

Spring 5-31-2003

Comparative studies of network traffic in actual and emulated wireless networks

Curtis Salley Jr.
New Jersey Institute of Technology

Follow this and additional works at: <https://digitalcommons.njit.edu/theses>



Part of the [Computer Engineering Commons](#)

Recommended Citation

Salley, Curtis Jr., "Comparative studies of network traffic in actual and emulated wireless networks" (2003). *Theses*. 633.

<https://digitalcommons.njit.edu/theses/633>

This Thesis is brought to you for free and open access by the Electronic Theses and Dissertations at Digital Commons @ NJIT. It has been accepted for inclusion in Theses by an authorized administrator of Digital Commons @ NJIT. For more information, please contact digitalcommons@njit.edu.

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen

The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

ABSTRACT

COMPARATIVE STUDIES OF NETWORK TRAFFIC IN ACTUAL AND EMULATED WIRELESS NETWORKS

**by
Curtis Salley Jr.**

Wireless networks are becoming more popular in all types of environments from home office, to business infrastructure, to mobile computing. As more of these networks flourish, so does the attraction for hackers. The thesis examines the bandwidth of wireless networks and the effects of intrusions in three different scenarios. First, it looks at a wireless network that is connected to a wired infrastructure. Next, the thesis examines a wireless ad hoc network and how bandwidth plays a valuable part in the communication with neighboring nodes. And finally, the thesis attempts to simulate a wireless network by using a bandwidth configuration method on a dynamic switch.

The thesis looks at the three different scenarios in two ways. First, it examines the bandwidth utilization without any intrusion attempts. This was done in order to obtain a baseline for analysis. Secondly, it introduces intrusions into the scenarios and examines the effects. Data is collected for both types of scenarios and compared to determine if there is a noticeable effect on the utilized bandwidth.

**COMPARITIVE STUDIES OF NETWORK TRAFFIC
IN ACTUAL AND EMULATED WIRELESS NETWORKS**

**by
Curtis Salley Jr.**

**A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering**

Department of Electrical and Computer Engineering

May 2003

Blank Page

APPROVAL PAGE

**COMPARATIVE STUDIES OF NETWORK TRAFFIC
IN ACTUAL AND EMULATED WIRELESS NETWORKS**

Curtis Salley Jr.

Dr. Constantine Manikopoulos, Thesis Advisor
Associate Professor, Department Electrical and Computer Engineering, NJIT

5/21/2003

Date

Dr. George Antoniou, Committee Member
Professor, Department of Computer Science, Montclair State University

5/21/2003

Date

Dr. Bin He, Committee Member
Senior scientist, XPRT Solutions Inc.

05/21/03

Date

BIOGRAPHICAL SKETCH

Author: Curtis Salley Jr.

Degree: Master of Science in Computer Engineering

Date: May 2003

Date of Birth:

Place of Birth:

Undergraduate and Graduate Education

- Master of Science in Computer Engineering
New Jersey Institute of Technology, Newark, NJ, May 2003
- Bachelor of Science in Computer Engineering
New Jersey Institute of Technology, Newark, NJ, August 2001

Major: Computer Engineering

To my beloved wife Danielle

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Constantine Manikopoulos, who not only served as my research advisor, providing valuable and countless resources, insight, and guidance, but also constantly gave support, encouragement, and reassurance. Special thanks goes to the many graduate students who worked side by side with me during my research.

TABLE OF CONTENTS

Chapter	Page
1 INTRODUCTION.....	1
2 NETWORK INTRUSION SIGNATURES AND ANALYSIS.....	2
2.1 The TCPdump tool.....	2
2.2 The Mitnick Attack.....	5
2.3 Fragmentation.....	8
2.4 ICMP.....	10
2.5 Denial of Service.....	14
2.6 Summary.....	15
3 NETWORK SETUP AND TESTING.....	17
3.1 Initial Network Setup.....	17
3.2 Testing With Background Traffic.....	19
3.2.1 HTTP Traffic.....	19
3.2.2 FTP Traffic.....	20
3.3.3 Client Server Traffic.....	21
4 DATA COLLECTION.....	24
4.1 Baseline Background Traffic.....	24
4.2 Attack Traffic.....	26
4.2.1 Node Identification.....	27
4.2.2 Bandwidth Consumption.....	29
5 DATA ANALYSIS AND COMPARISON.....	31

TABLE OF CONTENTS
(Continued)

Chapter	Page
5.1 Analysis of Background and Attack Traffic in Same Scenario	31
5.1.1 Infrastructure Network	31
5.1.2 Ad hoc Network	33
5.1.3 Dynamic-Link Switch Network	35
5.2 Comparison of the Different Scenarios	37
6 CONCLUSION	38
7 USER MANUAL	39
APPENDIX A INFRASTRUCTURE NETWORK	43
APPENDIX B AD HOC NETWORK	44
APPENDIX C DYNAMIC-LINK SWITCH NETWORK	45
REFERENCES	46

LIST OF FIGURES

Figure	Page
2.1 Sample Packet	4
2.2 Attacker and Victim Nodes	7
2.3 Smurf Attack	12
2.4 Tribe Flood Network Attack	13
3.1 Infrastructure Setup	17
3.2 Link Info	18
3.3 Server Program	21
3.4 Client Program	22
3.5 Client Server Communication	23
4.1 Windump	25
4.2 Ethereal	26
4.3 File name request	27
4.4 Scan Options	28
4.5 File Locator	28
4.6 FLOODz Interface	29
5.1 Infrastructure Background Data	32
5.2 Infrastructure Attack Data	32
5.3 Ad hoc Background Data	34
5.4 Ad hoc Attack Data	34
5.5 Dynamic-Link Switch Background Data	36

LIST OF FIGURES
(Continued)

Figure	Page
5.6 Dynamic-Link Switch Attack Data.....	36

CHAPTER 1

INTRODUCTION

The objective of this thesis is to show the effects of intrusion attempts on wireless networks. Bandwidth analysis is used to determine how the network is impacted. First, I explain the tools used to understand network traffic. Then, different types of attacks are examined and their signatures are identified in the traces. Next, I demonstrate what was done during my research and the type of analysis used.

Normal traffic is captured in order to give a baseline for comparison. Then attack traffic is added and more traffic is collected to determine the effects. There are many different types of attacks used during the simulations and their effects are different on the network. There are several wireless setup tested including a simulated network using a dynamic-link switch. I examine three different types. The first type that is examined is infrastructure. This is a setup where you have wireless network as a subnet in a wired environment. This allows nodes in the wireless subnet interact with the nodes in the wired subnets. Secondly, I examined a wireless ad hoc network. In this setup the access point is not used as the central point of communication for nodes. The nodes seek each other out and communicate independently. And the final network topology used was a dynamic-link switch, which can simulate a wireless network. Varying the available bandwidth during testing does this.

Comparisons are done between the three different setups with respect to the difference between their baseline and attack traffics.

CHAPTER 2

NETWORK INTRUSION SIGNATURES AND ANALYSIS

2.1 The TCPdump Tool

In this chapter I will present many different aspects of Intrusions, Intrusion Signatures, and Analysis. The beginning will be broader to give a general flavor of the subject. Then I will tighten the focus to give a deeper understanding on a given topic. This will be shown in a clear format within the progression of this paper.

This being graduate level work I will not get too involved in the basics. I just want it to be understood that the foundation upon which the theory is about to be represented is TCP/IP. I do not have a mastery of this subject, yet my understanding is adequate enough to carry on an intelligent conversation/paper. With this foundation laid, I will now start relevant information on my topic.

TCP dump and Windump are tools that help people understand the traffic on the network. It is important to understand these items because they are the foundation that will help you get a quick understanding of other commercial tools that may be available. The focus at this point is on the UNIX TCPdump. The command used to run it is `tcpdump`. This command reads the traffic from the network interface, usually the default interface, and displays it on the screen. The problem with this simple format is that there is a lot of information that will show on the screen and it can be very hard to follow because of the rapid change. For this reason there are many different commands that can change this default display.

Filters could be used to enable `tcpdump` to only collect the data desired. For example, say we were only interested in TCP records. We can just specify that we want

TCP. What the filter does, or has the option to do, is check the field(s) in the IP datagram that we may want checked and look specifically for what we want, in this case TCP. The command to do this is `tcpdump `tcp'`. This is just an example of one specific field. This process can easily become very complex, as well as restrictive, when different combinations of fields and traits are sent. TCPdump also has a way of letting us know that a filter is stored in a file. This is good because now we do not necessarily have to write the total command. The option is `-F filename`. This *filename* is where the filter is located.

Again the default behavior of TCPdump is to put the output onto the screen. If someone is here and looking for a specific record this may be acceptable. But, most of the time there is no one physically monitoring this information; it is data being collected for later analysis. During this period we may want data to be collected in binary format, sometimes called raw output. In fact, the screen output has been converted from binary format to a human-readable format. For the purpose of analysis binary format is best because it stores all data, not only data for the screen output. The command to collect data in a binary format is `tcpdump -w filename`. To read this data, use the command `tcpdump -r filename`.

Another option that must be considered is the amount of data tcpdump will collect. The entire datagram is not collected usually. This is mainly due to the great volume this would generate. The header is normally what is collected. The standard number of bytes is 68. The reason for this number is shown in the picture below.

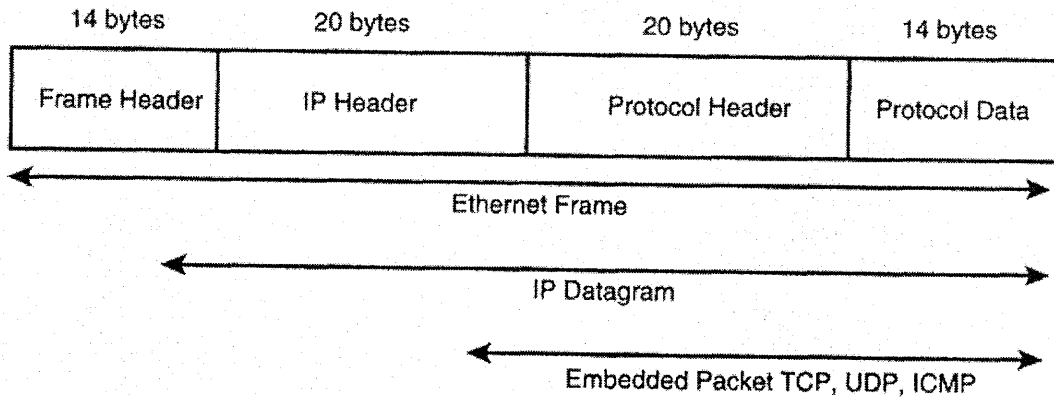


Figure 2.1 Sample Packet

The length can be set by the command `tcpdump -s length`, where *length* is the number of bytes.

Now I will describe three types of TCP intrusion attempts. The first is a port scan. There are many different reasons why this may be done. The main reason, however, has been determined to be that someone may want to know if a particular host, or group of hosts, have a service. Once this service is found on a host, a hacker may try to exploit the possible weakness of this service. The fact that the ports are being scanned is obvious from the TCPdump. A hacker that attempts this believes that no one is monitoring this network or that the host being attacked has no way of being traced back. A person who reads the TCPdump and understands how TCP works can easily notice this act.

The second TCP attack is another scan. The reason I call these attacks TCP attacks is because this is the protocol on which they travel and, in fact, the strange behavior from normal behavior of TCP is the signature that helps categorize them as such. In the example in one of the books, a look at the TCPdump may show a host is trying to see that hosts on a particular subnet(s) has telnet available. But again, with an understanding of TCP one can notice something is wrong. TCP has a 3-way handshake

before a connection is made. During this process no information, in the form of data packets, is sent. The TCPdump in this case has 4 data packets with each scan. These packets are not dropped instead they are added/included to the data after the connection is complete. This is a way of going around an Intrusion Detection System. Because data is only checked after the connection is complete.

The third is the concept of hijacking a TCP session. TCP is considered to be a safe protocol by many. This is mainly because of the need to establish a session and other parameters needed in order for the exchange of data to occur. Yet there is software that can sniff this session and take it over. It is easy for them to see users IDs and passwords that are encrypted. Once a session is established the only items needed to authenticate a host is IP number, port numbers, sequence numbers, and acknowledgement numbers. A sniffer can easily monitor the session and get this information and take over the session.

2.2 The Mitnick Attack

Now I will give a short review of one of the most famous system intrusions. This was when Kevin Mitnick successfully hacked into Tsutomu Shimomura's system. There were two different techniques used in this attack. They were SYN flooding and TCP hijacking. The SYN flooding was a method used in order to stop a host from transmitting. While this was done TCP hijacking was used to allow Mitnick to assume the identity of that host. The key was that he noticed a trust relationship between two computers. Once he saw this fact, he used it to exploit that relationship to the best of his ability.

Earlier in the paper I mentioned TCP hijacking so anyone reading should be able to follow that concept at this point. I mentioned the establishment of connections earlier,

but now I will explain a little more in order to help others follow my explanation of this attack just in case they do not fully understand TCP. A connection in TCP is established via a 3-way handshake. The sender sends a SYN packet. The receiver replies with an ACK. Then the sender replies to this reply and a connection is formed. These connections, or sessions, are kept in a database. The size of this database is limited. Once the database is full it cannot create any more sessions. Therefore when SYN flood is setup by an attacker they try their best to create as many requests for the connection as possible. The key is that they do not want to complete the connection, just flood the queue. Of course each connection has a certain time limit. Once this time limit is reached the connection, or attempt, is dropped. The key is, however, in the fact that once the queue is full it can be kept full.

Mitnick was smart. He did not want to be found. His code included a fake source address and destination address. He even went a step further to make sure that the address that he created was one that was routable to, but not active. What would happen is that when an address is entered the program would automatically ping it to make sure it was routable, but not active. Before he began his complex attack he did a lot of "intelligence gathering", called recon probes. Again, his program would send a SYN. If the host responded, he would send a RESET. Then he would decrement by one and do it again. The purpose of this was to get an understanding of the sequence number generator. With this understanding, he could then predict the behavior and take the place of the machine. In this particular case 128,000 was added to the sequence number to create the next one.

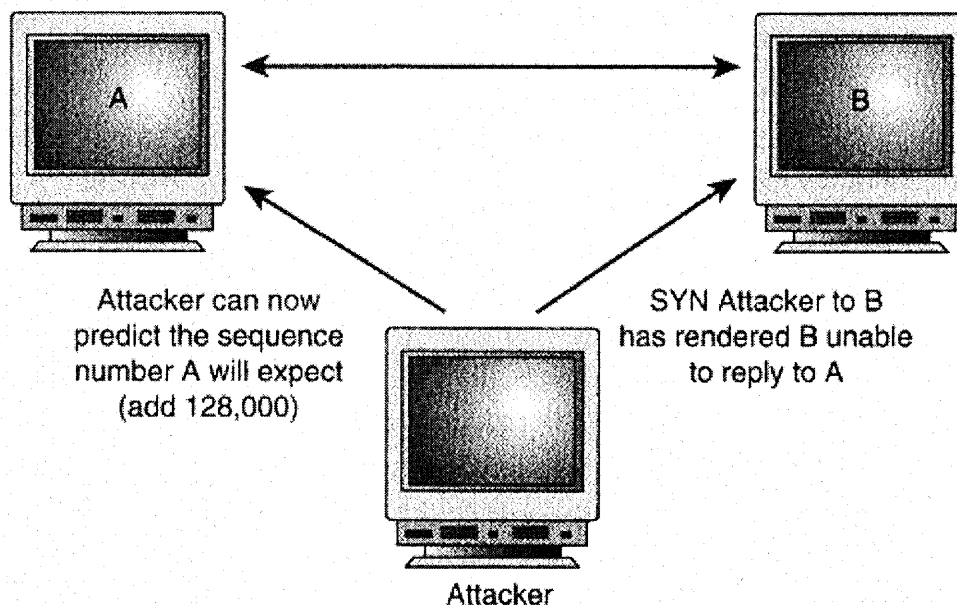


Figure 2.2 Attacker and Victims Nodes

A problem that allowed this attack to work is the fact that things were only checked once, usually during startup of the connection. Mitnick took the name of the computer and the system did not notice the change of IP addresses. A way to tell that spoofing is going on is when a route or MAC address changes during a TCP session. Also the TTL could be another indication. If it changes dramatically during the session spoofing may be occurring. Yet, this was a TCP attack. The IP address remained the same. Once Mitnick took over the appearance of the host, a server in this case, the key was only to generate the correct sequence numbers.

He sent a SYN packet to the host. The host is supposed now to respond with an ACK. He could not see this ACK because it went to the correct server. Under normal operation the server would have noticed that it never sent a SYN packet. Therefore it would send a RESET packet to the host. This is where the SYN flooding comes in. The true server is flooded and cannot respond to the SYN. The trusted connection, the ghost

server, then sends a UNIX command that tells the host to trust all computers and users of these computers. Once this command is sent, a FIN is sent to end the connection. The attack is now complete and Mr. Mitnick can log on to any computer. He did clear some of the SYN's that were flooding the true server. This was done in case other people tried to logon to the server and got an error message he knew it would arouse suspicion.

This attack could have been prevented at many steps along the way. A firewall, with the proper configuration or a good packet-filtering router is an easy ways to prevent this. Also it is good to verify all information about the connection periodically during the connection, not just once. In looking at the TCPdump, we could have noticed the initial sequence number incremented by one, which is not done by a sequence number generator. Other obvious things were mentioned earlier.

2.3 Fragmentation

It has been said that fragmentation has been used to both hide and carry probes and attacks. Not all Intrusion Detection Systems and packet-filtering devices reassemble packets. So if the signature is carrier over several datagrams they may not be detectable. This is the reason I will discuss fragmentation at this time.

Fragmentation is normally done when an IP datagram is traveling through the network (Internet) and the MTU of the router/network is smaller that the packet. At this point the packet must be broken down into pieces. Each of these fragments must reach their destination. As these fragments travel they can come across a network with even a smaller MTU and be fragmented again. It is up to the final destination to reassemble the packet. Each fragment must carry specific information in order to allow the reassembly to

occur correctly. Some of these important parts include the fragment IP, the offset, and the length of data being carried by the fragment. All fragments, except the last, must also carry a more fragment (MF) bit to tell the destination that more data is coming. All this information is in the IP header.

The first fragment will carry the protocol information. So devices attempting to block fragments will block this first datagram but let the others through. This is because the device does not keep fragment ID. They look at each fragment individually and do not associate them with previous ones. Although this is not necessarily good practice, economically and time wise this is the better choice. To operate the other way would mean that each fragment must be examined and stored, which cost money. Then fragments must either be allowed or rejected, which takes more time and resources.

TCP dump can be used to view fragmented datagrams. The book shows an example of an ICMP echo request that was sent using three fragmented Ethernet packets. The word frag in the output to let you know that the packet is a fragment. Each piece has the same id, letting you know they are part of the same original datagram. A + is used to let you know that the more fragment flag was set and there are other fragments to follow. The last fragment has no +, yet it does have offset, as does the second one. TCPdump does have an option not to fragment a datagram (DF). When this is set a fragmented datagram is not allowed on the network. If one is no the network it is dropped. Either way an ICMP error message is sent back to the sender letting them know the MTU of their network so that the datagrams use that number at their maximum from the beginning.

Fragmentation has opened the door for many to carry out malicious acts. Some Intrusion Detection Systems cannot be configured to handle fragmentations. A good one,

however, correctly maintains the state, reassemble fragments and then gives you some sort of assessment. I will describe two of the most famous ones when I discuss Denial of Service. For now I will give a simple example. The TCP header can be fragmented in order to try to avoid detection. The book gave an example of a fragment tcp packet of 16 bytes. (We know the minimum is the 20-byte header.) The next packet had 4 bytes, the rest of the header. Then a final fragment followed with data. Some detection systems will not reassemble this packet, allowing it to go through.

2.4 ICMP

I chose to add the discussion on ICMP to my paper to give a good, well-rounded feel of the different ways that intrusions can occur. This is another simple protocol that was created in order to have a way to relay error conditions and grant simple request. It has been altered, however, for negative purposes and the effects have been seen on some networks. I chose not to get too deep in the theory of ICMP because many generally know it. I will give a review of mapping, then discuss the normal activity, and finally malicious activity.

When planning the strategy of an attack, mapping is a very crucial part. The main purpose, usually, of an attacker is to find live host in the network that you want to attack. This is important because if mapping is not done, a lot of unnecessary traffic can be generated on the network causing attention that could have a negative effect for the attacker. Echo request is the most common mappings used. The echo reply comes back from the target if they receive the request. An example of this is ping. Knowing this most administrators have blocked ping. This alone is not good enough because of some TCP

examples I have shown earlier. This mapping can be done using not only host addresses individually, but also broadcast addresses of networks, subnetworks, and routers when trying to find their subnet mask.

Normal activity for ICMP consists of actions and responses to these actions and error messages. Host unreachable is an example of an error message generated in response to a request to find a specific host for the IP address given. A router sends this response because the host obviously cannot. When a requested port on a host is not listening a port unreachable message is sent back to the requesting host. An administration-prohibited message may be sent to a host who tries to access another host but they are not on the access list of the router that needed to give this passage. A router may send a redirect if it determines that there is a more optimal path to the desired destination. Need to fragment is another ICMP message that is sent when a desired host cannot be reached because the datagram is too large and fragmentation cannot be done. This message is sent back to the sending host along with the MTU of the network so the datagram can be of that size or smaller when they are generated. And the final one I will discuss is the time-exceeded in-transit. This is generated in response to a TTL value that is lower than the thresholds of the current/next hop.

Now I will give examples of malicious attacks. The first one is the Smurf attack, which is displayed below:

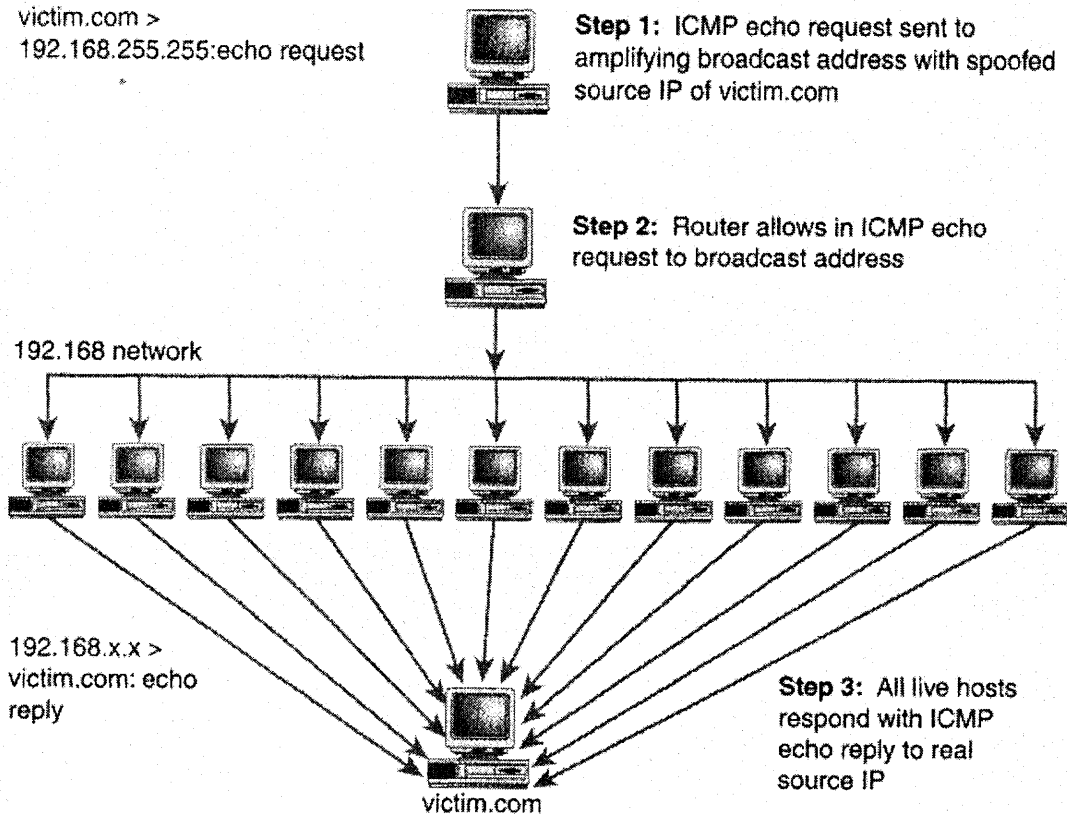


Figure 2.3 Smurf Attack

As shown in the picture a malicious host with the broadcast address of local network of victim.com created the ICMP echo request. All the hosts on this network will respond to victim.com. If there are a lot of host on this network then victim.com will be overwhelmed, causing a denial-of-service.

Another attack is Tribe Flood Network (TFN). This is also a denial-of-service attack. The difference with this attack is that distributor host are recruited for the attack.

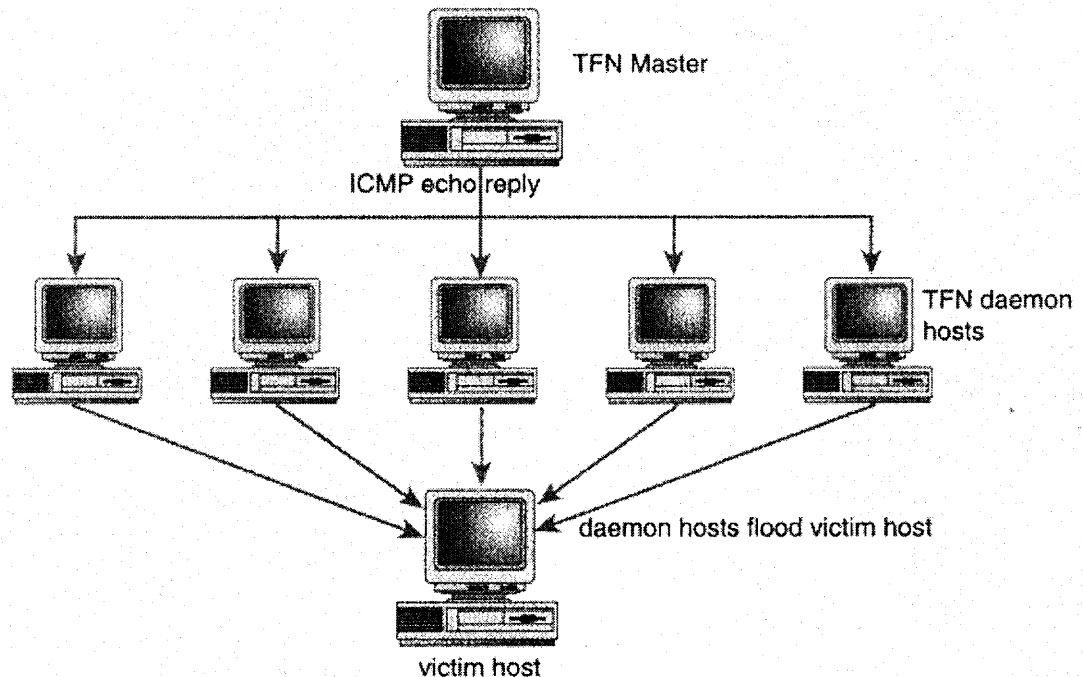


Figure 2.4 Tribe Flood Network Attack

For this attack there is a TNF master and daemon hosts. The different hosts are instructed by the master to attack a victim via echo replies. These attacks can actually be UDP flood, a TCP SYN flood, an ICMP echo request flood or a Smurf attack. The ICMP identification number field in the header of echo reply gives which action to take. The data portion is the part that sends the arguments.

The WinFreeze attack is characterized by the flooding of ICMP redirect messages. As the host attempts to reset their routing table it becomes overwhelmed and the performance of the host goes down. Loki is the final one I will discuss in this section and is probably the worst of the ones mentioned. It uses ICMP as a tunneling protocol to install Loki server. Once installed it will respond to Loki clients. These clients could send request for passwords, files, etc.

2.5 Denial of Service

Almost two years ago this was a very important topic. The main reason for this was the shut down of major Internet sites such as eBay and CNN. The main focus of these attacks is to stop the normal operations of the system. Spoofing is a key factor because it makes finding the true attacker very hard. By spoofing I mean creating a false source address. The denial of services will be explained in two categories. The first is brute-force attacks and the second contains well-known attacks with elegant kills.

Brute-force attacks are well known by many major Internet companies. Precaution should and can be taken by all, so if these attacks affect today's companies then they are partially to blame themselves. I say this because the remedies are published, easy to understand, and implemented by most. The key ones represented are Smurf and Echo-Chargen. I will not go into detail about Smurf because this was done earlier. Echo-Chargen is an attack that focuses on UNIX systems and uses them as amplifiers. It is described as being similar to an audience at a tennis match or ping-pong. UDP port 7 is used. This port echoes back whatever it receives. UDP port 19 is also used. You can send this port a character and it responds with a random string of characters. This port is a character generator. That's how the name Echo-Chargen was formed for this attack. A back and forth would go on consuming more bandwidth and CPU cycles as time continues. This can be prevented by not allowing packets to go to either of these ports.

Now I will discuss attacks that have a little more finesse. The first of these is the Teardrop. It exploits the point that the protocol stack is not the greatest at math, particularly negative numbers. Fragmentation is the medium used to cause havoc. What happens is that, for example, a fragment is sent with 20 bytes and offset of 0. The next

fragment is also 20 bytes with an offset of 10. In this case the operating system would have to rewind. The negative number can possibly be translated into large positive numbers. This may cause the operating system to write over information in other parts of the memory. If this is done a couple times, the system will be corrupted.

The Ping of Death is another attack with finesse. It also is a one-packet kill like Teardrop. Although they are packets of size one, they travel in about 30 datagrams. The code to get this to work is:

```
ping -1 65510 target.ip.address
```

This will crash an unsuspecting system. The key to get this to work is to create a packet that is of larger size than is allowed in an ICMP packet. This size is any number greater than 65,535. Because of fragmentation the larger ICMP packet can be sent. The receiver will not realize until total reassembly that the packet is too large. In fact, the last fragment is what can cause the problem. This is because if the offset is correct then it will be allowed to continue reassembly no matter how many packets are contained in this last fragment. The overflow can possibly cause the system to crash as well as other problems.

2.6 Summary

In this chapter I showed different aspects of attacks and signatures. The purpose was to bring to light a beginning of understanding so that some that may not know much, or anything, about what an attack or intrusion is and how it is carried out can have a little knowledge. I learned a lot about this subject and I hope to carry on my interest via research and study in other classes. I started with a general understanding of TCPdump and the attacks that manipulate TCP, especially “The Mitnick Attack”. Then I discussed

how fragmentation is used to help disguise attacks by spreading them over several datagrams. Another protocol used to masquerade attacks is ICMP. I explained this and showed a couple of examples as well. The next subject was Denial of Service. Many attacks seem to fall under this category and I only listed a few. This could have been the subject of the paper by itself because of the number and effects of the attacks that have been tried and still exist. Finally I discussed intrusion detection systems.

Overall this paper is a good summary of the many different aspects of the subject. This is, however, is not meant to be the overall summary of the entire different topic that could fall under the category of intrusions and detection. Hopefully enough explanation was given to the different topics I did choose to discuss. If they're any gaps to fill please feel free to seek myself or read the references cited at the end of this paper. Thank you.

CHAPTER 3

NETWORK SETUP AND TESTING

3.1 Initial Network Setup

The initial setup of the wireless network began with two laptops. They were used to verify connectivity both between each other and the wired network. The laptops came with Windows 95 already installed and this was sufficient for the tests that needed to be done at this stage of the project. D-Link wireless equipment was used and the access point was the first thing that was connected to the already existing wired network. A new subnet was created in order to distinguish the wireless network from the wired when necessary. A layer three switch was the tool used to inter-network the different subnets.

The design of the network was one in which the access point, which is the pathway through which the laptops communicate with the wired subnets, was connect to a hub, which connected to the layer three switch. Then the switch would forward packets to what ever network necessary (see Figure 3.1).

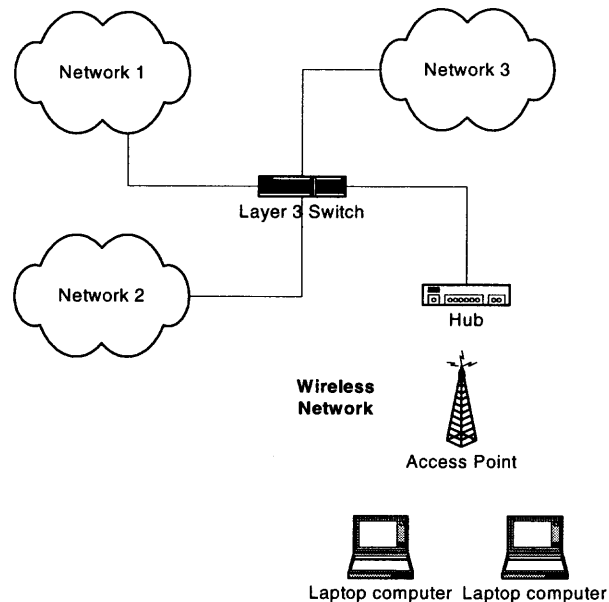


Figure 3.1 Infrastructure Setup

Internet Protocol addresses were assigned to each of the laptops. The subnet was already created and I made sure that the addresses given fell within the correct range. The D-Link software was next installed on the laptops in order to configure the wireless cards. This configuration was needed to enable communication with either the access point, which was connected to the wired network, or with each other. The D-Link products that were chosen followed the IEEE 802.11b standard. After the software was installed I went into the D-Link control utility to set the SSID. This is what the wireless cards used to verify that they are part of the correct network.

Under the configuration tab I made sure that the mode chosen is Infrastructure and the SSID is Coe259. The transmittal rate is set to fully automatic to allow flexibility regarding the flow of data. To verify that I can reach neighbor nodes I sent out ICMP ping packets. This is done from the laptops to all the nodes in the network. Once this test completed successfully from close proximity, I send a continuous ICMP ping (ping -t) and test the distance limitation of the laptops with respect to the access point. Under the Link Info tab of the D-Link control utility I can see both the link quality and signal strength (see Figure 3.2).

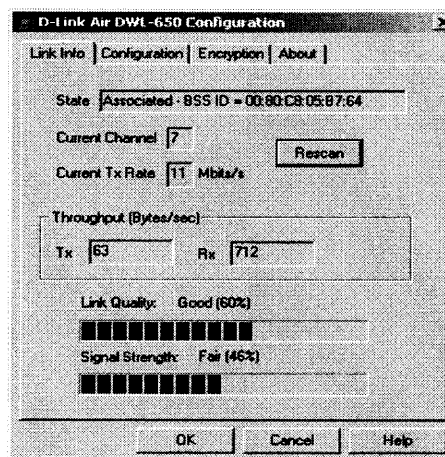


Figure 3.2 Link Info

From the computer lab I could go about 100 feet. Even when the link quality and signal strength was very low, there was no loss of ICMP ping packets. Once I stepped out of range, however, I was not able to communicate with the other nodes. This was a great test because it showed that the way to test packet loss was not just based on the distance the laptop was away from the access point, unless I stepped out of range completely. Once this was done for the two laptops another was added to increase the amount of nodes in the wireless environment. The same types of test were ran on the new node as well.

3.2 Testing With Background Traffic

In order to determine the effects of the attack traffic on the network there must be a baseline to represent the normal flow of traffic. I use background traffic to represent this necessary information. Three forms of background traffic were used and I tested each to verify that they would be sufficient and that they were compatible with what I want to do regarding the behavior of normal traffic flow. FTP, HTTP, and Client Server were the three choices selected.

3.2.1 HTTP Traffic

In the wired network background traffic was used in a similar manner in which I intended to use it. One such traffic was HTTP. This was generated via a request for a file that resided on a remote computer. The request was made through a web browser and the file was a picture, either bitmap or jpeg. It was rather simple to generate this traffic. All that was needed to be done was to input the ip address of the remote computer that has the

desired file and the file name. This information was put into the web browser address line and when go was pressed, or enter, the image was retrieved and displayed on the screen.

There were many images available and I wanted a continuous stream, so I created a script that would constantly request these image files. The script would request images one after the other with minimal delay all the way to about 100 different images. Then it would delete the images and start the request all over again. This would continue until I stopped the script. The script was created using a program call recorder, which allowed a user to either capture keyboard strokes and mouse clicks, or create a script in notepad. For this traffic, I created a program in notepad because it was much easier.

This script seemed to work very well, but it did not allow you use the computer for other tasks once it was started. The traffic seemed okay and the overall this process was not bad. There was no user involvement once the script was started which allowed for consistent results each time it was ran.

3.2.2 FTP Traffic

Another form of traffic that was generated across the network was ftp. This setup was very similar to HTTP traffic. The same files were requested from remote computers, just in a different manner. To test this type of traffic I used the DOS prompt to transfer files one by one. There were no problems with this task. Again, I tested the range to ensure that the traffic could travel even at the edge of link availability within the wireless network. Overall the tests were a success. Next I created a script to automatically run the ftp traffic. It worked in a manner similar to the HTTP script I had generated for the previous test. The files were transferred over the network, deleted, and then they were

requested again. Minimum delay was used to generate higher amounts of traffic on the network.

3.3.3 Client Server Traffic

Automatic background traffic generation is very important because the focus can be placed on other factors that may be more relevant to your research. A Ph.D. candidate, Jun Li, had created a program that used Pareto's distribution to determine the message length. This was client server program in which the client sent messages to the server and the server would return messages to the client. The range and size of the messages varied, which made for a good example of traffic that would normally travel in a computer network environment.

There were two programs, client and server, that worked in conjunction with each other. The server program had to be running before the client program was started. To start the server program was simple. All that was required was that the user specified on which port the server would be listening, figure 3.3 show an example of this.

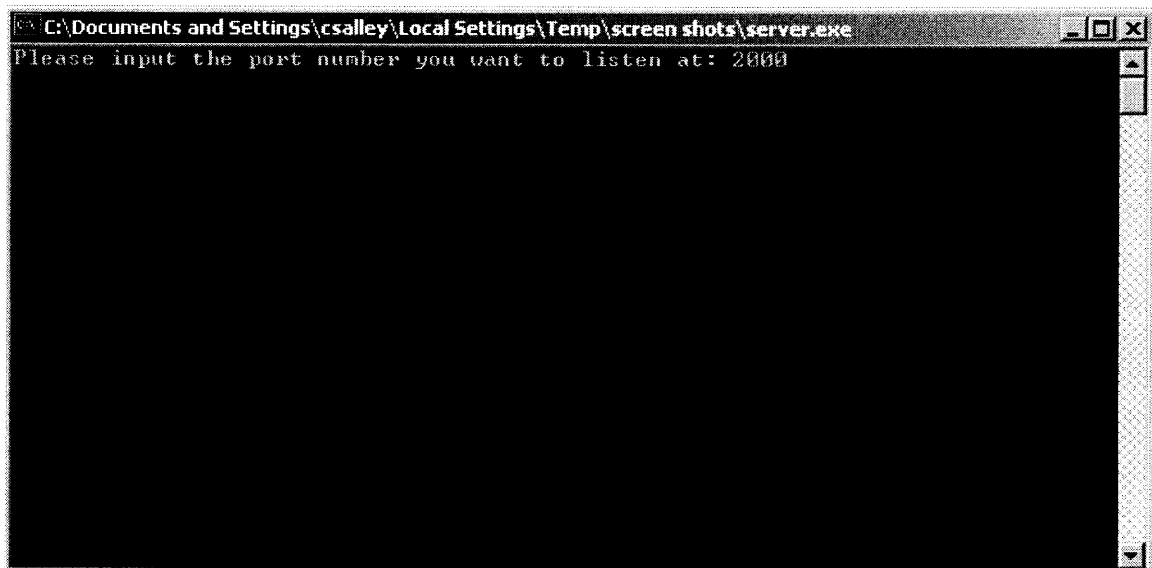
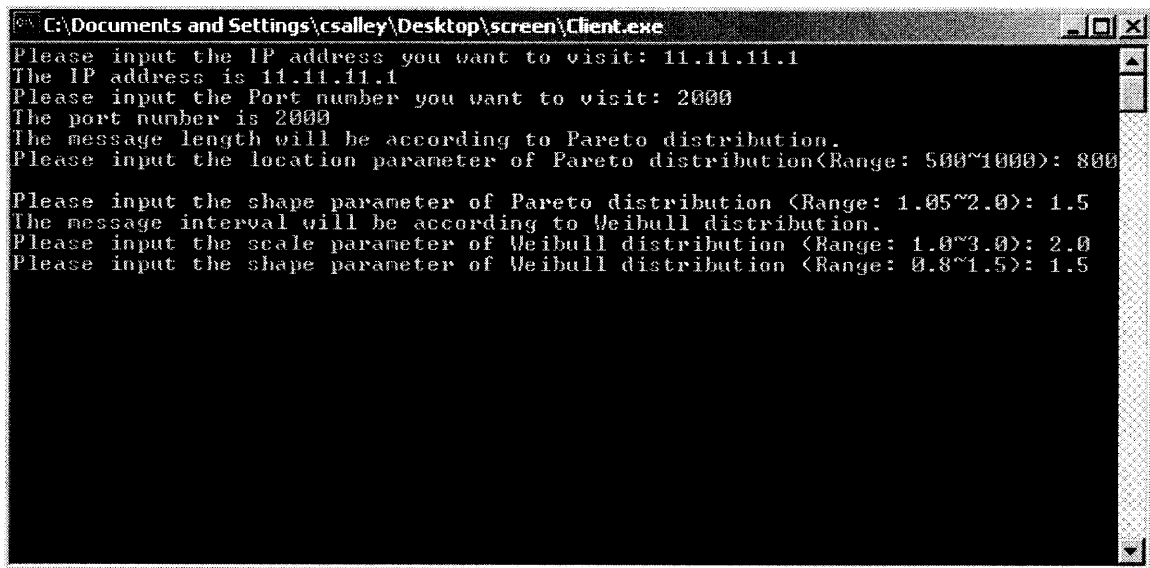


Figure 3.3 Server Program

Once the server program was started the client program then had to be started. There were a few more steps in getting this program started. There were a series of 5 questions that had to be answered before traffic was generated across the network. First the IP address of the server program needed to be inserted. Next, the port on which the server was listening had to be specified. After that there were three questions regarding the parameters that needed to be set in order for the Pareto Distribution message pattern to be set. Figure 3.4 shows the sequence along with a sample users input.



```
C:\Documents and Settings\csalley\Desktop\screen\Client.exe
Please input the IP address you want to visit: 11.11.11.1
The IP address is 11.11.11.1
Please input the Port number you want to visit: 2000
The port number is 2000
The message length will be according to Pareto distribution.
Please input the location parameter of Pareto distribution (Range: 500~1000): 800
Please input the shape parameter of Pareto distribution (Range: 1.05~2.0): 1.5
The message interval will be according to Weibull distribution.
Please input the scale parameter of Weibull distribution (Range: 1.0~3.0): 2.0
Please input the shape parameter of Weibull distribution (Range: 0.8~1.5): 1.5
```

Figure 3.4 Client Program

The directions are rather simple and very user friendly. Once all the requested information is given, message packages of random size are sent to the server. Then the server responds with random message packets of its own. This communication between the client and server continues until it is stopped by the user. The great thing about this program is that it is scalable. There are many different ways this can be setup. One example is where you can have many clients talk to one server. Another is where you can have several servers on different nodes or even on the same node listening at different

ports. In order to generate more traffic more clients could be added. Figure 3.5 shows a sample of communication between the two programs. Here we see the information that is received by the server. The same type of information is generated on the client side.



Figure 3.5 Client Server Communication

CHAPTER 4

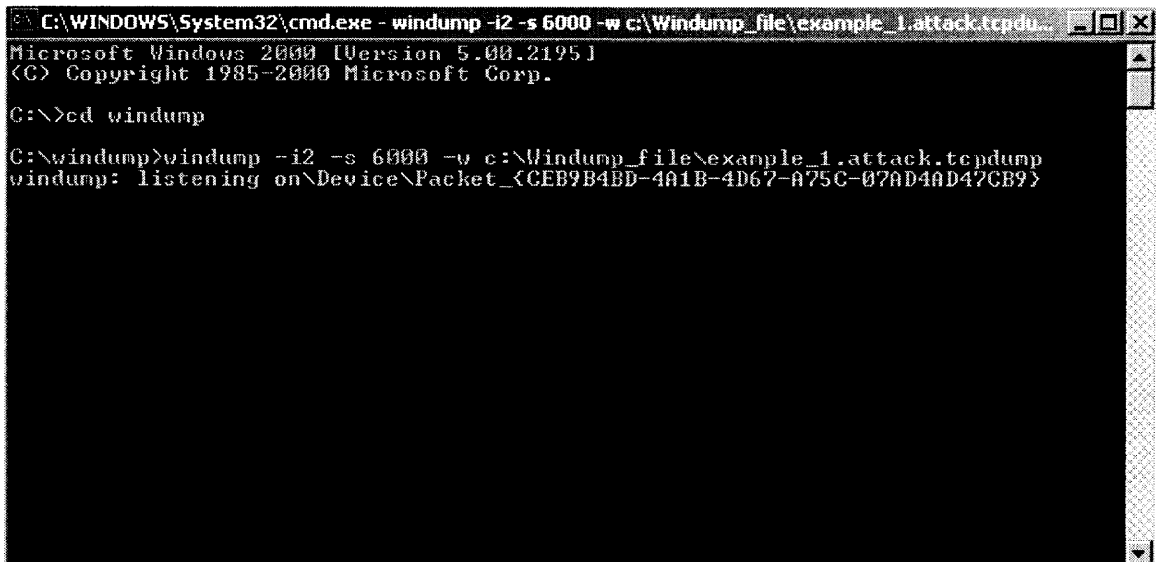
DATA COLLECTION

In order to determine the effects of intrusions on the network, I had to first collect data. This data fell into two categories, baseline traffic and attack traffic. The baseline was needed as a frame of reference that would be used to determine if the attack traffic could be noticed. It would represent the normal traffic that would occur on the network. The collection of data was done in all three scenarios, infrastructure, ad hoc, and dynamic switch. Attack traffic was also captured in the same manner the baseline traffic was captured. The comparison of the two different types of traffic would occur, not only within the scenario collected, but also with the other scenarios.

4.1 Baseline Background Traffic

The baseline traffic would represent the normal traffic that occurs across the network. In a live network environment, this would be file transfer traffic, HTTP traffic, Internet traffic, etc. As stated in chapter 3, I tested three different types of traffic on the network. I collected the data on each network type and compared them side-by-side. The traffic that seemed to be the most useful was the client server traffic. There were many reasons it was a better choice and the main ones were mentioned earlier. They included scalability and ease of use.

The way in which traffic was collected was rather simple. I would run a program called Windump and it would listen for incoming packets and collect all the data that came. It would run in the background and needed no assistance once started. One simple



```

C:\WINDOWS\System32\cmd.exe - windump -i2 -s 6000 -w c:\Windump_file\example_1.attack.tcpcdu...
Microsoft Windows [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>cd windump

C:\windump>windump -i2 -s 6000 -w c:\Windump_file\example_1.attack.tcpcdu
windump: listening on\Device\Packet_{CEB9B4ED-4A1B-4D67-A75C-07AD4AD47CB9}

```

Figure 4.1 Windump

command had to be executed in order for the data collection to begin. Figure 4.1 shows an example of the necessary command. The key was to specify the file that would store the data. The program would be started, and then I would start the client and server programs on the nodes that would be involved in the data collection. Data collection would go on for some while before being stopped.

It was important to make sure that traffic was being sent and received by each node that participated in my research. This is what made the client server programs so important. I created a server on one node and made all the other nodes clients. In this manner all the nodes would be communication. The clients would send information to the server and the server would then return some information to the client. This two-way communication would continue until I stopped it.

The next step was to read the data and get statistic from it. I used a program named Ethereal to read the information that was collected by Windump. Figure 4.2 showed the main screen of Ethereal. All the important packet information could be found

here. This included source address, destination address, sequence number, time, etc. Ethereal also offered summary statistics such as packet count, average packets/sec, average bytes/sec, elapsed time, etc. The information obtained from the data that was collected was sufficient to determine the baseline for my comparison. This test was repeated for all the scenario types.

The screenshot shows the Ethereal interface with a list of captured packets. The first packet is an ARP request from the local interface to the destination. The subsequent packets are TCP RST, ACK responses from the destination to the source IP (172.16.2.11).

No.	Time	Source	Destination	Protocol	Info
1	0.000000	D-Link_00:29:a2	D-Link_da:4e:42	ARP	20.20.20.1 is at 00:05
2	0.003728	172.16.2.11	20.20.20.4	TCP	http > 43995 [RST] Seq
3	0.316359	172.16.2.11	20.20.20.4	NBNS	Name query response NB
4	0.326559	172.16.2.11	20.20.20.4	TCP	1466 > 3550 [RST, ACK]
5	0.331993	172.16.2.11	20.20.20.4	TCP	579 > 3551 [RST, ACK]
6	0.333236	172.16.2.11	20.20.20.4	TCP	465 > 3552 [RST, ACK]
7	0.334746	172.16.2.11	20.20.20.4	TCP	533 > 3553 [RST, ACK]
8	0.336228	172.16.2.11	20.20.20.4	TCP	1520 > 3554 [RST, ACK]
9	0.640898	172.16.2.11	20.20.20.4	TCP	1466 > 3555 [RST, ACK]
10	0.642462	172.16.2.11	20.20.20.4	TCP	579 > 3556 [RST, ACK]
11	0.644024	172.16.2.11	20.20.20.4	TCP	465 > 3557 [RST, ACK]
12	0.645567	172.16.2.11	20.20.20.4	TCP	533 > 3558 [RST, ACK]
13	0.647044	172.16.2.11	20.20.20.4	TCP	1520 > 3559 [RST, ACK]
14	0.942454	172.16.2.11	20.20.20.4	TCP	1466 > 3560 [RST, ACK]
15	0.943350	172.16.2.11	20.20.20.4	TCP	579 > 3561 [RST, ACK]
16	0.944575	172.16.2.11	20.20.20.4	TCP	465 > 3562 [RST, ACK]

Frame 1 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:05:5d:00:29:a2, Dst: 00:05:5d:da:4e:42
Address Resolution Protocol (reply)

```

0000  00 05 5d da 4e 42 00 05 5d 00 29 a2 08 06 00 01  ..].NB.. ]).
0010  08 00 06 04 00 02 00 05 5d 00 29 a2 14 14 14 01  ..... ]).
0020  00 05 5d da 4e 42 14 14 14 04 00 00 00 00 00 00  ..].NB..
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....

```

Figure 4.2 Ethereal

4.2 Attack Traffic

Once the baseline was established, I worked on the different type of attacks that I would attempt to use. I thought of several approaches. Attacks come in many forms and fall under many categories, but the one I chose to focus on was Denial of Service in the form of bandwidth consumption. In order to implement the attack I had to use two steps. The

first step was to determine what nodes were reachable and available for attack. The second step was the actual attack itself.

4.2.1 Node Identification

In this, the first step, I represented a hacker who wanted to know what nodes were available to be accessed. This information could have been found in many different ways. I could have determined just the IP addresses of the nodes, what operating system they were running, what ports were open, etc. All of the above choices seemed very valuable; therefore I made them all available to be chosen. This was done through a script I created which asked a few simple questions regarding the type of information that you wanted to capture. The script not only asked you what information you wanted, but it also saved the information in several different files.

The script was created through a program called recorder and used the services of two other programs called Nmap and Windump. The recorder program would have to be started by the user and then the correct script would have to be loaded. It started off by asking you for a name that would be used label all the files that would be collected as shown in Figure 4.3. All the screens were similar to this one. The user would input the

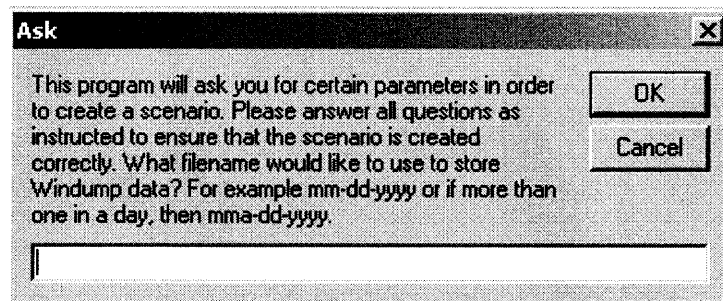


Figure 4.3 File name request

information in the box provided and then pressed OK. The next screen would ask for the ip address of the network or particular node you would like to explore. The user would input the necessary information again and pressed OK. Then it would give the user a list of options regarding the type of scan it they would like to perform. Figure 4.4 shows this screen and as you can see there are eight different choices to choose from. The next screen would vary depending of the type of scan you had chosen.

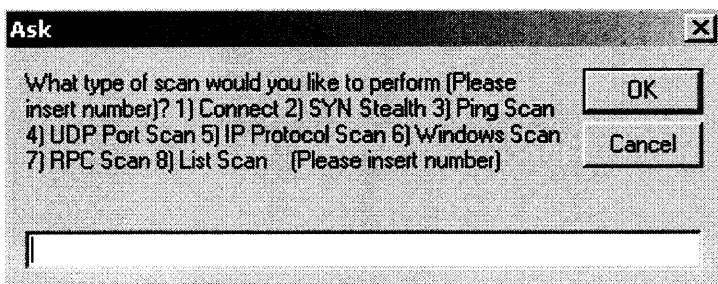


Figure 4.4 Scan Options

After a few more screens that helped to better define the type of attack that was done, the script displayed a final screen, which gave the location of the different files that were saved (see Figure 4.5).

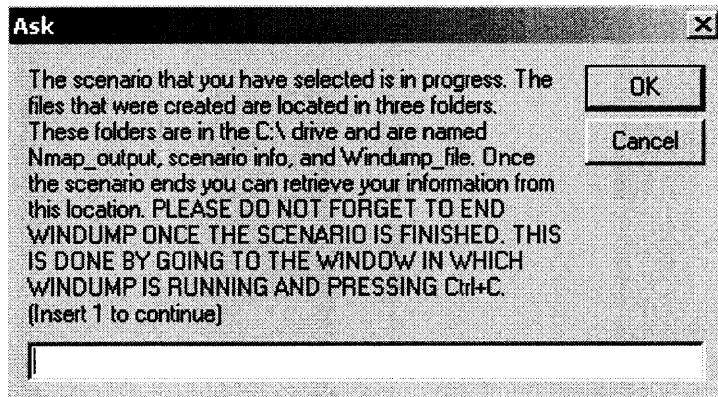


Figure 4.5 File locator

Each file held different information. The Nmap_output file held the results of the scan that was done. The scenario file held all the information that you requested to be done in

the attack. The Windump_file held all the packets that came to the node, which were the results of the Windump program. All the different files still had the same name just different extensions. The information that was collected was then used for the next step of the attack process.

4.2.2 Bandwidth Consumption

Now that all nodes were identified with the necessary information, the next step of the attack process was to use this information to attempt to create a denial of service to the node, or nodes in question. There were many different tools available for me to use to generate enough traffic to get the results I wanted. My main package of choice, after many tests, was called Devine Intervention. This one package had many different tools that generated attacks in many different forms. It could attack a specific port of a node and could send packets of variable sizes in specified amounts, etc.

I used a tool named FLOODz from the Devine Intervention package. In this tool I am able to specify the ip address of the victim, the packet size, and the number of threads I would like to have sent. Figure 4.6 shows what the interface looks like. It is a rather

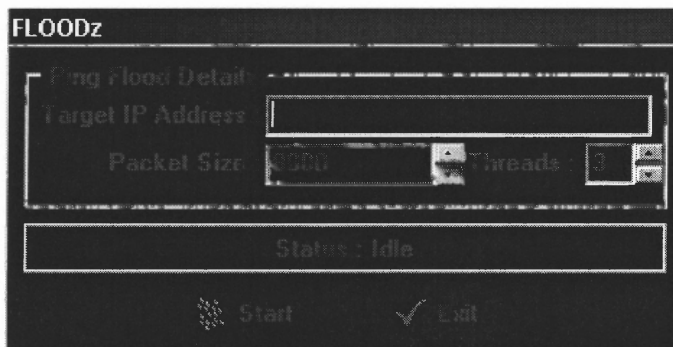


Figure 4.6 FLOODz Interface

simple program that is began by pressing start and ended by pressing halt, which is in the same position as start once the program is started. Packets of the size specified are continuously sent to the designated ip address causing the result of a lot of traffic being sent to the specified host. I ran this program in all three different scenarios and the data was collected. At the same time this program was running, the background server client programs were running. Therefore, the data is a combination of both background and attack traffic. The key now would be to determine if the results would show any difference on the bandwidth of the network.

CHAPTER 5

DATA ANALYSIS AND COMPARISON

5.1 Analysis of Background and Attack Traffic in Same Scenario

Two types of data were collected for each scenario. The first type was the baseline network traffic and the second type was attack and baseline traffic. The baseline network is a representation of what the normal traffic pattern would look like during normal operation. This would be the bar by which the attack traffic would be measured. An analysis of each type of traffic is done for each scenario and finally a comparison is done between the different scenarios.

5.1.1 Infrastructure Network

The first network on which I collected data was the infrastructure network. This network, which is shown in Appendix A, had one of the nodes in the wireless network setup as the server while the other nodes represented clients and sent messages to the server. The server would then answer with messages of variable length. Figure 5.1 below shows a sample data time plot. As you can see, the data remained relatively consistent throughout this period of the test and was in line with what I expected. There were no changes while the client server programs were running. The server node was on a wireless laptop, so I roamed the hallways near the lab to test the range. At no time did I go outside the range. The lowest I let the signal strength and link quality get was 6%. No packets were dropped and there was no delay noticed during the analysis of this data collected. The average traffic was 23 kbps, while the maximum value was 1.4 Mbps. It can easily be seen in the

figure that the traffic follows the same pattern through the collection, with only a few packets going far away from the average.

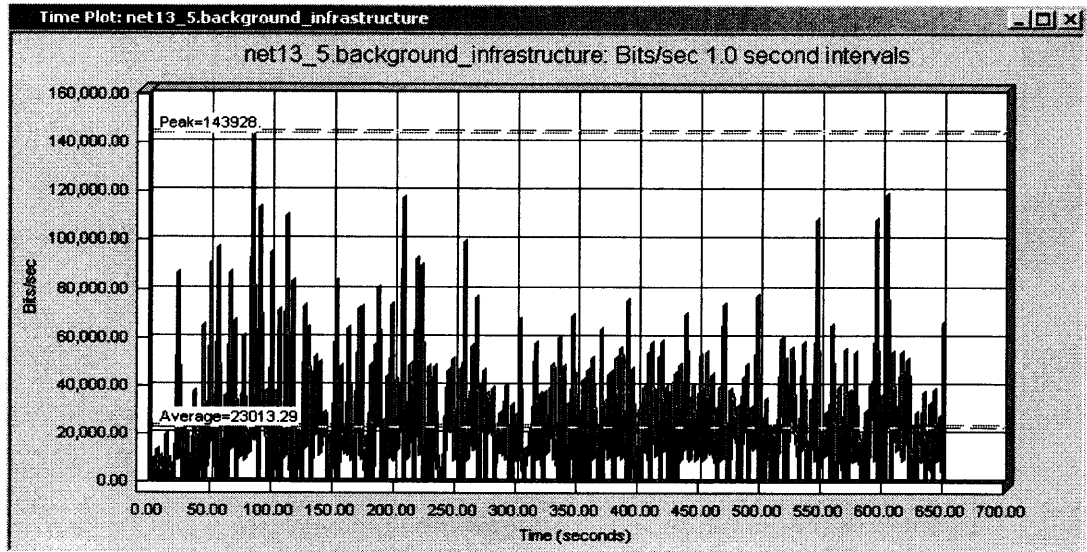


Figure 5.1 Infrastructure Background Data

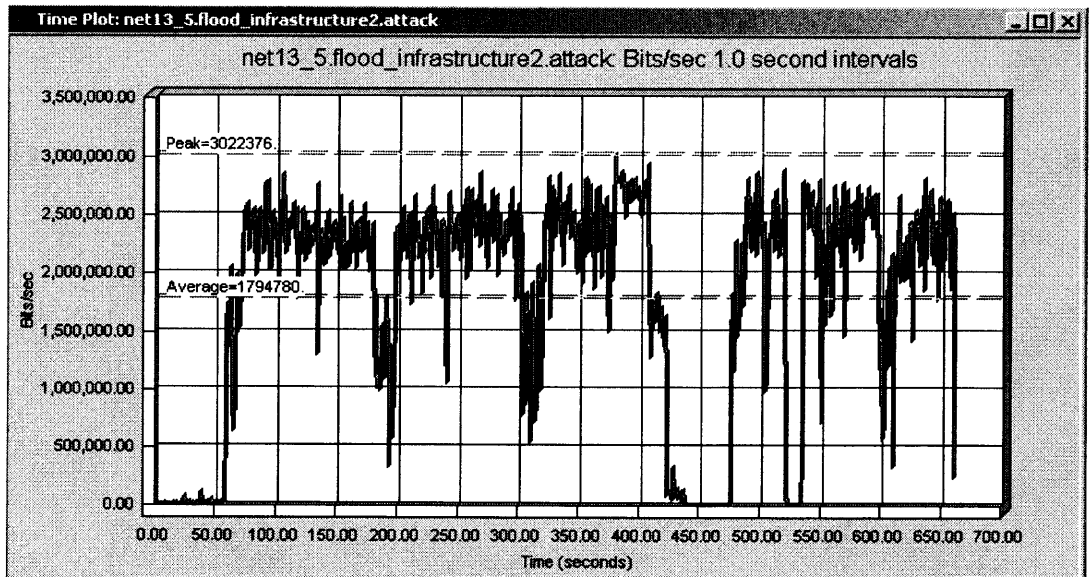


Figure 5.2 Infrastructure Attack Data

The attack traffic was collected in the same manner that the background traffic was collected. As stated in Chapter 4, I used the FLOODz program to implement the attack. Figure 5.2 shows the results of the data collected from the attack. Looking at the

figure there are two main things to notice. The first is that the average traffic was 1.8 Mbps. This average is very high compared to the background traffic and is a strong indication that there is suspicious traffic on the network. The second is the gaps in the data pattern. These gaps represent a loss of data throughput. The periods between 0 and 54 seconds and between 440 and 473 have 0 or minimum throughput of bits. The attack was so effective that it inhibited the flow of data during these, as well as other, periods.

5.1.2 Ad hoc Network

This network functioned without the use of an access point or communication with other networks. The network setup, which is in Appendix B, shows that the individual nodes interacted with each other independently. Each node scanned the different channels every 5 seconds for other nodes that had the same SSID. This was the determining factor on whether they could interact or not. Unlike when the wireless laptops were in infrastructure mode, there is not a link quality and signal strength meter to determine the values of these parameters. This made roaming a little more difficult.

Data was collected in the same manner it was collect for infrastructure. The only difference in this scenario is the amount of nodes that was generation the traffic. In both data collections there were only a total of three nodes. One of the nodes was the server while the other two nodes were the clients. Figure 5.3 shows that the data was somewhat consistent. The average was about 12 kbps, which is half that of infrastructure. But this was to be expected due to the reduced number of nodes. I roamed the halls with the server in the same manner used with the infrastructure scenario and tried to say within the

same bounds. Again the data was consistent throughout the collection and there were no surprises with the background traffic.

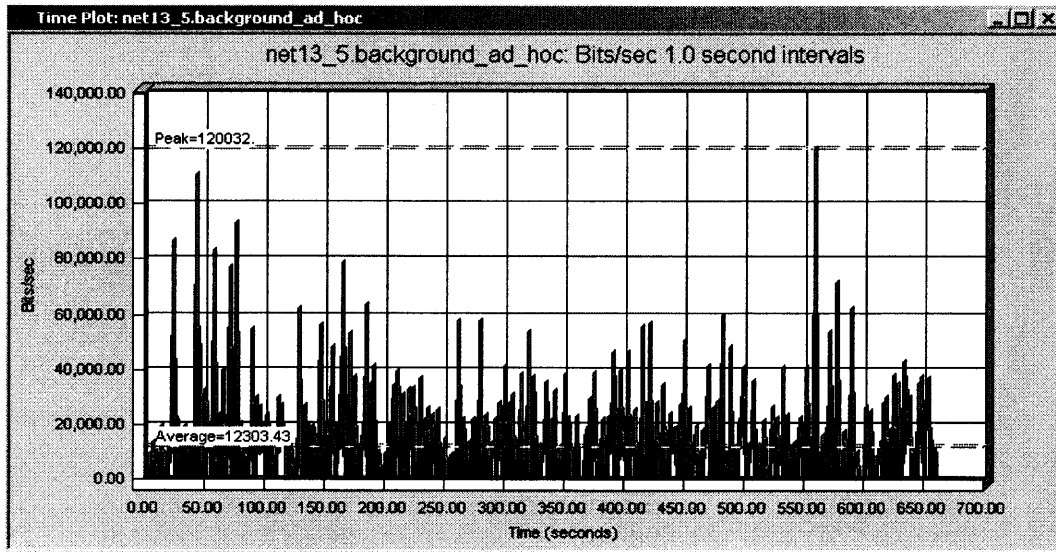


Figure 5.3 Ad hoc Background Data

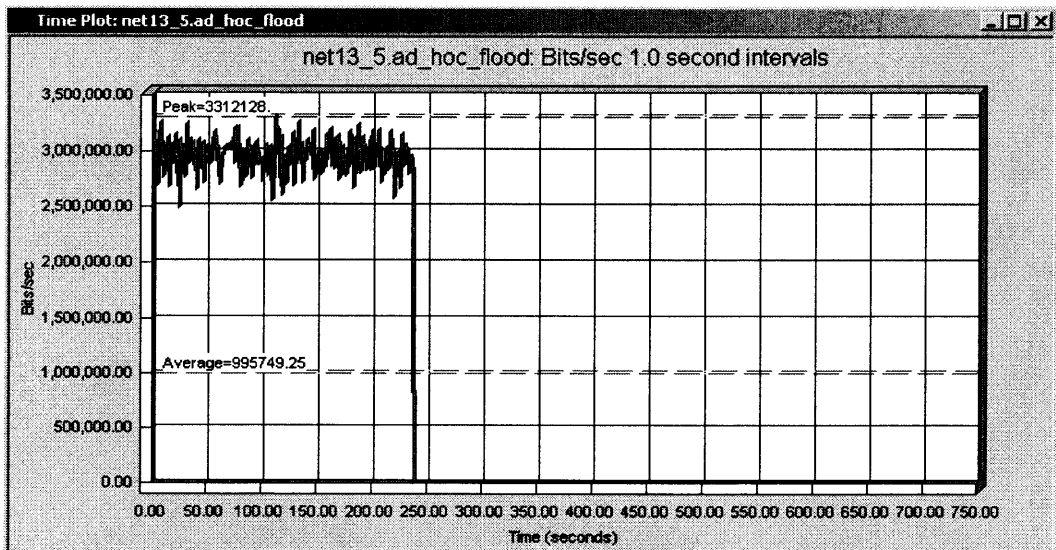


Figure 5.4 Ad hoc Attack Data

The attack also came from two nodes going to one. As Figure 5.4 shows, after 240 seconds the throughput goes to 0. This continues for a while and eventually comes back up (which is not shown in this figure). The attack crippled the ad hoc network for a

period of time showing that it was not as robust I would have expected. Others attempts at attacks using fewer packets eventually showed the same results.

5.1.3 Dynamic-Link Switch

In this network setup, which can be seen in Appendix C, the Dynamic-Link switch served as the gateway through which the different networks communicated. Software is supposed to be able to configure the bandwidth that is specified by the user. The problem with this setup is that there must be a problem with the program because through numerous experiments I was not able to determine the correct bandwidth that was setup to be allowed through the switch.

The way the switch worked, in order for packets to be allow through they had to have enough tokens to cover the total number of bits in the packet. There were three items that were configurable. They were the arrival rate of the packets, the number of tokens that could be held in the token buffer, and the number of bytes a token can support. The allowable bandwidth was supposed to be a simple multiplication of the arrival rate and the number of bytes per token.

Figure 5.5 show the data collected when I attempted to run only background traffic. This data was for the most part consistent with what I noticed the past. The average rate was 21 kbps, and since the setup was similar to the infrastructure setup, it did not bother me that the numbers were close. After this data was collected I then attempted to collect data for the attack traffic. This data is shown in Figure 5.6. I noticed the gaps in the test and tried to do some bandwidth calculations to allow a high bandwidth throughput. This is where the problems started with this scenario. My

calculations did not agree with the results I received. I checked with others who were working with the switch and they were having similar problems. We concluded that there is currently no way to determine, definitely, the allowable bandwidth once configured.

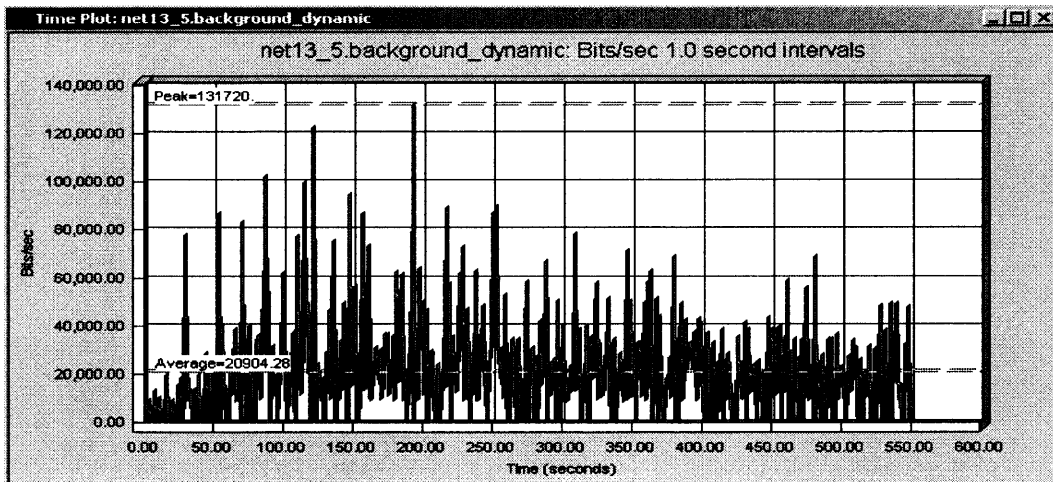


Figure 5.5 Dynamic-Link Switch Background Data

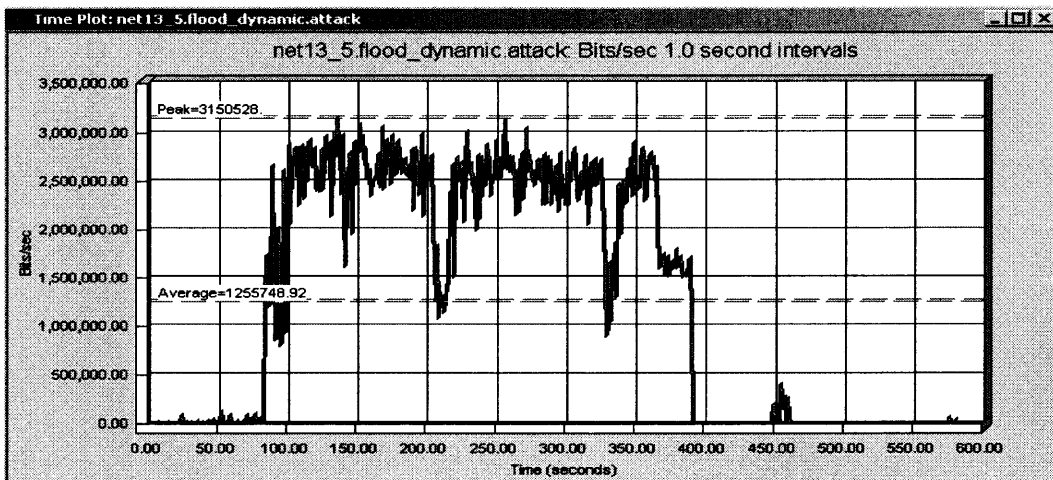


Figure 5.6 Dynamic-Link Switch Attack Data

Due to this conclusion I could not determine if the switch is functioning correctly or not in this scenario. Yes, just looking at the data I can see that a possible attack has occurred just from the variance from the background data and the attack data. But the accuracy of the data and the possibilities is not sufficient to come to a definite conclusion.

5.2 Comparison of Different Scenarios

In each of the scenarios it can easily be determined from the data collected that an attack has occurred. The variance in the data from the background traffic to the attack traffic shows that something did occur. More bandwidth was utilized in each of the scenarios and the data seemed to be consistent, at least in the first two networks. Overall the variance from background data to attack data was similar in all three different network setups, which was satisfying. They were not exactly the same, but they were not different by an obscure amount. The scenarios are consistent. More needs to be done with the dynamic-link switch, however, to ensure that it is correct in achieving its main purpose, bandwidth configuration.

CHAPTER 6

CONCLUSION

The research was a success. The goal was to determine if network intrusions had bandwidth effects of three different types of network setups. These setups were infrastructure, ad hoc, and dynamic-link switch. Through data analysis it was determined that the effects could be seen in all three different setups. There is some question on the validity of one network setup, dynamic-link switch, due to the fact that I was not able to confirm that the bandwidth configuration worked correctly. Keeping that in mind, the results were still consistent with what was found in the other network setups.

CHAPTER 7

USER MANUAL

This is a simple user manual that can be used to guide you through the many different steps I've done during this research. There are three different parts. The first part is setup. This involves making sure that the different both have the correct IP addresses and can communicate with each other. If you know the network to which the node belongs, check the IP address to ensure it is part of that network. This can be done from MSDOS.

On the command line of MSDOS type ipconfig. This will give you the IP address of the node. If the address is part of the correct network then try to ping the other nodes in the same network and in other networks to ensure connectivity. If it is not part of the correct network then you must give the node the correct address. This is done by pressing the right button of the mouse while it is over the *my network places* icon on the desktop. This brings up a list of options that can be selected. Click the work properties and this brings up a window. In this window there is an icon, which is labeled Local Area Connection. Right click this icon and again select properties. This action will bring up another window with at least one, maybe more, tab.

Under the general tab there are several components that can be selected. Highlight the component that reads Internet Protocol (TCP/IP). Then click the properties button in this window. Again another window will show which will allow you to change the IP address. Click the radio button next to the statement Use the following IP address. Input the correct IP address, subnet mask, and default gateway that will allow it to be part of the network. Once this is done click the okay button to close the windows, the first

window that was opened must be closed by pressing the X in the top right corner. Once again go to MSDOS and attempt to ping nodes in the same network and other networks.

The next step is to start the client server programs. Both if these programs are started from MSDOS. Find the location of the programs and get to that directory in MSDOS, the names `client.exe` and `server.exe`. Type the name of the program you would like to start, please remember that the server program must be started before the client program. If you type `client` and press enter on the keyboard, it will start the client program. This program will ask you several questions such as the IP address of the server, the port on which the server is listening, etc. The questions are rather simple and easy to follow. If you type `server` and press enter on the keyboard the program will only ask on what port would you like to listen.

In order to collect data Windump must be started to capture all the packets. This is also started in MSDOS. The command line should be pointing to the directory in which the program `windump.exe` is located. Then you must type “`windump -i2 -s 6000 -w c:\folder\filename.tcpdump`”. In this example *folder* represents the folder that will store the file and *filename* represents the name of your file. Once you press enter on the keyboard, the program will start listening for any traffic that will come to the node. To stop the program, which can be done at any time you must press `Ctrl + C`. The data will be automatically stored in the file specified. To read the data you must open a program called `ethereal` and load the file from the location where it's stored. The `ethereal` program either has a shortcut on the desktop or in the programs menu of the windows start icon.

The next item explained is using the scan script I created using recorder. To use this script is rather easy. The only problem is that in order to transport it to another node

involves a little manipulation of the program. It is currently setup on one of the nodes and the instructions give below apply to this computer. The instructions will be the same for any other node you transport it to, but you must make sure that the same folders exist in the same areas on the new node. The recorder program can be retrieved from either the shortcut on the desktop or under programs in the windows start menu. Once the program is open you must press the load button. This will allow you to find the program you would like to run. This will automatically bring up a folder called Curtis. In this folder there is a program titled Nmap scenario. Click on this program and press ok. This will load the program into the recorder application.

Pressing the replay button starts the script. This will start the first of many windows in which you will have to put information. The first window will request the filename in which all the other files will be saved. The next screen asks you the IP address of the network you would like to scan. Then there are several screens that will follow requesting information regarding what type of scan you would like to run. In order to understand the many different possibilities of attacks that are available, read the help file of the Nmap program, which has a short cut on the desktop. The last screen of the program gives you the location and names of the different files that are created. This will help you find the files when you are ready to retrieve them.

Now the program works on its own to complete the requested scan. It does this by opening and starting windump, opening and starting Nmap, and saving all the scenario information to the appropriate files. Once the scenario has stopped you must press Ctrl + C to stop windump. All the files that were created are now available for your viewing.

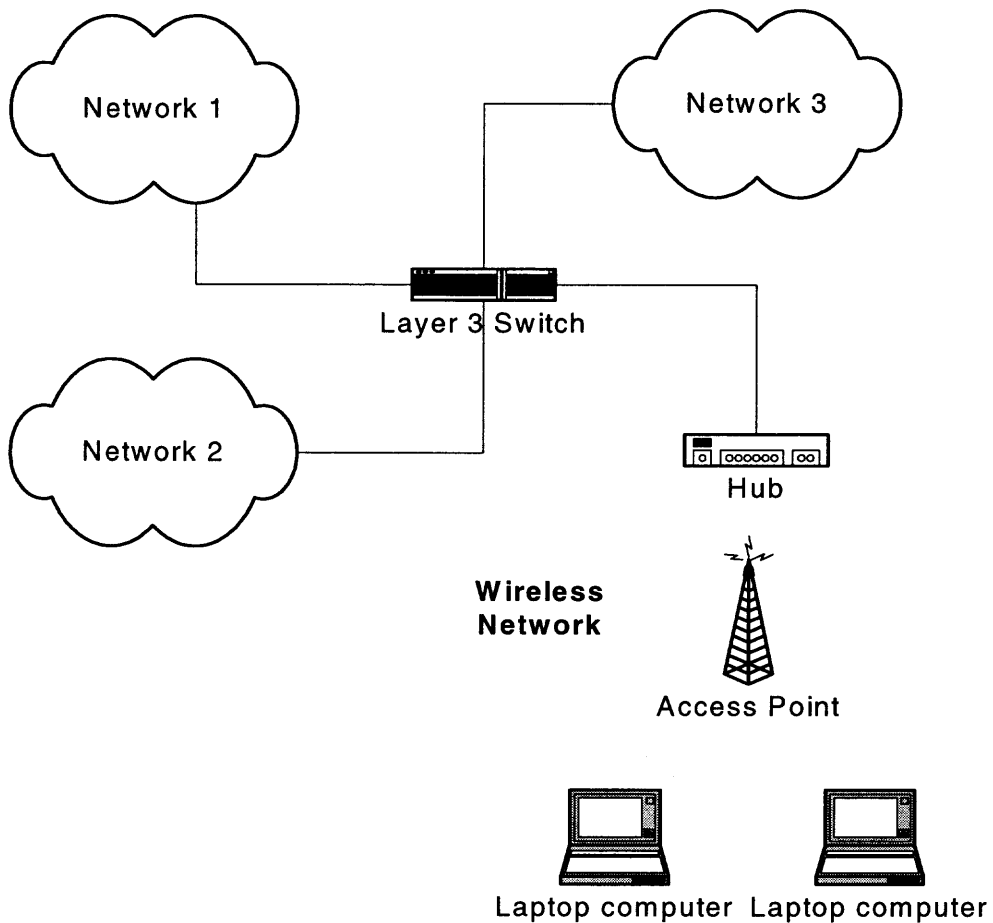
These files contain the tcp data in windump, the scenario setup information, and the results of the scenario.

The last part of the experimentation involved using the data gathered from the nap scenario to attempt other kinds of attacks. There are many different types of attacks that can be attempted and I leave the choices up to you. So far I believe that there are about 200 attacks available, please see the professor, for your use. Many of them seek for weaknesses in the operating systems that are present on the nodes. The choice is yours; just remember to start windump before implementing the attack to ensure that you collect the data. Running background traffic is also an option and if chosen should be started prior to any attack attempts.

APPENDIX A

INFRASTRUCTURE NETWORK

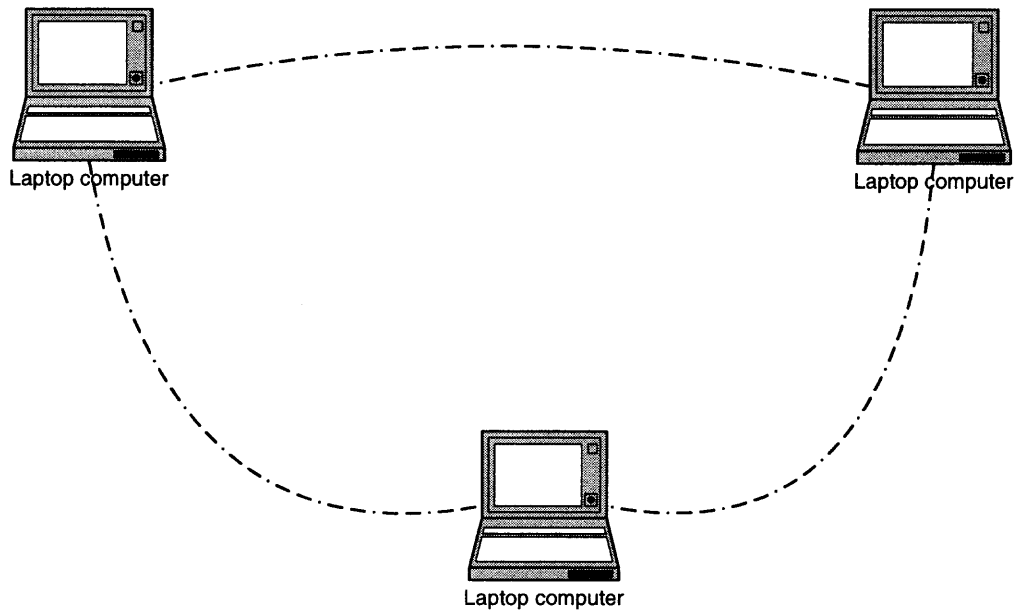
This is the first network that was tested in research. There are four different networks Inter-connected by a layer 3 switch. The wireless network is connected via an access point. This unit is connected to a hub, which is then connected to the layer 3 switch.



APPENDIX B

AD HOC NETWORK

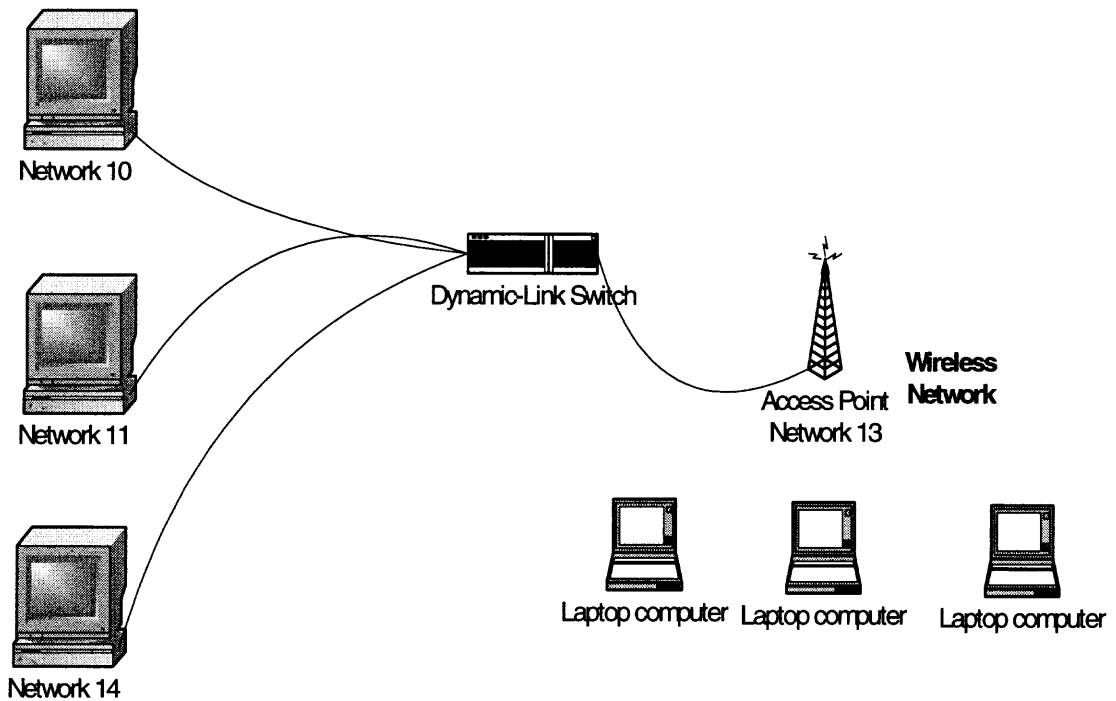
This network setup features the different laptops interacting with each other without the use of an access point. The laptops scan the different channels every 5 seconds in search of another laptop with the same SSID so they can communicate.



APPENDIX C

DYNAMIC-LINK SWITCH NETWORK

This network setup uses a computer with a dynamic-link software that in suppose to inter-connect different networks. The switch should also be configurable it order for the user to set the desired bandwidth. All the different networks connect to a port in an Ethernet card on the switch.



REFERENCES

1. Goerzen, J. (2000). Linux Programming Bible. Foster City, CA: IDG Books Worldwide, Inc.
2. Lin, T., Midkiff, S. F., & Park, J. S. (2002). A Dynamic Topology Switch for the Emulation of Wireless Mobile Ad Hoc Networks. IEEE Computer Society Digital Library, LCN'02, 791-798.
3. McClure, S., Scambray, J., & Kurtz, G. (2001). HACKING EXPOSED Network Security Secrets & Solutions (3rd ed.). New York: Osborne/McGraw-Hill.
4. Northcutt, S., Cooper, M., Fearnow, M., & Frederick, K. (2001). Intrusion Signatures and Analysis. Indianapolis, In: New Riders.
5. Northcutt, S., Novak, J. (2001). Network Intrusion Detection An Analyst's Handbook (2nd ed.). Indianapolis, In: New Riders.
6. Rappaport, T. S. (1996). WIRELESS COMMUNICATIONS Principles & Practice. Upper Saddle River, NJ: Prentice Hall.
7. Schwartz, M. (1987). Telecommunication Networks Protocols, Modeling and Analysis. Reading, MA: Addison-Wesley Publishing Company.
8. Tanenbaum, A. S. (1996). Computer Networks (3rd ed.). Upper Saddle River, NJ: Prentice Hall.