3-2008

Digital Signal Processing Leveraged for Intrusion Detection

Theodore J. Erickson

Digital Signal Processing Leveraged for Intrusion Detection

THESIS

Theodore J. Erickson, First Lieutenant, USAF

AFIT/GCE/ENG/08-04

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GCE/ENG/08-04

Digital Signal Processing Leveraged for Intrusion Detection

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Theodore J. Erickson, B.S.C.E.

First Lieutenant, USAF

March 2008

AFIT/GCE/ENG/08-04

Digital Signal Processing Leveraged for Intrusion Detection

Theodore J. Erickson, B.S.C.E.
First Lieutenant, USAF

Approved:

/signed/	26 Feb 2006
B.E. Mullins, PhD (Chairman)	date
/signed/	26 Feb 2006
R.F. Mills, PhD (Member)	date
/signed/	26 Feb 2006
G.L. Peterson, PhD (Member)	date

Abstract

This thesis describes the development and evaluation of a novel system called the Network Attack Characterization Tool (NACT). The NACT employs digital signal processing to detect network intrusions, by exploiting the Lomb-Scargle periodogram method to obtain a spectrum for sampled network traffic. The Lomb-Scargle method for generating a periodogram allows for the processing of unevenly sampled network data. This method for determining a periodogram has not yet been used for intrusion detection. The spectrum is examined to determine if features exist above a significance level chosen by the user. These features are considered an attack, triggering an alarm.

Two traffic statistics are used to construct the time series over which the periodogram analysis is accomplished. These two statistics are packet inter-arrival time and payload size. The traffic source for this research is the 1999 DARPA intrusion detection data set developed by MIT Lincoln Laboratories. Three specific attacks from this data set are examined; the Processtable attack, the Dictionary attack and the Teardrop attack. Of the three attacks the NACT was able to detect the Processtable attack with an accuracy of 100%. The Dictionary and Teardrop attacks were also detected with 100% and 85% accuracies respectively. This success in detecting these attacks establishes that digital signal processing methods can be a successful technique for network intrusion detection.

Acknowledgements

I would like to thank my wife, for her support and patience during my long hours of research. I would also like to thank my parents for always encouraging me to take the next step, even if it's not the easy one. Finally I would like to thank my advisor, Dr. Barry Mullins for allowing me the freedom to pursue a unique topic following my own approach.

Theodore J. Erickson

Table of Contents

Page

Page

List of Figures

List of Tables

List of Abbreviations

Digital Signal Processing

Leveraged for Intrusion Detection

I. Introduction

The Internet has revolutionized the way information and ideas are transmitted around the world. This is information that we rely on to do business, and to protect and run our country. With the rapid growth of this medium has come the rapid development of attacks against systems designed to transmit that information. In order to prevent this, system administrators are faced with the difficult task of defending their networks against these attacks. To do this they employ tools and best practices designed to mitigate these threats. One such tool is the network intrusion detection system (NIDS), designed to alert the network administrators to the presence of an attack.

NIDS tools range in complexity from simple pattern matching systems to complex behavioral models [LPN]. In order to improve the accuracy of these systems, new and perhaps non-traditional methods must be explored. Digital signal processing techniques represent an avenue of detection not yet fully realized.

1.1 Objectives

This thesis focuses on an alternative means for network intrusion detection. In order to make intrusion detection systems more accurate, and thereby more reliable, it is necessary to explore methods for detection outside the traditional approaches. To

that end this research will establish digital signal processing as an additional technique capable of detecting compute network attacks.

The tool developed in this research applies signal processing to obtain frequency information from network traffic. This makes it possible for intrusion detection to take place without prior knowledge of the attacks. This addresses a weakness in current NIDS technology, which rely on up-to-date databases of attack signatures for detection.

1.2 Implications

Digital signal processing applied to network intrusion detection provides another tool in the arsenal for network defense. The addition of this method may be able to increase the accuracy of current NIDS. This research demonstrates the feasibility of this approach and provides a foundation for future work.

1.3 Preview

Chapter II presents background material important to the understanding of future chapters. The methodology, or the approach to the problem, is covered in Chapter III. Chapter IV presents the results of this research, including interpretation of those results. Lastly, Chapter V restates the problem, identifies the contributions made by the research and their significance, identifies directions for future research and ends with a summary.

II. Literature Review

This chapter lays the foundations for the understanding of intrusion detection systems (IDS) specifically Network based Intrusion Detection Systems (NIDS), digital signal processing (DSP) and some of the recent research in these fields.

Section 2.1 presents an overview of IDS discussing current IDS technology and methods. Section 2.2 discusses the design of IP protocols as they relate to NIDS. Section 2.3 provides a background on Bayesian statistics as they apply to false alarm rates, as well as providing some research on human interaction with alarms from similar fields. In Section 2.4, relevant topics in the field of DSP are discussed with an emphasis on those techniques applied in this research. Section 2.5 discusses the Defense Advanced Research Projects Agency (DARPA) intrusion detection data set, and Section 2.6 covers the tools used in this effort. Finally, Section 2.7 focuses on relevant, current research.

2.1 Intrusion Detection Systems

Amoroso defines intrusion detection as "...the process of identifying and responding to malicious activity targeted at computing and networking resources" [Amo98]. Malicious activity can then be classified as attempts to disrupt the three basic tenets of computer security: confidentiality, integrity and availability [Esc98]. Activities of malicious users typically fall into one or more of those categories, and it becomes the responsibility of network and system administrators to prevent those breaches from occurring. To do this they use many different tools, including intrusion detection

tools. Throughout this section the term attacker is used to reference a user that is inside or outside the network with malicious intent.

Computer networks are vulnerable to a myriad of threats including, but not limited to, pre-packaged or scripted attacks delivered by readily available tools such as Metasploit, WinNuke and others [Ref08]. Some other vulnerabilities include denial of service attacks that exploit the load balancing characteristics of different networks and servers, as well as viruses and worms. Intrusion detection systems are designed to sense when these threats occur so that a network administrator or, in some cases, even the network itself can take defensive action.

Intrusion detection systems can be broadly classified into two different categories, anomaly detection and signature based detection [And80]. Most modern intrusion detection packages employ both of these techniques in order to strengthen the system [ACF+00]. There are exceptions to these classifications [SGF+02]. However, this section will differentiate only between these two groups.

2.1.1 Anomaly Based Intrusion Detection. Anomaly-based detection is based on the premise that a computer system has a relatively normal set of behaviors. These behaviors might include browsing the Internet or the occasional file download and printing. Anomaly-based detection establishes a normal profile for the activity of a network and continuously compares that to what is actually occurring on the network. It seeks to filter out ordinary results for network administrators, and typically establishes a baseline from which deviations from ordinary behavior can be

measured [CRBM01]. Benefits to this type of system include the ability to detect attacks that may not have a pre defined pattern or attacks that perhaps violate security policy in a way that is difficult to define [Axe00a].

There are a number of approaches for achieving the goal of anomaly based detection. One of these is data-mining or the compilation of all sorts of attack data in the hopes of finding a pattern. Another is machine learning, which is a subset of artificial intelligence. Yet another approach is Immunity-based detection which approaches the problem of intrusion detection as if it were a disease being fought by an artificial immune system. All of these approaches are being used to increase the accuracy of NIDS [CRBM01].

Problems with this category of intrusion detection are high false alarm rates due to the fact that some of the unique user traffic may not in fact be an intrusion. This new and unique activity could represent a user with a different task or perhaps a fresh project within the network. Since this traffic is outside the baseline, it would be categorized as an intrusion, until a new baseline is established. The ramifications of these false alarms are discussed further in Section 2.3.1. Anomaly-based detection also tends to be computationally intensive, requiring the intrusion detection system to process and store large amounts of data to maintain and compare against the baseline.

2.1.2 Signature-Based Intrusion Detection. Signature or misuse based intrusion detection relies on knowing what undesirable behavior looks like. This type of detector looks for clues to behavior that the designers determine is consistent with

malicious intent [Axe00b]. This malicious behavior is cataloged and used to compare against network behaviors during a period of time. One of the ways this can be accomplished is through bit-by-bit comparison of the payloads against known attack patterns or even filtering of the content for suspicious strings. For example the sequence "9090909090" represents a no-operation sled; an IDS may look for this pattern in a packet and filter it out [CRBM01]. Another example is filtering for character strings like "top-secret" or specific paths such as "/etc/shadow" where malicious activity is likely to occur [Amo98].

Difficulties with this type of NIDS includes the requirement for a current database of known attack signatures. Once this database is out of date, the NIDS will fail to detect newer exploits. Attackers can also use inflexibility of the design as a weakness by creating polymorphic code, which is code that changes the way it looks without altering its function, thereby altering the signature of the attack [CRBM01]. There is very little room in this design for errors or gaps, since the performance of the detector is only as strong as the signatures it employs [Axe00b].

2.1.3 Modeling Intrusion Detection. Denning's model of an intrusion detection system [Den87] lays the framework for the understanding of how these systems are implemented. This model is designed to help quantify a system that had the desirable properties of a low false alarm rate and a high probability of detection. Denning's model is a "rule-based pattern matching system" [Den87] that outlines the components necessary for the implementation of a real-time intrusion detection sys-

tem. The subject and object system put forth by Denning is the basis for how rules are written for an IDS. Denning's model marks the beginning of intrusion detection as it is implemented in today's systems. Her theoretical Intrusion Detection Expert System (IDES) exists, in some form or another, in almost every operational IDS today. Figure 2.1 represents a basic outline of the major components combined to make an effective IDS.

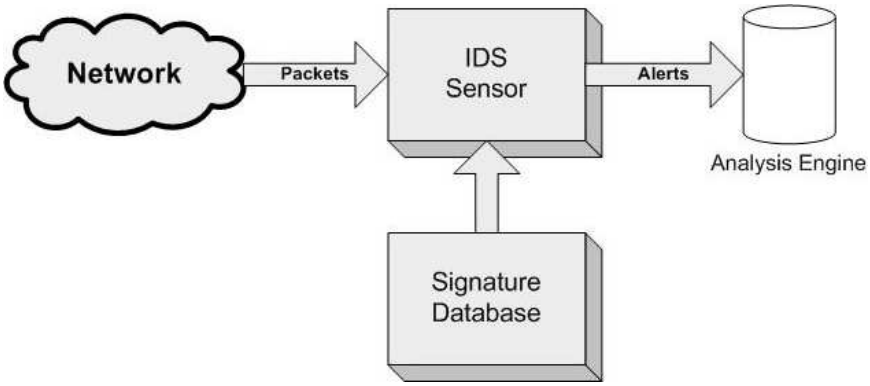


Figure 2.1: Illustration of the basic data flow through an IDS.

2.2 Networking Protocols

Since most all Internet Protocol (IP) packets must contain the same information and are arranged in much the same way, it becomes more evident how useful information can be extracted from them without knowing or caring what the entire message contains. Section 2.2.1 discusses how an IP packet is organized and the benefits of that deliberate organization. Section 2.2.2 discusses how those packets fit into the bigger picture of network devices across the Internet.

2.2.1 Packet Design. Internet communication is defined in a number of publications called Request for Comments (RFC). RFC 791 governs the construction of packets for Internet Protocol Version 4 (IPv4), which is the most widely used version. Internet Protocol Version 6 has been in the works for a number of years and promises to come into effect soon changing the design of the basic packet, but keeping the same structured qualities.

Figure 2.2 illustrates the information contained in an IP packet as well as how it is arranged. The packet is noticeably well structured allowing for different fields to be easily examined or compared. There is a great deal of research being accomplished on efficient and fast algorithms for the inspection of large amounts of IP traffic. Baker and Prasanna propose efficient means for implementing this inspection process onto field programmable gate arrays (FPGA) [BP05]. All of this is designed to make looking at the increasing amounts of data, upwards of 10 Gb/s, feasible. This speed is critical to the timely detection and correction of security threats that may take only a fraction of a second to compromise important data.

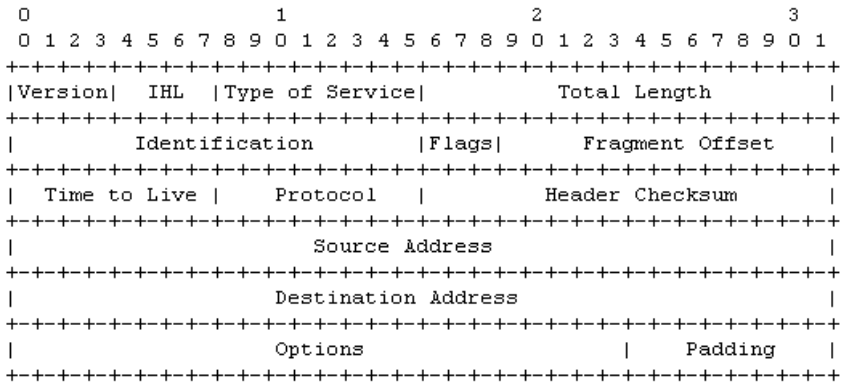


Figure 2.2: Makeup of an IP packet. [Pos81]

There are a plethora of useful statistics to be found not only in the construction and content of the packets themselves, but in how those packets are traveling through the network. Packet inter-arrival times, packet sizes, payload sizes and the payload itself are only some of the many statistics available for analysis.

2.2.2 Layered Network Architecture. The layered approach to network architecture is a way to abstract away some of the complexities involved in connecting many different types of computer systems. The model presented by Kurose and Ross for this layering is included in Figure 2.3, and illustrates the five layers. The application, transport, network, link and physical layers will not be explored individually, however, it is useful to note how they relate to one another.

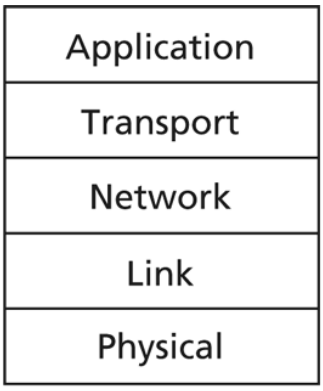


Figure 2.3: Internet protocol stack. [KR05]

These layers allow network designers and programmers to modify sections of the network architecture without affecting the other layers. For instance the physical layer is responsible for transmitting the electrical impulses across the wire. The link

layer which sits above the physical layer, does not know how those impulses get onto the wire. The link layer, does not need to know this information to function; it simply sends the bits down to the physical layer to be sent. In turn, the network layer does not need to know how the link layer forms the bits, it simply passes the information down. Figure 2.4 illustrates how a message "M" is wrapped and passed down the layers in order to travel across the network [KR05]. The message is successively encapsulated by a series of headers. Each individual header contains the information needed for the message to be interpreted by that layer. As the message progresses through the different layers it accumulates all the information it needs to be routed correctly across the network to it's destination host and process.

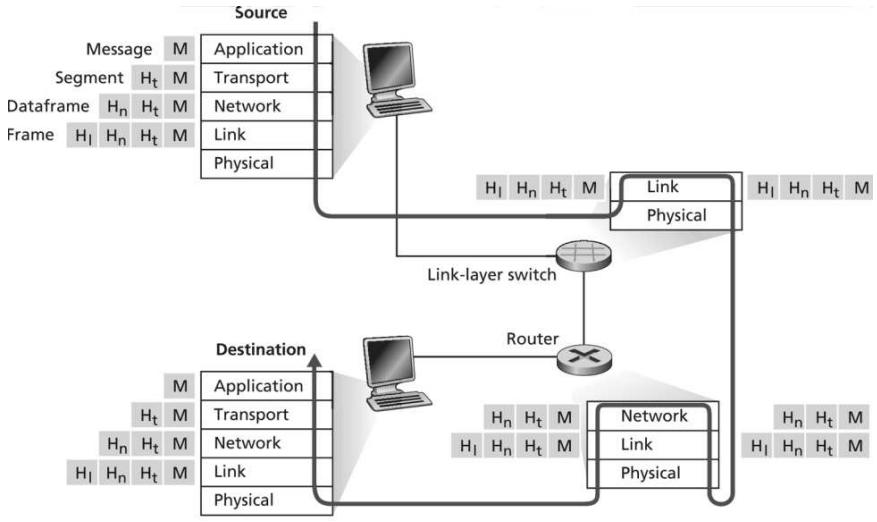


Figure 2.4: How the Internet protocol stack is used across multiple networked connections. [KR05]

The NIDS fits into this architecture by monitoring the traffic coming across the wire at the physical layer. This allows each packet to be examined using either signature or anomaly based methods for detection [Esc98].

2.3 Bayesian Statistics

This section discusses Bayes theorem and how it motivates research into more accurate intrusion detection systems. Bayes theorem relates the conditional and marginal probabilities of the events A and B [MA03]. Conditional probability is the probability of the occurrence of some event A given that some event B did or did not occur. The marginal probability is the probability of event A occurring, without regard to whether or not B occurs.

Bayes Theorem is

$$P(A \mid B) = \frac{P(A) \cdot P(B \mid A)}{P(B)} \tag{2.1}$$

where $P(A)$ is the marginal probability of event A, $P(B \mid A)$ is the conditional probability of event B given A, and $P(B)$ is the marginal probability of event B. Accounting for all the possible outcomes yields

$$P(A_j)B = \frac{P(A_j) \cdot P(B \mid A_j)}{\sum_{i=1}^{n} P(A_i) \cdot P(B \mid A_i)} \tag{2.2}$$

This becomes useful for IDS through the following example taken from Axelsson's examination of the Base-Rate Fallacy and it's applications to Intrusion Detection [Axe00a]. First is the discussion of an intuitive example, continuing on to an example with specific IDS ramifications.

Axelsson invokes the image of a patient visiting his doctor. The doctor administers a test that is 99% accurate, meaning that when the test was administered to a group where everyone had the disease, the test reported that they were sick 99% of the time. So it follows that when the test is administered to a group where no one has the disease, the test would indicate that they were not sick 99% of the time as well. Upon visiting the doctor to receive the results, the patient learns that the test did in fact indicate he had the disease; however, the doctor informs the patient that there is still good news. That news is that the rate of incidence for the disease within the population is only $\frac{1}{10{,}000}$.

Applying 2.2 to the example yields

$$P(S|R) = \frac{P(S) \cdot P(R|S)}{P(S) \cdot P(R|S) + P(\neg S) \cdot P(R|\neg S)}. \tag{2.3}$$

where S indicates sick, $\neg S$ indicates not sick, R indicates a positive test result (indicating the disease) and $\neg R$ indicates a negative result. Substituting for the values referenced in the example leads to

$$P(S|R) = \frac{\frac{1}{10000} \cdot 0.99}{\frac{1}{10000} \cdot 0.99 + (1 - \frac{1}{10000}) \cdot 0.01} = 0.00980... \approx 1\%. \tag{2.4}$$

So the chance of actually having the disease is approximately 1%, despite the positive test result. This leaves the patient with a considerably better chance of not having the disease, even though the test was positive and accurate 99% of the time.

This is all due to the rate of incidence in the population being small as compared to the accuracy of the test.

Figure 2.5 illustrates the medical testing example. This representation makes it easier to see how the relative size of the region encompassing the positive test and not being sick (R & $\neg$ S) is much larger than that of the region covering a positive test result and actually being sick (R & S). This larger region can also be described as the region encompassing false alarms.

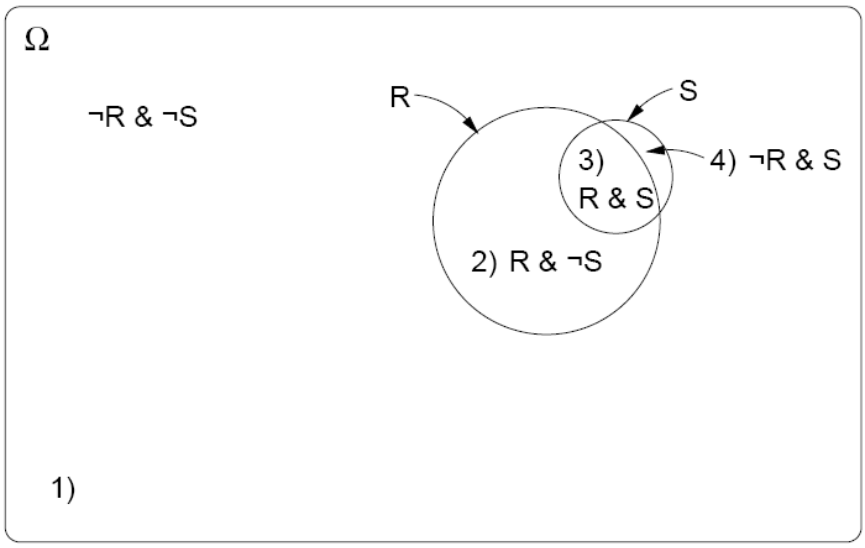


Figure 2.5: Venn diagram illustrating the medical test example, not to scale. [Axe00a]

This example can be extended to network monitoring activities by simply changing some of the participants. The patient becomes a computer on the network to be defended, the test is a NIDS, and the disease is an attack. Even if the detection system is 99% accurate, the sheer volume of traffic combined with the relatively low amount of attacks makes the rate at which a false alarm is reported higher than expected. It

follows then, that in order to reduce the false alarms, the accuracy of the detection system must be increased. This research proposes a tool to accomplish that goal.

2.3.1 False Alarms. One of the primary difficulties associated with the design of any system is interfacing that system with it's human operators. Figures of merit for an IDS can be measured using metrics for effectiveness, efficiency, ease of use, security, and interpretability [Axe00a]. An effective NIDS must address each of these metrics. However, without human interaction, the NIDS provides only another data point. The system and network administrators must interpret and take action based on the information. So in order for the user to trust the system, the NIDS needs to provide a reasonably low level of false alarms.

Accuracy, along with rates of true and false positive allow for the analysis of overall performance along with the performance against specific attacks. Equations 2.5, 2.6 and 2.7 are all used to calculate those metrics.

$$True\ Positive\ Rate = \frac{True\ Postives}{True\ Positives + True\ Negatives} \tag{2.5}$$

$$False\ Positive\ Rate = \frac{False\ Positives}{False\ Positives + True\ Negatives} \tag{2.6}$$

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Positives + Negatives} \tag{2.7}$$

True positive rate is the rate at which the system successfully detects a target. False positive rate is the opposite of true positive rate and is the rate at which the system generates an alarm when nothing is there, also known as the false alarm rate. Accuracy is a way of quantifying how close the system comes to correctly detecting every event; a perfect system would achieve 100% accuracy. These measures are all useful in measuring overall system performance.

2.4 Digital Signal Processing (DSP)

Roberts describes a signal as "...any time-varying physical phenomenon which is intended to convey information" [Rob04] and by that definition network traffic constitutes a signal. DSP techniques can therefore be applied to network traffic. The difficulty here is defining where exactly that signal exists, since information exists in a binary state, unlike radio signals that vary across a spectrum. In order to create a signal from network traffic, various parameters such as packet inter-arrival times, packet sizes and even packet destinations can be chosen to create a time varying signal. For instance the time the packet arrived would be plotted on the x-axis and the size of the payload would be represented on the y-axis, creating a signal that varies over time.

DSP techniques, specifically the discrete fourier transform (DFT), break signals into multiple overlapping sinusoidal components, allowing for an analysis of the segments of the signal. Figure 2.6 illustrates how a DFT allows the signal to be represented as a waveform in the time domain (upper signals) and as a spectrum (lower

signals). This reveals information about the signal that is not readily available using simple visual inspection techniques.

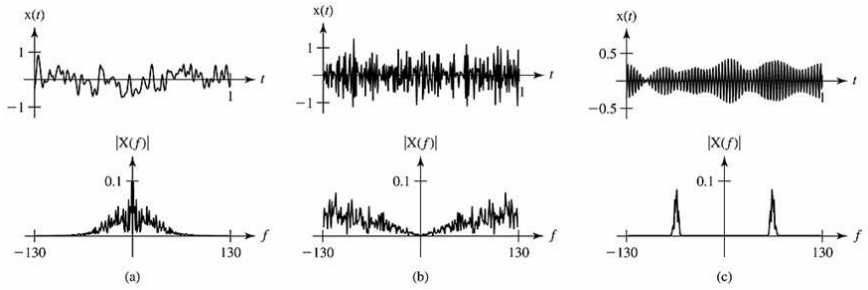


Figure 2.6: Examples of signals represented in the time (upper) and frequency domain (lower). [Rob04]

DFT techniques have been used successfully to search large quantities of time-series data, such as electrocardiographic signals, for similarities [YLAA00]. This research is relevant for a number of reasons. First, it uses DFT techniques in computer databases, which dovetails nicely with the idea of network traffic statistics presumably buffered in the same sort of database. Secondly, this technique searches for similarities, making it useful for comparisons against known patterns of network traffic. As shown in Figure 2.7, despite the discrete nature of the signal, useful frequency information is still obtained from the data stream.

Research in a similar field has been done to isolate similarities and points of interest in electrocardiographic signals [CFPCAGN02]. This research outlines methods for searching large amounts of continuous data, not unlike network statistics, for interesting features. Automating this search allows doctors to maximize their time for diagnosis, removing the need to sift through large amounts of data. Extending that idea to network traffic analysis would allow the network administrator to take the

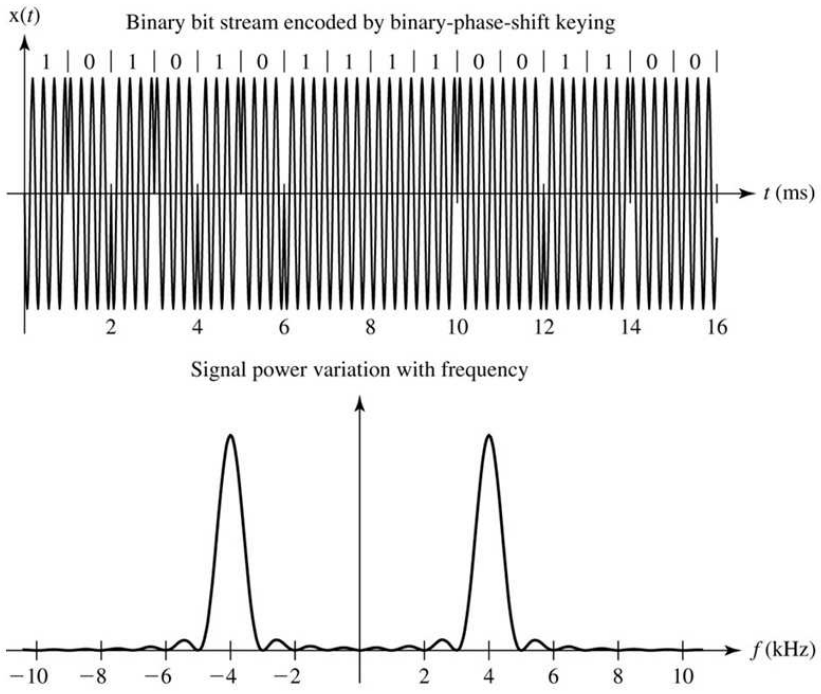


Figure 2.7: This illustrates how DFT techniques can transform even binary data. [Rob04]

place of the physician, using the algorithm to remove the tedium of sifting manually through large amounts of network data. This is not to say that there are no current methods to automate the task of sifting through the network data, however, there are few methods using DSP techniques.

2.4.1 Periodogram. One of the unique problems faced while using DSP techniques to process network traffic data is that the traffic data is not uniformly sampled. A uniformly sampled data set is characterized by values existing at known intervals over a sampled period. For instance a sine wave might be sampled at 100 Hz or 100 times/second, so the sampled values would exist in the data set at uniform

intervals of 1/100 of a second. The difficulty with network traffic is that the sample exists whenever the packet arrives, leaving potentially large gaps in the data.

Unfortunately, these large gaps make processing the data with traditional DSP techniques unwieldy and often inaccurate. The discrete fourier transform, discussed in Section 2.4, relies on the sampled data values existing at a known rate. This does not occur in the network traffic data sets, forcing manipulation of the data in one of several alternative ways. Analysis can be accomplished by stitching zeros in between the sampled values, providing a sample at known intervals. Another option is to approach the data analysis using a different technique.

Stitching zero's in between the sampled values can lead to errors and bias in the data [PCJ+02]. Another drawback to this approach is that with a network data set consisting of minutes or hours of data, a file that was originally kilobytes in size quickly balloons to gigabytes making the files too large to compute in a reasonable amount of time using Fourier Transform techniques.

The Lomb-Scargle method for computing a periodogram offers a number of benefits over padding the data with zero's. The Lomb-Scargle method only computes the frequency content at measured values, removing the possible error introduced by interpolation. The Lomb-Scargle periodogram is equivalent to least-squares fitting a sinusoid of frequency ω to the given unevenly spaced data [Lom76]. The least-squares fitting is a method for linear regression where the sum of the squared residuals is minimized. The fundamental equation for the Lomb-Scargle Periodogram is

$$P_N(\omega) \equiv \frac{1}{2\sigma^2}\{\frac{[\Sigma_N(h_n - \bar{h})\cos\omega(t_n - \tau)]^2}{\Sigma_N \cos^2\omega(t_n - \tau)} + \frac{[\Sigma_N(h_n - \bar{h})\sin\omega(t_n - \tau)]^2}{\Sigma_N \sin^2\omega(t_n - \tau)}\} \quad (2.8)$$

where the N data points are unevenly sampled events at times t_i arranged as $h_i \equiv h(t_i)$, $i = 0, ..., N-1$. The mean $(\bar{h})$ and variance (σ^2) are calculated as

$$\bar{h} \equiv \frac{1}{N}\sum_{i=0}^{N-1} h_i \quad (2.9)$$

$$\sigma^2 \equiv \frac{1}{N-1}\sum_{i=0}^{N-1}(h_i - \bar{h})^2 \quad (2.10)$$

$$\tan(2\omega\tau) = \frac{\sum_j \sin 2\omega\tau_j}{\sum_j \cos 2\omega\tau_j} \quad (2.11)$$

Equation 2.8 then renders the Lomb-Scargle normalized periodogram where spectral power is a function of angular frequency. The relationship $\omega \equiv 2\pi f > 0$ allows the conversion into Hz.

Additionally, because the Lomb-Scargle method produces a $P_N(\omega)$ that is normally distributed [Pre02, Lom76] it is possible to calculate the probability that a peak frequency value is above a given significance level. This is leveraged in

$$P(> z) \approx Me^{-z} \quad (2.12)$$

where M is the number of frequencies calculated. This leaves z, which is the spectral power for a chosen frequency. Solving for z gives the algorithm the ability to determine which, if any, frequencies contain significant information that might possibly indicate an attack.

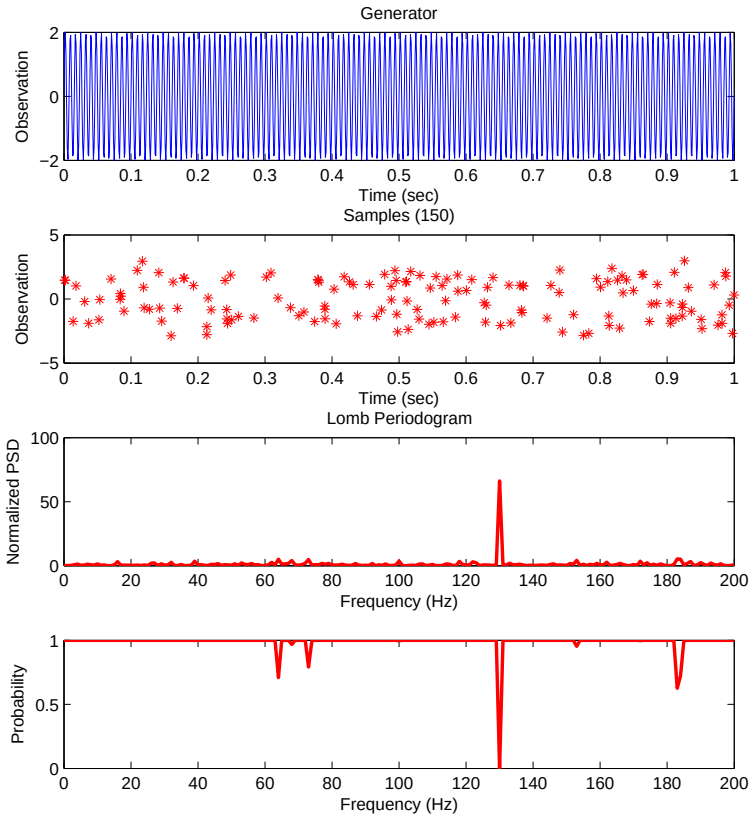


Figure 2.8: Demonstration of the Lomb-Scargle periodogram.

A Lomb-Scargle periodogram is illustrated in Figure 2.8, where the top panel indicates the raw signal to be sampled. This particular signal is a 130 Hz sin wave of amplitude 2. The second panel illustrates 150 samples taken at random times between 0 and 1 second. The samples are then analyzed using the Lomb-Scargle periodogram

method, with a regression being performed on each whole frequency from 1 Hz to 200 Hz. Stated another way, the calculation performed a least squares regression at every frequency from 1 Hz to 200 Hz. The third panel illustrates the periodogram which contains the results from the least-squares fitting, showing a distinct peak at 130 Hz, as expected. The fourth and final panel demonstrates the power of Equation 2.12, illustrating the significance of each peak in the periodogram. The significance is akin to a p-value, where a lower number indicates greater confidence that the value in question did not occur randomly. The p-value for 130Hz approaches zero, indicating a high level of confidence in the 130Hz signal contained in the data. The Matlab code used to generate Figure 2.8 is contained in Appendix A.3.

2.5 DARPA Intrusion Detection Data Set

The Defense Advanced Research Projects Agency (DARPA) conducted a simulation of a medium sized military installation, with the intent of using the captured data with embedded attacks to test Intrusion Detection Systems [Lab99]. Some 200 instances of 58 different attacks were used to test the intrusion detection systems [LHF+00a]. The test, conducted in 1999, remains one of the best documented sources of network attack traffic.

Drawbacks to using this dataset include the age of the data and the fact that the traffic is simulated and not actual traffic [Mch00]. Despite these drawbacks there are few other alternatives that offer the same level of documentation and scrutiny that the DARPA data set provides.

2.6 Tools

This section is intended to familiarize the reader with some of the tools used to accomplish this research. Included below are brief overviews of the Wireshark network capture tool in Section 2.6.1 and the Matlab computing environment in Section 2.6.2.

2.6.1 Wireshark. Wireshark is a network capture and analysis tool developed originally as Ethereal [Com08]. The tool allows for the easy capture of network traffic along with the display and filtering of that capture traffic. For this research Wireshark is used to open, display and filter the captured traffic from the DARPA data set. Wireshark, 0.99.6a, is the version used to complete this study.

2.6.2 Matlab. Matlab is a high performance numerical computing environment based on matrices. Produced by The Mathworks, this program is used to compute and display the periodogram results. The built in graphics processing ability make it a natural choice for this research. Matlab version 7.4.0.287 is used for this research.

2.7 Relevant Research

Network defense is becoming an increasingly important field of research. With this increased focus on security there are many unique developments in the field of network intrusion detection. Sections 2.7.1 and 2.7.2 discuss developments relevant to the frequency-based approach taken in this research.

2.7.1 Frequency-Based Intrusion Detection. Mian Zhou and Sheau-Dong Lang were able to demonstrate the appearance of features corresponding to attacks in the spectra of network traffic data [ZLC+03]. Their IDS is outlined conceptually in Figure 2.9, and took as input network traffic captured in the 1999 DARPA intrusion detection evaluation data set [Lab99].

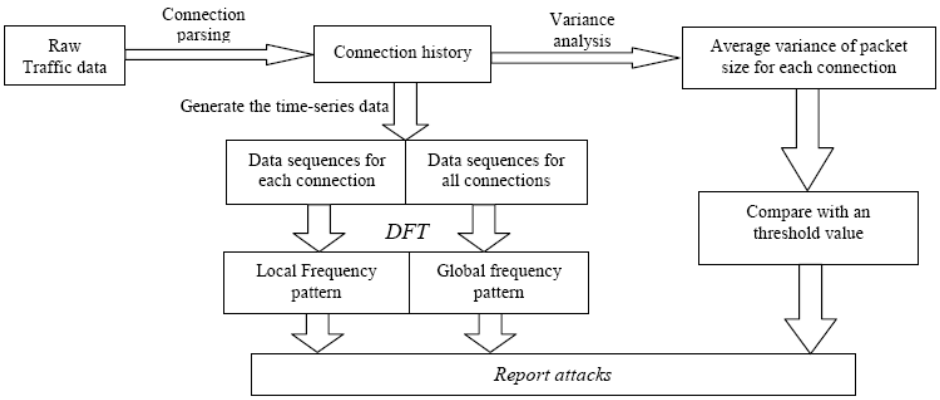


Figure 2.9: This diagram describes the conceptual approach for frequency-based intrusion detection. [ZLC+03]

The simulation traffic was parsed to include only the nodes of interest, this allowed the traffic to be replayed using the OPNET simulator. The ProcessTable attack was chosen as the attack of interest. The authors were then able to analyze the variance in packet size, using that metric to gain further insight into the nature of the attack. Connections between the victim and the potential attackers with sufficiently low variance were tagged for further inspection. These connections were then analyzed using a traditional DFT, the resulting spectrum was then interpreted visually for unique features such as spikes. The authors were able to demonstrate that these spikes

in the frequency domain, illustrated in Figure 2.10, corresponded to the connections representing the attacks.

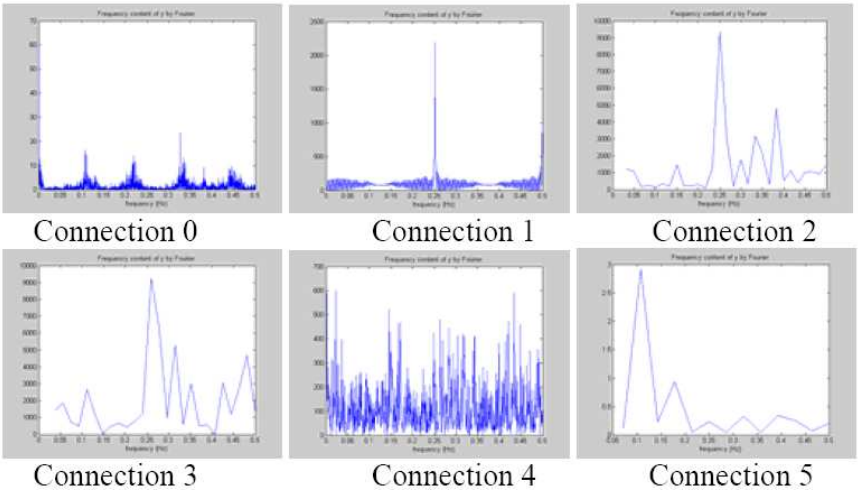


Figure 2.10: Frequency patterns for the detection of the ProcessTable attack. [ZLC+03]

Connections 0 and 1 in Figure 2.10, represent connections of interest. Zhou and Lang were able to show that these connections represented both an attack and a network probe, both of which represent events worth knowing about. This research was able to demonstrate the merit of using DSP techniques to analyze network traffic.

This method for frequency-based intrusion detection is limited by its implementation within the OPNET simulation environment. The network architecture must be known well enough to represent it within the simulator, and the traffic must be parsed to allow for the OPNET simulation engine to read it. In addition to using OPNET, this approach uses the DFT algorithm to obtain the spectrum for the data. This method requires evenly sampled data, and can be time consuming to compute.

2.7.2 Signal Processing Techniques Used for Wireless Traffic Analysis. Partridge, et al. [PCJ+02], developed a method for the analysis of wireless data traffic using the Lomb-Scargle periodogram. Their method centers around modeling the network as observed from a network tap. A network tap is essentially a network hub allowing the monitoring device, usually an Intrusion Detection System (IDS) of some sort, to capture all of the traffic flowing across a link. These devices are necessary in switched networks where traffic destined for a specific IP address would not normally be visible on an adjacent link because the switch can intelligently route packets across the correct wires.

The research took encrypted wireless traffic, observed as if through a network tap, and was able to determine some characteristics of the networks in question. This was accomplished despite the encryption of the network, which typically obfuscates packet information normally used to determine this information. Researchers were able to calculate round trip times of the TCP connections as well as routing information using coherence analysis [PCJ+02]. Routing information is of particular value when dealing with wireless networks as this can lend valuable insight into their operation, active nodes, and weaknesses.

Partridge chose the Lomb-Scargle periodogram for the analysis of the network traffic for many of the same reasons outlined in Section 2.4. The data used for this effort was accumulated using the *ns-2* network simulator and a simulated network containing four nodes. The network tap was only able to hear transmissions originating at two of the four nodes. Despite this configuration and the network encryption,

the periodograms obtained using the Lomb-Scargle method show peaks for each of the traffic flows within the network.

2.8 Summary

Contained in this chapter are discussions of material relevant to the understanding of the subsequent chapters. Of particular interest were the discussions of Intrusion Detection Systems in general contained in Section 2.1, and the discussion of Digital Signal Processing contained in Section 2.4.

III. Methodology

3.1 Problem Definition

3.1.1 Goals and Hypothesis. This research establishes the usefulness of DSP techniques in the realm of network intrusion detection. Current technology applies signature and anomaly based techniques against network traffic which results in unacceptably high false alarm rates when faced with new and unique threats rendering the systems unreliable [LHF+00b]. DSP techniques, when added to the existing framework, may be able to increase the accuracy of these systems. The methodology that follows shows that DSP techniques, specifically the measurement of packet inter-arrival times along with payload size, are useful statistics for network intrusion detection.

Packet inter-arrival times along with payload sizes and other statistics contain features in their frequency patterns that distinguish attack traffic from normal traffic. These features represent signals within the sampled data, signals that this research uses to successfully detect network intrusions.

3.1.2 Approach. The approach to this problem is unique in that transforming the network traffic into a different domain is not something that has been widely accomplished in this field. However this method of signal transformation has been well defined for many years, and now can be used in a different area. The initial difficulty of using DSP techniques on network data is the construction of a time series signal on which to perform the DFT algorithm. It is the construction of this time

series using the correct statistics that holds the key to generating a useful signal to analyze.

The 1999 DARPA data set containing simulated intrusion traffic at a DOD installation is well documented [Lab99]. DSP techniques, specifically the discrete Lomb-Scargle periodogram, are applied to this data. Zhou and Lang accomplished a similar study using packet inter-arrival time and the discrete fourier transform (DFT) [ZLC+03]. This research expands on Zhou and Lang and applies the Lomb-Scargle periodogram against time series data derived from both packet inter-arrival time and payload size.

Once the data has been transformed into the frequency domain, the periodogram is examined for features differentiating the selected traffic from normal traffic. If the periodogram does contain a feature or a spike in power, the selected traffic is said to contain an attack. These attack detections are then compared to the data set documentation to determine whether or not the detection was accurate.

3.2 System Boundaries

The system under test is the intrusion detection system proposed by Zhou and Lang and modified for this research. This system is described in Figure 3.1. Where the input to the system is the raw TCPDUMP data for any captured network traffic. In this case the data is obtained from the DARPA intrusion detection evaluation data set [Lab99]. The components of the system are the script designed to parse individual connections out of the much larger data set, the Lomb-Scargle algorithm

itself and the signal probability calculation. The component under test is the signal probability calculation tool used to classify the individual frequency patterns as benign or malicious. The system then outputs a classification of the traffic in question as either traffic that is believed to be attack based or traffic that does not have attack characteristics.

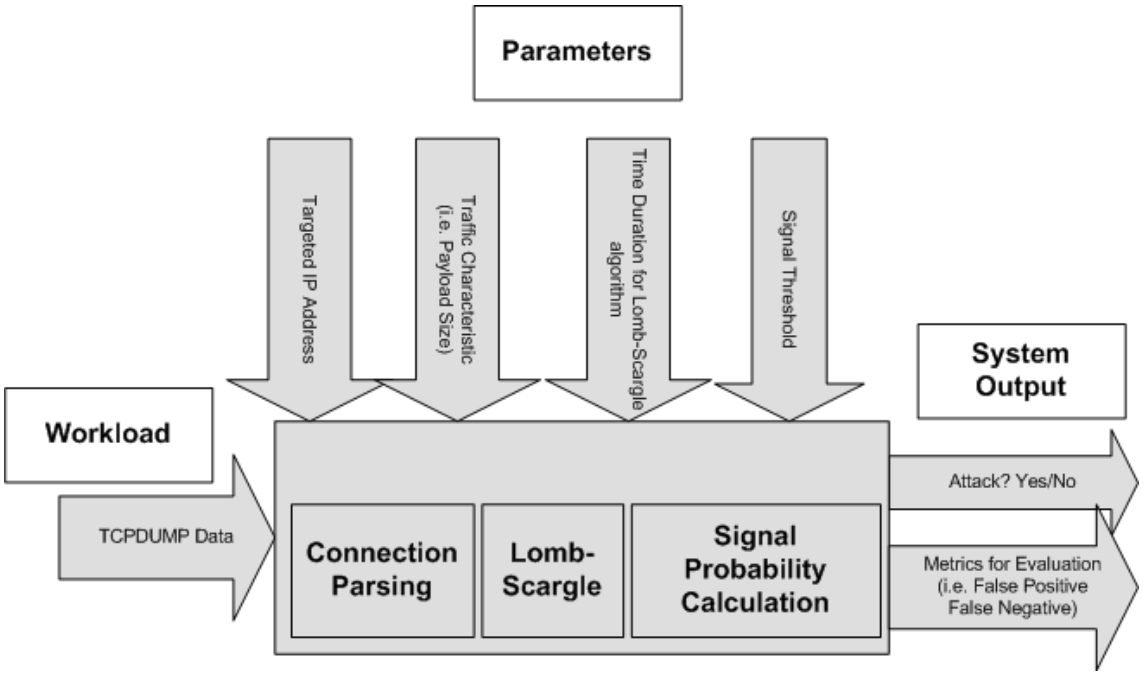


Figure 3.1: System under test.

This system is limited in the sense that it requires a well documented workload in order to produce the performance metrics. While any TCPDUMP traffic is accepted by the system, not knowing if that traffic contained attacks yields unverifiable results from the system. Another workload restriction is knowledge of the sampled network's topology. In the case of a well documented network, the target IP addresses are easy to obtain for the initial scripting. If the traffic is not well documented, it becomes

difficult for the system to narrow the scope of the traffic to a point at which it could perform the analysis.

3.3 System Services

This Network Attack Characterization Tool receives a file containing TCP-DUMP data from a network capture program. After obtaining that data, the user chooses an IP address whose connections are of interest. The target address for this study is for the computer under attack. This allows the system to trim the network traffic to only inbound and outbound connections from that specific address. This is accomplished in order to simulate a monitor residing on the link attached to the target computer, much the same way as a firewall might be placed in line to protect a computer. This placement is illustrated in Figure 3.2.

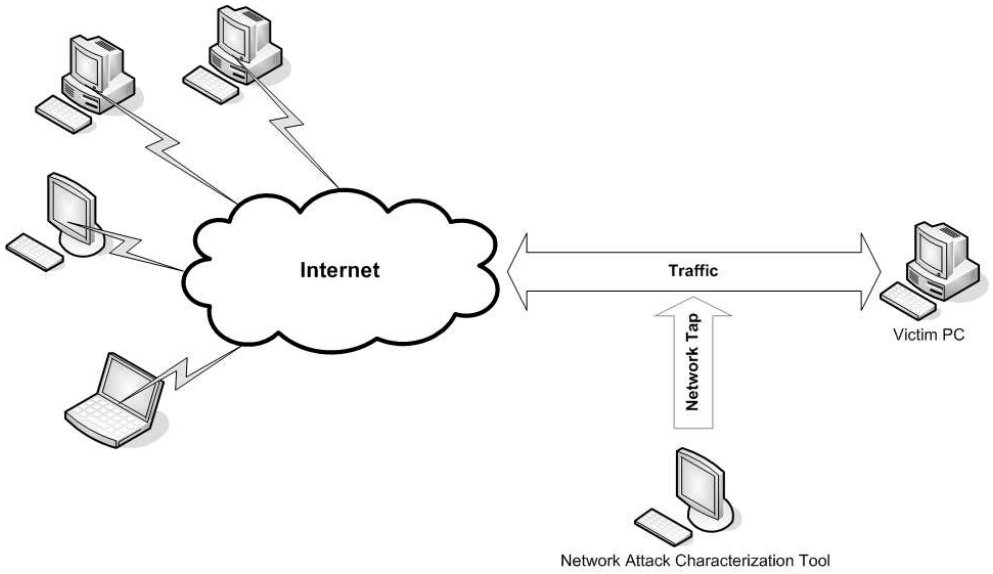


Figure 3.2: Placement of the Monitoring System.

Once the traffic has been culled, a time series is constructed using packet arrival times as well as the selected traffic characteristic. The traffic characteristic for this study includes packet inter-arrival time along with payload size. This information creates the time series over which the Lomb-Scargle algorithm is run. The Lomb-Scargle algorithm is run over a user-defined time window, allowing for a variable resolution to be achieved. Once the periodogram has been generated for the time series, (2.12) is applied generating significance levels for the detected frequencies. These significance levels in the frequency pattern determine whether or not that traffic contains an attack.

Once the NACT has analyzed the selected traffic, the system is able to generate metrics based on the known characteristics of the TCPDUMP data set. False positive, true positive and accuracy rates are used to determine the performance of the tool. The performance in detecting attacks can then be compared to existing NIDS tools to determine if in fact this type of detection technique is worth pursuing.

3.4 Workload

The workload for this study is the 1999 DARPA intrusion detection evaluation study data set. The data was collected based on a simulation of network traffic at a DOD installation of medium size. The traffic was simulated with known embedded attacks along with background traffic. The simulated traffic was then captured and input into various intrusion detection systems to determine their effectiveness against embedded attacks.

The advantage of this data set is that the attacks are known and well documented with respect to connections, ports, and target IP addresses. This allows for the ease of verification of the experimental setup. This data set has also been used in previous studies, allowing for the validation of results [ZLC+03].

The disadvantages of this workload include age, this data set is over 7 years old. The data is based on a capture of simulated traffic, and unfortunately DARPA has not completely documented their configurations, making the simulation of the network difficult [Mch00].

From this workload, specific attacks were chosen over which to run the NACT. These attacks fall generally into the (DoS) category of attack types. This type of attack usually seeks to overwhelm its target, typically causing it to be unresponsive or slow. The specific attacks chosen for analysis are discussed further in Section 4.2.

3.5 Performance Metrics

The Network Attack Characterization Tool's performance is judged on accurate detection of attack traffic. This accuracy is determined using the truth data from the supplied workload. System outputs are compared against the known content of the workload and this data is used to generate rates of false positive, true positive and accuracy. Quality of the traffic characteristic can then be measured using rates of false positive and true positive, or simply put, how often the system was correct or incorrect in detecting attack traffic.

3.6 Parameters

3.6.1 System Parameters. The first system parameter involves parsing the incoming data based on connections. This defines the scope of the traffic the system is processing, and can thereby reduce the amount of noise input into the system. The system view can be limited to a single link, thereby only allowing traffic destined for a target personal computer (PC) to pass into the system. The decision could also be made to look at the aggregate traffic, presenting a picture of all the packets traveling across a link.

Another system parameter is the selection of a traffic characteristic with which to build the time series. Some examples of relevant characteristics are packet inter-arrival times, payload sizes and message fragmentation. Selection of characteristics that lend insight into attack traffic will greatly increase the effectiveness of the network attack characterization tool.

The time duration over which the Lomb-Scargle algorithm is run composes another system parameter. Since the choice of a time interval can dramatically affect what information is presented in the frequency domain, this parameter must be chosen carefully. Too large a time duration can cloud the data with unnecessary noise, and too small a time duration may fail to cover the duration of an attack.

The final system parameter is the signal threshold, or the significance level at which an attack is said to have occurred. The Lomb-Scargle algorithm allows, by virtue of its normal distribution, the calculation of the significance of signals present

in the sampled data. The signal threshold is set so that only signals of a certain significance are labeled as attacks.

3.6.2 Workload Parameters. The workload for the network attack characterization tool is the TCPDUMP data input into the system. The parameters for this workload are limited to what traffic is presented to the system. This traffic may or may not contain attacks, it may contain multiple attacks, and it may contain multiple concurrent attacks. For the purpose of this study the well defined TCPDUMP data from the 1999 DARPA data set is used. This data set contains known attacks at known times allowing for the system outputs to be verified.

TCPDUMP data containing no attack traffic is used to determine a system baseline. Traffic from the DARPA data set documented as containing no attacks is used for this purpose. This information is used to set the signal threshold to its initial level for the pilot studies.

3.7 Factors

The first factor to be varied for this study is the time duration over which the Lomb-Scargle algorithm is computed. This duration could range anywhere from the total time of the simulated data, to millisecond slices of that data. For this research, the time duration for the data is set to the total duration of the chosen attack. The attack duration is determined from the data set documentation. This allows for the complete analysis of all the possible features contained within the attack.

The other factor for this study is the signal threshold. The initial threshold value is determined after pilot studies are accomplished to establish a baseline for normal traffic. These pilot studies showed that a high degree of significance for a chosen signal is 0.001 or 0.1%. So a signal found to have a greater significance, or a value of less than 0.001 triggers an alarm.

3.8 Evaluation Technique

Evaluation of this method is accomplished using analysis of the simulated TCPDUMP traffic contained in the DARPA data set. Since the traffic is simulated the results of this study will not extend directly to actual network traffic. However, the data is based on actual attack instances, providing confidence in the results. Simulated data is chosen as the best documented data available. Analysis of this simulation data provides a more easily controlled environment than measured data.

The experimental configuration begins with the selection of the workload data set. In this case the DARPA data set TCPDUMP file is chosen. Based on what is known about the specifics of the data set, an IP address for the computer in question is chosen. This computer is the target of a specific attack or attacks at known times. These times are recorded as truth data for producing performance metrics.

Once the IP address is found, the TCPDUMP file is parsed to create a time series containing packet arrival times, along with the payload size. This time series is cut to include only the attack traffic. This cut traffic is then submitted to the

Lomb-Scargle algorithm. The Matlab signal processing toolbox is used to compute the periodogram along with the significance levels.

The output of the Lomb-Scargle algorithm contains the frequency domain representation of the power spectral density of the time series data. This frequency representation is then analyzed to determine whether or not it contains any signals above the threshold. If features above the threshold are found, the traffic is flagged as containing an attack.

Once the traffic classification has been made, performance metrics are calculated by determining the rates of false positive and true positive. The performance of that configuration is then recorded, and modifications to the factors are performed in order to obtain the best performance of the system.

Validation of the simulation results are accomplished by duplicating experiments from the Zhou and Lang study [ZLC+03], and comparing results from their DFT based analysis.

3.9 Experimental Design

This study is conducted using a full factorial experimental design. Pilot studies are performed to determine initial attack signal characteristics. Once relevant characteristics have been identified, levels are chosen over which to perform the analysis.

System performance is determined for each run of the experiment, and these performances are in turn used to make adjustments to the attack signal characteristics. This results in a final system configuration with the best possible detection rate.

3.10 Methodology Summary

This study intends to determine what traffic characteristics are valid markers for network intrusion detection. It accomplishes this goal by selecting two characteristics, specifically, packet inter-arrival time and payload size. The study then uses those statistics to create a signal to analyze.

This signal analysis is performed using DSP techniques, specifically the Lomb-Scargle algorithm for generating a periodogram. This transformation technique takes the time series data and represents it in the frequency domain. Once the data has been transformed into this domain it can be further analyzed. This analysis consists of searching the periodogram for certain signals above a significance level. If a signal is found, the traffic is considered to contain an attack.

Once this determination has been made, that information is compared with what is known about the original data. In this case the original data is well documented, allowing for further analysis. Metrics for performance are determined based on the systems accuracy in classifying the traffic, and the experiment is repeated against multiple attacks to determine performance.

IV. Analysis and Results

This chapter outlines the procedures used to establish that digital signal processing is capable of intrusion detection. This process begins with how the network traffic is parsed to isolate attacks in Section 4.1. Section 4.2 discusses which attacks showed characteristics worth analyzing. The results of the Periodogram analysis are contained in Section 4.3; this section demonstrates that the chosen network traces did in fact contain enough frequency information to detect attacks. Finally, overall system performance is discussed in Section 4.4.

4.1 Data Preparation

The network trace data used to accomplish this research was obtained through the Massachusetts Institute of Technology's (MIT) Lincoln Laboratory website [Ref01]. This network data is organized by weeks. Weeks one, two and three contain training data for the test while weeks four and five contain the simulated attack data. This simulated attack data is used to construct the different time series and complete the analysis.

The network trace files are further broken down into days, with each day lasting from about 0800 to 0600 the following morning. The traffic is separated into two sections with traffic captured outside the firewall and inside the firewall being stored in separate files. For the purpose of this research, only the traffic allowed through the firewall was examined. This corresponds to a network architecture with the NIDS

system located behind the firewall. This may have excluded some attacks or pieces of attacks, and might represent an avenue for future research to consider.

Each day network traffic is stored in a zipped archive, ranging in size from 80 MB-200 MB. The files are stored as a TCPDUMP output file. These files are the output of a program, also called TCPDUMP, designed by the Lawrence Berkeley National Laboratory (LBNL). These capture files preserve the information in every packet that traveled through the network, making it possible to analyze their contents and timing.

Once the capture file is unpacked another tool called Wireshark, described more in Section 2.6.1, is used to view the data and ultimately isolate the traffic of interest [Com08]. For the purposes of this research, Wireshark's filtering ability is used to target specific IP addresses and times, in order to isolate the desired attacks. This allows the display of all connections inbound and outbound to the victim PC and the corresponding traffic. That traffic is then used to construct the time-series for analysis. Further filtering of the traffic provides an individual time series for each connection to the victim PC during the attack; these individual time series are the object of further periodogram analysis. The time series data is then output from Wireshark into a text file in the standard Wireshark output format.

Before the time series can be analyzed they must first be parsed. This parsing strips the unnecessary information from the file and organizes the data into columns containing pertinent information. Parsing the files allows for easier import into the

Matlab program for further analysis. The file is organized into columns where the first column is packet number, the second is arrival time, third is the IP address of the source computer, fourth is the inter-arrival time and fifth is the payload size. This is accomplished by way of a small C++ program executed on the command line. It is provided in Appendix B along with sample inputs and outputs.

After the time series has been generated, the files are then input into Matlab where the *networklomb.m* file is run against both statistics. This file outputs three graphical representations of the data, a mapping of the chosen statistic versus time (top), the Lomb-Periodogram output itself (middle) and the significance levels mapped versus frequency (bottom). The *networklomb.m* file is the heart of the Network Attack Characterization Tool. Along with the graphical output, the program makes a simple determination of whether or not a time series contains an attack. This output is determined by looking to see if any frequency in the periodogram over 5 Hz exists with significance greater than 0.001. If such a frequency is present, the alarm is set to true as an indication of an attack. Frequency content below 5 Hz was discarded because pilot studies showed a great deal of noise below that level. A significance level of 0.001 was chosen to represent a high degree of confidence. Future work may find that altering this significance level may tune the NACT, increasing or decreasing its sensitivity as discussed in Chapter V.

4.2 Attack Types

The attacks used to validate the use of DSP techniques for intrusion detection were extracted from the DARPA data set [Lab99]. Generally the attacks that demonstrated the highest level of frequency content were those that fell into the Denial of Service (DoS) category of attacks. This type of attack lends itself to frequency detection because it typically involves a larger number of packets and at some level is trying to overload the abilities of the target computer. This large number of packets offers greater amounts of data over which to run the algorithm, offering the benefit of higher confidence in the results. The DoS attacks targeted are the Processtable attack and the Teardrop attack.

4.2.1 ProcessTable Attack. The ProcessTable attack makes use of a vulnerability typically found on Unix/Linux machines where, if a service is mis-configured or simply vulnerable, will spawn a new process every time a user requests a connection. Most systems have a limit to how many processes might be started by users on the systems; however, since most services run as root this safeguard is often bypassed. Therefore the ProcessTable attack requests hundreds of connections with a service in the hopes of filling the victim's process table to the point where it can no longer answer new requests.

4.2.2 Dictionary Attack. The Dictionary attack is another attack that has good frequency characteristics, however, it is different than the previous attack in that it is not intended to be a DoS type attack. The Dictionary attack is simply an

attempt to remotely guess the passwords on a target system [Ref01]. It does this through repeated attempts to log in with different user names and passwords. This attack lends itself to frequency detection because it typically sends out a fixed number of attempts with a fixed delay between attempts.

4.2.3 Teardrop Attack. The Teardrop attack is another DoS type attack. This exploit uses some operating systems inability to re-assemble fragmented packets sent to them, causing the server to fail. This attack is the shortest attack, lasting only several seconds, making it difficult to obtain much frequency information from the connection.

4.3 Detection Results

Outlined here are some of the findings demonstrating the DSP technique used in this research; complete results are listed in Appendix C. For all three of the attacks listed in Section 4.2, two types of periodogram analysis are accomplished, one against the inter-arrival time of the packets and another against the payload size of those same packets. This results in two alarms, a timing alarm and a payload alarm. The timing alarm is set to true if the time series, using the inter-arrival statistic, triggers an alarm, and the payload alarm is set to true if its time series triggers an alarm.

4.3.1 Processtable Attack Results. The Processtable attack demonstrated the highest degree of success using this detection technique. Tables 4.1 and 4.2 include the connection listing for two separate instances of the attack against the same victim.

The connection column lists the IP address of the incoming connections while the packets column lists the total number of packets in the captured time series. These time series are constructed using all the packets traveling to and from the source and victim computer for the duration of the attack. The timing and payload alarm columns are indications of whether the Network Attack Characterization Tool flagged an attack versus the final column which is an indication of which connection actually contained the attack.

Table 4.1: Summary for ProcessTable Attack against 172.16.113.50, 31 March

Connection	Packets	Timing Alarm	Payload Alarm	Attack
172.16.112.10	76	False	False	False
172.16.112.20	905	False	**True**	False
172.16.112.149	158	False	False	False
172.16.113.84	319	False	False	False
172.16.113.204	104	False	False	False
172.16.114.148	324	False	False	False
172.16.114.207	465	False	False	False
172.16.118.60	782	**True**	False	**True**
194.7.248.153	611	False	False	False
195.115.218.108	968	False	False	False

Table 4.2: Summary for ProcessTable Attack against 172.16.113.50, 7 March

Connection	Packets	Timing Alarm	Payload Alarm	Attack
172.16.112.10	30	False	False	False
172.16.112.20	837	False	**True**	False
172.16.112.50	62	False	False	False
172.16.112.100	37	False	False	False
172.16.114.50	55	False	False	False
172.16.117.52	1871	**True**	False	**True**
196.227.033.189	1705	False	False	False

Further inspection of Tables 4.1 and 4.2 show that the attack was successfully detected through the observation of the packet inter-arrival time and that there were

no false alarms. The payload statistic was not as successful; it failed to detect either attack and generated one false alarm in both cases. Figures 4.1, 4.2, 4.3 and 4.4 illustrate the results of the Network Attack Characterization Tool graphical output.

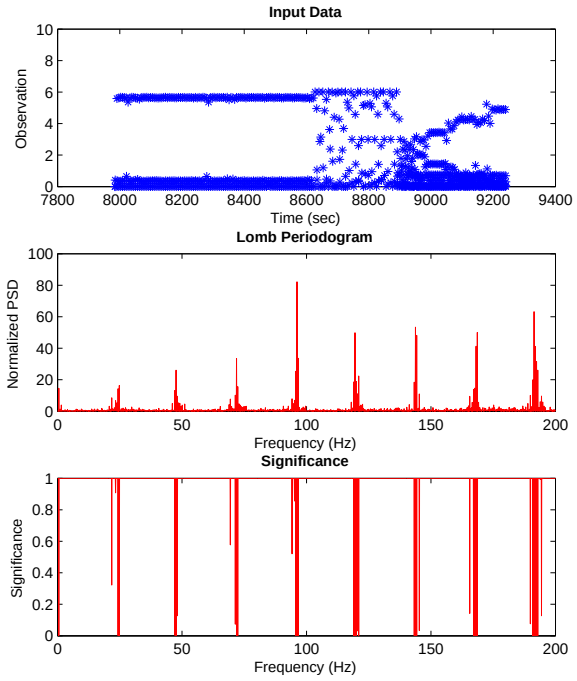


Figure 4.1: Processtable Attack Source=172.16.118.60, Inter-Arrival: Alarm **True**

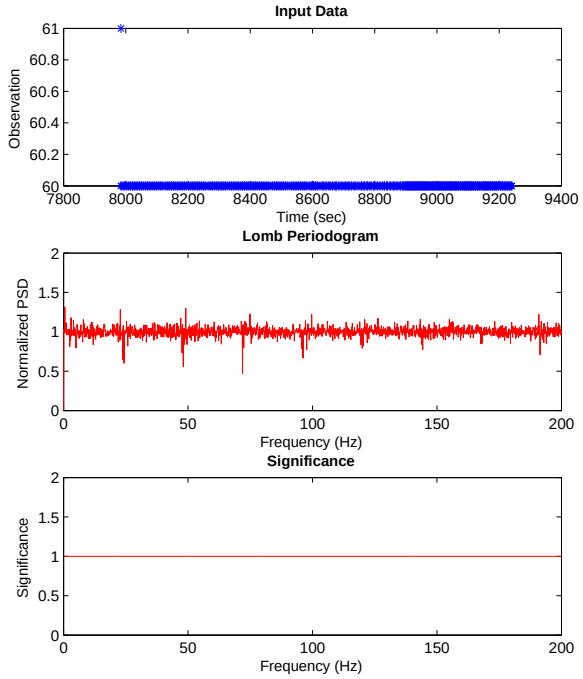


Figure 4.2: Processtable Attack Source=172.16.118.60, Payload: Alarm False

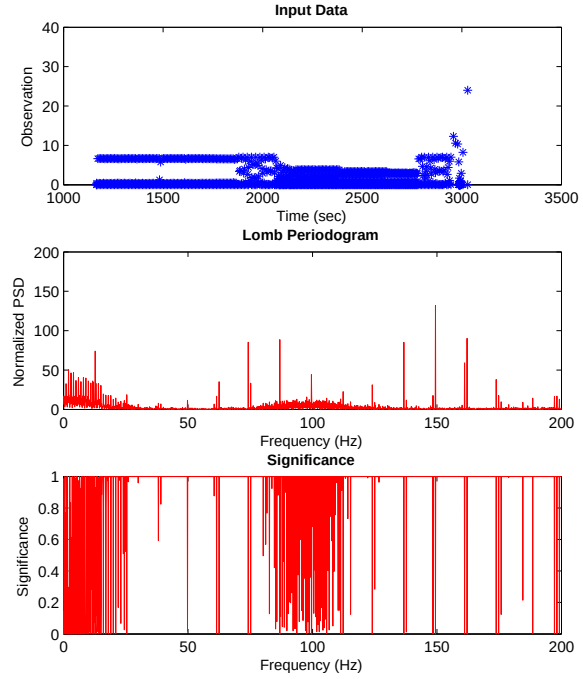


Figure 4.3: Processtable Attack Source=172.16.117.52, Inter-Arrival: Alarm **True**

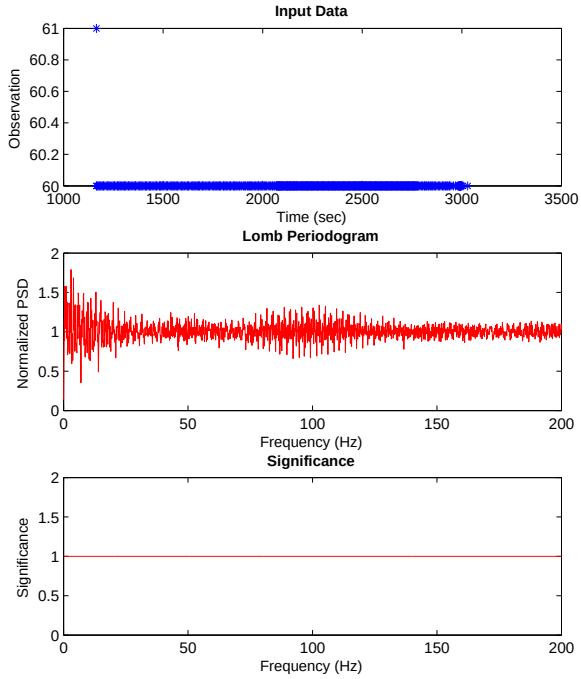


Figure 4.4: Processtable Attack Source=172.16.117.52, Payload: Alarm False

In Figure 4.1, the frequency features of the attack that are noticeable in the middle panel are much more pronounced than those in the same panel in Figure 4.3. The frequency spikes occur roughly every 25 Hz in Figure 4.1. A greater amount of noise is present in 4.3 which seems cloud the significance levels. The spikes in the frequency pattern are quantified using their significance level. This is shown by the strongest significance levels approaching zero in the bottom panel.

The performance of the NACT against the Processtable attack was determined using the equations from Section 2.3.1, calculated across both instances of the attack. The true positive rate for the timing alarm was 100% as compared to a false positive rate of 0%. This led to an overall accuracy of 100%, achieved without knowing any

of the specifics of the attack other than over what period it occurred. Conversely, the payload alarm had a true positive rate of 0% and a false positive rate of 13%, leaving an accuracy of 80%. This result seems counterintuitive, since the payload statistic failed to detect the attack at all. However, the accuracy still remains high because of the relatively small number of traces combined with the few false alarms.

4.3.2 Dictionary Attack Results. This section contains the results for the Dictionary attack. Unfortunately, this attack was launched only once in the data set making it difficult to draw any conclusions about future instances. Here again the inter-arrival time showed the greatest success, detecting the attack with zero false alarms as shown in Table 4.3. The Payload alarm was also unsuccessful again, failing to detect the attack at all and producing one false alarm.

Table 4.3: Summary for Dictionary Attack against 172.16.114.50

Connection	Packets	Timing Alarm	Payload Alarm	Attack
172.16.112.20	228	False	False	False
172.16.112.50	764	False	False	False
172.16.113.50	289	False	False	False
172.16.113.105	156	False	False	False
172.16.114.169	128	False	False	False
172.16.118.10	3438	**True**	False	**True**
195.115.218.108	2094	False	**True**	False

Figures 4.5 and 4.6 illustrate the attack connection, where the timing alarm in Figure 4.5 was true, indicating an attack. The features in this attack are not as pronounced as those in the processtable attack, however, they do translate into strong significance levels.

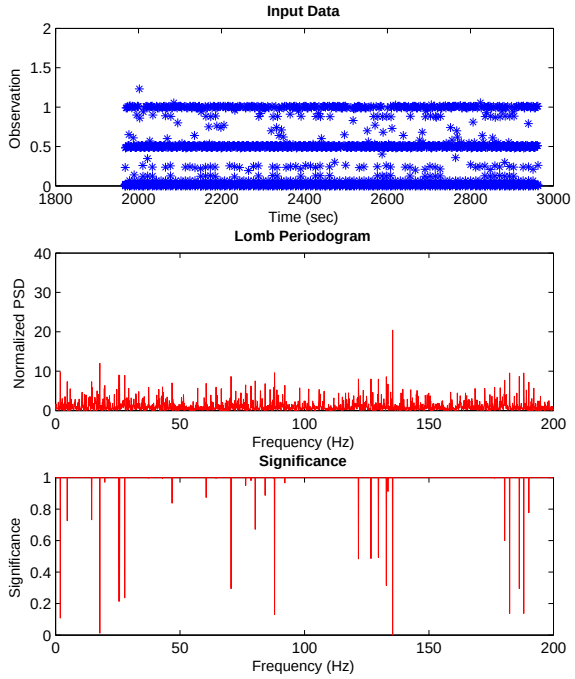


Figure 4.5: Dictionary Attack Source=172.16.118.10, Inter-Arrival: Alarm **True**

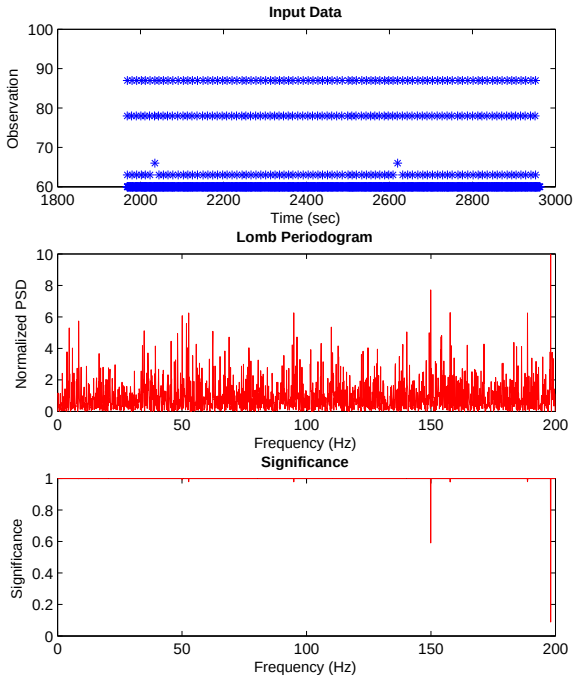


Figure 4.6: Dictionary Attack Source=172.16.118.10, Payload: Alarm False

The NACT performance against the dictionary attack is summarized here. For the timing alarm, the accuracy was again 100%, and for the payload alarm the true positive rate was 0% with a false positive rate of 16%, leaving an accuracy of about 71%. It is worth noting that in Figure 4.4.6 the payload alarm significance comes very close to accurately detecting the attack; this is indicated by the significance line coming very close to zero near 200 Hz in the bottom panel.

4.3.3 Teardrop Attack. The Teardrop attack is the shortest attack examined and the NACT achieved mixed results while attempting to detect it. Tables 4.4 and 4.5 indicate that the tool was successful only one time in detecting the attack despite the two instances in the data set. There were, however, no false alarms issued by either statistic.

Table 4.4: Summary for Teardrop Attack against 172.16.114.50, 6 April

Connection	Packets	Timing Alarm	Payload Alarm	Attack
172.16.112.10	17	False	False	False
172.16.114.148	24	False	False	False
206.48.44.50	493	False	False	False
207.230.54.203	90	**True**	False	**True**

Table 4.5: Summary for Teardrop Attack against 172.16.114.50, 8 April

Connection	Packets	Timing Alarm	Payload Alarm	Attack
135.16.216.191	14	False	False	False
172.16.113.105	41	False	False	False
199.227.99.125	20	False	False	**True**

Figures 4.7 and 4.8 demonstrate just how small the attack is with only 90 packets over which to run the analysis. It takes a strong embedded signal to trigger the alarm with this few packets to construct the time series. The top panel in Figure 4.7

illustrates the time series created by the inter-arrival times in the attack traffic. The horizontal lines indicate a consistency in the data, this consistency is mirrored in the second panel of that same figure where spikes in the spectrum are visible.

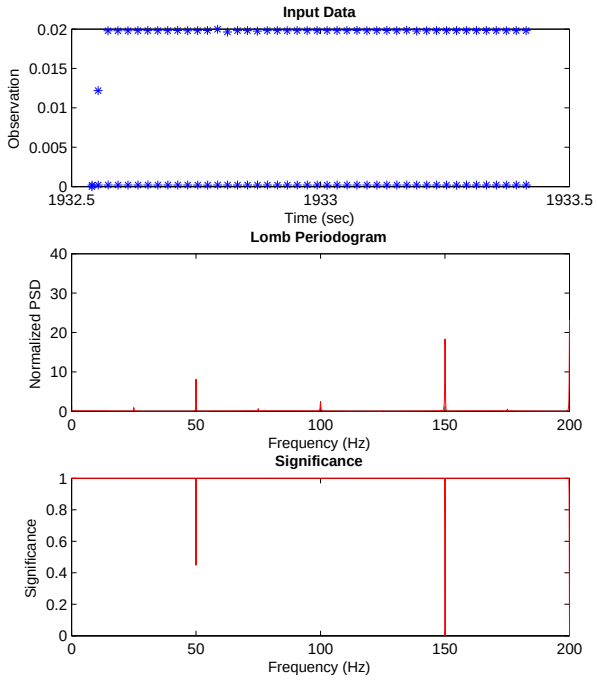


Figure 4.7: Teardrop Attack Source=207.230.54.203, Inter-Arrival: Alarm **True**

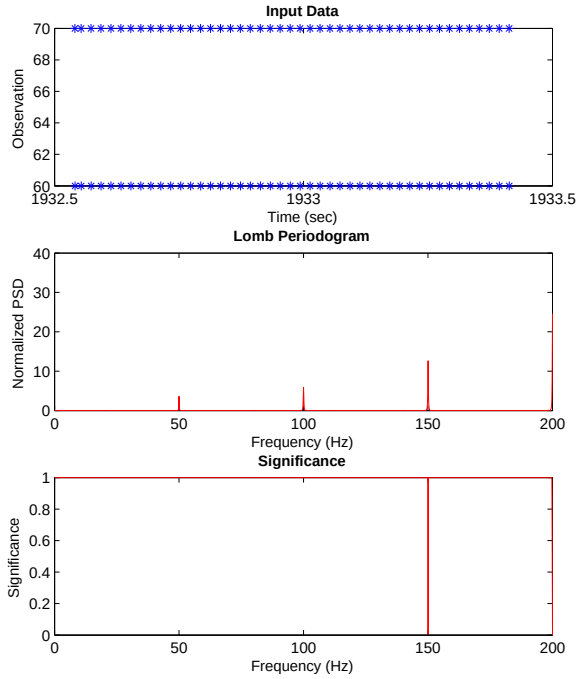


Figure 4.8: Teardrop Attack Source=207.230.54.203, Payload: Alarm False

The attack in Figure 4.9 is shorter still, probably making the difference in missing the attack. Features similar to that in Figure 4.7 are present in Figure 4.9. These peaks in frequency, found in the middle panel of both figures, occur at 50 Hz and 150 Hz. Unfortunately, because of the low packet count, the significance level never reaches 0.001 in Figure 4.9. This is explained after examining (2.12) in where M varies based on the number of frequencies calculated by the periodogram algorithm. This in turn depends on how many data points exist in the time series of interest. Fewer sampled packets makes it more difficult to be certain of an embedded signal.

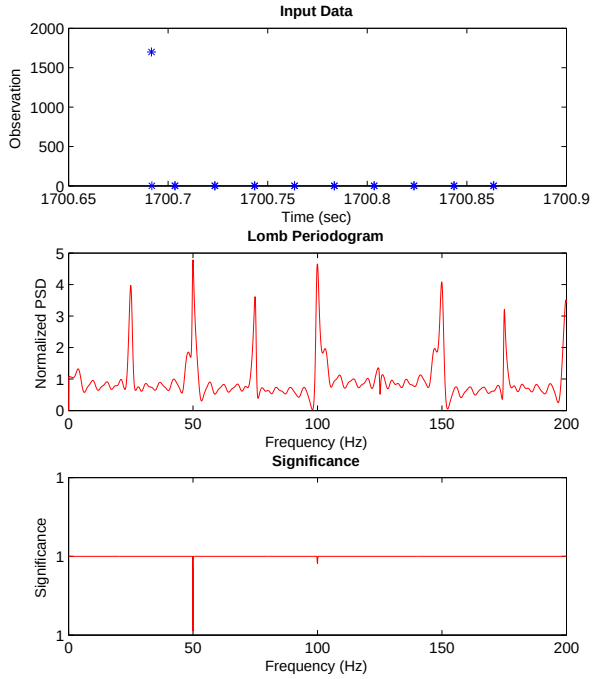


Figure 4.9: Teardrop Attack Source=199.227.99.125, Inter-Arrival: Alarm False

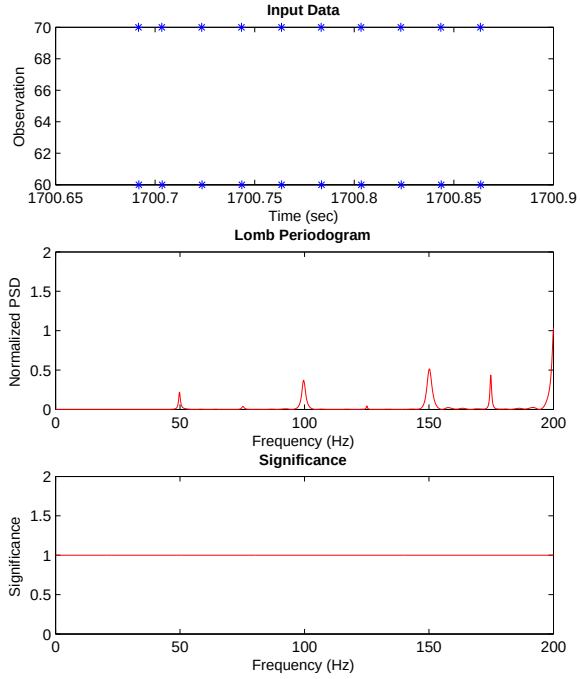


Figure 4.10: Teardrop Attack Source=199.227.99.125, Payload: Alarm False

NACT performance against the teardrop attack is a bit misleading, due mostly to the small number of connections examined. For the timing alarm, the true positive rate was 50% and the false positive rate was 0%, since there were no false positives. The accuracy for the timing alarm against the teardrop attack was 85%. For the payload alarm the true positive and false positive rates were both 0%, since no alarms were raised. The accuracy turned out to be only slightly lower than the timing alarm at 71%.

4.4 System Performance

Overall performance of this system is measured by the ability of the NACT to successfully detect attacks. In this case, the performance is measured across all three types of attacks to include all instances. This was accomplished by totalling the true and false positive rates along with the true and false negative rates for both timing and payload. For the timing alarm the NACT had an overall true positive rate of 80%, a false positive rate of 0% and an overall accuracy of 96%. The payload alarm had a true positive rate of 0% as it never accurately identified an attack, a false positive rate of 11% and an overall accuracy of 74%. These results are summarized in Table 4.6.

Table 4.6: Combined Results for All Attacks

Attack/Statistic	True Pos.	True Neg.	False Pos.	False Neg.	Accuracy
Processtable/Timing	2	15	0	0	100%
Processtable/Payload	0	13	2	2	80%
Dictionary/Timing	1	6	0	0	100%
Dictionary/Payload	0	5	1	1	71%
Teardrop/Timing	1	5	0	1	85%
Teardrop/Payload	0	5	0	2	71%
Total/Timing	4	26	0	1	96%
Total/Payload	0	23	3	5	74%

These measures for system performance are only intended to compare the two alarms and demonstrate that detection is possible with frequency analysis. The accuracy and rates of true and false positive are significantly biased by the low number of trials run, this makes it difficult to compare the accuracy of the NACT to other NIDS. The results are, however, useful in tuning the NACT along with future intrusion detection systems. For instance it seems that using payload size as a statistic may not be an avenue worth pursuing in future iterations of the system.

4.5 Results Summary

Section 4.1 outlined the process through which the data flowed in order to accomplish the analysis, ending with the *networklomb.m* Matlab file which produces the ultimate results. The attacks over which the analysis was performed were discussed in Section 4.2. Periodogram results were listed in Section 4.3, with the most successful detection occurring over the Processtable attack. Complete results, including all summary graphs, are contained in Appendix C. Finally the overall system performance

was discussed in Section 4.4. This chapter served as the foundation for the assertion that DSP techniques can be leveraged for network intrusion detection.

V. Conclusions

5.1 Problem Restatement and Conclusions

The stated goal for this research effort is to establish that digital signal processing (DSP) techniques can be leveraged for use in network intrusion detection. Current intrusion detection techniques employ many different strategies in order to successfully detect attacks against their networks. Most of these strategies require some knowledge of the attack, typically in the form of a signature or behavior pattern.

Chapter IV demonstrates that through the use of the Network Attack Characterization Tool (NACT), some of these attacks are detected using DSP techniques. The technique specific to this research is the Lomb-Scargle periodogram, a method for obtaining the spectrum of unevenly sampled data [Pre02]. Once this spectrum is obtained the significance of features within the spectrum is used to trigger an alarm.

The NACT was approximately 96% accurate against the Processtable, Dictionary and Teardrop attacks; when using the inter-arrival time of the packets to construct the time series of interest. This indicates that using DSP techniques to detect network intrusions can be successful. A weakness of these results, however, is that the attacks chosen for analysis were attacks known to contain large amounts of structured traffic. Attacks that make use of less traffic are much more difficult for the system to detect.

5.2 Contributions and Significance

The main contribution of this research effort is the demonstration of a successful detection tool constructed using digital signal processing techniques. This technique for network intrusion detection represents an alternate means of detecting attack traffic. This approach is not intended to replace current NIDS equipment or techniques, but rather to augment those systems in order to increase their accuracy.

5.3 Directions for Future Research

In the process of accomplishing this research, a number of avenues for future research were discovered. The two statistics sampled to construct the time series were only a few of many other options. Some others could include packet fragmentation, protocol type, and port numbers. Any of these other statistics, or combinations of those statistics, may hold further insight into different attacks. The statistics used in this research could be merged, forming a hybrid time series over which to perform the same type of analysis.

The significance threshold at which an attack is triggered was not varied for this research. Future work may find that tuning this value increases or decreases the sensitivity of the detector. Along with this threshold value there was the frequency cutoff of 5 Hz; this number was chosen after several test runs. It may be worth exploring over which values the tool is most successful. Future research may find that successful attack detections occur only within a certain frequency range. Limiting this range may in fact limit false alarms, and increase accuracy.

Finally, a much more exhaustive test is needed. What was accomplished in this thesis was only meant to provide impetus for further research. The few attacks that were chosen are only a sliver of the threats available, and further testing is the only way to determine the ultimate usefulness of the NACT.

5.4 Summary

This work demonstrates an intrusion detection tool employing digital signal processing to detect network intrusions. The NACT employs the Lomb-Scargle periodogram method to obtain a spectrum for the sampled network traffic. Any features that exist in the spectrum above a significance level are considered an attack, triggering an alarm.

Two traffic statistics were used to test the ability of the NACT to accurately detect intrusions. The statistics were packet inter-arrival time and payload size. Detection using inter-arrival time achieved an accuracy of 96%, while using the payload statistic never actually detected an attack. These results demonstrate that digital signal processing techniques can be used to detect network intrusions.

Appendix A. Matlab Code

This appendix includes the implementation of three important Matlab files referenced throughout this research.

A.1 Lomb-Scargle Periodogram Implementation in Matlab

Listing A.1: Matlab implementation of the Lomb-Scargle periodogram method. This file is called by both subsequent files in this appendix.
(appendix1/lomb.m)

```
%
%       [Pn, Prob] = lomb(t, y, freq)
%
%               Uses Lomb's method to compute normalized
%               periodogram values "Pn" as a function of
%               supplied vector of frequencies "freq" for
%               input vectors "t" (time) and "y" (observations).
%               Also returned is probability "Prob" of same
%               length as Pn (and freq) that the null hypothesis
%               is valid.
%
%               Caveat: x and y must be the same length.
%
%
%       Authors: This implementation was designed for the "12.747:
%       Modeling, Data Analysis and Numerical Techniques for Geochemistry"
%       course at the Woods Hole Oceanographic Institution.
%
%       Changelog: Created by the authors circa Fall 1996
%               Modified Jan 2008 - additional comments and formatting
%               changes were made by 1Lt Theodore Erickson, USAF
%


function [Pn, Prob] = lomb(t, y, freq)
%
%               check inputs:
%       throw error if t and y are not the same length
if length(t) ~= length(y); error('t and y not same length');
        exit; end;

%
%       subtract mean, compute variance, initialize Pn
z = y - mean(y);
var = std(y);
N=length(freq);
Pn=zeros(size(freq));

%
%       Execute main loop for all frequencies
for i=1:length(freq)
    w=2*pi*freq(i);
    if w > 0
       twt = 2*w*t;
       tau = atan2(sum(sin(twt)),sum(cos(twt)))/2/w;
       wtmt = w*(t - tau);
       Pn(i) = (sum(z.*cos(wtmt)).^2)/sum(cos(wtmt).^2) + ...
                (sum(z.*sin(wtmt)).^2)/sum(sin(wtmt).^2);
```

```
        else
            Pn(i) = (sum(z.*t).^2)/sum(t.^2);
51      end
    end

    %
    %       and normalize by variance, compute probabilities
56  Pn=Pn/2/var.^2;
    Prob = 1-(1-exp(-Pn)).^N;
    for i=1:length(Pn)              % accomodate possible roundoff error
        if Prob(i) < .001
            Prob(i) = N*exp(-Pn(i));
61      end
       end
```

A.2 Periodogram Analysis using Matlab

Listing A.2: Matlab implementation of periodogram analysis using the Lomb-Scargle Periodogram. This file is the heart of the NACT. (appendix1/networklomb.m)

```
 1  %
    %       networklomb(fileToRead1,column)
    %
    %               column 4 contains inter arrival time and column 5 contains
    %       payload sizes
 6  %
    %       This program calls lomb.m
    %
    %       Authors: This implementation was designed for the "12.747:
    %       Modeling, Data Analysis and Numerical Techniques for Geochemistry"
11  %       course at the Woods Hole Oceanographic Institution.  Adapted from
    %       the algorithm found in the text "Numerical Recipes in C++"
    %
    %       Changelog: Created by the authors circa Fall 1996
    %               Modified Jan 2008 - additional comments and formatting
16  %               changes were made by 1Lt Theodore Erickson, USAF
    %

    function networklomb(fileToRead1,column)

21  %import data from the text file
    inputdata = importdata(fileToRead1);

    t = inputdata(:,2);
    y = inputdata(:,column);
26

    % plot the data
    subplot(311);
    plot(t,y,'*');
31  xlabel('Time (sec)');
    ylabel('Observation');
    title('Input Data','fontweight','b')

    %first create a vector of frequency bins
36  f=[0:.1:200];
    %now compute the lomb periodogram
    [Pn Prob]=lomb(t,y,f);

    %plot the periodogram
41  subplot(312)
    h=plot(f,Pn,'r');
    xlabel('Frequency (Hz)');
```

```
ylabel('Normalized PSD');
title('Lomb Periodogram','fontweight','b');

%plot the signficance levels
subplot(313)
h=plot(f,Prob,'r');
xlabel('Frequency (Hz)');
ylabel('Significance');
title('Significance','fontweight','b');
%
%       then sort for smallest values first
%               (ie. those points least likely to be random)
%
[p,ind]=sort(Prob);
most=ind(1:2000);                               % all the values
attack=[f(most)' Pn(most)' Prob(most)'];

% Test to see if any of the probabilities above 5Hz contain a
%   significance less that .001, if so output a true indicator
indicator = 0;
for k=1:2000

    if ((f(k)>=5.0) && (Prob(k)<=0.001))
        indicator=1;
    end
end
%attack
indicator
```

A.3 Lomb-Scargle Periodogram Demonstration Implemented in Matlab

Listing A.3: Matlab implementation of the Lomb-Scargle periodogram demonstration included in Chapter II.
(appendix1/lombtestthesis.m)

```
%    Tests the Lomb Periodogram algorithm and plots results
%
%       Authors: This implementation was designed for the "12.747:
%       Modeling, Data Analysis and Numerical Techniques for Geochemistry"
%       course at the Woods Hole Oceanographic Institution.
%
%       Changelog: Created by the authors circa Fall 1996
%               Modified Jan 2008 - additional comments and formatting
%               changes were made by 1Lt Theodore Erickson, USAF

%           first, create a random set of times from 0 to 1 seconds with
%               the rand function, and sort in ascending order
%
t=rand(150,1); t=sort(t);
tt=0:.001:1;

%                       choose two frequencies to build data
f1=50; f2=130;

%                       then create data, with some noise
y=2*sin(2*pi*f2*t)+0.5*randn(size(t));

%       next, plot the data

subplot(412); plot(t,y,'r*');           % plot data
set(gca,'FontSize',10);
xlabel('Time (sec)'); ylabel('Observation'); title('Samples (150)')

% Now Plot the generator function
```

```
subplot(411);
z=2*sin(2*pi*f2*tt);
plot(tt,z);
set(gca,'FontSize',10);
xlabel('Time (sec)'); ylabel('Observation');title('Generator');

%
%               Now do compute the Lomb normalized periodogram
%
%       first create a vector of frequency bins, 1 per Hz
f=[0:200];
[Pn Prob]=lomb(t,y,f);

%       and plot the periodogram
%
subplot(413)
h=plot(f,Pn,'r'); set(h,'LineWidth',2);
xlabel('Frequency (Hz)'); ylabel('Normalized PSD');
title('Lomb Periodogram'); set(gca,'FontSize',10);
%
%       and plot the probabilities
%
subplot(414)
h=plot(f,Prob,'r'); set(h,'LineWidth',2);
xlabel('Frequency (Hz)'); ylabel('Probability');
set(gca,'FontSize',10);
%
%       then sort for smallest values first
%               (ie. those points least likely to be random)
%
[p,ind]=sort(Prob);
most=ind(1:5);                          % just the first 5 values
ANS=[f(most)' Pn(most)' Prob(most)']
```

Appendix B. C++ Code

This appendix includes the implementation of the C++ parsing program.

B.1 C++ Parsing Program

Listing B.1: C++ Parsing Code.(appendix3/parse.cpp)

```
#include <fstream>                      //files streams
#include <iostream>                     //standard streams
#include <sstream>                      //string stream support
#include <string>                       //string variable support
#include <iomanip>
#include <list>
#include <vector>
#include "nr.h"

using namespace std;            //using standard library

void main(int argc, char **argv) {


        string inName, outName;//strings for the input and output file names
        unsigned int number;
        double time, interarrival;
        string test, test2, test3, test4, test5;//string to test if I've got a ...
            spare line
        string Line1;//use this to test inside my if statements
        string source;
        string framestring;
        bool first, second, third, debug;
        //vector<string> sources;

        if (argc <=1){
                //useful debugging message
                cout << "Program Usage: " << argv[0] << " <filename>" << endl;
                return; //fail the program
        }else{

                inName=argv[1];//save the filename

        }

        ifstream iFile(inName.c_str());

        if(!iFile.is_open()){

                cout << "Bad Input Filename" << endl;

                return;
        }

        cout.precision(12); //increases the decimal precision for the output
        debug=false;//debug on or off

        /*////////////////////////////////////////////
        //Here's the code for opening the input file
        ////////////////////////////////////////////
        cout << "Please enter new input filename:  ";
        getline(cin, fName);
        cin.ignore(1, '\n');
        ifstream iFile(fName.c_str());
```

```
        while ( iFile.fail()) {
                iFile.clear();
                cout << "File not found:  " << fName << endl;
57              cout << "Please enter new input filename:  ";
                getline(cin, fName);
                cin.ignore(1, '\n');
                iFile.open(fName.c_str());
        }*/
62
        ////////////////////////////////////////////
        //Here's the code for creating the output file
        ////////////////////////////////////////////
        outName="Output_" + inName;
67      ofstream outFile(outName.c_str());

        outFile.precision(16);//set the decimal precision for the outfile


72
        //initialize the testing counter
        int count = 0;
        first = false;
        second = false;
77      third = false;

        //Grab the first line to start the loop
        getline(iFile, test);

82      //Loop until the end of the file
        while (iFile) {

                istringstream In(test);

87              In >> test >> test2 >> test3 >> test4 >> test5;

                if(debug==true){
                cout << test << test2 << test3 << test4 << test5 << endl;
                }
92
                //The First Case
                if (test == "No."){
                        getline(iFile, Line1);
                        istringstream In(Line1);
97                      In >> number >> time >> source;
                        first = true;

                        if(debug==true){
                        cout << "First Case" << endl;
102                     }

                }

                //The Second Case
107             if ((test == "[Time") && (test5=="displayed")){
                        In.ignore(INT_MAX, ':');
                        In >> interarrival;
                        second = true;

112                     if(debug==true){
                        cout << "Second Case" << endl;
                        }
                }

117             //The Third Case
                if ((test == "Frame") && (test2=="Length:")){
```

```
                framestring = test3;

122             third = true;

                if(debug==true){
                cout << "Third Case" << endl;
                }
127         }

            //The Fourth Case
            if (first && second && third){
                cout    << setw(10) << number << setw(20) << time << setw...
                    (20) << source << setw(25) <<interarrival << setw(10) ...
                    << framestring <<endl;
132             outFile << setw(10) << number << setw(20) << time << setw...
                    (20) << source << setw(25) <<interarrival << setw(10) ...
                    << framestring <<endl;
                first = false;
                second = false;
                third =false;

137

                //Increment the testing counter
                count = count++;
                cout << "Record #: " << count << endl;

142             if(debug==true){
                cout<< "Fourth Case" << endl;
                }

            }
147

            //Reset the input streams
            In.clear();

152

            //Grab another line
            getline(iFile, test);
        }

157     iFile.close();


}
```

B.2 Sample Input

This section contains some sample input to the parse.cpp program. This input is obtained after viewing the desired traffic in Wireshark, and selecting the option to output to a text file the displayed packets.

```
No.     Time        Source                Destination           Protocol Info
  73111 4398.532380 172.16.112.10         172.16.113.50         NTP      NTP server

Frame 73111 (90 bytes on wire, 90 bytes captured)
    Arrival Time: Mar 31, 1999 10:13:27.607540000
    [Time delta from previous captured frame: 0.000478000 seconds]
    [Time delta from previous displayed frame: 4398.532380000 seconds]
    [Time since reference or first frame: 4398.532380000 seconds]
    Frame Number: 73111
    Frame Length: 90 bytes
    Capture Length: 90 bytes
```

```
    [Frame is marked: False]
    [Protocols in frame: eth:ip:udp:ntp]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
Ethernet II, Src: SunMicro_83:4a:82 (08:00:20:83:4a:82), Dst: SunMicro_09:b9:49 (08:00:20:09:b9:49)
Internet Protocol, Src: 172.16.112.10 (172.16.112.10), Dst: 172.16.113.50 (172.16.113.50)
User Datagram Protocol, Src Port: ntp (123), Dst Port: ntp (123)
Network Time Protocol

No.     Time        Source                Destination           Protocol Info
  75186 4462.517102 172.16.112.10         172.16.113.50         NTP      NTP server

Frame 75186 (90 bytes on wire, 90 bytes captured)
    Arrival Time: Mar 31, 1999 10:14:31.592262000
    [Time delta from previous captured frame: 0.000364000 seconds]
    [Time delta from previous displayed frame: 63.984722000 seconds]
    [Time since reference or first frame: 4462.517102000 seconds]
    Frame Number: 75186
    Frame Length: 90 bytes
    Capture Length: 90 bytes
    [Frame is marked: False]
    [Protocols in frame: eth:ip:udp:ntp]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
Ethernet II, Src: SunMicro_83:4a:82 (08:00:20:83:4a:82), Dst: SunMicro_09:b9:49 (08:00:20:09:b9:49)
Internet Protocol, Src: 172.16.112.10 (172.16.112.10), Dst: 172.16.113.50 (172.16.113.50)
User Datagram Protocol, Src Port: ntp (123), Dst Port: ntp (123)
Network Time Protocol

No.     Time        Source                Destination           Protocol Info
  77702 4526.485501 172.16.112.10         172.16.113.50         NTP      NTP server

Frame 77702 (90 bytes on wire, 90 bytes captured)
    Arrival Time: Mar 31, 1999 10:15:35.560661000
    [Time delta from previous captured frame: 0.000445000 seconds]
    [Time delta from previous displayed frame: 63.968399000 seconds]
    [Time since reference or first frame: 4526.485501000 seconds]
    Frame Number: 77702
    Frame Length: 90 bytes
    Capture Length: 90 bytes
    [Frame is marked: False]
    [Protocols in frame: eth:ip:udp:ntp]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
Ethernet II, Src: SunMicro_83:4a:82 (08:00:20:83:4a:82), Dst: SunMicro_09:b9:49 (08:00:20:09:b9:49)
Internet Protocol, Src: 172.16.112.10 (172.16.112.10), Dst: 172.16.113.50 (172.16.113.50)
User Datagram Protocol, Src Port: ntp (123), Dst Port: ntp (123)
Network Time Protocol

Continued...
```

B.3 Sample Output

This section contains some sample output obtained after parsing the sample input file.

```
    73111          4398.53238       172.16.112.10              0.0           0
    75186         4462.517102       172.16.112.10              63.984722       90
```

77702	4526.485501	172.16.112.10	63.968399	90
79336	4590.45371	172.16.112.10	63.968209	90
80239	4654.412086	172.16.112.10	63.958376	90
81156	4718.380345	172.16.112.10	63.968259	90
82297	4782.349253	172.16.112.10	63.968908	90
83547	4846.317122	172.16.112.10	63.967869	90
86244	4910.285995	172.16.112.10	63.968873	90
88644	4974.254602	172.16.112.10	63.968607	90
90366	5038.222591	172.16.112.10	63.967989	90
92854	5102.191042	172.16.112.10	63.968451	90
93688	5166.158629	172.16.112.10	63.967587	90
94437	5230.127581	172.16.112.10	63.968952	90
95031	5294.09686	172.16.112.10	63.969279	90
97323	5358.063556	172.16.112.10	63.966696	90
97621	5422.031773	172.16.112.10	63.968217	90
98767	5486.000149	172.16.112.10	63.968376	90
101438	5549.958484	172.16.112.10	63.958335	90
102698	5613.927384	172.16.112.10	63.9689	90
104177	5677.895122	172.16.112.10	63.967738	90
106055	5741.864053	172.16.112.10	63.968931	90
107954	5805.832306	172.16.112.10	63.968253	90
109647	5869.80002	172.16.112.10	63.967714	90
111665	5933.768394	172.16.112.10	63.968374	90

Continued...

Appendix C. Results Organized By Attack

This appendix includes the periodogram data for each connection of each attack that was analyzed in this research.

C.1 The ProcessTable Attack

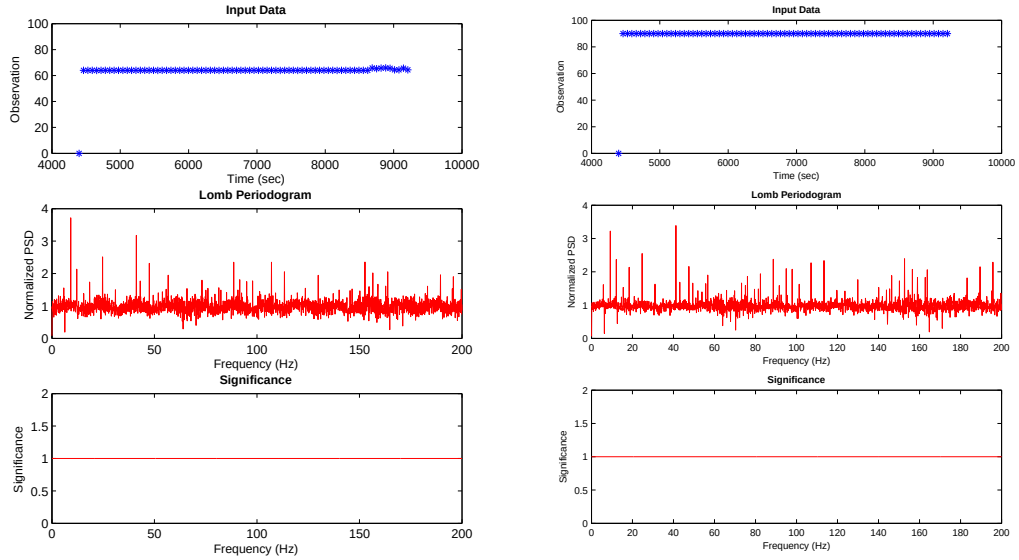
C.1.1 Victim: 172.16.113.50 on 31 March 1999. This section contains the pertinent information describing the ProcessTable attack mounted against IP address 172.16.113.50 by IP address 172.16.118.60 on 31 March 1999.

Table C.1: ProcessTable Attack against 172.16.113.50

Name	processtable
Date	03/31/1999
Category	Denial of Service
Duration	00:28:02
Attacker	172.16.118.60
Victim	172.16.113.50

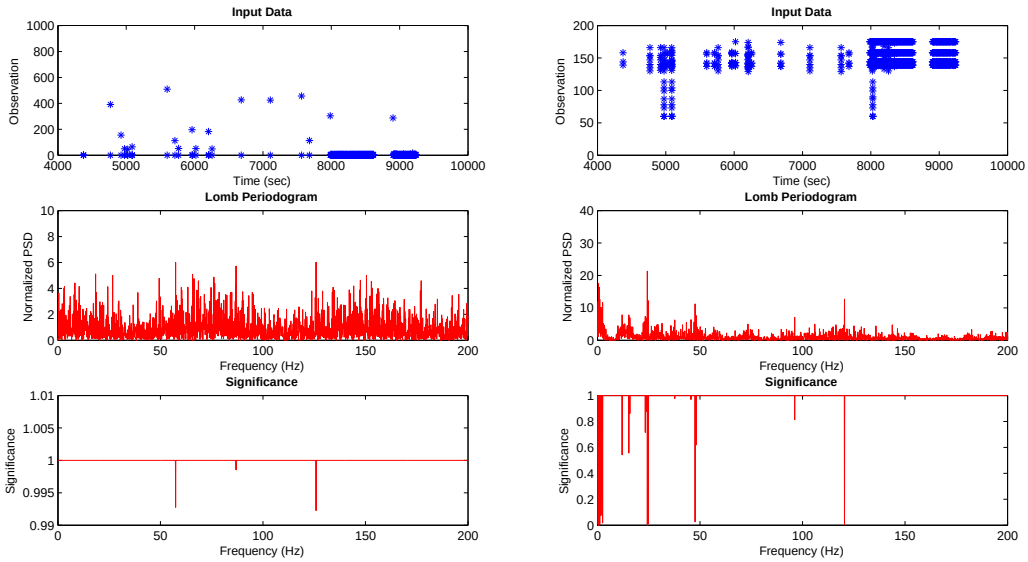
Table C.2: Summary for ProcessTable Attack against 172.16.113.50

Connection	Packets	Timing Alarm	Payload Alarm	Attack
172.16.112.10	76	False	False	False
172.16.112.20	905	False	**True**	False
172.16.112.149	158	False	False	False
172.16.113.84	319	False	False	False
172.16.113.204	104	False	False	False
172.16.114.148	324	False	False	False
172.16.114.207	465	False	False	False
172.16.118.60	782	**True**	False	**True**
194.7.248.153	611	False	False	False
195.115.218.108	968	False	False	False



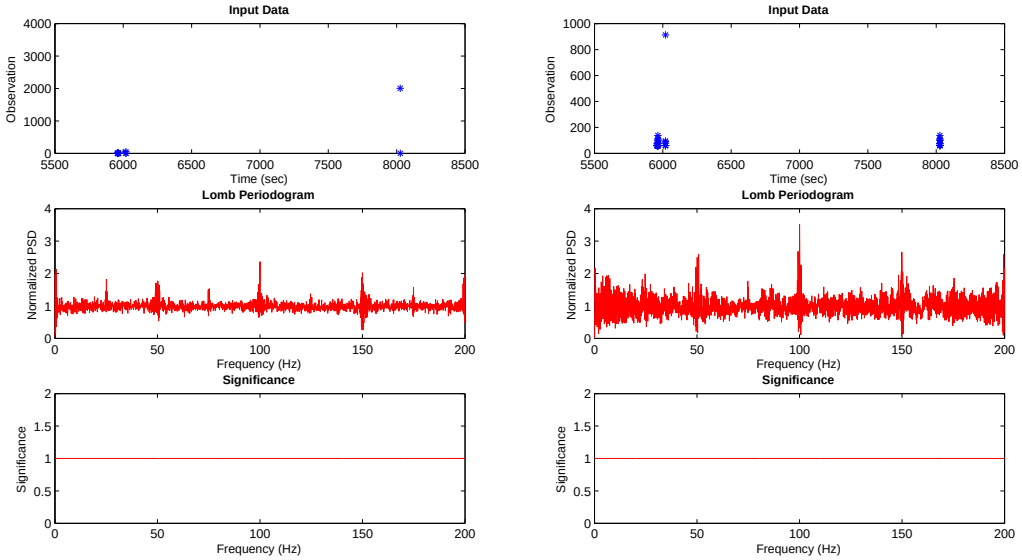
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.1: ProcessTable Attack Source=172.16.112.10



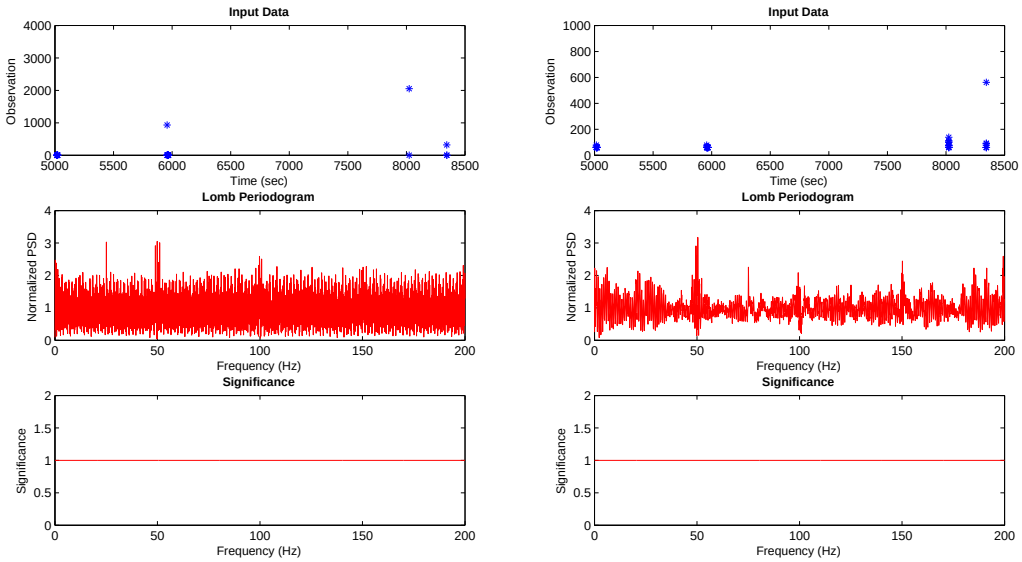
(a) Inter-Arrival: Alarm False (b) Payload: Alarm **True**

Figure C.2: ProcessTable Attack Source=172.16.112.20



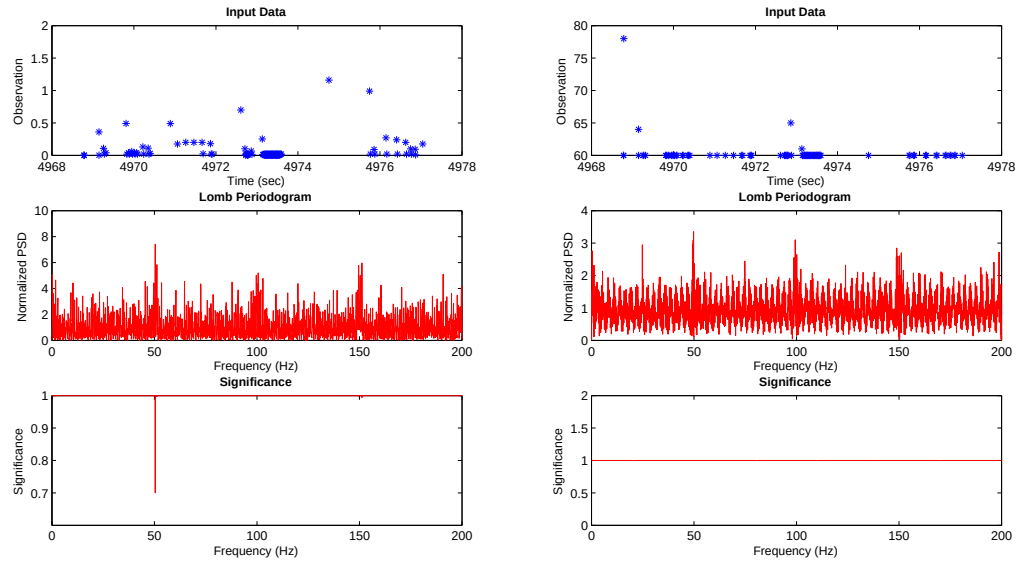
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.3: ProcessTable Attack Source=172.16.112.149



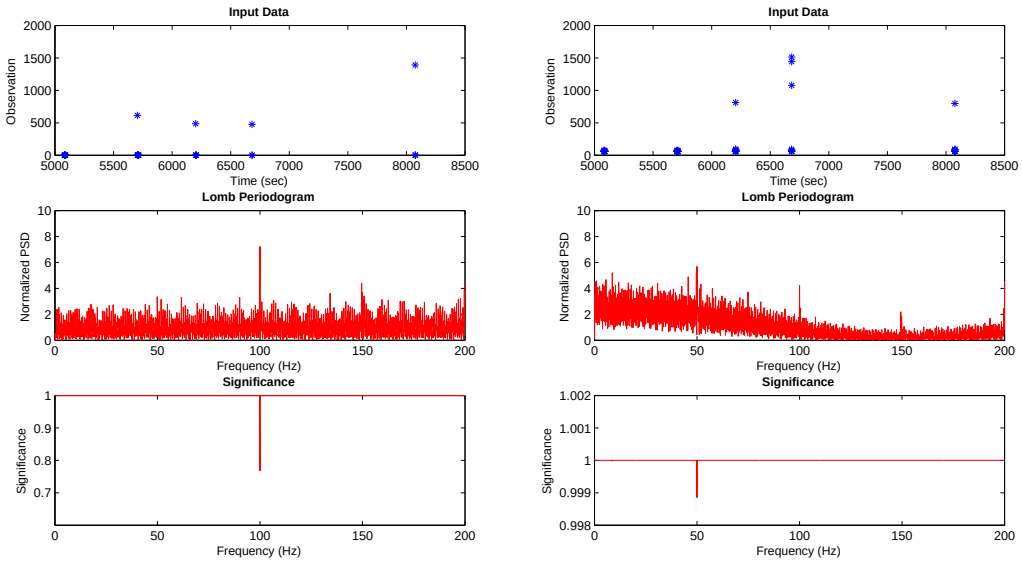
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.4: ProcessTable Attack Source=172.16.113.84



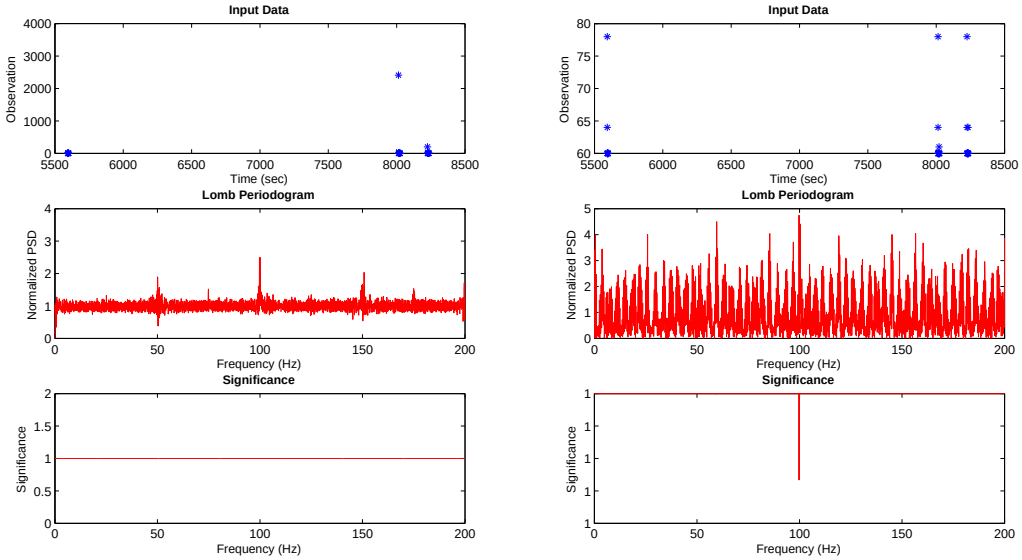
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.5: ProcessTable Attack Source=172.16.113.204



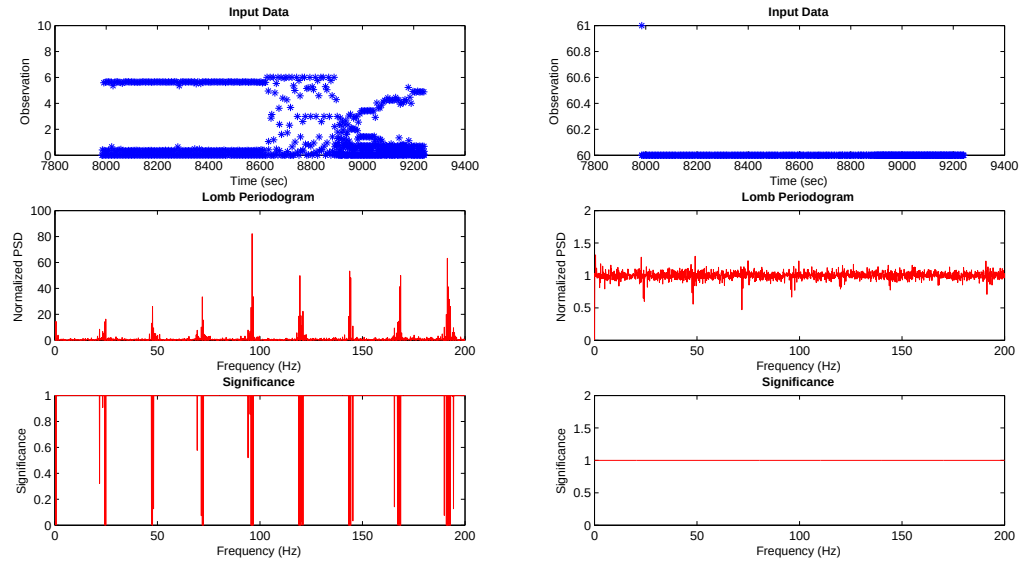
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.6: ProcessTable Attack Source=172.16.114.148



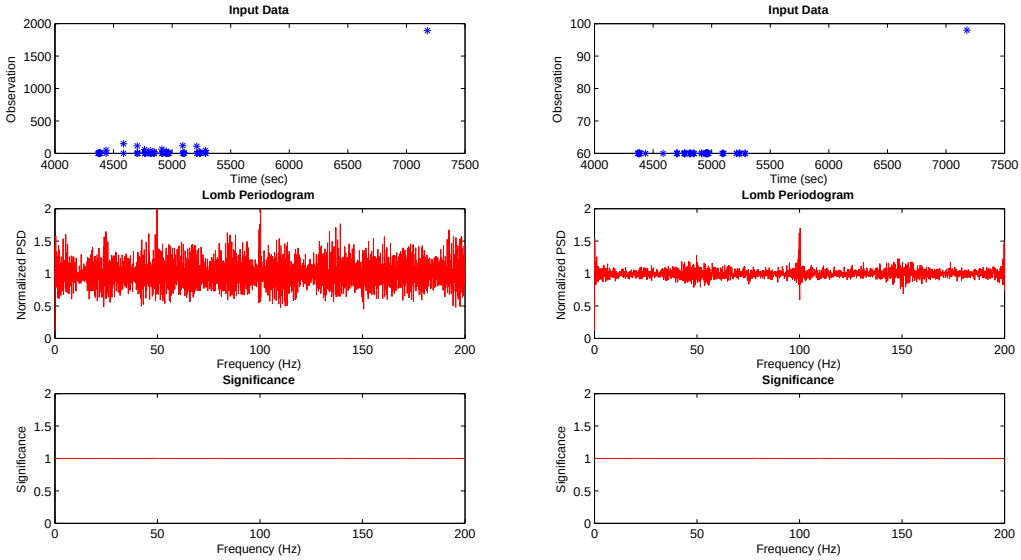
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.7: ProcessTable Attack Source=172.16.114.207



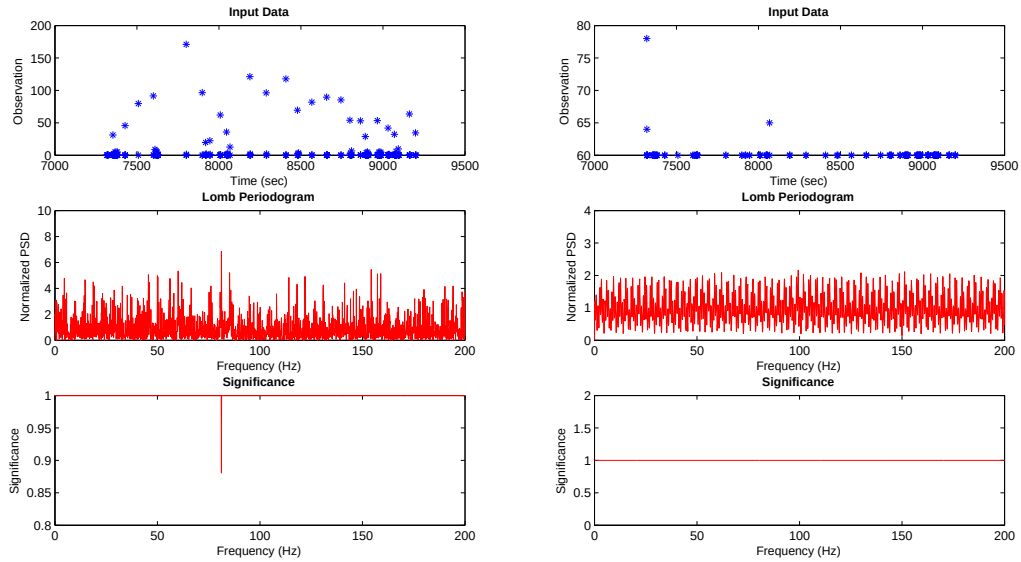
(a) Inter-Arrival: Alarm **True** (b) Payload: Alarm False

Figure C.8: ProcessTable Attack Source=172.16.118.60



(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.9: ProcessTable Attack Source=194.7.248.153



(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.10: ProcessTable Attack Source=195.115.218.108

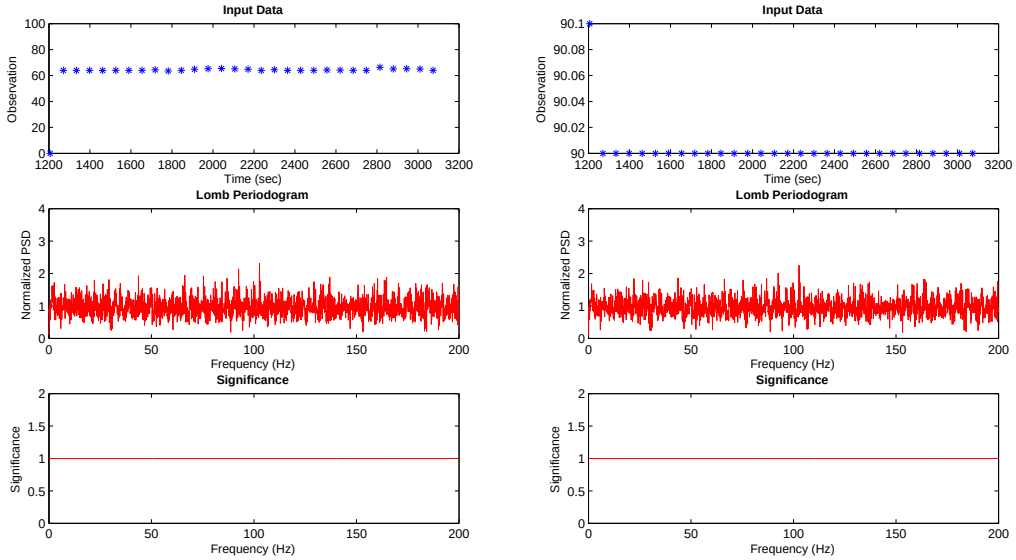
C.1.2 Victim: 172.16.113.50 on 7 April 1999. This section contains the pertinent information describing the ProcessTable attack mounted against IP address 172.16.113.50 by IP address 172.16.117.52 on 7 April 1999.

Table C.3: ProcessTable Attack against 172.16.113.50

Name	processtable
Date	04/07/1999
Category	Denial of Service
Duration	00:31:05
Attacker	172.16.117.52
Victim	172.016.113.50

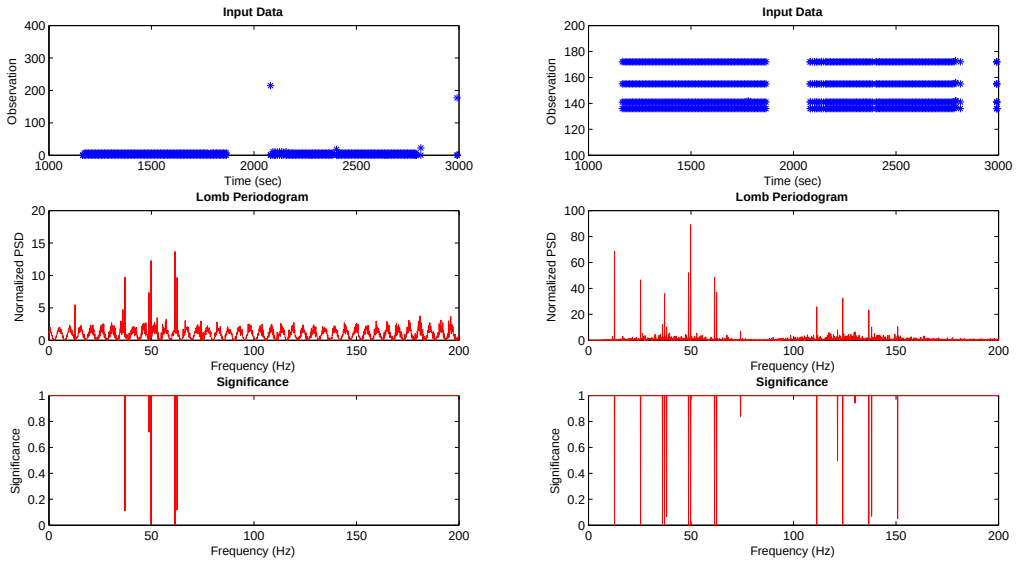
Table C.4: Summary for ProcessTable Attack against 172.16.113.50

Connection	Packets	Timing Alarm	Payload Alarm	Attack
172.16.112.10	30	False	False	False
172.16.112.20	837	False	**True**	False
172.16.112.50	62	False	False	False
172.16.112.100	37	False	False	False
172.16.114.50	55	False	False	False
172.16.117.52	1871	**True**	False	**True**
196.227.033.189	1705	False	False	False



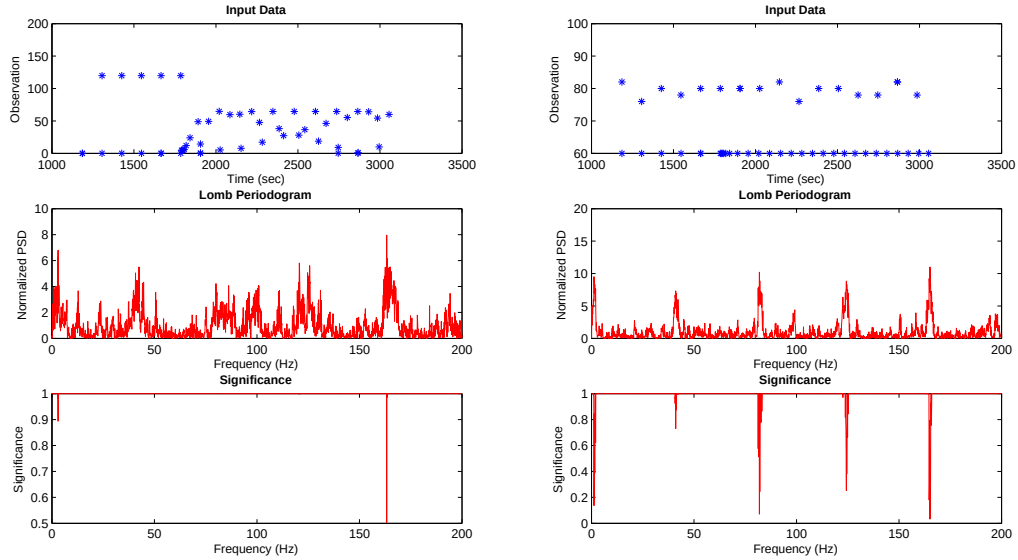
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.11: ProcessTable Attack Source=172.16.112.10



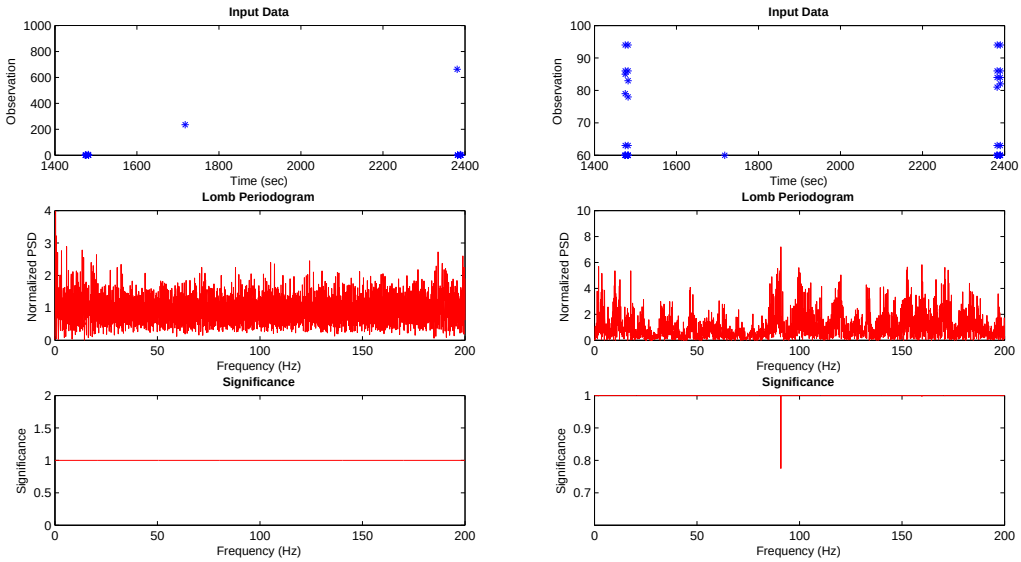
(a) Inter-Arrival: Alarm False (b) Payload: Alarm **True**

Figure C.12: ProcessTable Attack Source=172.16.112.20



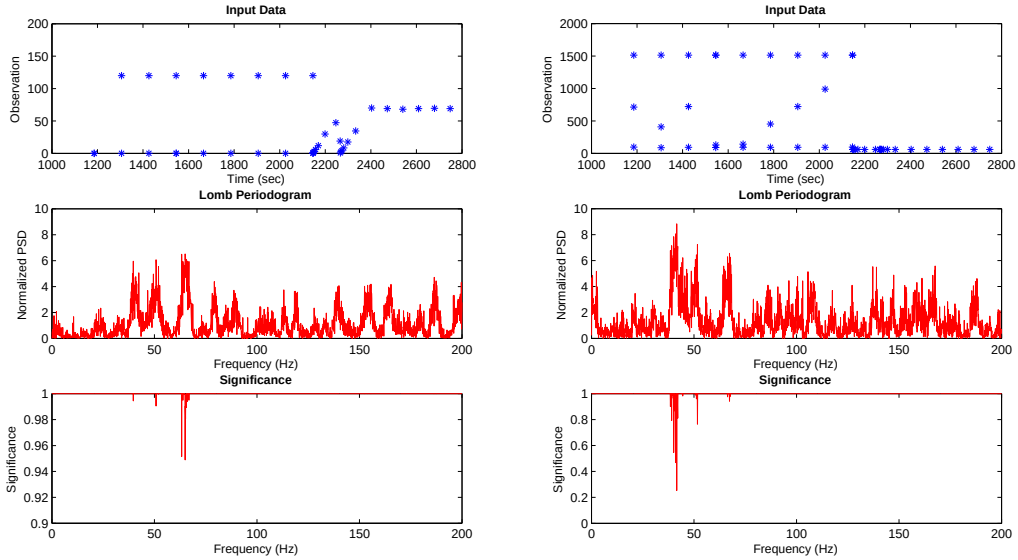
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.13: ProcessTable Attack Source=172.16.112.50



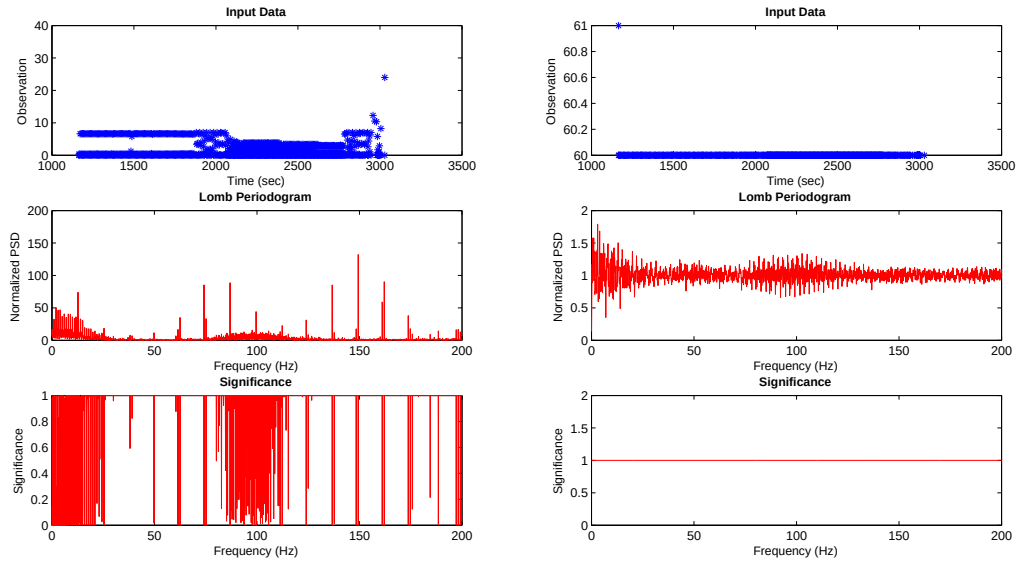
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.14: ProcessTable Attack Source=172.16.112.100



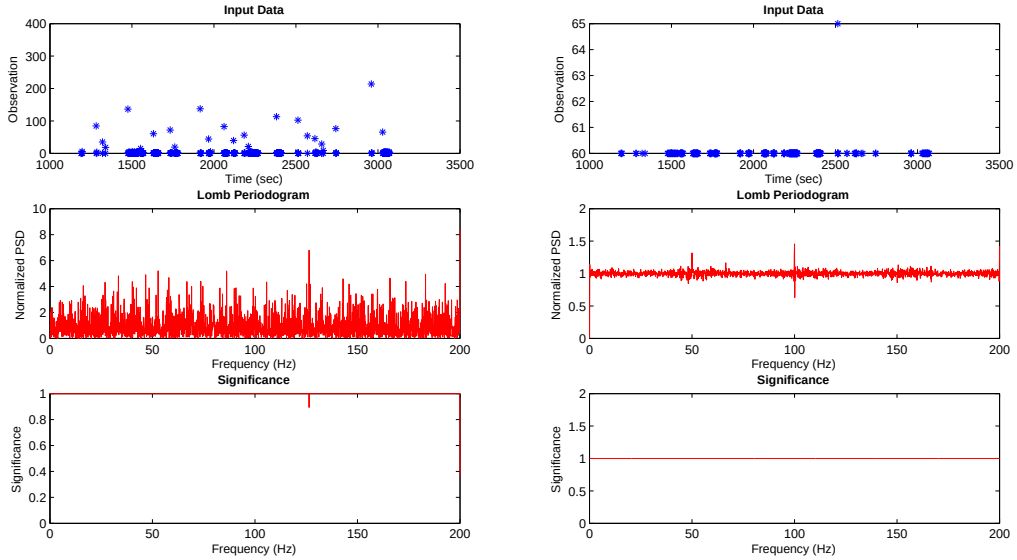
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.15: ProcessTable Attack Source=172.16.114.50



(a) Inter-Arrival: Alarm **True** (b) Payload: Alarm False

Figure C.16: ProcessTable Attack Source=172.16.117.52



(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.17: ProcessTable Attack Source=196.227.33.189

C.2 The Dictionary Attack

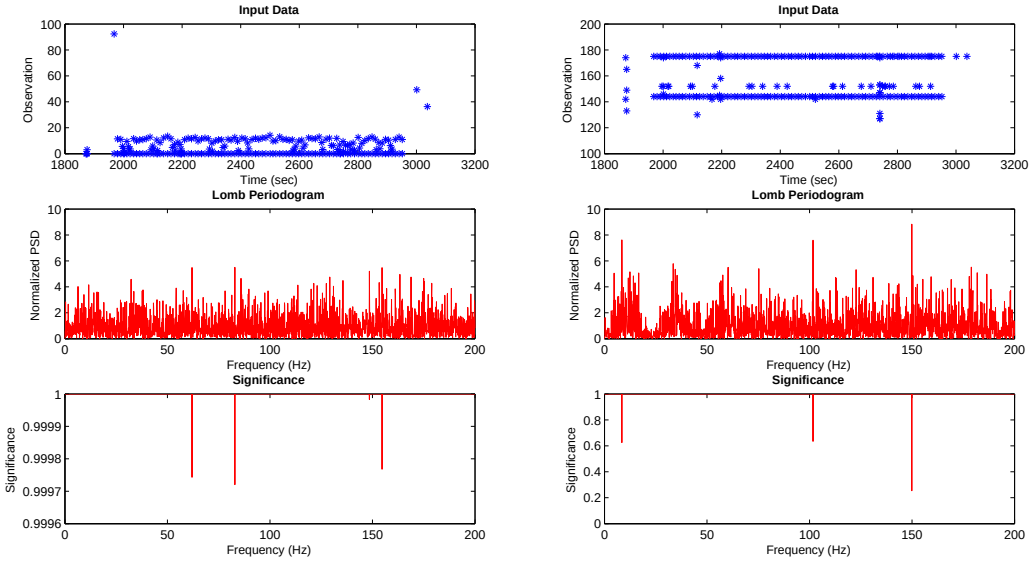
C.2.1 Victim: 172.16.114.50. This section contains the pertinent information describing the Dictionary attack mounted against IP address 172.16.114.50 by IP address 172.16.118.10 on 5 April 1999.

Table C.5: Dictionary Attack against 172.16.114.50

Name	dict
Date	04/05/1999
Category	Root to Local
Duration	00:16:35
Attacker	172.16.118.10
Victim	172.16.114.50

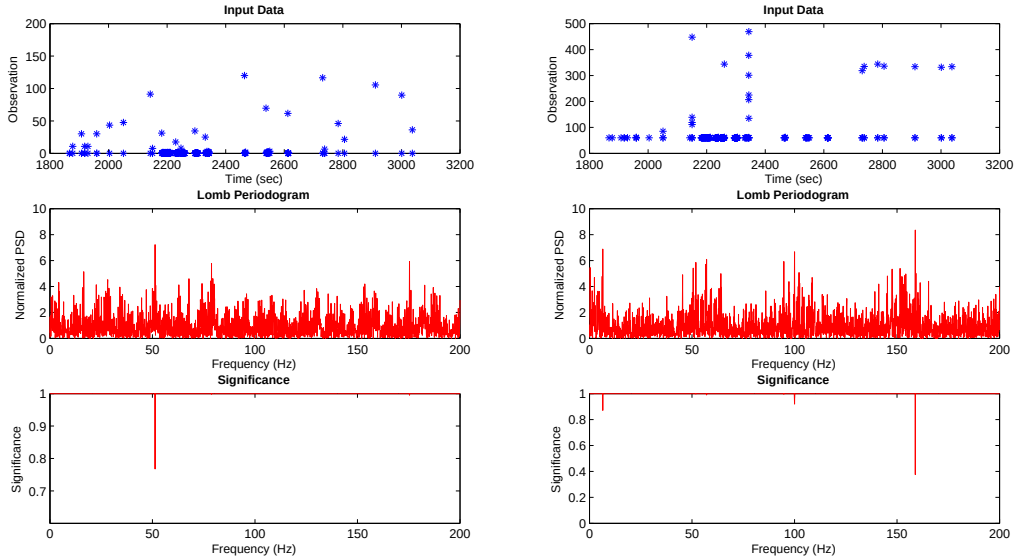
Table C.6: Summary for Dictionary Attack against 172.16.114.50

Connection	Packets	Timing Alarm	Payload Alarm	Attack
172.16.112.20	228	False	False	False
172.16.112.50	764	False	False	False
172.16.113.50	289	False	False	False
172.16.113.105	156	False	False	False
172.16.114.169	128	False	False	False
172.16.118.10	3438	**True**	False	**True**
195.115.218.108	2094	False	**True**	False



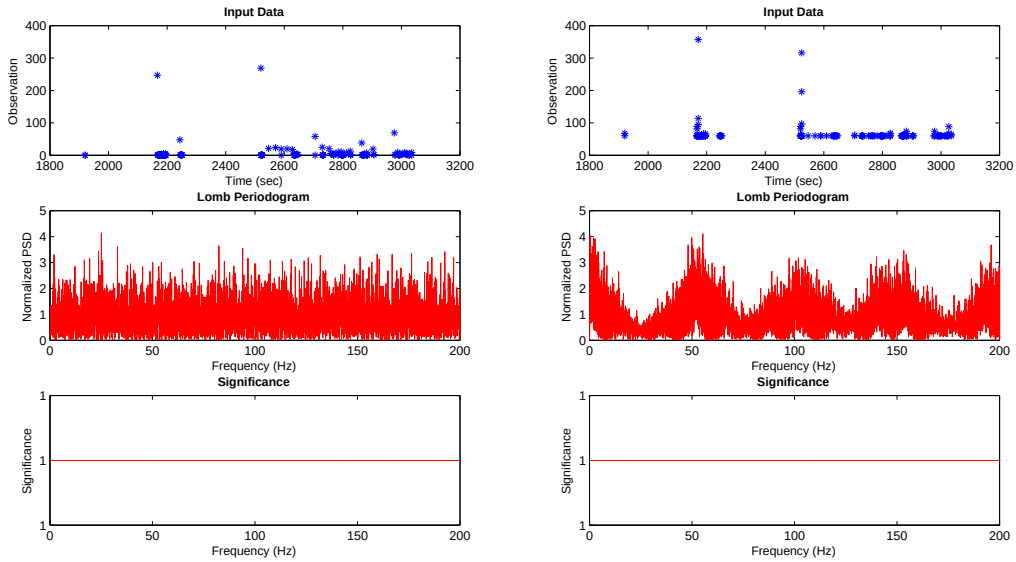
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.18: Dictionary Attack Source=172.16.112.20



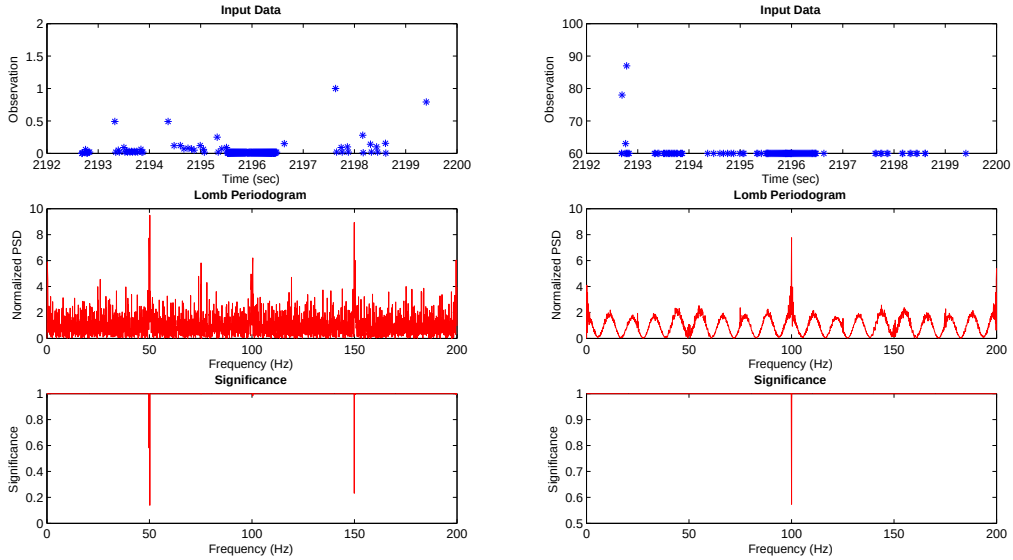
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.19: Dictionary Attack Source=172.16.112.50



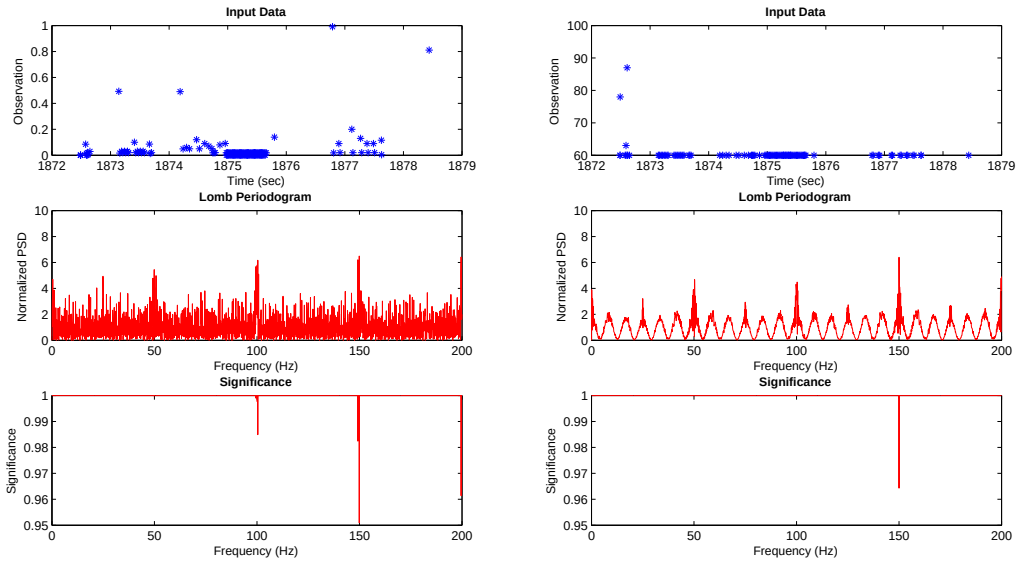
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.20: Dictionary Attack Source=172.16.113.50



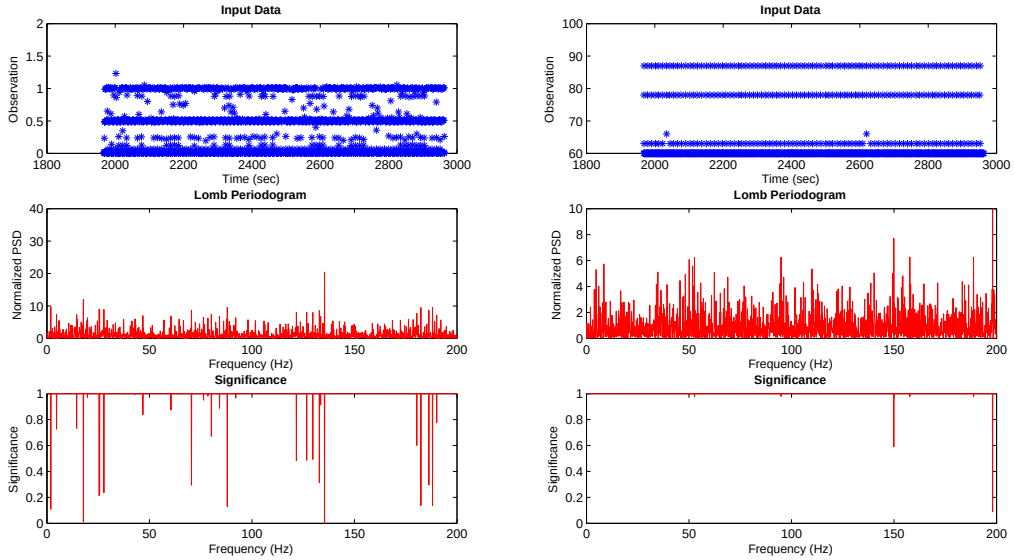
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.21: Dictionary Attack Source=172.16.113.105



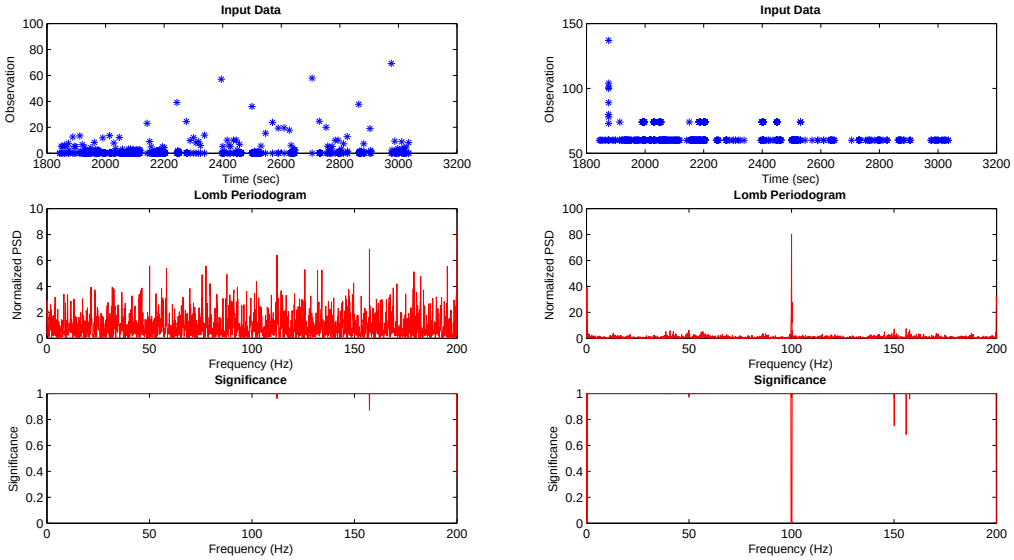
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.22: Dictionary Attack Source=172.16.114.169



(a) Inter-Arrival: Alarm **True** (b) Payload: Alarm False

Figure C.23: Dictionary Attack Source=172.16.118.10



(a) Inter-Arrival: Alarm False (b) Payload: Alarm **True**

Figure C.24: Dictionary Attack Source=195.115.218.108

C.3 The Teardrop Attack

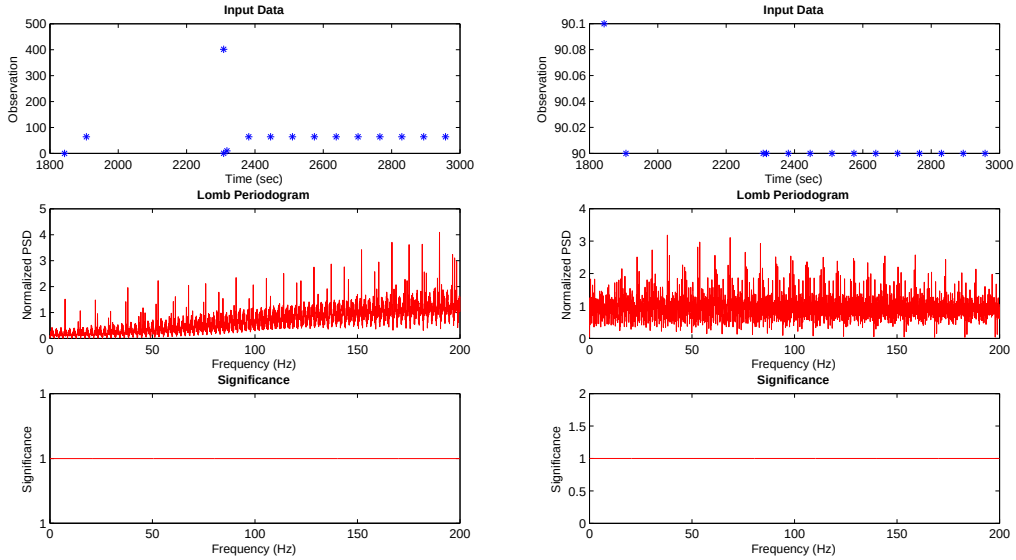
C.3.1 Victim:172.16.114.50. This section contains the pertinent information describing the Teardrop attack mounted against IP address 172.16.114.50 by IP address 207.230.54.203 on 6 April 1999.

Table C.7: Teardrop Attack against 172.16.114.50

Name	teardrop
Date	04/06/1999
Category	Denial of Service
Duration	00:00:01
Attacker	207.230.54.203
Victim	172.16.114.50

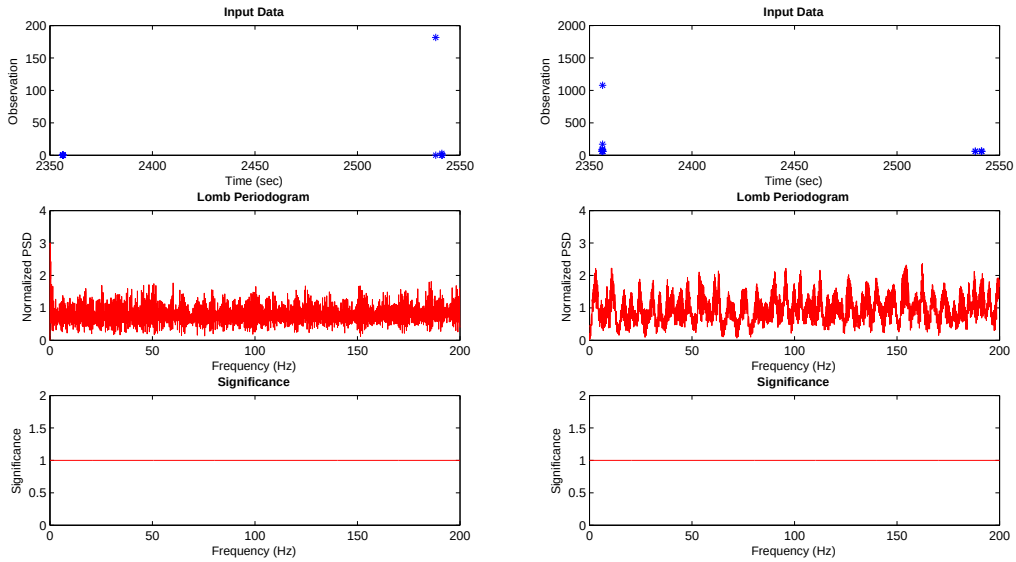
Table C.8: Summary for Teardrop Attack against 172.16.114.50

Connection	Packets	Timing Alarm	Payload Alarm	Attack
172.16.112.10	17	False	False	False
172.16.114.148	24	False	False	False
206.48.44.50	493	False	False	False
207.230.54.203	90	**True**	False	**True**



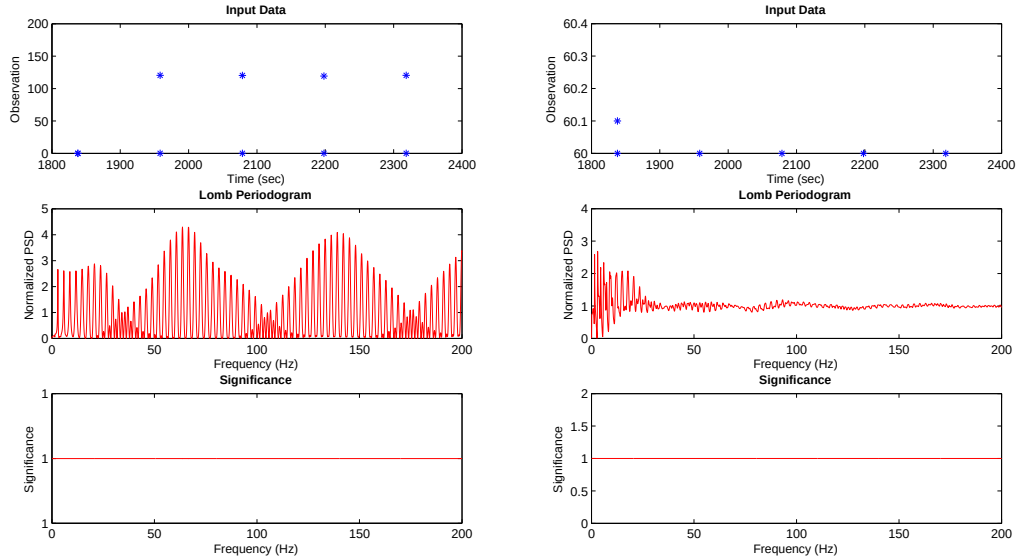
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.25: Teardrop Attack Source=172.16.112.10



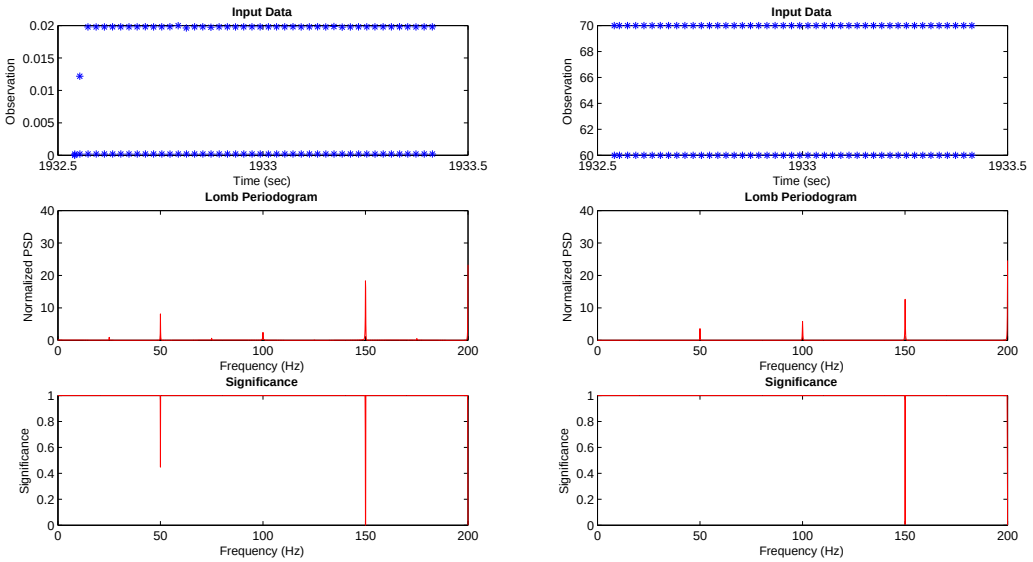
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.26: Teardrop Attack Source=172.16.114.148



(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.27: Teardrop Attack Source=206.48.44.50



(a) Inter-Arrival: Alarm **True** (b) Payload: Alarm False

Figure C.28: Teardrop Attack Source=207.230.54.203

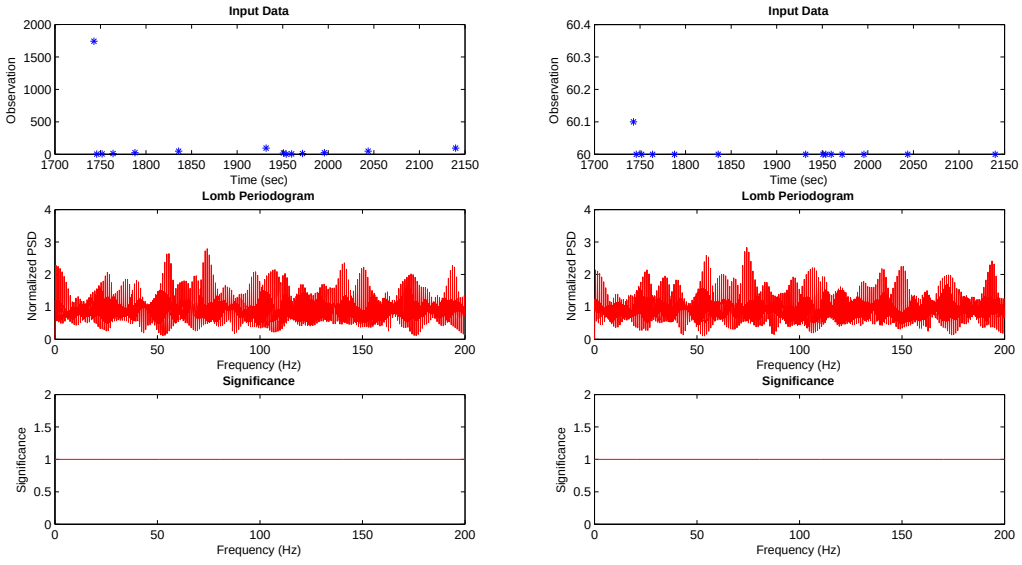
C.3.2 Victim:172.16.114.50. This section contains the pertinent information describing the Teardrop attack mounted against IP address 172.16.114.50 by IP address 199.227.99.125 on 8 April 1999.

Table C.9: Teardrop Attack against 172.16.114.50

Name	teardrop
Date	04/08/1999
Category	Denial of Service
Duration	00:00:01
Attacker	199.227.99.125
Victim	172.16.114.50

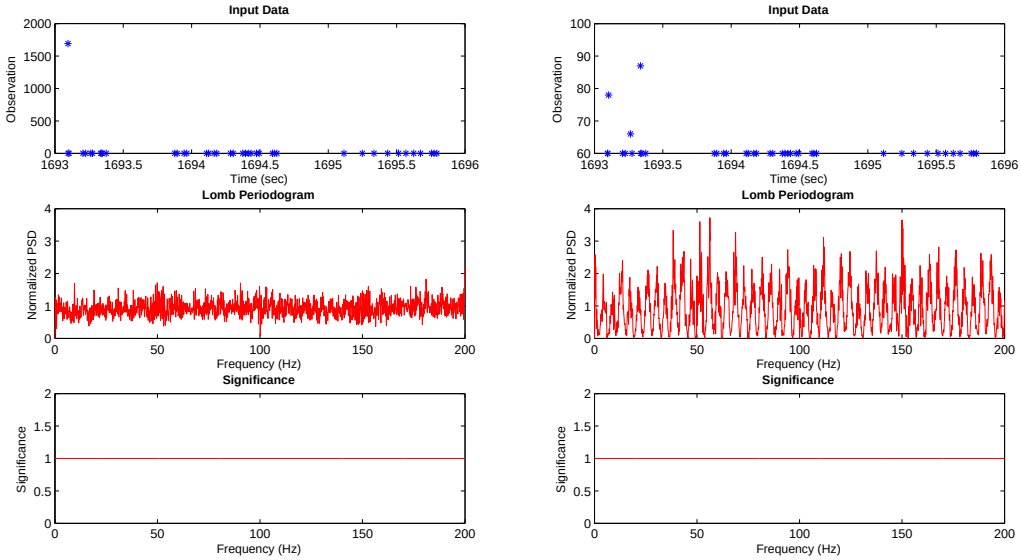
Table C.10: Summary for Teardrop Attack against 172.16.114.50

Connection	Packets	Timing Alarm	Payload Alarm	Attack
135.16.216.191	14	False	False	False
172.16.113.105	41	False	False	False
199.227.99.125	20	False	False	**True**



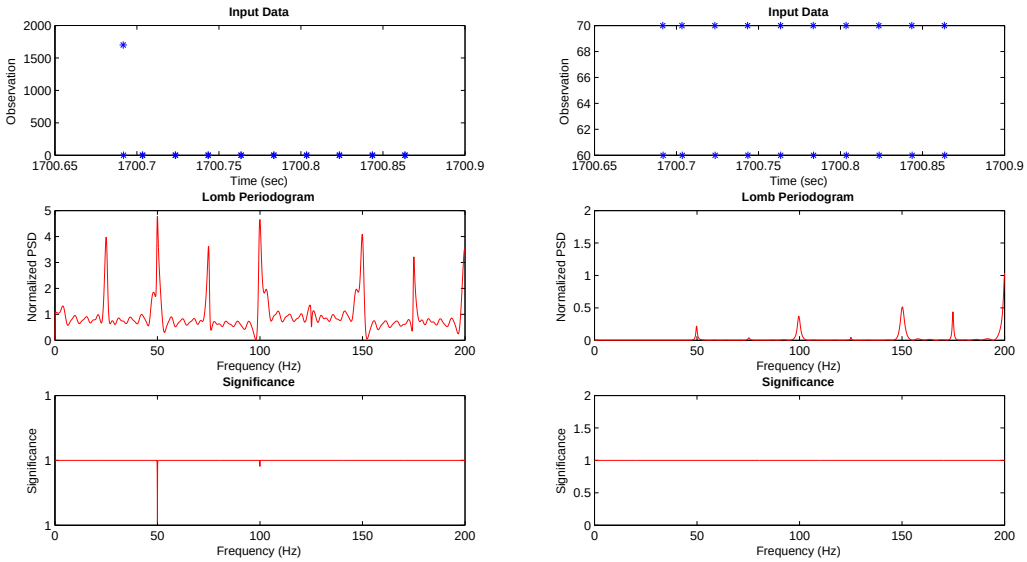
(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.29: Teardrop Attack Source=135.13.216.191



(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.30: Teardrop Attack Source=172.16.113.105



(a) Inter-Arrival: Alarm False (b) Payload: Alarm False

Figure C.31: Teardrop Attack Source=199.227.99.125

Bibliography

[ACF+00] J. Allen, A. Christie, W. Fithen, J. Mchugh, J. Pickel, and E. Stoner. State of the practice of intrusion detection technologies. Technical Report CMU SEI-99-TR-028, Carnegie Mellon University, Software Engineering Institute, 2000.

[Amo98] Edward G. Amoroso. *Intrusion detection : an introduction to Internet surveillance, correlation, traps, trace back, and response*. Intrusion.Net Books, Sparta, N.J., 1998.

[And80] J. P. Anderson. Computer security threat monitoring and surveillance. Technical, Washington, PA, James P. Anderson Co, 1980.

[Axe00a] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Trans.Inf.Syst.Secur.*, 3(3):186–205, 2000.

[Axe00b] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. 2000.

[BP05] Zachary K. Baker and Viktor K. Prasanna. High-throughput linked-pattern matching for intrusion detection systems. In *ANCS '05: Proceedings of the 2005 symposium on Architecture for networking and communications systems*, pages 193–202, New York, NY, USA, 2005. ACM Press.

[CFPCAGN02] D. Cuesta-Frau, J. C. Perez-Cortes, G. Andreu-Garcia, and D. Novak. Feature extraction methods applied to the clustering of electrocardiographic signals. a comparative study, 2002.

[Com08] Gerald Combs. Wireshark: Go deep., 2008.

[CRBM01] D. J. Chaboya, R. A. Raines, R. O. Baldwin, and B. E. Mullins. Network intrusion detection: automated and manual methods prone to attack and evasion. *IEEE Security & Privacy*, 4(6):36–43, 1101.

[Den87] D. E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13(0098):222, 1987.

[Esc98] T. Escamilla. *Intrusion detection : network security beyond the firewall*. John Wiley, New York, 1998.

[KR05] James F. Kurose and Keith W. Ross. *Computer networking : a top-down approach featuring the Internet*. Pearson/Addison Wesley, Boston, 2005.

[Lab99] MIT Lincoln Laboratories. 1999 darpa intrusion detection evaluation data set, 1999.

[LHF+00a] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. The 1999 darpa off-line intrusion detection evaluation. *Computer Networks,*, 34(4):579–595, 10 2000.

[LHF+00b] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. The 1999 darpa off-line intrusion detection evaluation. *Computer Networks,*, 34(4):579–595, 10 2000.

[Lom76] N. R. Lomb. Least-squares frequency analysis of unequally spaced data. *Astrophysics and Space Science*, 39(2):447–462, 1976.

[LPN] A. Lazarevic, D. Pokrajac, and J. Nikolic. Applications of neural networks in network intrusion detection. AN: 9453135.

[MA03] J. Susan Milton and Jesse C. Arnold. *Introduction to probability and statistics : principles and applications for engineering and the computing sciences*. McGraw-Hill, Boston, 2003.

[Mch00] J. Mchugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans.Inf.Syst.Secur.*, 3(4):262–294, 2000.

[PCJ+02] Craig Partridge, David Cousins, Alden W. Jackson, Rajesh Krishnan, Tushar Saxena, and W. Timothy Strayer. Using signal processing to analyze wireless data traffic. In *WiSE '02: Proceedings of the 1st ACM workshop on Wireless security*, pages 67–76, New York, NY, USA, 2002. ACM Press.

[Pos81] J. Postel. Rfc 791, internet protocol: Darpa internet program protocol specification. *Information Sciences Institute, http://www.ietf.org/rfc/rfc0791.txt*, 791, 1981.

[Pre02] William H. Press. *Numerical recipes in C++ : the art of scientific computing*. Cambridge University Press, Cambridge, UK ; New York, 2002. 019: 48930980; William H. Press ... [et al.]; Includes bibliographical references (p. 927-930) and index.

[Ref01] Darpa intrusion detection evaluation, 2001.

[Ref08] The metasploit project, 2008.

[Rob04] Michael J. Roberts. *Signals and systems :analysis using transform methods and MATLAB*. McGrawHill, Dubuque, Iowa, 2004.

[SGF+02] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification-based anomaly detection: a new approach for detecting network intrusions. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 265–274, New York, NY, USA, 2002. ACM Press.

[YLAA00] W. Yi-Leh, D. Agrawal, and A. E. Abbadi. A comparison of dft and dwt based similarity search in time-series databases, 2000.

[ZLC+03] Mian Zhou, Sheau-Dong Lang, N. Callaos, N. Chen, S. Wyanor, and A. Wolf. *A frequency-based approach to intrusion detection*, volume 3 of *SCI 2003. 7th World Multiconference on Systemics, Cybernetics and Informatics Proceedings*, pages 393–397. IIIS; WOSC: World Organization on Systemics and Cybernetics, 2003.

Vita

First Lieutenant Theodore J. Erickson graduated from Bishop Kelly High School in Boise, Idaho. After high school he attended the United States Air Force Academy, graduating with a degree in Computer Engineering in the summer of 2004. His first assignment was with the Space Vehicles directorate of the Air Force Research Lab located at Hanscom Air Force Base, Massachusetts. While stationed at Hanscom he traveled worldwide in support of the Scintillation Network Decision Aid (SCINDA) sensor system. His travels took him to Taiwan, Christmas Island, the Kwajalein Atoll, Cape Verde Africa and Ascension Island.

After his assignment to Hanscom he was selected to attend the Air Force Institute of Technology in pursuit of his Masters in Computer Engineering. Upon his graduation in March of 2008 he will be assigned to Los Angeles Air Force Base in California.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
27–03–2008	Master's Thesis	Sept 2006 — Mar 2008

4. TITLE AND SUBTITLE

Digital Signal Processing
Leveraged for Intrusion Detection

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Theodore J. Erickson, 1st Lt., USAF

5d. PROJECT NUMBER

08-130

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCE/ENG/08-04

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFIOC (James C. Collins)
404 Greig Street, Building 178
IOA Division, ATTN: J Collins / 5-3534
San Antonio, Texas 78226 / 5-3534
(210) 925–2412, james.collins@lackland.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT This thesis describes the development and evaluation of a novel system called the Network Attack Characterization Tool (NACT). The NACT employs digital signal processing to detect network intrusions, by exploiting the Lomb-Scargle periodogram method to obtain a spectrum for sampled network traffic. The Lomb-Scargle method for generating a periodogram allows for the processing of unevenly sampled network data. The spectrum is examined to determine if features exist above a significance level chosen by the user. These features are considered an attack, triggering an alarm. Two traffic statistics are used to construct the time series over which the periodogram analysis is accomplished. These two statistics are packet inter-arrival time and payload size. Three specific attacks from this data set are examined; the Processtable attack, the Dictionary attack and the Teardrop attack. Of the three attacks the NACT was able to detect the Processtable attack with an accuracy of 100%. The Dictionary and Teardrop attacks were also detected with 100% and 85% accuracies respectively. This success in detecting these attacks establishes that digital signal processing methods can be a successful technique for network intrusion detection.

15. SUBJECT TERMS

network intrusion detection, digital signal processing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Barry Mullins
U	U	U	UU	106	19b. TELEPHONE NUMBER *(include area code)* (937) 255–3636, ext 7979; barry.mullins@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18