existing connections	108	55.10%
3	FIN RST to close a connection	12	6.12%
4	PSH set in SYN	1	0.51%
5	Probe packets	1	0.51%
	Total	196	100.00%

hosts that were communicating at the time the traces were gathered. More thorough analysis showed that these packets had meaningless values in some other fields of the TCP header (i.e. had bad header lengths, unrealistically large data portion lengths, or carried unrecognized TCP options). A number of these packets also fall into other categories of protocol violations (e.g. one of the ports had a value of zero). Their checksums, however, were either correct or could not be verified due to insufficient snapshot length. We believe these packets were either malformed at the sending end or potentially corrupted during transmission (for those packets where checksums could not be checked). This category is composed of the following packet types:

1. Packets with IP addresses and TCP port numbers belonging to open connections but carrying invalid data. In this case, information about both endpoints carried in the packets seemed correct but either their checksums, TCP header length, or other fields had abnormal values.

2. Packets in which the "correct" source port number was used as the destination port, and the source port was set to a random value. A large portion of these packets had the source port number set to zero. This same error was described

earlier during our analysis of TCP packets containing zero port numbers (see section 3.2.1).

3. Packets with apparently correct values for the source port number (i.e. corresponding to an open connection) but incorrect values for the destination port number.

4. Packets with IP addresses belonging to communicating hosts, but neither one of the source or destination port numbers belonged to connections that were known to be open between the hosts. The TCP port numbers carried in these packets vary, but values 18245 and 21536 (as source and destination, respectively) were recorded in several packets.

Even though all these packets carry valid IP addresses, the first type of corrupted packets is the most innocent because the packets can interrupt at most one connection between one pair of hosts. The other errors introduce new port numbers and thus might be more harmful.

The next type of packets with invalid TCP flags from Table 3.13 are packets sent to close a connection but having had both the FIN and RST flags set. These packets were issued:

1. As RST packets in response to an initial SYN or other unsolicited packet;

2. After a FIN packet in the same direction where the other end did not close the connection after some amount of time (3–5 minutes). In some cases, more than one FIN RST packet was issued. All these FIN RST packets appeared to have been interpreted as RST packets, as no other packets that belonged to the corresponding connections followed these packets.

3. After a RST packet in the same direction.

There is no single, obvious explanation for all of these packets. A number of them, especially those RST packets sent in response to SYN and other unexpected packets, were most likely legitimate packets and meant by the sender to be pure RST packets. However, we believe that such packets should still be blocked at routers because they introduce violations of protocol specifications. The fact that these packets are blocked should not have an adverse effect even when the sender is not aware that the packet it sent was not correct — the host will resend the packet when it determines that the packet did not reach the destination (if the other end does not close the connection)[8].

Table 3.13 shows that SYN packets with the PSH flag set are also present on the network. A TCP packet can have the PSH flag set only when it carries data [21], but the packet from our data set was apparently accepted during connection establishment and communication continued. Normally such packets would not be expected to to cause damage, but nevertheless they should not be treated as valid packets at the destination.

The last group of packets having abnormal combinations of TCP flags consists of probe packets. The only packet in this category had the SYN, FIN, URG, and PSH flags set. It was issued along with other apparent probe packets which we believe were part of a fingerprinting attempt[9]. Thus, our empirical results suggest that a very small fraction of packets having invalid combinations of TCP flags are likely to be a result of malicious activity.

3.2.2.2 Local Network

The rate of packets having invalid TCP flags on the local link was approximately the same as the rate of similar packets obtained on the global link. The number of bytes of the packets captured on the local link that was saved in trace files (the "snap-shot length") was large enough to verify the checksums of all of the packets; therefore

[8] For more analysis on this filter rule refer to the Recommendations section 4.1.

[9] A popular network scanner `nmap` [5] is known to send packets with exactly the same combination of TCP flags when trying to determine operating system type of a remote host.

Table 3.14 TCP Packets with Invalid Combinations of TCP Flags from Local Link

Case	Type	Number	Percent
1	Invalid checksums	44	86.3%
2	Malformed packets	2	3.9%
3	FIN RST packets to reset connections	5	9.8%
	Total	51	100.0%

the checksum analysis in this subsection is more precise than the corresponding analysis of the packets from the global link (see section 3.2.2.1). The results provided here are conformant with the results obtained at the global monitoring point. The packets are categorized in Table 3.14 and explained below.

- **Case 1:** The largest group of packets having invalid combinations of TCP flags consists of corrupted packets. The rate of corrupted packets on the local link is much higher than the same rate on the global link. Most likely we underestimated the rate of corrupted packets on the global link due to the inability to verify all of the checksums. If we look at the tables more thoroughly, we can see that the percentage of these packets in Table 3.14 is approximately equal to the percentage of both packets with invalid checksums and malformed packets in Table 3.13. This tells us that our assumption about the invalidity of packets that we called malformed on the global link was probably correct, and that at least a portion of them were unlikely to be carrying valid checksums.

  An interesting fact is that the majority of the packets from this category (over 80% of all of the packets with invalid checksums) were very similar and did not look like they were corrupted in the network. All of them had the SYN, FIN, RST, and URG bits set, carried payload, and were sent from port 18245 to 21536 (note that such packets also appear on on the global link but they are not as common). We noticed only three machines on the network that received such packets from hosts outside the university. It is our opinion that they are more likely to be caused by a specific program or implementation than network corruption.

- **Case 2:** This group includes only two packets. One of them had an invalid header length, and we think it was ignored by the receiver because the packet did not appear to change the connection's state as communication continued as usual. The other packet had both the SYN and FIN flags set in addition to carrying data and reporting unacceptable sequence numbers. It also apparently did not affect the connection it corresponded to and apparently was ignored by the receiver.

- **Case 3:** We believe that the majority of the FIN RST packets included in this category could have been intended as RST packets. At the local link, one FIN RST packet was sent after a connection close (FIN packets corresponding to that connection were seen in both directions) and all other packets from this group were responses to SYN packets that were sent to reset the connections. In the latter case, it seems likely that the sender of the SYN packets did not interpret these FIN RST packets as RST packets and therefore repeated attempts to establish a connection several times.

3.2.3 TCP Header Reserved Bits

The last element of the TCP header considered in this work is the reserved bits. In this subsection, we provide results from the validation of the reserved bits in the TCP headers of packets seen on the network. The two upper flag bits and the other reserved bits are analyzed separately, with the results combined together for comparison. As shown later in this subsection, packets that were recorded as having non-zero reserved bits can be divided into a very few groups according to their characteristics — the

Table 3.15 Packets with Non-zero TCP Reserved Bits on Global Link

Type	Upper	MBZ	Both	Total	Percent
Invalid Checksum	92	324	67	349	33.3%
Listed in previous sections as invalid	90	102	83	109	10.4%
ECN-capable SYN packets	451	0	0	451	43.1%
RST packets with one upper bit set	92	0	0	92	8.8%
Other	29	41	24	46	4.4%
Total	754	467	174	1,047	100.0%

majority of them were either corrupted packets or legitimate attempts to establish ECN-capable connections. The percentage of packets that fall into each category varies by link.

3.2.3.1 Global Link

Our results after processing the data from the global link are shown in Table 3.15. It can be seen that the first two categories represent packets that we would not consider valid. The first group consists of packets that failed checksum verification tests, and the second group is composed of packets that have already been declared invalid in previous subsections. In the latter case, we deal with packets carrying zero ports, forbidden combinations of TCP flags, and/or incorrect values in other header fields. It should be noted that these corrupted or malformed packets are likely to have non-zero values in either or both of the upper flag and MBZ bits. Combined together, these two groups comprise a large fraction (over 43%) of all of the packets having non-zero values in the reserved bits.

The next large group of packets that we observed were ECN-capable SYN packets that tried to negotiate the usage of ECN with the receiving end. We observed several hosts that were ECN-compliant and willing to use this feature for their communication. However, none of the attempts from our data set were successful or led to establishment of an ECN-capable connection. All of the packets from this group are legitimate and compose another large portion of the packets in this category. The MBZ bits are properly set to zero in all of these SYN packets.

Packets from the next group had the RST bit as well as one upper flag bit (the most significant bit of byte 14 of the TCP header) set. In most cases these RST packets were sent following FIN packets, which adds another instance of non-conformance with the protocol specifications. We were not able to determine what these packets might mean.

The last category listed in Table 3.15 includes the smallest number of packets with the prevailing number of them being invalid. Non-zero values in both the upper flag and MBZ bits are common. We were unable to verify the TCP checksums of large data packets due to insufficient snapshot length. However, most of the packets from this category had control flags set together with a non-empty data payload, which makes our assumption of their corruption more plausible. We observed packets that, in addition to having non-zero reserved bits, had some of the following characteristics:

- Packets with their data length entered as the destination port number in RST packets with a non-zero length data portion.

- Data packets with control bits (FIN, SYN, RST) set (a number of them also report bad header lengths).

- Packets with invalid port numbers that could not have belonged to open connections because there was no other communication between those pairs of hosts on the ports specified.

From all of the types of packets described above, only the RST packets appeared to be interesting from a security point of view. The corrupted or otherwise malformed packets are likely to be discarded at the destination host. They normally would not

Table 3.16 Packets with Non-zero TCP Reserved Bits on Local Link

Type	Upper	MBZ	Both	Total	Percent
Invalid checksums	44	46	43	47	77.0%
Other malformed packets	3	4	3	4	6.6%
Normal RST packets	0	7	0	7	11.5%
ECN-capable SYN packets	3	0	0	3	4.9%
Total	50	57	46	61	100.0%

trigger a response, making the probability of the sender gaining information about the destination machine small. All of the ECN-capable packets, in turn, comply to existing standards and were designed to be compatible with implementations that do not support ECN. Therefore the dominant number of packets from this category (either legitimate or corrupted) can be treated as innocuous as they are unlikely to purport malicious intent.

3.2.3.2 Local Network

Our results from the local link are presented in Table 3.16. As one can see, the majority of the packets from this category are corrupted as well — either packets with invalid TCP checksums or packets with correct checksums but improper values in other header elements that we believe were rejected by the destination hosts. The RST packets listed as "normal" in the table appeared to have random values in the MBZ bits. They were sent, as best we can tell, in response to SYN packets and were all from the same host. The rest of the packets that we captured were legitimate attempts to establish TCP connections that support ECN.

Table 3.17 Checksum Statistics

Protocol	Link	Analyzed Packets	Invalid Checksums	Error Rate
IP	Local	54,696,049	0	N/A
	Global	247,873,366	0	N/A
TCP	Local	30,701,761	2,178	$7.09\ 10^{-5}$
	Global	95,429,544	46,466	$4.87\ 10^{-4}$
UDP	Local	20,805,660	6	$2.88\ 10^{-7}$
	Global	14,897,627	49	$3.29\ 10^{-6}$

3.3 Checksum Statistics

This section combines the results of checksum verification of IP, TCP, and UDP packets. In the case of transport layer protocols (TCP and UDP), only the packets that were large enough to perform checksum verification were included in the statistics that correspond to that transport layer protocol. Table 3.17 lists the number of packets that had invalid checksums for each protocol considered in the analysis and their error rates. It combines the results of checksum verification of packets from global and local links.

It can be seen that we did not record any packets with invalid IP checksums on either link. The number of bytes covered by the IP checksum is not large[10], which reduces its probability of failure. Also, many routers verify IP checksums and discard the packets having failed checksums, which reduces the number of packets with corrupted IP headers that can be seen on a given network. Therefore we believe that our results are realistic, and the rate of packets that fail IP checksum verification

[10] The IP checksum covers only the IP header. Compare this with, for example, the TCP checksum which covers the entire TCP header, the TCP data, and a part of the IP header.

is indeed insignificant.

Packets with failed TCP and UDP checksums were present in our trace files, and the results are also shown in Table 3.17. As mentioned earlier, a number of traces from the global link had truncated packets. Consequently, not all of the checksums for packets from the global link could be verified, and we realize that our statistics may be biased. We believe, however, that the results from the global link shown in Table 3.17 represent the lower bound of the rate of corrupted packets. Large packets are more likely to be corrupted during transmission than short packets, making the actual error rate on the global link probabilistically higher than the error rate of the verifiable packets that were included in our statistics.

Comparing results from the global and local links, one might quickly notice that the error rates on the global link are almost an order of magnitude higher than those on the local link. The error rates on the global link may in reality be even higher if we in fact underestimated the actual number of corrupted packets at the global link, as hypothesized above. A large number of the packets traversing the local link do not leave the local area network and are not as likely to be corrupted during transmission as packets that travel through the global internet and traverse heterogeneous networks.

The error rate of UDP packets is much lower than the rate of corrupted TCP packets on both global and local link. The difference in the rates is similar on both links. We see two possible explanations of the difference between the rate of the TCP and UDP failed checksums. First, the UDP standard does not require all UDP datagrams to use checksums, and the checksum field is to be set to all zeros if the transmitter generates no checksum [18]. If no checksum was used for a UDP packet, our software counts the packet toward the packets with correct checksums, lowering the actual rate of UDP packets with corrupted checksums. We determined that over 40% of UDP packets with verifiable checksums (i.e. where the captured packet length was long enough to perform checksum verification) on the local monitoring point did not include checksums. The percent of this category of UDP packets on the global link was significantly smaller and comprised only 2% of verifiable packets on average.

Second, we estimated that an average TCP packet would be longer than an average UDP packet, which makes the probability of TCP packets being corrupted higher than that for UDP packets. Our analysis showed that TCP packets on the local link are as twice as large as UDP packets captured on that link. The difference in length is even larger for packets on the global monitoring point — the average length of TCP packets is equal to 3–4 average-sized UDP packets on that link.

Using the reasons described above, we can conclude that if checksum calculation were performed for all UDP packets and these UDP packets were of the same length as TCP packets, the rate of UDP packets with corrupted checksums would be approximately four times as large as listed in Table 3.17 for both links. This diminishes the gap between the rate of TCP and UDP packets with failed checksums but nevertheless leaves the rate of corrupted UDP packets much smaller than the rate of TCP packets that failed checksum verification.

3.4 Packet Distribution Analysis

This section summarizes results of out analysis detailed in the previous sections and draws more general conclusions about packet error rate. In particular, we evaluate distribution of packets that triggered warnings over the time of day and also their distribution by the possible cause. Moreover, we divide all traffic according to the direction of the packets — incoming, outgoing, or local — and analyze distribution of the packets that triggered warnings over time for each data stream.

3.4.1 Packet Distribution Over Time

The packet traces that we included in this study were captured at different times of the day. In particular, the data from the global link was being captured for rather brief periods of time five times a day over a course of seven days. The data gathering process always started at the same times (at 3AM, 6AM, 9AM, 3PM, and 11PM), which allows us to build error rate distribution over time and easily compare the results.

Packets from the local link were also captured at different times of the day, but their starting times were not as systematic as at the global link. Consequently, it would be difficult to build equivalent distribution for packets from the local link since the times when data capturing started vary from trace to trace. We also think that our distribution for the local link could be inaccurate or biased due to the inability to obtain approximately the same number of packets for each period of time included in the distribution. Therefore, in this subsection we take into consideration only packets from the global link and do not build distribution over time of the day for packets from the local link. We, however, provide limited results for the local link when we analyze distribution of packets going in each direction later in this section.

3.4.1.1 Global Error Rate

Figures 3.1 and 3.2 illustrate the distribution of packets that triggered errors over time at the global link. Figure 3.1 gives us overall error rates over all traces obtained at a particular hour, and Figure 3.2 provides a finer view where every data point corresponds to a single packet trace. Each value on the graphs represents the number of packets that generated warnings divided by the total number of packets captured on the link at that hour. We believe that ratios calculated in this fashion provide a good basis for data comparison over time.

As one can see from these figures, the error rates are always higher during business and evening hours and drop significantly at night. Since a large number of erroneous packets were presumably caused by poor implementations or misconfiguration, we expect that a lot of software that issues such packets involves user intervention and is not likely to be system software that runs constantly. Another reason, which is even more important than the first one, is that networks are busier during the day, and errors in data transmission are more likely to occur at that time than during the night. Since a lot of errors are due to failed checksums, we believe that our results

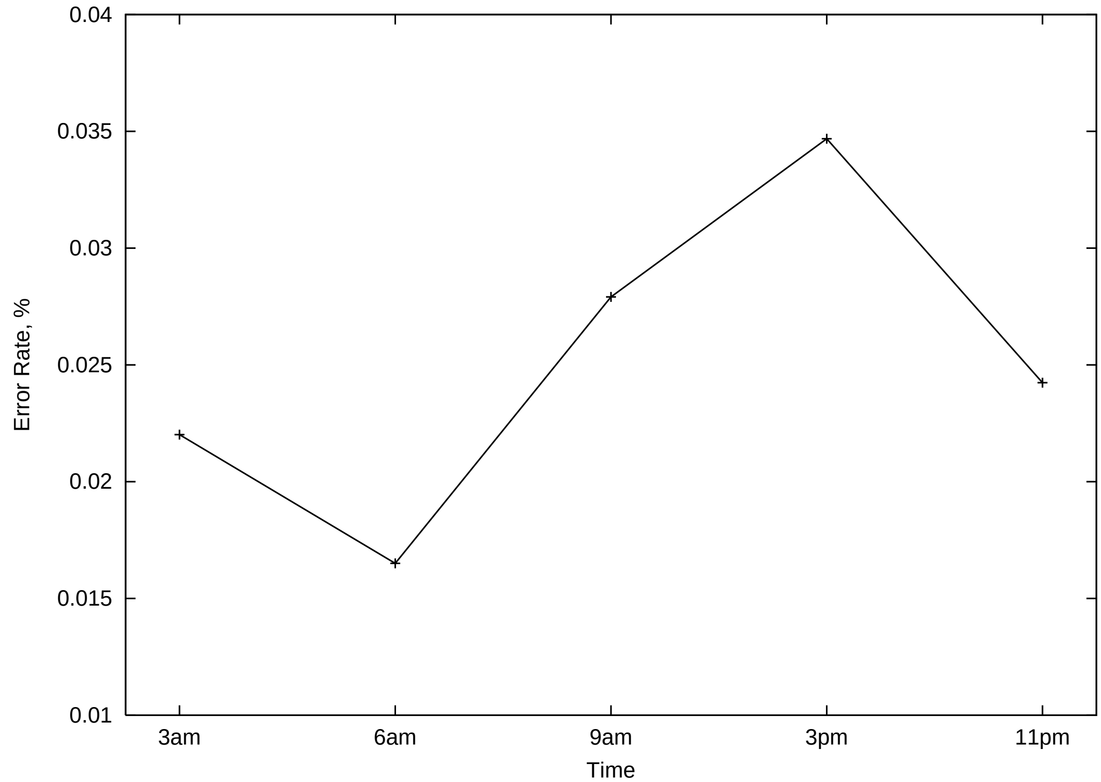

Figure 3.1. Combined Distribution of Erroneous Packets over Time on Global Link

are realistic.

3.4.1.2 Error Rate of Packets Going in Each Direction

In order to study the influence of the packet direction on the distribution of error rate over time, we divide all traffic into incoming, outgoing, and local and then compare error rates for each direction. We used physical addresses from the Ethernet frames to determine direction of a packet. For the local link, all packets that come from or go to the router were considered incoming and outgoing respectively, and all other packets (both broadcast and those that target a single host) were considered local. For the global link, we divide all packets into incoming and outgoing and do not have the local category. We start our description with the local link.

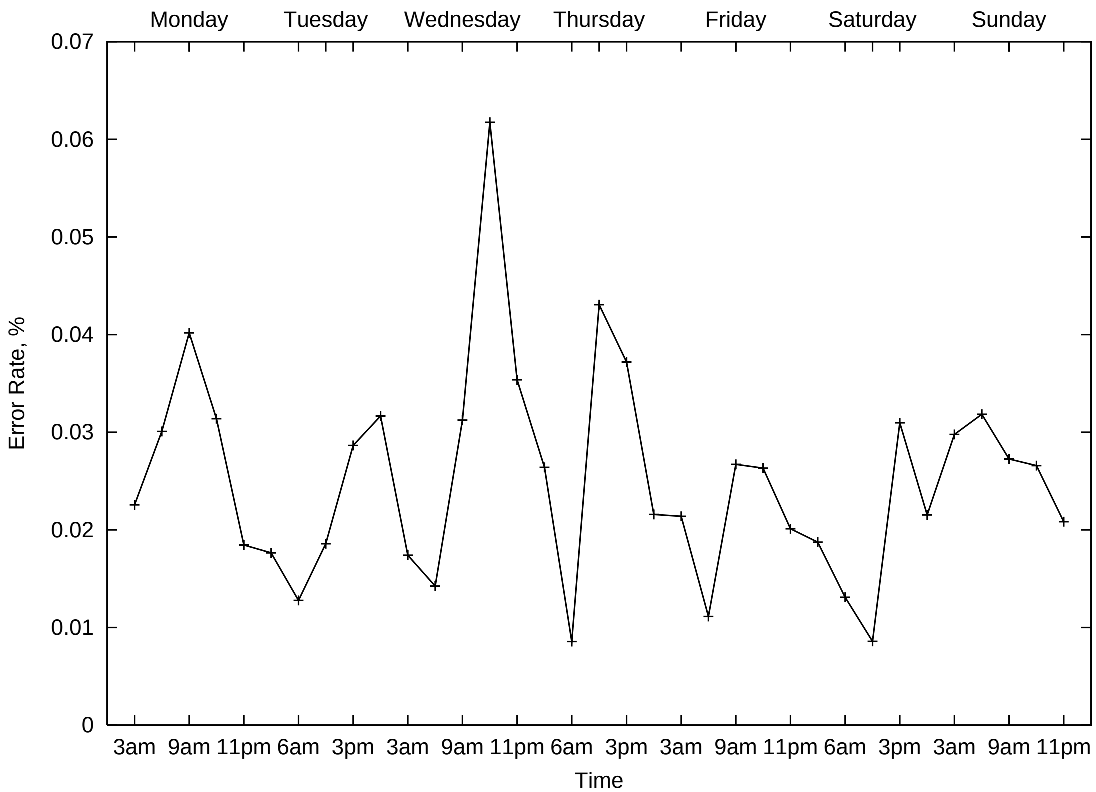

Figure 3.2. Distribution of Erroneous Packets by Day of Week on Global Link

Table 3.18 Distribution of Erroneous Packets by Direction on Local Link

Type	Errors	Error Rate	Fraction of Errors	Fraction of Packets
Incoming	230,360	9.36×10^{-3}	92.20%	45.02%
Outgoing	101	3.89×10^{-6}	0.04%	47.56%
Local	19,377	4.78×10^{-3}	7.76%	7.42%
Total	249,838	4.57×10^{-3}	100.00%	100.00%

Table 3.19 Distribution of Erroneous Packets by Direction on Local Link (Adjusted)

Type	Errors	Error Rate	Fraction of Errors	Fraction of Packets
Incoming	2,306	9.35×10^{-5}	10.59%	45.02%
Outgoing	101	3.89×10^{-6}	0.46%	47.56%
Local	19,377	4.78×10^{-3}	88.95%	7.42%
Total	21,784	3.99×10^{-4}	100.00%	100.00%

Table 3.18 shows the number of packets that were recorded as erroneous at the local link for each data direction and their error rate. This table also provides the percent of errors for each direction named "Fraction of Errors" (the number of erroneous packets going in one direction divided by the total number of erroneous packets) and percent of packets that traversed the link in each direction named "Fraction of Packets" (the number of all packets going in one direction divided by the total number of packets seen on the link). The DDoS attack described in section 3.1.2.1 affects the results dramatically and, in our opinion, this data does not provide realistic statistics. Therefore, we performed the same calculations excluding the DDoS packets and show the results in Table 3.19.

One can see from Table 3.19 that the largest number of packets that triggered warnings are local packets (almost 90% of all warnings generated by the system), even though the number of packets that do not leave the network is relatively small (about 7.5% of total traffic). The majority of violations in local packets are related to invalid IP addresses (packets with private addresses[11], packets in which none of the addresses belongs to the OU address space, etc.). Incoming packets generated the

[11] Note that private IP addresses are allowed for local communication in public internets if the packets do not leave the link.

Table 3.20 Distribution of Erroneous Packets by Direction on Global Link

Type	Errors	Error Rate	Fraction of Errors	Fraction of Packets
Incoming	5,844	5.00×10^{-5}	9.40%	47.13%
Outgoing	56,312	4.30×10^{-4}	90.60%	52.87%
Total	62,156	2.51×10^{-4}	100.00%	100.00%

second largest number of warnings, and packets sent by the local machines to global addresses had the smallest number of erroneous packets. Notice that the error rates differ by several orders of magnitude.

Table 3.20 shows distribution of packets by direction at the global link[12]. We found it surprising that the error rate of outgoing packets on the global link is an order of magnitude greater than the the error rate of incoming packets. This phenomenon can not be easily explained, especially if we take into consideration the fact that the majority of packets that triggered warnings are corrupted. These results also differ substantially from what we have seen on the local link.

Figure 3.3 shows distribution of the error rate of incoming and outgoing as well as combined traffic over time on the global link. The shape of the global error rate curve is governed by the outgoing traffic, which is expected since the error rate of the outgoing traffic is significantly higher in all trace files than the error rate of incoming traffic.

[12]Note that the number of erroneous packets in Table 3.20 is slightly smaller than the number of packets provided in Table 3.1 in the beginning of this chapter. This can be explained by the fact that in this section we include only unique packets in our analysis, while Tables 3.1 and 3.2 show unique packets for each type of violations. The total number of packets in Table 3.1 will be higher when the same erroneous packet falls into more than one of the categories listed in that table.

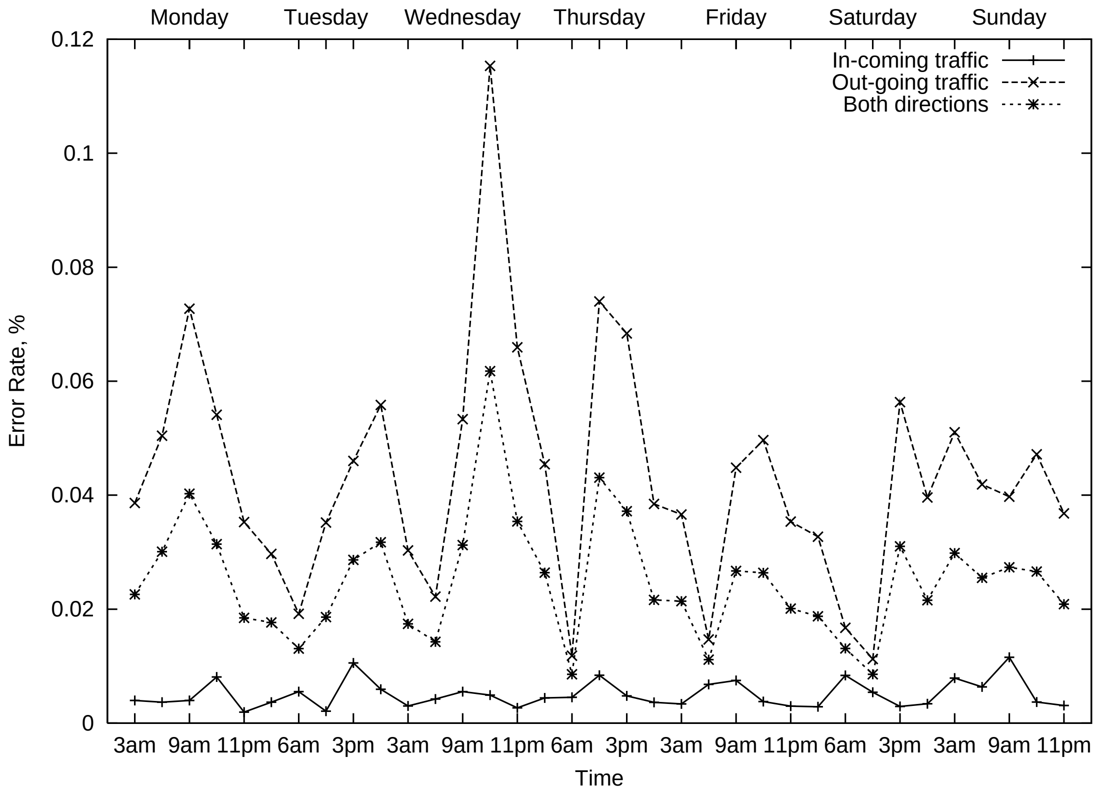

Figure 3.3. Distribution of Erroneous Packets by Time for Different Packet Directions on Global Link

3.4.2 Packet Distribution by Possible Cause

In sections 3.1 and 3.2 we analyzed all packets that were recorded as abnormal, divided them into categories, and tried to provide meaningful explanation and possible cause for each category. In this section we divide the same packets into different categories according to their possible cause derived from our analysis and show the results. Tables 3.21 and 3.22 summarize our findings for the global and local link respectively. The results presented in Table 3.22 do not include packets that correspond to the DDoS attack described in section 3.1.2.1 and were adjusted accordingly[13].

[13]With the attack included, the “Malicious User” category would comprise over 91% of all abnormal packets from the local link and percentage of all other categories listed in Table 3.22 would scale to fill the remaining space.

Table 3.21 Distribution of Erroneous Packets by Possible Cause on Global Link

Type	Packets	Percent
Legal Packets	451	0.72%
Corrupted Packets	47,234	75.83%
Poor Implementation	372	0.60%
Misconfiguration	11,096	17.81%
Backscatter Packets	1	0.00%
Malicious User	18	0.03%
Unknown	3,115	5.00%
Total	62,287	100.00%

From all of the categories listed in the tables, only packets included in the "Malicious User" and "Unknown" groups might purport malicious intent. The "Malicious User" category consists of only those packets that we, with some degree of certainty, view as sent intentionally. All of them are probe packets that were a part of various scans. The packets that could either be issued by an attacker or had a different origin (and we were unable to determine the cause of such packets) are united as "Unknown". All other categories — to the best of our knowledge — are either the result of a mistake or are not related to human factors, and therefore we believe that they were not sent by an attacker.

Table 3.21 shows that only 5% of all packets that triggered warnings on the global link might signal malicious intent. In reality, the actual number of packets sent by attackers is smaller than the total number of packets listed in these two categories. The largest portion of packets in the table (about 76%) was considered corrupted, mostly due to failed TCP checksums. A large number of errors (about 18%) were also caused by different types of misconfiguration. The remaining categories seem to

Table 3.22 Distribution of Erroneous Packets by Possible Cause on Local Link

Type	Packets	Percent
Legal Packets	2,598	11.93%
Corrupted Packets	2,286	10.49%
Poor Implementation	15,840	72.71%
Misconfiguration	16	0.07%
Backscatter Packets	1	0.00%
Malicious User	0	0.00%
Unknown	1,043	4.79%
Total	21,784	100.00%

be unsubstantial source of erroneous packets on the global link.

Our results from the local link are different from the results collected on the global link and summarized in Table 3.22. Table 3.22 does not list any packets as being sent by an attacker[14], and the total number of packets that might have had malicious intent is also below 5%. According to our analysis, the largest portion of erroneous packets that we recorded on the local link (about 73%) was caused by incorrect implementations or poor coding. It can be seen that the same category counts a much smaller number of packets on the global link and is below 1% (see Table 3.21). This allows us to conclude that the packets from this category are not likely to propagate far beyond their local network and in the majority of cases they are broadcast packets. The next two largest categories of packets that triggered warnings on the local link are legitimate and corrupted packets that comprise about 12% and 10% of all errors from the local link respectively.

[14]We again would like to draw the reader's attention to the fact that the DDoS attack was excluded from the analysis.

As Tables 3.21 and 3.22 suggest, only a small portion of all packets recorded by the system can be considered intrusive. Therefore, we conclude that the method described in this paper is not very efficient for network intrusion detection on its own. Work conducted during this research shows that if packet header analysis is performed as a part of network intrusion detection, it should be correlated with at least another more efficient method. This method, however, might be used as an additional source of data since all abnormalities detected by our system violate widely accepted communication rules[15].

[15]ECN-capable packets, while included in our research, do not violate existing standards. They should be removed from the analysis if the system is to be run in a production environment.

4. CONCLUSIONS

In this chapter we summarize the results of the analysis performed, draw general conclusions about the packets we recorded, and provide directions for future work. We also give recommendations on securing a site with carefully built access lists based on our observations.

4.1 Recommendations

In many cases, it would be beneficial to secure a site at the router against possible attacks that use invalid values of IP or TCP header fields through proper router configuration. Adequate router software should allow the administrator to filter out the following categories of packets:

1. Packets carrying private IP addresses where either the source or destination address is private. The rules should include all ranges of private IP addresses that are not being used within a network (10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16) and apply to packets traveling in either direction, both to prevent packets with spoofed source addresses from leaving the network and by this possibly contributing to a network attack, and to block packets with untraceable source addresses that enter the network.

   In addition to these ranges, we recommend that the range of private Microsoft addresses (169.255.0.0/16)[1] is also added to the filters. Since the cause of those packets carrying the private Microsoft addresses appears to be Windows machines that can not obtain legitimate IP addresses, it is preferred that such packets are filtered out on inbound interfaces of routers as close to the edge of

[1] For more discussion on the private Microsoft addresses, see section 3.1.2.1.

the network as possible. Both packets coming from and destined to IP addresses from that range should be blocked. Packets that come to the network from the global internet and carry the private Microsoft addresses as their source IP are appropriate, in turn, to be filtered on a global link. This scheme is also applicable to other ranges of private IP addresses.

2. Packets with neither a source nor destination address from the address range of the current network (except for networks where in-transit packets are allowed). Such filtering can be performed at all levels of traffic aggregation and is essential on the border and network edge routers to prevent widespread address spoofing.

3. Packets with inappropriately-used "special" types of IP addresses, as described in section 3.1.3. In particular, the following categories of packets are to be blocked:

   - packets sent to the all-zero IP address;
   - packets sent to the all-zero network with a non-zero host number;
   - packets sent from the limited broadcast address (255.255.255.255);
   - packets sent from the directed broadcast address (the host portion of the address consists of all ones);
   - packets containing one of the internal host loopback addresses (127.x.x.x), regardless of which addresses — source, destination, or both — are set to that value;
   - packets sent from multicast addresses (224-239.x.x.x)[2].

   In addition to these types, limited broadcast packets should not be allowed to travel through most gateways, and the forwarding of limited broadcast packets should often be disabled at the router.

[2]We have not discussed multicast addresses earlier in this paper but nevertheless recommend that they are included in the rules installed on a router.

4. TCP or UDP packets with zero port numbers. A large percentage of these packets are caused by poor implementations, and in the majority of the observed cases a zero port number may indicate that the packet is malformed. Since these packets violate TCP and UDP specifications (this port is reserved in both of these protocols), it is safe to drop packets with zero port numbers. Such packets can potentially be used to contribute to malicious activity when the port number is irrelevant (for instance, the source port number can be set to zero in attack packets that have no need of a response). Note that the source port number field is optional for UDP [18].

5. Source routed packets with either the strict or loose routing option. While source routed packets do not violate existing standards, they were designed for testing purposes and should not appear in a production environment. The source routing options in conjunction with spoofed IP addresses can be used to deliver response packets back to the sender of forged packets correctly and thus should be blocked at routers.

6. Zero-length data packets. These packets are not useful to the destination host since they do not carry data, but this filter might help to prevent DoS attacks using small packets.

7. Packets with invalid combinations of TCP flags. We recommend that at least the following combinations are considered for the router filters:

   - SYN RST;
   - FIN RST;
   - SYN FIN RST.

   SYN FIN packets can not easily be discarded because of deployment of T/TCP [3].

> We believe that packets with invalid combinations of TCP flags can be discarded at routers even if the sender is not aware of the fact that the packet it sent was incorrect. The sending host will resend the packet when it determines that the packet did not reach the destination — immediately if it knew that the original packet was malformed, and after a timeout otherwise. This policy might seem unfair to the flows that set the TCP flags to an incorrect value and still expect the protocol to work — such implementations might not set the flags to correct values on packet retransmission either. In this case, the decision of whether to block packets with improper combinations of TCP flags or let them through is left to the network management personnel.

These filters will noticeably reduce the number of warnings generated by our detection system. Some of the remaining errors can be analyzed further without human intervention. For instance, if the system is capable of determining whether a TCP packet belongs to an open connection or not, then based on its decision some packets with invalid TCP flags may be blocked while others may be allowed to go through.

Summarizing these results, we should say that we do not consider a large portion of the packets that generated warnings harmful and believe they could be caused by poor IP or TCP implementations or other similar errors. On the other hand, we were able to catch a number of cases where erroneous packets could not belong to legitimate traffic. Such cases can be analyzed so that knowledge obtained about them can be integrated into an IDS as an additional source of data. The technique described in this work is not a very efficient way for detecting a wide range of network intrusions but it can contribute to the intrusion detection process and make it more efficient.

4.2 Future work

As previously mentioned, we did not include the contents of packets in our analysis and considered only the packet headers. Future research should expand the analysis to cover the data itself. Examples of future directions include comparison of original and retransmitted packets for cases when these packets do not agree on the data contents. This is often done to subvert monitors for the purpose of network attacks. We would also like to detect fragments that carry overlapping data such that the contents of the overlapping region are different in the different fragments.

In our research, we did not attempt to detect TCP packets carrying acknowledgments for data that had not been sent. Such packets are not hard to detect and would be interesting to analyze. We believe that this bears future research. We also leave more detailed analysis of packets with small TTL values for future study.

Lastly, this work can be expanded to include analysis of data from different networks with more diverse traffic and cover more protocols.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] ALLEN, J., CHRISTIE, A., FITHEN, W., MCHUGH, J., PICKEL, J., AND STONER, E. State of the practice of intrusion detection technologies. Tech. rep., Carnegie Mellon Software Engineering Institute, Jan. 2000. http://www.sei.cmu.edu/publications/documents/99.reports/99tr028/99tr028abstract.html.

[2] AMERICA ONLINE, INC. *Webmaster Info, Article 14*, Oct. 2001. http://webmaster.info.aol.com/index.cfm?sitenum=2&article=14.

[3] BRADEN, R. T/TCP — TCP extension for transactions functional specification. RFC 1644, July 1994.

[4] DROMS, R. Dynamic host configuration protocol. RFC 2131, Mar. 1997.

[5] FYODOR. `nmap`. http://www.insecure.org/nmap/.

[6] FYODOR. Remote OS detection via TCP/IP stack fingerprinting, Oct. 1998. http://www.insecure.org/nmap/nmap-fingerprinting-article.html.

[7] GULA, R. How to handle and identify network probes, Apr. 1999. http://www.network-defense.com/papers/probes.txt.

[8] INTERNET SECURITY SYSTEMS. *Real Secure Attack Signatures*, 2000. http://documents.iss.net/literature/RealSecure/Signatures_5.0.pdf.

[9] JACOBSON, V. `traceroute`, 1989. ftp://ftp.ee.lbl.gov.

[10] JACOBSON, V., LERES, C., AND DMCCANNE, S. `tcpdump`, June 1989. http://www.tcpdump.org.

[11] LUDWIG, R., AND KATZ, R. The Eifel Algorithm: Making TCP robust against spurious retransmissions. *ACM Computer Communications Review 30*, 1 (Jan. 2000).

[12] Microsoft Corporation. *Automatic Private IP Addressing.* http://www.microsoft.com/TRAININGANDSERVICES/content/training/samples/2152/Webfiles/Mod03/03m10.htm.

[13] Moore, D., Voelker, G., and Savage, S. Inferring internet denial-of-service activity. In *10th USENIX Security Symposium* (Aug. 2001).

[14] NetBIOSWG. Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods. RFC 1001, Mar. 1987.

[15] NetBIOSWG. Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications. RFC 1002, Mar. 1987.

[16] Ostermann, S. `tcptrace`, 1994. http://www.tcptrace.org.

[17] Paxson, V. Bro: A system for detection network intruders in real-time. *Computer Networks 31*, 23–24 (Dec. 1999), 2435–2463.

[18] Postel, J. User datagram protocol. RFC 768, Aug. 1980.

[19] Postel, J. Internet control message protocol. RFC 792, Sept. 1981.

[20] Postel, J. Internet protocol. RFC 791, Sept. 1981.

[21] Postel, J. Transmission control protocol. RFC 793, Sept. 1981.

[22] Ramakrishnan, K., Floyd, S., and Black, D. The addition of explicit congestion notification (ECN) to IP. Proposed Standard, June 2001.

[23] Rekhter, Y., Moskowitz, B., Karrenberd, D., de Groot, G. J., and Lear, E. Address allocation for private internets. RFC 1918, Feb. 1996.

[24] Reynolds, J., and Postel, J. Assigned numbers. RFC 1700, Oct. 1994.

[25] Sekar, R., Guang, Y., Verma, S., and Shanbhag, T. A high-performance network intrusion detection system. In *Proceedings of the 6th ACM conference on Computer and communications security* (1999), pp. 8–17.

[26] Srisuresh, P., and Egevang, K. Traditional ip network address translator (traditional NAT). RFC 3022, Jan. 2001.