Combining Statistical and Spectral Analysis Techniques in Network Traffic Anomaly Detection

by

Stevan Novakov, B. Eng. (CSE)

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada K1S 5B6

The undersigned recommend to
the Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Combining Statistical and Spectral Analysis Techniques in Network Traffic Anomaly Detection

submitted by

Stevan Novakov, B. Eng. (CSE)

in partial fulfillment of the requirements for

the degree of Master of Applied Science in Electrical and Computer Engineering

Chair, Howard Schwartz, Department of Systems and Computer Engineering

Thesis Co-Supervisor, Chung-Horng Lung

Thesis Co-Supervisor, Ioannis Lambadaris

Carleton University
April 2012

Abstract

Recent computer attacks target networks and there is a need to detect these network anomalies quickly and efficiently. This area has been widely studied and solutions typically use data not freely available. A labeled available dataset, Kyoto2006+, has been recently created. Most existing works using Kyoto2006+ apply various clustering approaches. Our research leverages existing spectral analysis and statistical analysis techniques.

One well known spectral analysis technique is Haar Wavelet filtering analysis. It measures the amount and magnitude of abrupt changes in data. Another popular approach is a statistical analysis technique, Principle Component Analysis (PCA). PCA describes data in a new dimension to better express the data. A modified PCA which incorporates time shifting to account for changes over time is considered. Both approaches have strengths and limitations. In response, this thesis proposes Hybrid PCA – Haar Wavelet Analysis. This approach uses PCA to describe the data and Haar Wavelet filtering for analysis.

Based on prototyping and measurement, an investigation of the Hybrid PCA – Haar Wavelet Analysis technique is performed. Experimental results are presented to demonstrate the accuracy and precision of the combined approach compared to the two algorithms individually. Furthermore, tests to examine the impact of algorithm parameters are discussed.

Acknowledgements

I would like to express my sincere thanks to my thesis supervisors, Professor Chung-Horng Lung and Professor Ioannis Lambadaris, for all their guidance, and support throughout the completion of this thesis. Furthermore, I would like to thank the entire SCE office staff and SCE technical support for all their help. In addition, I would like to thank Nic Falvo and the Graduate Student Association for all their support. I am grateful to Carleton University, Ontario Center for Excellence (O.C.E.) and the Government of Ontario for providing financial support for this research. I would also like to acknowledge Solana Networks for their financial support and collaboration in this research.

Last but not least, I would like to thank a handful of very special people who supported me these last several semesters. Without their kind words or few moments of care, I could not have accomplished my work: Mike Floyd, Angad Gakhar, James Green, Norman Lim, Andre Loiselle, Allana Krolak-Lee, Hannan Mahmoud, Ziyad Muslat, Quynh-Chi Nguyen, Simon Skowronski, Mario Valiente, Patrick Visinski, Tina Woods and my most of all, my mother, Anna Buraczewski. I am blessed to have such strong and good people in my life.

Table of Contents

List of Tables

Table of Figures

List of Equations

1.0 Introduction

The way networks are being used is rapidly changing and a by-product of this change is the amount of network traffic volume is rapidly increasing. Similarly, the types of computer attacks are rapidly evolving. For example, malicious attacks are no longer limited to desktop computer viruses, but can target a network itself. These attacks are designed to create failures in the system. A failure brings upon anomalous network behaviour and can be categorized into two forms, soft failures and hard failures. A soft failure can be thought of as a performance degradation; whereas, a hard failure would be elements ceasing to operate [1]. Depending on the network, these failures can cause mild inconveniences, loss of productivity, loss of economic activity, or even, loss of public well being. This research applies statistical analysis and spectral analysis techniques to network traffic data in order to identify potential malicious network attacks. Specifically, this thesis focuses on a hybrid technique to provide information to a network operator such that the source of malicious behaviour can be isolated.

1.1 Motivation of Thesis

There is a need for effective and scalable approaches to maintain network stability and to detect anomalous network traffic behaviour created by attacks. The state of the art to manage massive amounts of data transmission is Cisco's NetFlow protocol [2]. This protocol resides on routers and each packet that passes through is examined for a set of IP packet attributes. The output of NetFlow is a multi-tuple record, called a flow. Some core features of a flow are: Source IP address, Destination IP address, total bytes, etc (See Figure 2.1 for an illustration of NetFlow collection). NetFlow does not provide any information if the flow is a part of abnormal or malicious behaviour [3].

Current intrusion detection systems (IDS), such as Bro, use existing knowledge base and identify attacks through patterns for signatures [4]. As a result of the high volume of traffic, inspecting individual signatures or traces of known hazards is time consuming and inefficient. Furthermore, the turnaround from discovery to updating the knowledge base can be extensive. Another approach, called anomaly detection, attempts to detect patterns which do not conform to expected behaviour [5]. Anomaly detection does not have to be dependent on an existing knowledge base. Some statistical models and signal processing algorithms do not require an existing knowledge base. These methods can be applied relatively quickly to create relationships and discover patterns from a range of data types and sizes. The types of computer attacks rapidly change, and hackers are constantly coming up with new techniques. As a result, no matter how refined the detection method becomes; a comprehensive IDS will require a significant amount of human expertise [5].

1.2 Proposed Solution

This thesis proposes a hybrid solution, based on statistical and spectral analysis techniques, which provide the network administrator time slices of potential network traffic intrusions. To the best of our knowledge, no such hybrid techniques has deployed by systems described in existing literature.

The statistical analysis studied is a Modified Principal Component Analysis (PCA) technique to determine abnormal behaviour [6]. Components are extracted by comparing feature data similarity. For example, consider a two-dimensional data set, a line of best fit can be considered as a component (See Figure 2.4). A ranked subset of components, selected by comparing sparsity of projected data, is used to create a subspace that describes anomalous behaviour. Time grouped data is projected onto this space and spectral analysis is applied. The Modified PCA algorithm is described in more detail in section 3.1.1. The feature with the most spread out data is considered in spectral analysis portion.

The spectral analysis technique adopted is Haar Wavelet decomposition [7]. This type of wavelet decomposition uses a Haar basis function to decompose the input dataset set into core time functions. Thresholds are applied to remove noise and highlight network traffic anomaly characteristics. The signal is reconstructed and a weighted score to describe the magnitude of fluctuations within each time slice is calculated. The Haar wavelet algorithm is described in more detail in section 3.1.2. A high score represents a large change in a time window and suggests to the network administrator that abnormal and potentially malicious behaviour is present.

This research uses a hybrid technique and illustrates how much more accurate and informative it is compared to the statistical and spectral analysis techniques independently. This hybrid approach examines network traffic data grouped into time bins and summarized. Next, it applies statistical analysis to reduce the complex nature of the network traffic. It then applies spectral analysis to determine time slices of interest where there is a change in behaviour. Network traffic data can be grouped by start time into bins and features describing each time bin behaviour are extracted (See Table 3.1.2 and Table 3.1.3 for a complete outline of features). The time binning approach is described in more detail in section 3.1.

To evaluate the performance of the algorithms, a prototype test bed is used. Furthermore, a novel labeled dataset, called Kyoto2006+, was used in analysis [8]. The design and implementation of the test bed is discussed in chapter 4 and an outline of the dataset is presented in Section 2.2.1.

1.3 Scope of the Thesis

This thesis focuses on the hybrid approach which is based on a modified PCA approach and a Haar wavelet approach. As discussed in Section 1.2, network traffic data is grouped into time bins, and descriptive features are derived. Then a hybrid analysis approach is applied is obtain a measure of potential anomalies for each time slice. The algorithm can provide a time slice, and features of interest as breadcrumbs, to lead to the offending source. As no specific host is highlighted, manual investigation of the network traffic data is required. A subsequent technique applied on the time slice to isolate the core source of the anomalous behaviour is beyond the scope of this thesis and can form a direction for future research. Further discussion is provided in Section 7.2.

1.4 Contribution of the Thesis

The main contributions of the thesis are presented next.

- A hybrid analysis: a novel technique which uses a statistical analysis technique, Principle Component Analysis, to reduce the complexity of the time binned dataset and produces an anomalous subspace. Next, a spectral analysis technique, Haar wavelet decomposition, is applied to highlight network traffic anomalies and to produce a measure of potential malicious behaviour.
- Independent applications of statistical analysis and spectral analysis applied on Kyoto2006+ dataset. The Kyoto2006+ dataset is a network traffic dataset which provides intrusion information and anti-virus alerts from existing open source detectors. To the best of our knowledge, none of the works using the Kyoto2006+ dataset have focused on Haar wavelet decomposition or Principle Component Analysis.
- Investigation of several algorithms for performance measurement and studying the impact in order to automatically indicate a set time slices which may contain potentially malicious behaviour.

- A prototype test bed is built and the performance of each of the algorithms is evaluated through various measurements made by the system. A number of insights into the algorithm behaviour and performance are obtained from the experimental results that include the following:
    - A demonstration of the effectiveness of each of the adopted approaches for identifying network anomalies is presented. The performance improvement of the hybrid approach compared to only the statistical approach and only the spectral approach is discussed.
    - The effect of different algorithm parameter values on the performance of the approaches is presented.

1.5 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 provides background information on network traffic and anomaly detection, and discusses related works. Chapter 3 outlines the statistical analysis algorithm, the spectral analysis algorithm and the hybrid algorithm in more detail. Chapter 4 describes the test bed prototype design and operation. Chapter 5 outlines each step of the experimental approach in detail. Chapter 6 presents and discusses the results of the performance analysis on the hybrid approach, on only the statistical approach and on only the spectral approach. Finally, Chapter 7 concludes the thesis and discusses possible directions for future research.

2.0 Background and Related Works

This chapter presents background information on network traffic. It then describes spectral and statistical analysis techniques in intrusion detection. In addition, it provides a background about the algorithms used in this research. Lastly, the chapter discusses related works.

2.1 Intrusion Detection in Network communication

Network traffic intrusion detection is identification of observed activities that do not conform to an established norm in computer communication [9]. There exist two main types of intrusion detection methods: misuse detection and anomaly detection. Misuse detection is based on an existing knowledge of known attacks and identifies attacks through patterns or signatures [9]. As a result of the high volume of traffic, inspecting individual packets for signatures or traces of known hazards is time consuming and inefficient. Anomaly detection is not dependent on an existing knowledge base and identifies potential network threats by finding deviations from normal behaviour.

Spectral analysis, also known as frequency analysis, looks at time-varying signals to reveal special characteristic [10]. Wavelet analysis is one form of spectral analysis and divides a complicated function into several simple ones. The term wavelet comes from the characteristic of converging to zero, and is illustrated by waving above and below the horizontal axis [11]. It represents a signal by a finite sum of components of different resolutions. Individual components can be examined for anomalies [12]. Statistical analysis deals with the discrimination, classification and interpretation of collected datasets [13]. In this research, principle component analysis is considered. PCA aims to represent data in a lower dimensional space through a transformation of the original features. The new variables or features are uncorrelated orthogonal linear combinations of the original dataset. They are a ranked in order of significance and describe trends in data with greater details [13]. Anomaly detection techniques are applied to these components. These various anomaly detection techniques can be applied relatively quickly to create relationships between observed variables over a range of data types and determine deviations.

2.2 Network Traffic Communication

Traditional computer network communication use a specific protocol to route data from an origin to a destination. A packet is a unit of data which contains user application information and control information used to properly deliver the data. A network protocol is a set of standards used to format communication between source and destinations [14]. A widely used protocol is Transmission Control Protocol (TCP). A session or flow is a set of packets sent from the same source and source port to a specific destination and destination port with the same network protocol. The network traffic software collects the data for the flow until the flow is terminated. It can be terminated with either a timeout or a finish flag, FIN. [2]

A flow can be described by a number of characteristics or features and are measured by software on network routers or network probes [3]. These probes examine the control information of data packets. An example is Cisco's NetFlow, an industry standard protocol, which describes a flow through seven features: source IP address, destination IP address, source port, destination port, protocol type, Class of Service, and Router or switch interface [2]. The observation software is loaded on Cisco routers (See Figure 2.1).

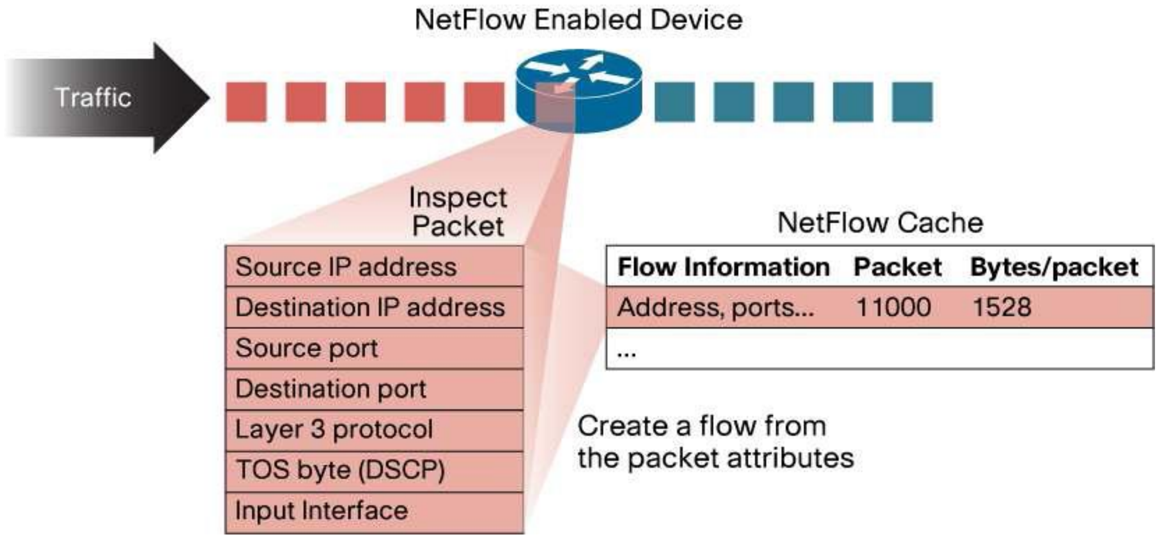


Figure 2.1: Creating a flow in the NetFlow cache [2]

2.2.1 Network Traffic Data Sources

A number of the data sources in anomaly detection research are from large research networks. An example of this is the Geant research network based in Europe [15]. This network connects over 30 European countries and connects national and education computer networks [15]. Although there are many research networks which create datasets, it has become extremely difficult to obtain data due to privacy concerns and competitive issues.

In 1999, a competition was held at the Knowledge Discovery and Data Mining annual conference to build a network intrusion detection system that would detect anomalies. Network traffic data was made up of nine weeks of raw TCP dump data. The data simulated network traffic on a typical U.S. Air Force LAN and takes about 743 MB of memory. The nine weeks were broken up into seven weeks of training, and two weeks of testing data. Unlike many research networks, which only provide flow data or sanitized raw TCP data, each observation contains anomaly labels and descriptions. This data source has been widely used in research. While it is very commonly used in algorithm evaluation; there are two major drawbacks. The data is more than ten years old and does not reflect current threats [8]. Even though the there are 41 features describing a flow, the most common descriptors, such as source port, are omitted. This is a major drawback as many popular approaches, like in [16] and [6], use various entropy measurements for their analysis. [17]

In response to this major drawback, Song et al. [8] created a new evaluation dataset called Kyoto2006+. This novel dataset was used in this research. It contains over three years of real traffic data and described by 24 characteristics or features. The primary 14 statistical features are derived from the KDD'99 dataset. The authors analyzed the features of this dataset and kept only those they believed relevant. Then the authors added six features of connection information, which was previously omitted. An example is source address, source port, destination address, etc. The remaining four features provide detector information that, to the best of our knowledge, no other research dataset provides. The authors used open source tools; an intrusion detection software tool called SNORT, an anti virus program called ClamAV and the Ashula detection software to detect exploit code. Along with the output information from these programs, the authors provide a "label" feature is a cross correlation between their research findings and that of the open source tools. (See Table 2.1 for an outline of the features). [8]

Feature Name	Description
Duration	The length (seconds) of the connection.
Service	The connection's service type, e.g., http, telnet.
Source bytes	The number of data bytes sent by the source IP address.
Destination bytes	The number of data bytes sent by the destination IP address
Count	The number of connections whose source IP address and destination IP address are the same to those of the current connection in the past two seconds.
Same srv rate	Percentage of connections to the same service in Count feature
Serror rate	Percentage of connections that have "SYN" errors in Count feature
Srv serror rate	Percentage of connections that have "SYN" errors in Srv count(the number of connections whose service type is the same to that of the current connection in the past two seconds) feature

Dst host count	Among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose source IP address is also the same to that of the current connection.
Dst host srv count	Among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose service type is also the same to that of the current connection
Dst host same src port rate	Percentage of connections whose source port is the same to that of the current connection in Dst host count feature
Dst host serror rate	Percentage of connections that have “SYN” errors in Dst host count feature
Dst host srv serror rate	Percentage of connections that “SYN” errors in Dst host srv count feature
Flag	The state of the connection at the time the connection was written.
IDS detection	Reflects if IDS triggered an alert for the connection; ‘0’ means any alerts were not triggered, and an arabic numeral means the different kinds of the alerts. Parenthesis indicates the number of the same alert.
Malware detection	Indicates if malware, also known as malicious software, was observed in the connection; ‘0’ means no malware was observed, and a string indicates the corresponding malware observed at the connection. Parenthesis indicates the number of the same malware.
Ashula detection	Means if shellcodes and exploit codes were used in the connection; ‘0’ means no shellcode nor 35 exploit code was observed, and an arabic numeral means the different kinds of the shellcodes or exploit codes. Parenthesis indicates the number of the same shellcode or exploit code.
Label	Indicates whether the session was attack or not; ‘1’ means the session was normal, ‘-1’ means known attack was observed in the session, and ‘-2’ means unknown attack was observed in the session.
Source IP Address	Means the source IP address used in the session. The original IP address on IPv4 was sanitized.
Source Port Number	Indicates the source port number used in the session.
Destination IP Address	It was also sanitized.
Destination Port Number	Indicates the destination port number used in the session.
Start Time	Indicates when the session was started.
Duration	Indicates how long the session was being established.

Table 2.1: Kyoto2006+ dataset features [8]

2.2.2 Network Traffic Volume

The volume of network traffic data is very high and there is need for efficient and scalable solutions for network anomaly detection. Considering traffic data alone can be excessive, and overly granular. Depending on the data source, raw traffic data can consist of millions of observations. One approach to limit data size, which would not add any extra overhead, is to create smaller summaries or profiles. A profile consists of composite features, such as summations, or statistical features, such as entropy. The most common approach, as seen in [16] and [6], is to create a number of time slices and assigning the flows, by start time, to each time slice. Then, a set of features describe each of these time bins. For example, in [6], the TCP traffic in a time slice was summarized by seven features: total bytes, total packets, total number of flows, destination address entropy, unique number of hosts, and unique number of destinations. Alternatively, Wanner [18] modeled each observation around a node. Each unique source node in the dataset was described through 41 composite features. These features consisted of total bytes received by the host, total bytes sent by the host, total number of unique ports, etc. Both approaches produce an observation table much smaller than the primary flow data set. In this research, time bin profiling is used. The number of observations comes down from millions of flows to a few thousand hosts, or a few hundred time bins.

2.3 Wavelet Analysis

Recently wavelets have become popular due to the effectiveness of representing time varying signals in a compact fashion. Unlike Fourier transforms, which use the sine and cosine function as bases, wavelets use signals localized in time and frequency [10]. Wavelet decomposition provides accurate resolution both the time and frequency domain [19]. Wavelets can represent a signal by a finite sum of component functions [12]. The components can be examined for anomalies (See Figure 2.2 for block diagram of wavelet data analysis). A natural extension of collecting metrics is to organize the observations by time. The input time signal is first decomposed using wavelet decomposition into several orthogonal components. Next, threshold techniques are applied. According to a set of parameters and depending on the type of analysis, various values are filtered out or set to zero. Lastly, the signal is reconstructed to produce a filtered time signal, where parameter dependant behaviour is illustrated. [11]

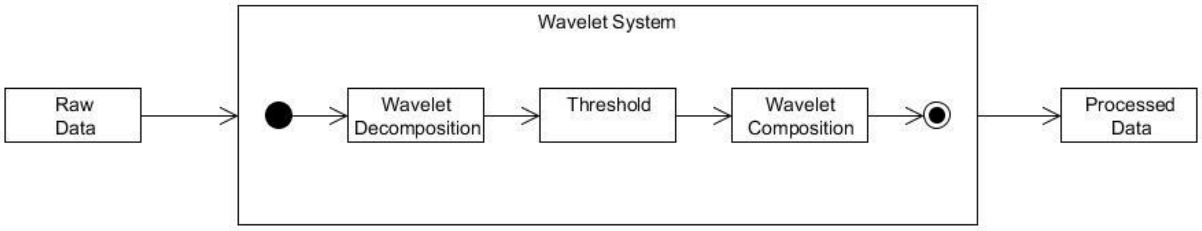


Figure 2.2: Data analysis by Wavelets [11]

There are different types of wavelet approaches, and each has their own basis function and decomposition approaches. The simplest wavelet analysis, used in this research, is the Haar wavelet analysis [10]. The Haar wavelet basis function, $\psi(x)$ (See Equation 2.1), and the scaling function, $\phi(x)$ (See Equation 2.2), are used to create a piecewise decomposition of the input time signal. The decomposition function is a sum of the scaling function and basis functions. (See Equation 2.3 for the decomposition function). The scaling function is used to make the decomposition discrete, such that an infinite amount of pieces are not needed to represent the input signal. In the decomposition, versions of the Haar function, denoted $\psi_{jk}(x)$ (See Equation 2.4), are translated and dilated to represent different frequency resolutions, specified by *j* and *k*. These components are orthonormal which means that they are independent, non-overlapping and analysis produces distinct results. As shown in [11], in order satisfy that the frequency components are orthonormal, the *const* from equation 2.4 must equal $2^{j/2}$.

$$\psi(x) = \begin{cases} 1 & 0 \leq x < 1/2 \\ -1 & 1/2 \leq x < 1 \end{cases}$$

Equation 2.1: Haar Wavelet basis function [11]

$$\boldsymbol{\phi(x) = 1 \;\; 0 \leq x \leq 1}$$

Equation 2.2: Haar Wavelet scaling function [11]

$$f(x) = C_{00}(x) + \sum_{j=0}^{n-1} \sum_{k=0}^{2^j-1} D_{jk}\psi_{jk}(x)$$

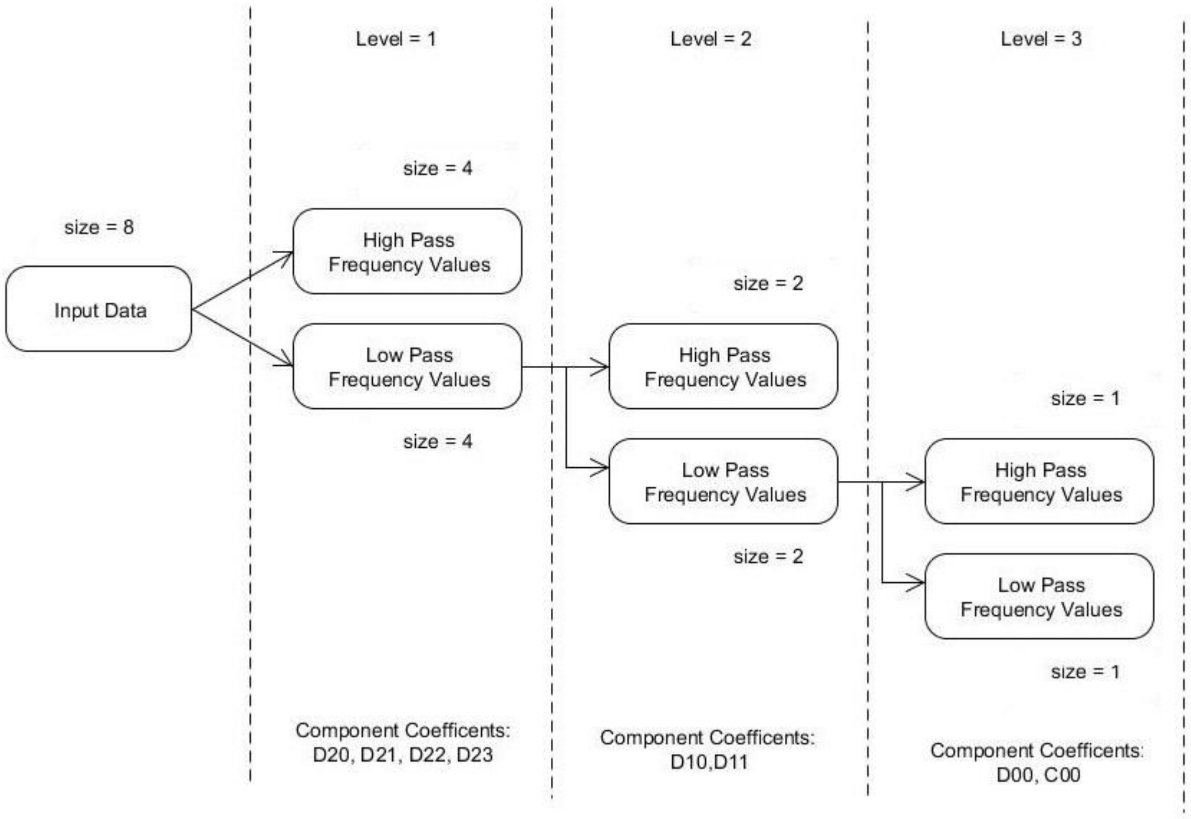
Equation 2.3: Haar Wavelet decomposition function [11]

$$\psi_{jk}(x) = const * \psi\left(2^j x - k\right)$$

Equation 2.4: Haar Wavelet frequency adjusted basis function [11]

The input signal size is directly proportional to the complexity of the decomposition function. The larger the input signal, the more coefficients are produced. Usually one can assign a frequency range to each component. Each coefficient scales a specific component. Haar wavelet filtering applies threshold techniques to these coefficients. Depending on the type of filter, a number of coefficients are assigned to zero. Typical filtering approaches include high pass frequency filters, low pass frequency filters and medium frequency filters. The user determines the filtering by providing a specific frequency range. Filtering occurs through the coefficients outside this range being set to zero. After filtering is applied, the signal is then reconstructed by reversing the decomposition approach. The result is a filtered version of the original signal.

The Haar Wavelet Filtering approach bisects the input data between higher and lower frequency values given a certain input. The lower pass frequency values are then fed back into the system until a defined limit is reached. A visualization of Equation 2.3 can be seen in Figure 2.3. Each iteration is referred to as a level and has its own set of components which correspond to a certain frequency range. It is possible to describe filters or frequency ranges by the number of levels applied. For example, in Figure 2.3, a dataset of eight observations is filtered. The four high pass values can be considered to be in level one, the two mid pass values can be considered in level two and the two remaining are low pass in level three. In order to apply filtering, the component coefficients for a specific level are set to zero. For example, in Figure 2.3, a high pass filter would set the coefficients at levels two and three to zero, and then reconstruct the signal.



Level = 1
Level = 2
Level = 3
size = 8
Input Data
size = 4
High Pass Frequency Values
Low Pass Frequency Values
size = 4
size = 2
High Pass Frequency Values
Low Pass Frequency Values
size = 2
size = 1
High Pass Frequency Values
Low Pass Frequency Values
size = 1
Component Coefficents: D20, D21, D22, D23
Component Coefficents: D10,D11
Component Coefficents: D00, C00

Figure 2.3: Haar Wavelet decomposition

2.4 Principle Component Analysis (PCA)

Principle Component Analysis derives new variables or features that are linear combinations of the original features and are uncorrelated [13]. It is a means of finding a reduced set of features which describe the dataset by examining the relationships between features. PCA is a transformation that can be considered as a rotation of the original axes to a new set of orthogonal axes. It can be used as feature selection, as it determines a new set ranked component. For example, in a two-dimensional dataset, a line of best fit on a Cartesian Graph would be considered a principle component (See Figure 2.4 for an illustration) [20].

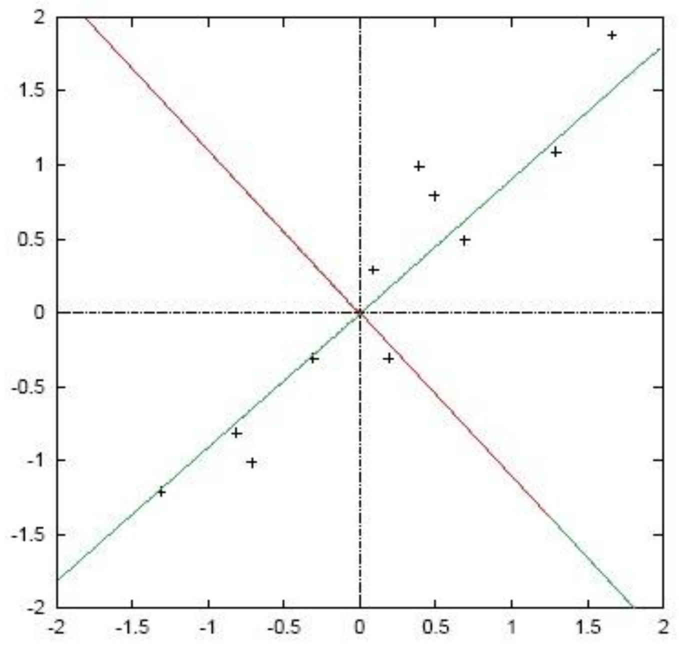


Figure 2.4: Eigenvalue Components [20]

There are a number of methods in deriving the set of principle components, ξ. In this research, eigenvalue decomposition approach is used. Typically, the new components will not have a zero mean. Therefore, the data must be standardized in order to create a zero mean. This requires that each original features set are shifted to have a zero mean. The principle components can be obtained by using Equation 2.5, where A is a matrix of observation coefficients, x is a matrix of random vectors, $x_1..x_p$, or features, and u the sample mean.

$$\xi = A^T(x - \mu)$$

Equation 2.5: Principle Component Formula [16]

As shown in [20], using eigenvector decomposition to determine the principle components, ξ, requires calculating the covariance matrix of (x – μ), as illustrated in equation 2.6. A covariance matrix is produced to illustrate the variance, or measure the spread of data, between every component combination (See Equation 2.6). The eigenvectors of Σ are the principle components. The nature of eigenvector decomposition produces the eigenvector and eigenvalues to be ordered from greatest to least. The eigenvalues are the measure of variance when the dataset get projected onto its respective eigenvector [21]. As a result, the list of eigenvectors is a ranked from most to least significant. These eigenvectors are the set of axis that characterize the data. Considering the previous example of two-dimensional data, an eigenvector would be the actual line of best fit.

$$Covariance\ Matrix = \frac{1}{n - N - 1} x^T x$$

Equation 2.6: Matrix of covariance estimates [16]

There are many components determined from decomposition. The number of components can equal the number of initial features. The components are ranked from most to least significant. The principle components are based on the type of selection criteria applied. The most basic selection criterion is 'Top K' components. The first *K* many elements are considered as the principle components. Another approach is based on the distribution of the data projected onto each of the components. Variance is the square root of standard deviation, and a standard deviation of one suggests that the dataset is normally distributed. This approach considers any eigenvector a principle component if its corresponding eigenvalue is greater than one. Lastly, an approach considered is to build a space which covers 99% of the data. The space starts with the most significant principle component. One by one, components are added to the space until the sum of the variances in the space is equal to or greater than 99% of the sum of all the variances. The type of selection criteria used directly affects the number of principle components extracted.

2.5 Comparison with Related Works

Intrusions Detection Systems are widely used in practice, as well as a subject of research. They are seen as one of the most promising countermeasures to defend computer systems and networks against attackers [22]. Existing systems detect intrusions by examining and comparing raw traffic with an existing knowledge base. Although theses have systems have contributed a great deal to the network traffic security domain, a number of weaknesses still exist [23]. Open source IDS, such as Snort, monitor network traffic to detect invalid communication using predefined attacks [24]. Anomaly detection methods can examine large amounts of data in a real time. Machine learning techniques require less expert knowledge and provide good performance; however, a truly comprehensive IDS will always require human expertise [5]. A number of systems were examined from literature and most proposed multi step approaches. The most recent approach, proposed by Zheng et al. [25], described a multi step approach based on Support Vector Machines(SVM) on time binned NetFlow data. After identifying malicious time bins, the authors' used Kullback-Leibler distance between the test set and training set to identify the features contributing to the anomaly [25]. Another approach, proposed in [22], involved data collection, conversion into session data and feature extraction. Training data was used to build a normal profile and anomaly detection was conducted using a variety of clustering algorithms. In [26] a similar approach was adopted; however, clustering was used to filter data and Support Vector Machine algorithms were used to detect anomalies. In [27], a multi step detector using wavelets was proposed. Adaptive threshold methods were first applied for quick detection and then finer analysis was applied using continuous wavelet transforms. In [28], a five step approach was proposed. It involves data conversion, feature analysis, wavelet transform, statistical analysis and threshold techniques, wavelet synthesis and anomaly detection. The types of anomaly detection methods in anomaly detection systems vary in literature; but, to the best of our knowledge, all propose multi-step approaches.

The first steps consistently involved data collection and data conversion. None of the machine learning techniques reviewed analyze raw network traffic data. There are various methods to characterize the data for analysis. As seen in [29], raw TCPDUMP data from 1999 DARPA ID dataset is converted into network flow logs which based on different basic metrics. In Barford et al. [7], analysis is only applied to byte and packet counts, over five minute sampling intervals, from wide-area routing links. Although [17] does not reflect current network situations; it is still commonly used in research [8]. Another method used in literature is NetFlow and it is the industry standard to collect and characterize network traffic data [2]. In Lakhina et al. [16], NetFlow data was grouped into a time series and projected onto an anomalous subspace. In [30], the time bin features used in [16] were used and discussed how easily they can be scaled to larger and more diverse datasets. Brauckhoff et al. [6] used three weeks of NetFlow data from August 2007 by aggregating the traffic at 15 minute intervals. Alternatively, [31] show that there exists different ways of time binning data which offer good performance for tracking per-flow data. It presented an alternative to NetFlow.

It is common for Intrusion Detection Systems to consider raw network traffic that has been characterized different ways, such as NetFlow. The data is further profiled in order to reduce data volume and detection algorithms are applied. These detection methods can be performed either real-time or offline [5]. Alarms can be generated through inspecting the results of the algorithm and is required for automated detection. For example, if a database is used, SQL triggers which initiate a predefined action for a specific condition can be used [5]. Typically, threshold techniques are used to anomalies automatically. Yet, considering the importance of the network administrator, visual plotting of the output is beneficial.

In this research, Principle Component Analysis was used. It has been used in pervious works, such as [16], [30], [6] and [15]. Lakhina et al. [16] first made this approach popular. Ringberg et al. [15] examined [16] and critiqued it. Through extensive analysis, they showed that selection criterion to build the anomalous subspace directly impacts the error rate. More specifically, the PCA algorithm is very sensitive to top k parameter, and it varies across different networks or aggregations [15]. The authors found that sufficiently large anomalies can contaminate the operation of the algorithm. Lastly, they observed that the PCA algorithm has different effectiveness under different traffic types and measurements. In [15], it was concluded that it is difficult to pinpoint exact anomalous flows. Brauckhoff et al. [6] also examined the works proposed by [16], and suggested that the standard PCA analysis lacked temporal consideration. They suggested implementing a time shift technique in order to compensate for data trends over time.

This research also looked at spectral analysis and wavelet analysis. There are different variations of wavelet analysis, and there have been different works on it. In [27], the continuous wavelet transform was proposed to detect Denial of Service attacks. Other works were based on the Discrete Wavelet Transform. As outlined in [32], an approach which uniquely applied four algorithms which were based on different basis functions. In [7], an approach used to identify frequency characteristics of anomalous network traffic methods. It focused on identifying anomalies by removing predictable, ambient parts and only then employing statistical methods [7].

There have been few works comparing the spectral and statistical analysis techniques in the same way our research does. One hybrid approach, as outlined in [33], proposed a multi-step approach that used a Kalman filter to filter out normal traffic and wavelet filtering to process the residual data. The authors concluded that their approach did not perform very well. There has been some work done comparing the statistical and spectral analysis approach used in our work. In [19], the authors examined the PCA and wavelet anomaly detection methods. The authors observed that wavelets are just as good as entropy-PCA in most cases. Considering a Distributed Denial of Service attacks, it was found wavelets performed better. The authors observed that the effectiveness of entropy-PCA depends strong on top k parameter [19]. The authors performed extensive analysis and concluded that there exists no method consistently better than the other in detecting all types of anomalies. There were scenarios where both methods were inconclusive.

3.0 Anomaly Detection Data, Algorithms, and Threshold Techniques

This chapter describes profiling approaches applied to the network traffic data. The three anomaly detection algorithms are described: the Haar wavelet decomposition algorithm, the modified principle component analysis algorithm and the hybrid analysis algorithm. These algorithms are applied to profiled network traffic data in order to get a measure of anomalous behaviour. Lastly, the threshold techniques used to automatically isolate potential anomalous time slices are discussed.

3.1 Network Traffic Data and Data Profiles

The network traffic dataset used in this research is the Kyoto2006+ dataset [8]. This dataset provides session information, anti-virus and intrusion detector information and source–destination routing information (See Table 2.1 for a complete listing of features). Different data profiles were created to reduce the traffic volume. One type of profile was for wavelet analysis, and the other for PCA analysis. The hybrid approach used profiles created for PCA analysis. Each profile describes a certain period of time or time bin, and no time bins overlap. Each of these time bins have a start time, end time and various aggregate features derived from flows which fall within that particular time bin. For example, total bytes is the total amount of bytes transmitted by all flows with a start time after the bin start time and before the bin end time (See Table 3.1 and Table 3.2 for a complete listing of features for the wavelet and PCA profiles). The anomaly detection algorithms were applied to these profiles in order to determine potential time slices with malicious behaviour.

Features	Description
Start_time	The beginning of the time bin. All flows beginning by this time and ending by the corresponding end time are placed into the particular bin.
End_time	The finish of the time bin. All flows beginning by the corresponding start time and ending by this time are placed into the particular bin.
TCP_Bytes	Total number of bytes of the TCP protocol in this particular time bin.
TCP_Flows	Total number of flows of the TCP protocol in this particular time bin.
TCP_Source_IP_Address_Entropy	Entropy of source IP addresses from TCP protocol flows in this particular time bin.
TCP_Destination_IP_Address_Entropy	Entropy of destination IP addresses from TCP protocol flows in this particular time bin.
TCP_Unique_Source_IP_Address_Count	Total number distinct source IP addresses from TCP protocol flows in this particular time bin.
TCP_Unique_Destination_IP_Address_Count	Total number distinct destination IP addresses from TCP protocol flows in this particular time bin.
OTHER_Bytes	Total number of bytes not of the TCP protocol in this particular time bin.
OTHER_Flows	Total number of flows not of the TCP protocol in this particular time bin.
OTHER_Source_IP_Address_Entropy	Entropy of source IP addresses not from TCP protocol flows in this particular time bin.
OTHER_Destination_IP_Address_Entropy	Entropy of destination IP addresses from TCP protocol flows in this particular time bin.
OTHER_Unique_Source_IP_Address_Count	Total number distinct source IP addresses not from TCP protocol flows in this particular time bin.
OTHER_Unique_Destination_IP_Address_Count	Total number distinct destination IP addresses not from TCP protocol flows in this particular time bin.

Table 3.1: PCA Profile Features

Features	Description
Start_time	The beginning of the time bin. All flows beginning by this time and ending by the corresponding end time are placed into the particular bin.
End_time	The finish of the time bin. All flows beginning by the corresponding start time and ending by this time are placed into the particular bin.
TCP_Bytes	Total number of bytes of the TCP protocol in this particular time bin.
TCP_Flows	Total number of flows of the TCP protocol in this particular time bin.
TCP_Source_IP_Address_Entropy	Entropy of source IP addresses from TCP protocol flows in this particular time bin.
TCP_Destination_IP_Address_Entropy	Entropy of destination IP addresses from TCP protocol flows in this particular time bin.
TCP_Source_Port_Entropy	Entropy of source ports from TCP protocol flows in this particular time bin.
TCP_Destination_Port_Entropy	Entropy of destination ports from TCP protocol flows in this particular time bin.
OTHER_Bytes	Total number of bytes not of the TCP protocol in this particular time bin.
OTHER_Flows	Total number of flows not of the TCP protocol in this particular time bin.
OTHER_Source_IP_Address_Entropy	Entropy of source IP addresses not from TCP protocol flows in this particular time bin.
OTHER_Destination_IP_Address_Entropy	Entropy of destination IP addresses from TCP protocol flows in this particular time bin.
OTHER_Source_Port_ Entropy	Entropy of source ports not from TCP protocol flows in this particular time bin.
OTHER_Destination_Port_ Entropy	Entropy of destination ports not from TCP protocol flows in this particular time bin.

Table 3.2: Wavelet Profile Features

3.2 Wavelet Analysis in Anomaly Detection

Wavelet analysis examined how quickly data changed and the magnitude of this change over time. The wavelet transformations reveal the composition of a signal in terms of the building blocks, or translated basis functions, of the transformed domain [10]. The decomposition resulted in a set of orthogonal signals or components which combined, describe the signal entirely. In our application, the input into the system was the wavelet time bin profiles

(See Table 3.2 for an outline of the input features). The wavelet transformation applied in this thesis was the Haar wavelet decomposition. Haar wavelets are the most basic wavelets and are able to approximate any continuous function [10]. The Haar wavelet analysis can only examine single dimensional signals. In this research, the input signal was each of the feature observations. Each of the features from Table 3.2 were applied independently as input signals. Each was decomposed and frequency components of the signal were examined in an attempt to detect patterns which could be previously missed.

The process of applying Haar wavelet decomposition to network anomaly detection consists of three main steps. First, a Haar wavelet high pass filter, low pass filter and mid pass filter were each independently applied to the input data (See Figure 3.1 for a filtering diagram). As outlined in [7] the motivation for frequency filtering in anomaly detection was that the high frequency values highlight the start and end times of the anomalous behaviours and the mid pass frequency values highlight the duration of the potential attacks. Next, each of the filtered datasets was normalized by using Equation 3.1. The local variability over a sliding window was calculated on each of the normalized datasets. The Haar wavelet filtering occasionally produced an output of negative values which is not semantically meaningful in the network traffic domain. The local variability provided a positive measurement of how sparse the data in the window was (See Equation 3.2). Specifically, a larger local variability indicates a greater change of values in the window. Lastly, in order to obtain a single scalar value in time, the deviation score was calculated. The deviation score was a summation of weighted local variability values for each point in time. This deviation score was a measure of how often and how much values change at a point in time (See Equation 3.3). Either a plot of these scores was produced for visual inspection, or threshold techniques were applied for automated classification of anomalous time bins (See Figure 3.2).

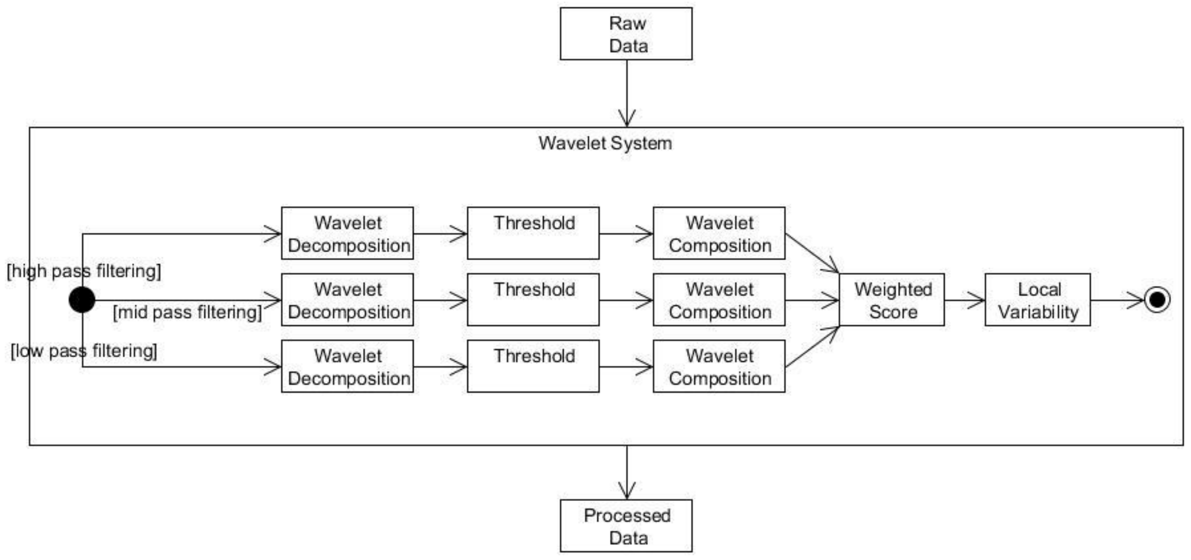


Figure 3.1: Activity Diagram of Haar Wavelet Filtering Approach

$$Normalized_i = \frac{obs_i - mean(obs)}{standardDeviation(obs)}$$

Normalizedi – normalized observation element i
obs – set of observations
Equation 3.1: Normalization Formula

$$DeviationScore_i(H, M, L) = H \times highpass_i + M \times midpass_i + L \times lowpass_i$$

highpass – set of normalized highpass values
midpass – set of normalized midpass values
lowpass – set of normalized lowpass values
H, M, L – set of weights
Equation 3.2: Deviation Score formula

$$LocalVariability_i(size) = Var(scores[i, i + size])$$

Size – size of window
Var – variance of input set
Score[x,y] – subset from scores starting at x to y
Equation 3.3: Local Variability formula

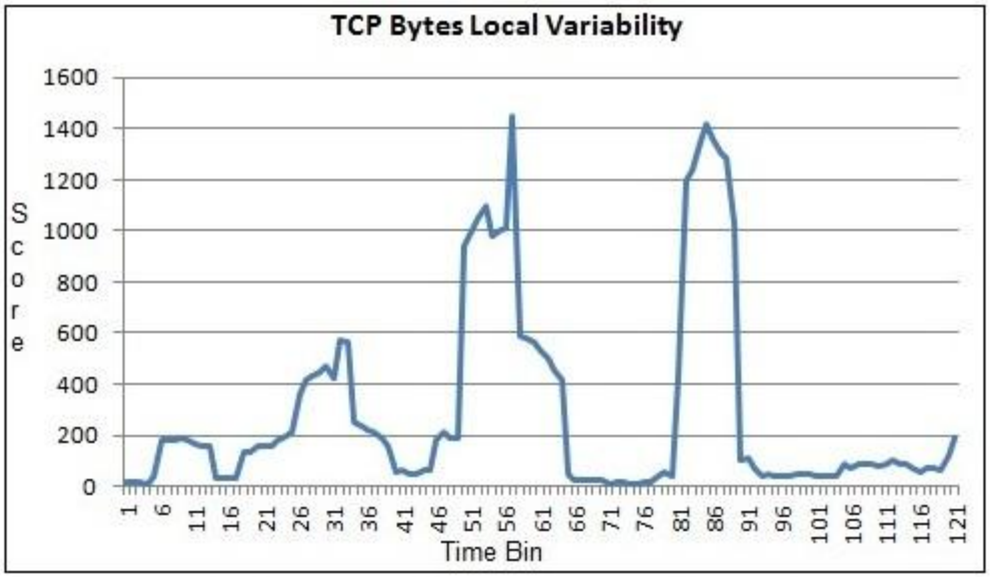
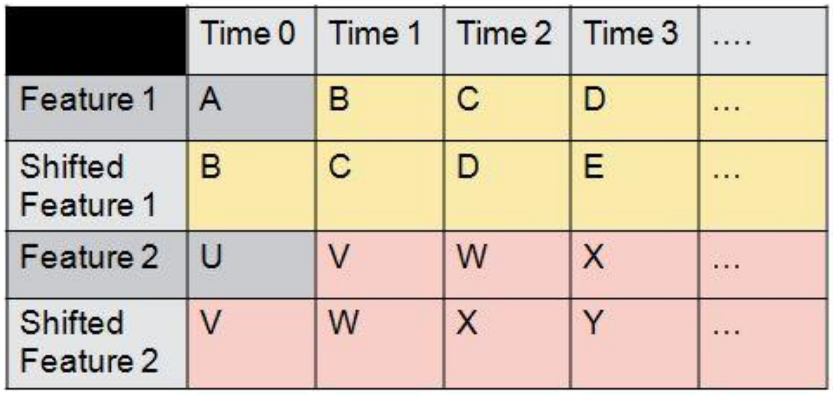


Figure 3.2: Local Variability score for March 8, 2009 with 5% infection rate

3.3 Modified Principle Component Analysis in Anomaly Detection

Principle Component Analysis (PCA) was designed to describe a set of observations from correlated variables into a set of new values of uncorrelated variables [13]. These variables were referred to as principle components and by definition are orthogonal. There are an equal number of principal components as original variables or features. The motivation was to create new independent components which describe the observations in more detail and uncover patterns and trends in data. For example, in a two-dimensional dataset, a line of best fit would be considered a principle component. In our application, the input into the system was the PCA time bin profiles (See Table 3.1 for an outline of the input features). As discussed in [6], applying straightforward PCA, as in [16], for anomaly detection does not account for temporal change. The type of PCA transformation applied in this work was a modified PCA approach, taken from [6]. In order to account for temporal changes, Brauckhoff et al. [6] proposed to create a spatio-temporal matrix. This matrix was a time shifted, by a variable amount of time bins, version of the original data matrix. For every feature and for every shift, a new “time shift” feature was created, and consisted of the subsequent observations. A number of time bins observations were dropped to compensate for the shifting (See Figure 3.3 for the time shifting illustration). Unlike wavelet analysis, PCA examined the entire input matrix as a whole, and performed an analysis on the entire dataset.



	Time 0	Time 1	Time 2	Time 3	....
Feature 1	A	B	C	D	...
Shifted Feature 1	B	C	D	E	...
Feature 2	U	V	W	X	...
Shifted Feature 2	V	W	X	Y	...

Figure 3.3: Time shifting Illustration

The modified PCA analysis was applied to the entire spatio-temporal input matrix. The result was a set of ranked principal components. The first component was the most significant and the last was the least significant. Depending on the selection criteria, a number of components were selected to build the anomalous subspace. This research, examined the Top K component selection criteria, the normal distribution selection criteria and the 99% of total space selection criteria. Applying one of these criteria built a component matrix, referred to as W. Each of the components was a column vector in the matrix W. Lakhina et al. [16] modeled the normal and residual space and projected the observations onto this space (See Equation 3.4 and Equation 3.5). For anomaly detection, the residual space was only considered. The original dataset was projected onto the residual space and a new transformed matrix was created. In order to get a measure of how anomalous each observation in time was, the Square Prediction Error (SPE) was calculated. The SPE is the Euclidean magnitude squared of the transformed observation in time (See Equation 3.6). Similar to the wavelet analysis, either a plot of these SPE scores could be produced for visual inspection, or threshold techniques were applied for automated classification of anomalous time bins.

$$Normal\ Space = WW^T y$$

W – column matrix of selected components
y – original data observations

Equation 3.4: Normal Space formula [16]

$$Abnormal\ Space = I - WW^T y$$

W – Column matrix of selected components
I – Identity matrix of size I
y – Original data observations

Equation 3.5: Abnormal Space formula [16]

$$\mathbf{SPE_i} = \left\|\tilde{\mathbf{C}}\mathbf{X_i}\right\|$$

SPEi – Square Prediction error at observation i
C – Matrix of anomalous subspace components
Xi – An observation, column vector from original dataset

Equation 3.6: Square Prediction Error (SPE) formula [16]

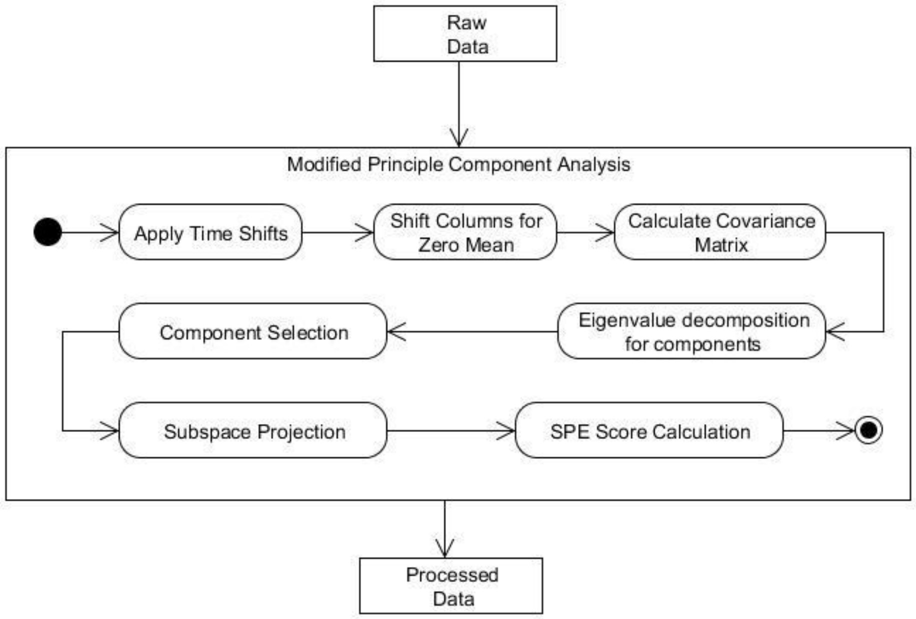

Raw Data
Modified Principle Component Analysis
Apply Time Shifts
Shift Columns for Zero Mean
Calculate Covariance Matrix
Eigenvalue decomposition for components
Component Selection
Subspace Projection
SPE Score Calculation
Processed Data

Figure 3.4: Activity Diagram of Modified PCA approach

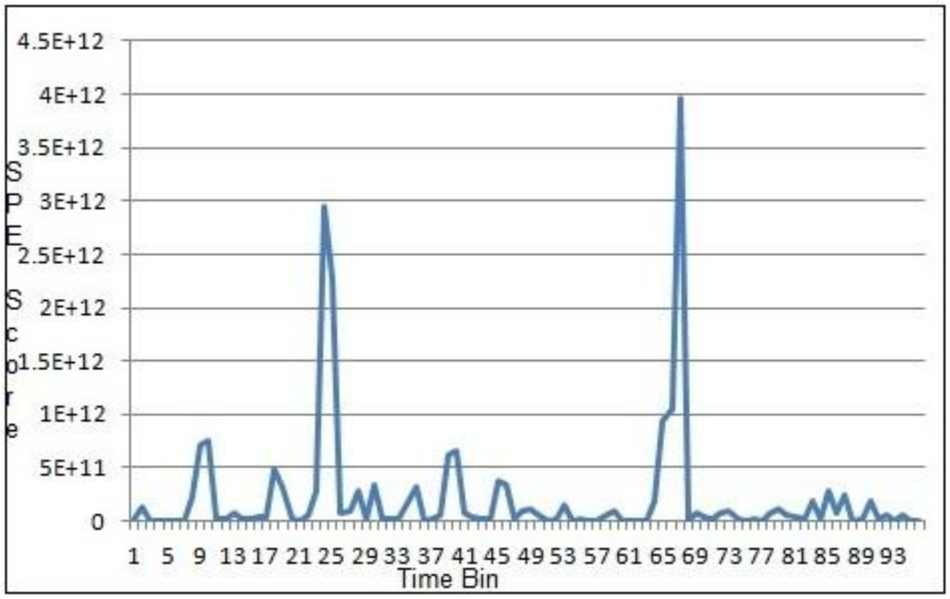


Figure 3.5: SPE score of March 8, 2009 for 5% infection rate

3.4 A Hybrid Analysis in Anomaly Detection

The hybrid analysis is a combination of the traditional Principal Component Analysis approach, as presented in [16], and Haar wavelet analysis, as used in [7]. Each of the approaches was described in sections 3.2 and 3.3 respectively. The hybrid approach uses the statistical and spectral analysis in a complementary fashion, and attempts to be less parameter dependant. The hybrid approach first aims to examine the entire dataset as a whole, as in PCA. Subsequently, a detailed localized frequency analysis to identify potential anomalous time bins is applied. In our application, the input into signal was the PCA time bin profile matrix (See Table 3.1 for an outline of the input features). Creating a time shifted spatio-temporal input matrix, as adopted in [6], was not considered. Next, the Haar wavelet analysis, examines the behaviour over time, and the concerns which shifting addressed are considered then.

The hybrid algorithm first applied PCA analysis to obtain a set of components. As in [16], the normal and residual spaces were modeled, and as before, only the residual space was considered. Unlike previous analysis in this research, for simplicity, only the single most significant component was considered. It was determined that the most significant component occupied a majority of the space. Limiting the selection criterion to "Top K=1 components" reduced the dependency on user specified parameters. As before, the original data input was projected onto the residual space in order to produce a transformed dataset for further analysis. The transformed dataset was comprised of the same amount of time bin observations as the input data. Applying the projection did not drop or create any new features. It did transform the existing ones, and strip them of their semantic meaning. The semantic meaning may have been lost; however, the new features were more representative of the behaviour and were able to potentially reveal patterns or trends.

The Haar wavelet analysis examines the degree of change of a single feature set at a time. This approach strives to be independent from user input parameters. The hybrid algorithm applies a selection criterion between features, based on variance. Variance is a measure of spread within a dataset. A dataset may have the same average, but the dataset with the greater variance is more spread out (See equation 3.7). In the hybrid approach, every feature dataset is examined and the feature with the greatest variance is selected for wavelet analysis. The motivation was that the most spread out data had the greatest potential for large changes over time. Next, as outlined in [7], the Haar wavelet high pass filter, low pass filter and mid pass filter are each independently applied to the input data. Each of the filtered dataset is normalized (see Equation 3.1) and a score, similar to deviation score, is calculated. This score is a summation of weighted normalized frequency values for each point in time. Local variability was not considered in order to be independent from user defined parameters. The score highlights the descriptive properties of the filtering and was sufficient enough to characterize the behaviour of the data. Examining data over a subset is offloaded to the threshold phase. Yet, the Haar wavelet filtering produced an output of negative values which is not semantically meaningful in the network traffic domain. In this case, the absolute difference, or delta, between two time bin scores (See equation 3.8) is examined. Either a plot of these delta scores was produced for visual inspection, or threshold techniques were applied for automated classification of anomalous time bins.

$$Var(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

Equation 3.7: Variance formula

$$DeltaScore_i = |score_i - score_{i+1}|$$

$Score_i$ – the anomalous score for a specific time bin

Equation 3.8: Delta score formula

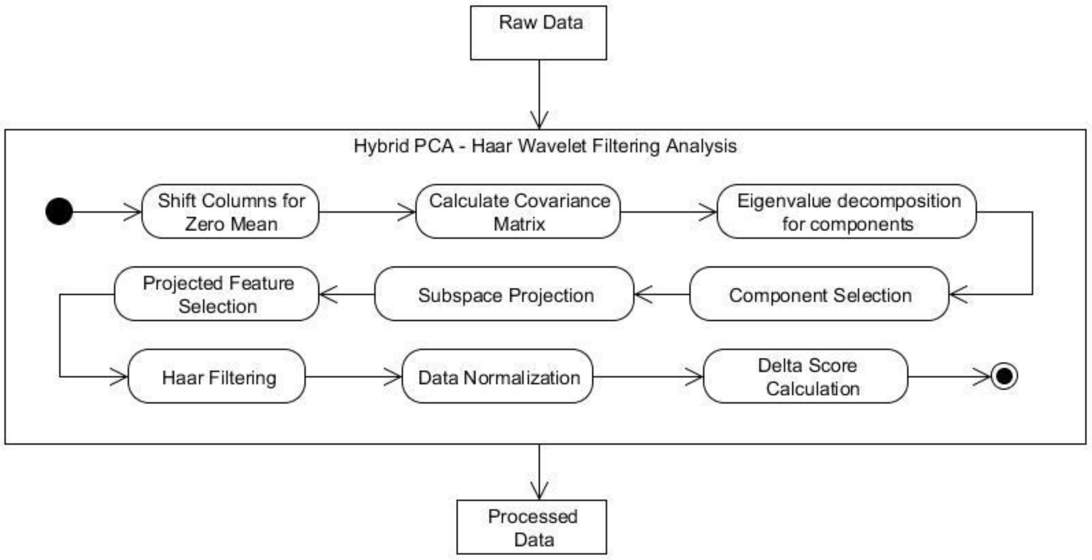

Raw Data
Hybrid PCA - Haar Wavelet Filtering Analysis
Shift Columns for Zero Mean
Calculate Covariance Matrix
Eigenvalue decomposition for components
Component Selection
Subspace Projection
Projected Feature Selection
Haar Filtering
Data Normalization
Delta Score Calculation
Processed Data

Figure 3.6: Hybrid PCA-Haar Wavelet Filtering activity diagram

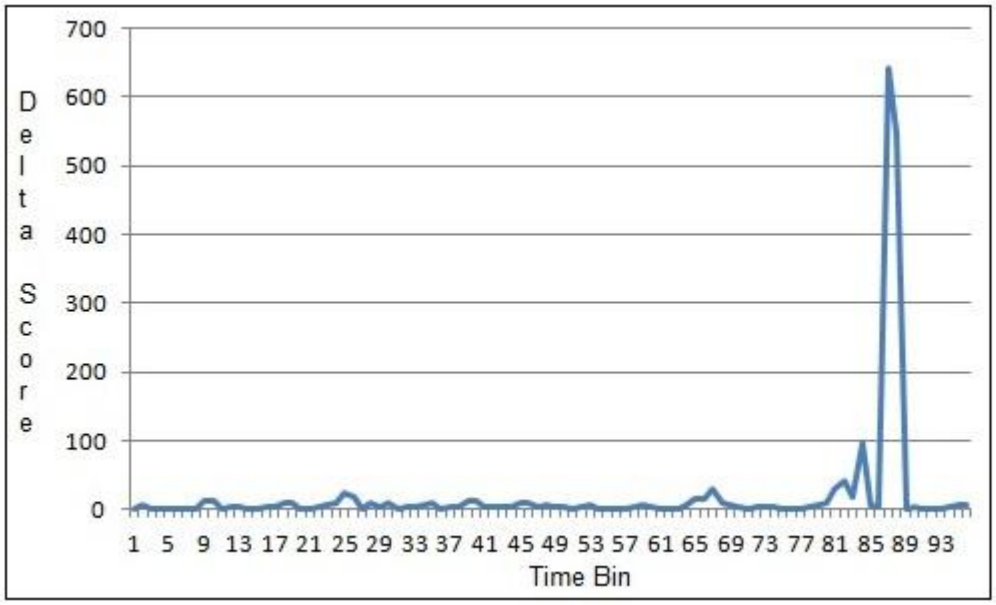


Figure 3.7: Delta score for March 8, 2009 for 5% infection rate

3.5 Threshold Techniques

A visual plot was a quick and easy way to present data. Yet, automating a specific task has many advantages, such as decreased detection time, reduction of human error and decreased operational complexity. The following subsections discuss various threshold techniques applied to the output plots of each of the anomaly detection methods.

3.5.1 Average

The quickest and simplest threshold calculation is the average. This calculation considers the entire set of scores and calculates the statistical mean (See Equation 3.9). However, an overwhelming majority of flows are less than one minute in length. It is problematic calculating an average from an analysis over a day or two. The behaviour the next day may be very much different and giving equal consideration to each time bin score may not be appropriate. A moving average, of size N observations, is a way of handling this range in time. The statistical mean for the first N elements is calculated and is the threshold value for the first N time bins. Then, the window slides over by one and the threshold value for time bin N+1 is the average. This repeats until the window reaches the end of observations (See Equation 3.10). Any score greater than either than the total average or moving average at this point in time is flagged as potentially anomalous for the network administrator.

$$Static\ Average = \frac{\sum scores}{n}$$

score – set of scores
n – number of elements in set
Equation 3.9: Static Average formula

$$Moving\ Average = \frac{\sum scores[i, i+n]}{n}$$

score[x,y] – set of scores starting at x to y
n – number of elements in set
Equation 3.10: Moving Average formula

3.5.2 Sliding Weighted Average

As discussed in the previous section, it may not always be appropriate to give equal weighting to each time bin score value. The sliding weighted average is alternative to the sliding moving average. This approach applies a weighting to each time bin score value and considers calculates the statistical mean. Each observation has a weight that is multiplied to the observation value. The weighting has an initial weighting of 1.0 and decrements evenly to 0. Similar to the moving average described in section 3.10, a window size, of N observations, is provided and those elements in the window are considered. This repeats until the window reaches the end of observations (See Equation 3.11). Any score which is greater than this values determined at this point in time is flagged as potentially anomalous for the network administrator.

$$WeightedMovingAverage = \frac{\sum_{j=0}^{n} W_j \times scores[i, i+n]}{n}$$

score[x,y] – set of scores starting at x to y
n – number of elements in set
W – set of weights

Equation 3.11: Weighted Moving Average formula

3.5.3 Average without outliers

Similar to the previously discussed anomaly detection algorithms, this threshold technique attempts to separate normal behaviour and residual values. The motivation is to build a curve which represents the normal behaviour and act as a threshold. There are two user defined parameters: the number of observations to be considered at a time, *N* and the growth tolerance, *GT*. The growth tolerance, *GT*, defines how much larger the time bin score can be from the threshold value. The first time bin score is the first threshold value. The average of the first two time bin scores is the second threshold value. The average of the first three time bin scores is the third threshold. This continues up until the first *N* threshold values are determined. The average of the first *N* many scores is considered the running threshold. If the next element is greater than the running threshold plus the tolerance, the corresponding time bin is flagged as potentially anomalous. Also, in this case, running threshold is not updated, and the threshold for that time bin is the current running threshold. However, if the element is less than or equal to the running threshold plus the tolerance, the running threshold is updated. The update calculates the average over the *N* elements, considering the new score and dropping the oldest score. The threshold value for that time bin is the new updated threshold value. Each element is considered one by one until all data has been examined. Any time bin scores greater than the corresponding threshold value are flagged as potentially anomalous for the network administrator.

4.0 Prototype Design and Implementation

A prototype test bed was developed to perform experiments. The algorithms and various functions outlined in previous chapters were implemented in order to obtain experimental results perform analysis. The prototype anomaly detection test bed is comprised of two components; the front end user interface and the backend server. It is implemented with client-server architecture. The front end is a browser based client interface and was implemented using an interactive open-source Web development framework, called Adobe Flex [34]. The browser is the responsible for handling user commands, making requests to the server and presenting data. The back end server is an open-source, Java based, web server called Apache Tomcat [35]. This server handles data lookup requests and applies various algorithms to specified datasets. The server performs various operations requested, and responds to the front end. The user is able to view the results analyze them and execute more operations (See Figure 4.1 for a system diagram). This chapter provides a high level overview of the test bed and describes the design and implementation of each of the two components.

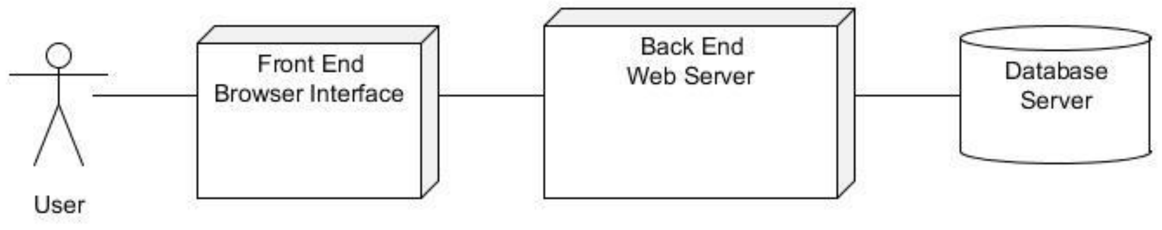


Figure 4.1: Test bed system diagram

4.1 Front-end Interface

Typically, desktop applications are stand alone programs which are platform dependant and sensitive to a number of environment parameters, such as hardware specifications, operating systems, etc. Applications can become complex to develop and maintain. Conversely, web pages are tremendously easy to implement. The framework front end is browser based, built using the Adobe Flex Web framework, and is very quick and easy to implement. The Adobe Flex framework is an open source technology used to create rich Internet applications. These applications are web pages that are dynamic, their content is reloadable and they act like desktop applications. Adobe Flex supports traditional Web framework functions such as POST and GET requests to servers. Furthermore, it provides a number of graphs and widgets which bind with XML data to render detailed illustrations. Adobe Flex is a powerful tool, does not require extensive programming experience to use and satisfies all of the client interface requirements. [34]

The network administrator operates the browser based front end. The interface enables the user to perform and examine all phases of the proposed anomaly detection system (See Figure 4.2 for a use case diagram of anomaly detection test bed). The operator can load raw network traffic data into the system, and drop any existing datasets. The operator can create subsets of data for analysis from loaded traffic data. These subsets can contain a desired infection ratio during a specified time segment. Data profiles can be created from the subsets and preserved for future analysis. The anomaly detection algorithms, described in chapter 3, can be applied to data profiles, and the results are visualized in a plot for manual inspection (See Figure 4.3 for a screen capture of a plot). For the results, in some cases, higher scores can overwhelm the behaviour of smaller scores. The user has the ability to remove specific scores such that the plot can be amplified on the remaining data. For manual investigation, the threshold function is plotted on top of the results. The program supports manual shifting of the threshold. The various threshold techniques, as outlined in section 3.5, can also be applied to provide an automatic classification of time bins (See Figure 4.4 for a screen capture). The Kyoto2006+ is a labeled dataset and the evaluation scores can be calculated to provide the user a scalar measure of how well the approach performs under the specified conditions (See Section 5.2 for an outline of the evaluation measures). The front end is comprehensive enough such that the user can perform their own inspection, or rely on the system automated results.

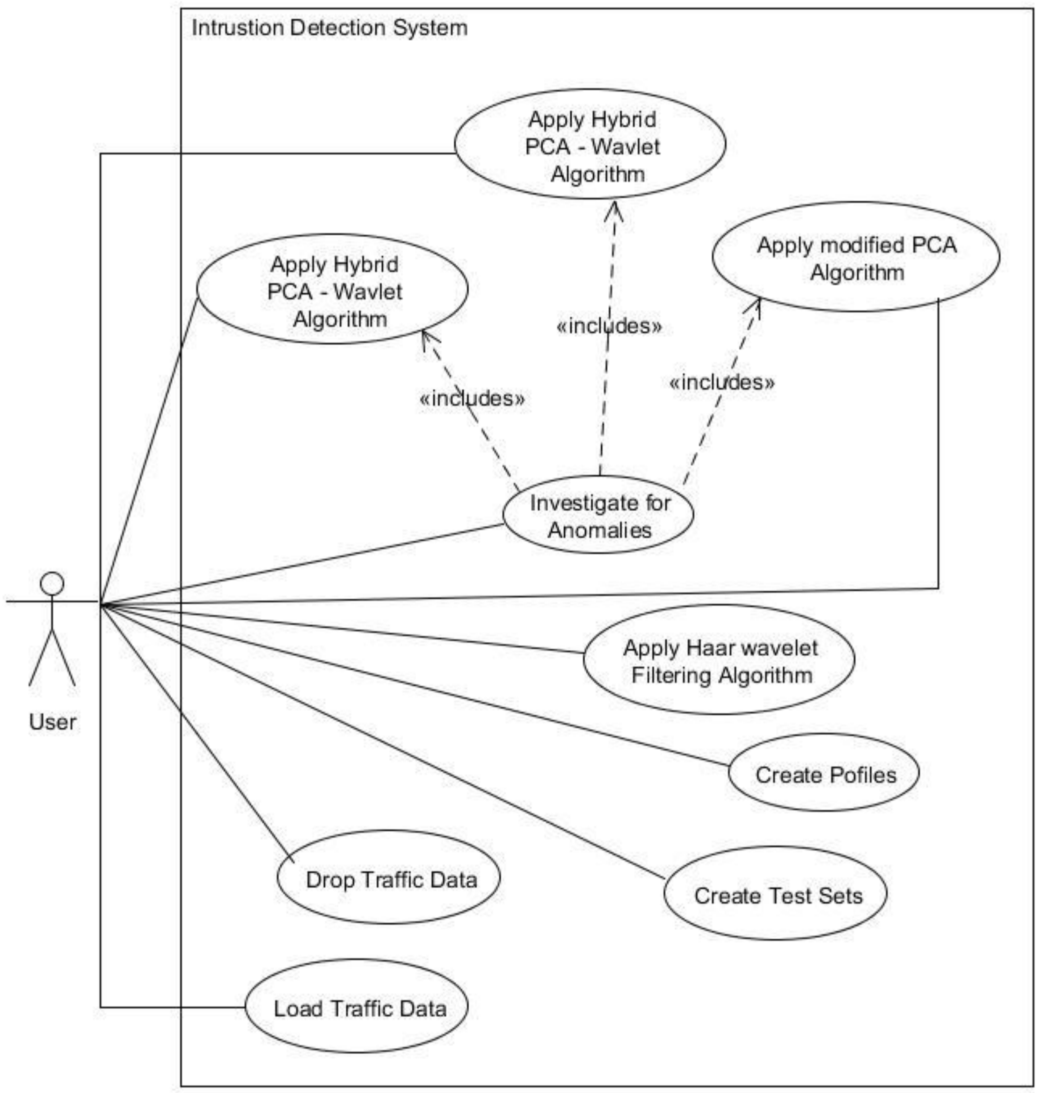

Intrustion Detection System
Apply Hybrid PCA - Wavlet Algorithm
Apply Hybrid PCA - Wavlet Algorithm
Apply modified PCA Algorithm
«includes»
«includes»
«includes»
Investigate for Anomalies
User
Apply Haar wavelet Filtering Algorithm
Create Pofiles
Drop Traffic Data
Create Test Sets
Load Traffic Data

Figure 4.2: Use case diagram of test bed system

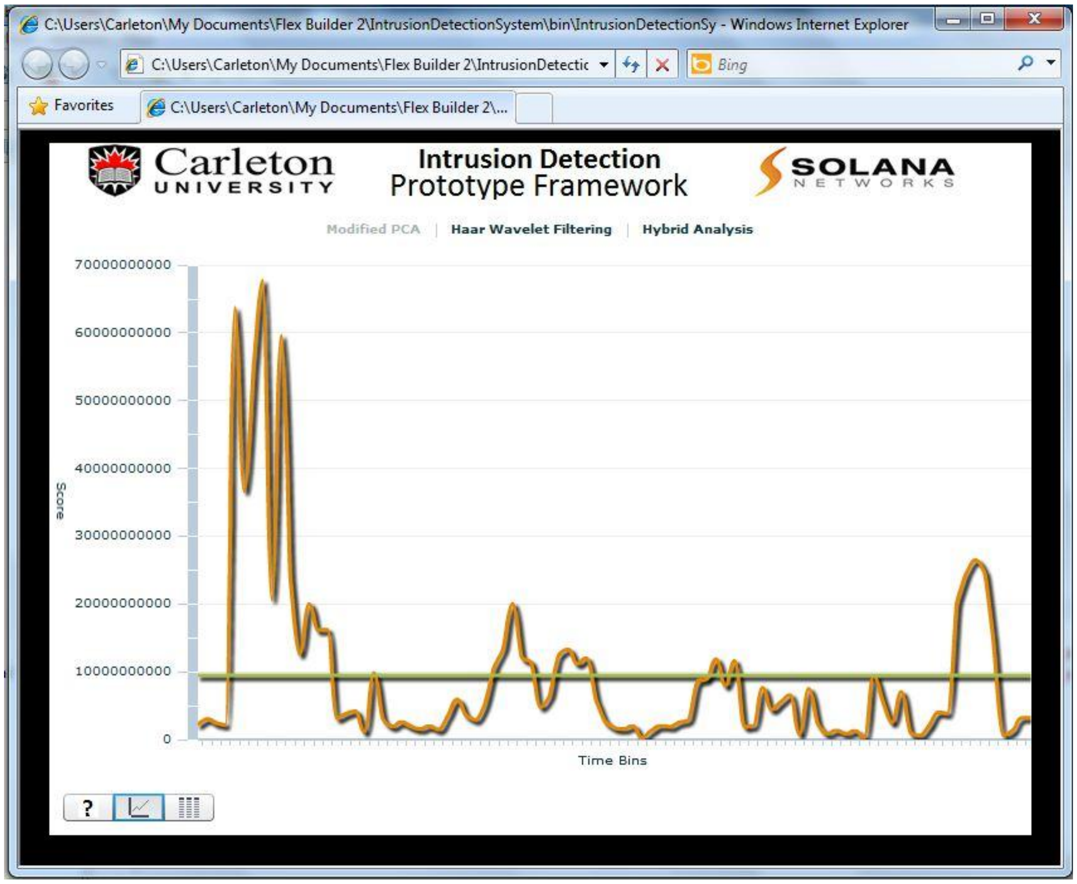
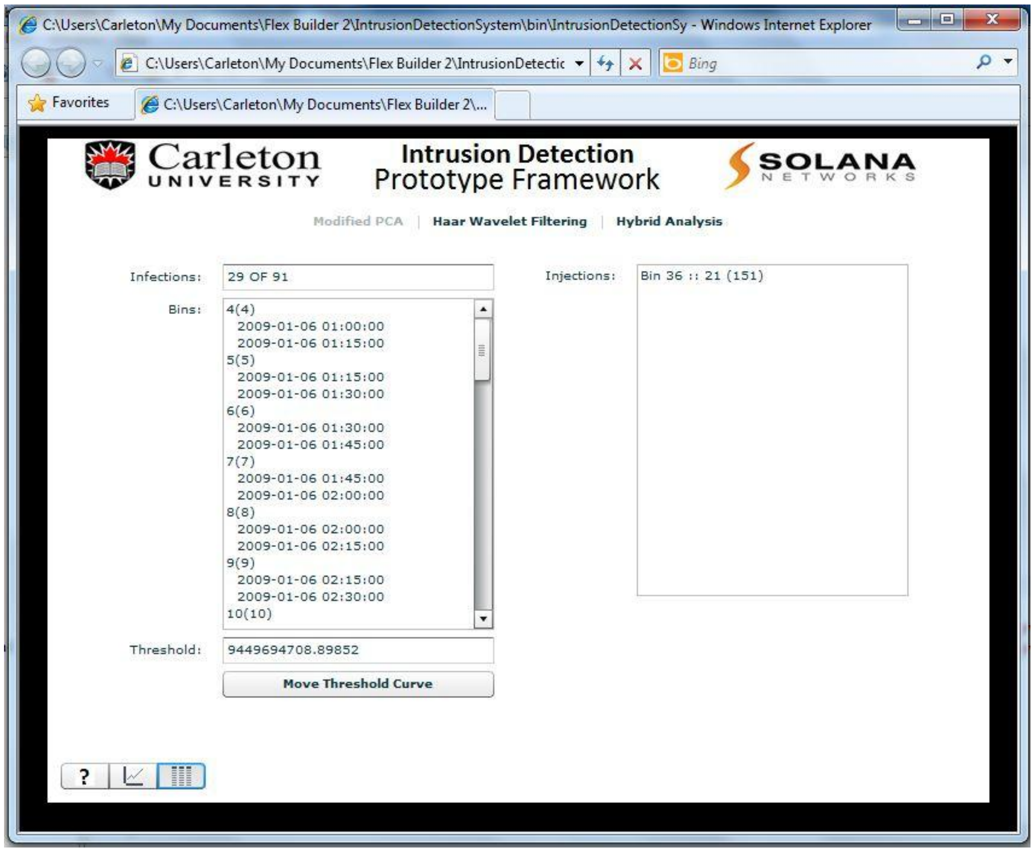

C:\Users\Carleton\My Documents\Flex Builder 2\IntrusionDetectionSystem\bin\IntrusionDetectionSy - Windows Internet Explorer
Favorites
Carleton UNIVERSITY
Intrusion Detection Prototype Framework
SOLANA NETWORKS
Modified PCA | Haar Wavelet Filtering | Hybrid Analysis
70000000000
60000000000
50000000000
40000000000
30000000000
20000000000
10000000000
0
Score
Time Bins

Figure 4.3: Screen capture of SPE scores plotted from test bed


C:\Users\Carleton\My Documents\Flex Builder 2\IntrusionDetectionSystem\bin\IntrusionDetectionSy - Windows Internet Explorer

Carleton UNIVERSITY | Intrusion Detection Prototype Framework | SOLANA NETWORKS

Modified PCA | Haar Wavelet Filtering | Hybrid Analysis

Infections: 29 OF 91

Injections: Bin 36 :: 21 (151)

Bins:
```
4(4)
  2009-01-06 01:00:00
  2009-01-06 01:15:00
5(5)
  2009-01-06 01:15:00
  2009-01-06 01:30:00
6(6)
  2009-01-06 01:30:00
  2009-01-06 01:45:00
7(7)
  2009-01-06 01:45:00
  2009-01-06 02:00:00
8(8)
  2009-01-06 02:00:00
  2009-01-06 02:15:00
9(9)
  2009-01-06 02:15:00
  2009-01-06 02:30:00
10(10)
```

Threshold: 9449694708.89852

Move Threshold Curve

Figure 4.4: Screen capture of information panel from test bed

4.2 Back-end Server

The back end serves the client user interface. The server sits idle until requests are made and responds to them in an XML format. The back end is an Apache Tomcat Web server, which implements Java servlets. To handle incoming http requests, for a pure Java Web serving solution [35]. All Java classes, interfaces and libraries are created, compiled, packaged together and then deployed on the Apache Tomcat server. Java satisfies the requirements for this application, as it is a powerful platform with the ability to connect with a local file system, a database and perform CPU intensive computations.

The server interfaces with various system components. As the network traffic data comes in text files, the system interfaces with the file system and a database. A database is used because it provides support for complex queries and enables specific data to be accessed easily. The server loads network traffic data into a database. Additionally, it poses queries to the database in order to create the profiles and save them into text files for future analysis. All of the anomaly detection algorithms described in Chapter 3 are implemented, and can be applied to a user specific dataset. In order for increased system stability, open source libraries, such as a Matrix library and a Haar wavelet filtering library, are leveraged (See Figure 4.5 for a class diagram of the system). Open source libraries from credible sources, such as Google projects, are preferred because they are maintained by a community and have been used extensively. The intense computations are preformed on the server side to remove workload from the user interface. Java is a powerful platform which is more appropriate to perform CPU intensive computations over a Web scripting language. The threshold techniques, evaluation operations and other utility functions were implemented in order to support all the features of the client interface. As required by Adobe Flex, all server responses are formatted in the XML format.

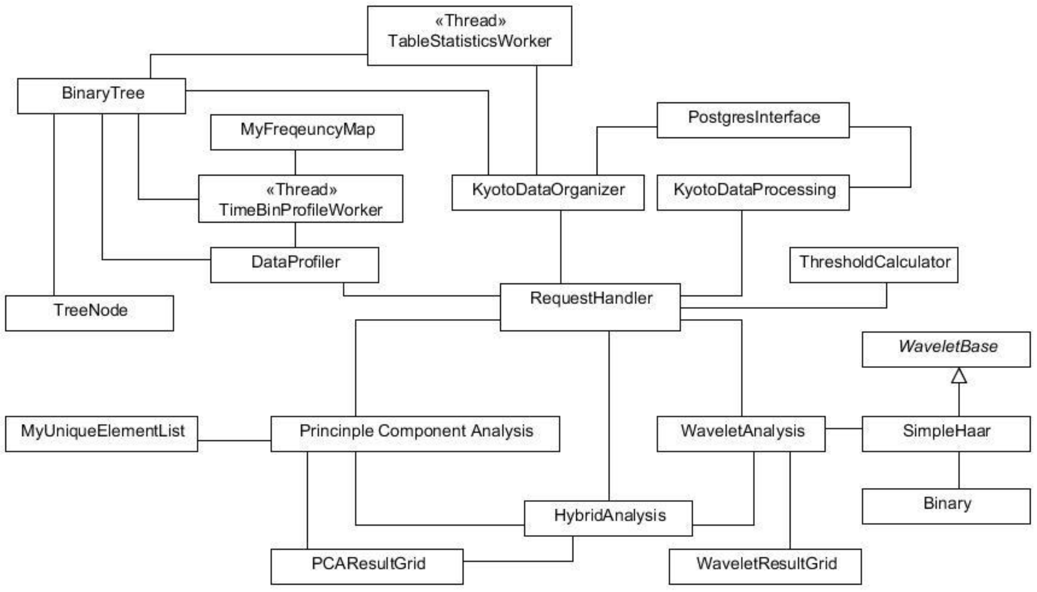


Figure 4.5: Class diagram of backend server from test bed

5.0 Experiments and Evaluation

This chapter presents the experiments conducted in this research, including the data sources used and data profiling. Furthermore, it outlines the intrusion detection algorithm step, threshold / anomaly detection step and lastly, the evaluation.

5.1 Experiment Overview

There were a number of experiments conducted with the anomaly detection test bed to determine the performance of the three anomaly detection algorithms presented in Chapter 3. Prior to applying the algorithms, datasets for testing needed to be created and profiled. A number of specific dates and infection times were selected for testing. These test sets were created and then both the PCA and the Wavelet profiles were extracted. The three algorithms were applied independently and using various parameters. Threshold techniques were applied to determine the potentially anomalous time bins. As the test sets were labeled, evaluation techniques were used to evaluate the performance of the intrusion detection algorithms.

5.2 Experimental Setup

This section outlines how the experiments were constructed and provides the rationale for the experiment decisions made. First, data setup is discussed and then evaluation techniques are described.

5.2.1 Data Source Setup

As discussed in section 3.1, the Kyoto2006+ data source was used in experimentation [8]. The Kyoto2006+ dataset provides session data for each day, starting in November 2006 until the end of August 2009. The volume of network traffic sessions increased with time. For example, November 2006 had 1,223,899 sessions, November 2008 had 2,374,504 sessions and July 2009 had 3,870,725 sessions. The data available in 2009 was the largest, which is why analysis began in January 2009.

As per the Kyoto2006+ specification, there are three features which outline the output of existing open-source detectors [8]. Also, the label feature indicates the authors' belief about the particular session is anomalous or not. For our research, normal flows were defined as flows the authors labeled as normal (1) and raised no detector alerts. Furthermore, infected flows were flows labeled anomalous (-1) by the authors and raised an alert with at least one of the three detectors. This approach used the detectors as validation to remove any uncertainty from the labeling. The session would more likely be normal or anomalous if all methods agreed.

One major limitation from this dataset is the amount of flows labeled as -1. Examining the flow counts (See Table 5.1), it can be observed that there is an overwhelming majority of these flows and this issue has been raised by Song et al. [26]. The aforementioned definition of anomalous flows was applied, and the number reduced so much that it became the minority. Next, the definition of normal flows was applied and observed that the number of normal flows also reduced. The result was that there were many more normal flows than anomalous, and the total amount of data was less than before. The approach used in this research, similar to [26], was to build a test set using some normal and some anomalous flows.

Our analysis was applied on 24 hours at a time. Throughout the 2009 dataset, the duration of sessions less than one minute was 98%. Performing analysis on 24 hours at a time would maintain small enough time bin duration such that anomalies would not be hidden. Although other works, such as [6] and [7], used a different data source with different features, they still used analysis over one day. A total of 16 24-hour time ranges, which had different behaviours, were selected. This research considered two attack characteristics: length of attack time and intensity of attack. Hence, experimentation was performed over four different infection scenarios:

- short length and high intensity.
- long length and high intensity.
- short length and low intensity.
- short length and high intensity.

The base case with no infection, 0% infection rate, was considered. For low infection rates, 5%, 10%, and 20% infection were created. For high infection rates, 75%, 100% and 150% infection were created (See Equation 5.1 for infection rate calculation). A total of 16 different time ranges, four for each scenario, were picked out. These ranges varied in days of the week, and actual time of day in which the attack occurred (See Table 5.2 for a breakdown).

Month	**Total Flows**	**Total Normal (label = 1)**	**Total Abnormal (label = -1)**	**Total our normal (label = 1 and no alerts)**	**Total our abnormal (label = -1 and at least one alert)**
January	3,670,176	2,216,820	1,453,356	2,177,273	129,571
February	3,321,532	1,947,619	1,373,913	1,911,817	113,398
March	3,398,114	1,861,103	1,537,011	1,828,827	139,291
April	3,767,136	2,637,097	1,130,039	2,534,233	106,136
May	3,747,527	2,153,355	1,594,172	2,108,499	107,484
June	3,743,655	1,852,554	1,891,101	1,807,121	101,633
July	3,870,725	2,086,698	1,784,027	2,026,854	81,632
August	3,710,941	1,640,651	2,070,290	1,568,229	223,476

Table 5.1: Kyoto2006+ flow count breakdown per month

Dates	Start Time	Analysis Duration
Short Duration and Low Infection Rates		
Tuesday, January 6, 2009	0:00	24 hours
Sunday, March 8, 2009	22:00	24 hours
Sunday, April 26, 2009	20:00	24 hours
Tuesday, May 5, 2009	21:30	24 hours
Short Duration and Low Infection Rates		
Friday, February 20, 2009	0:00	24 hours
Thursday, March 19, 2009	0:00	24 hours
Thursday, May 28, 2009	18:00	24 hours
Tuesday, July 14, 2009	19:00	24 hours
Short Duration and High Infection Rates		
Thursday, January 15, 2009	0:00	24 hours
Wednesday, April 1, 2009	20:30	24 hours
Monday, June 22, 2009	0:00	24 hours
Sunday, July 5, 2009	0:00	24 hours
Long Duration and High Infection Rates		
Monday, February 9, 2009	21:00	24 hours
Wednesday, June 10, 2009	22:00	24 hours
Sunday, August 9, 2009	20:00	24 hours
Friday, August 28, 2009	19:00	24 hours

Table 5.2: Dates in experimentation by scenario and analysis duration

$$InfectionRate = \frac{\#\ of\ injected\ Abnormal\ Flows}{\#\ of\ Normal\ Flows} \times 100\%$$

Equation 5.1: Infection rate formula

In order to maintain statistical independence, ten test sets were created. Similar to Song et al. [26], all normal flows and a subset of anomalous flows to create the desired infection rate in a test set were used. The test sets were derived from the original primary dataset. At each of the 16 time ranges, all the normal flows from the main collection were randomly and evenly distributed among the test sets. Each flow would be in only one test set, and all test set sizes had the same amount of flows. Next, depending on testing scenario, or desired infection ratio, a number of anomalous flows were injected. The flow characteristics were not changed. Due to the limited number of available anomalous flows, membership was not restricted to one test set. From these test sets, profiles were extracted and the algorithms were applied.

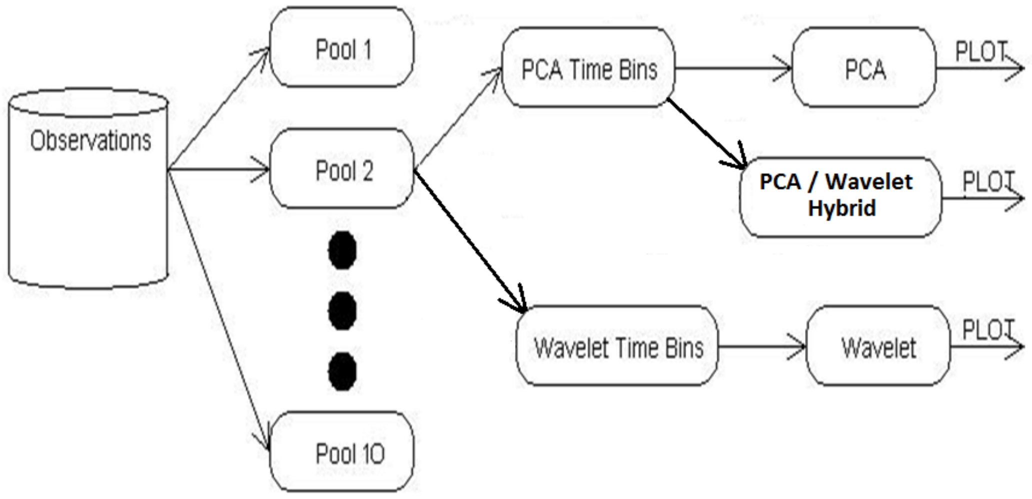


Figure 5.1: Experiment high level diagram

5.2.2 Performance Measurement Setup

The Kyoto2006+ is a labeled dataset, so it is possible to measure how well the algorithms perform. Two measurements were used to evaluate the performance of the algorithms: accuracy and precision. Accuracy is defined as how correctly the time bins are classified. It was measured using a hit miss count which counted the number of correct and incorrect classifications. The definition of precision is how exactly the algorithm can classify time bins. It was measured using a single scalar value to show how well the detection method performs.

5.2.2.1 Hit and Miss Count

The hit miss count compares the presence of anomalous flows within time bins. As part of the profiling phase, the number of anomalous flows and the number of normal flows per time bin preserved in the form of a list. If the anomalous flow count in a time slice was greater than zero, the bin was classified anomalous. Otherwise, it would be classified normal. After applying the anomaly detection algorithms and using the threshold techniques, a list of presumed abnormal and normal time bins was produced. The actual list and the experimental list were compared and the hit and miss counts were calculated. The hit count was the number of time slices in which the labels matched up in both lists. Conversely, the miss count was the number of time slices in which the labels did not match. These counts provide an indication of how accurately the algorithm performed.

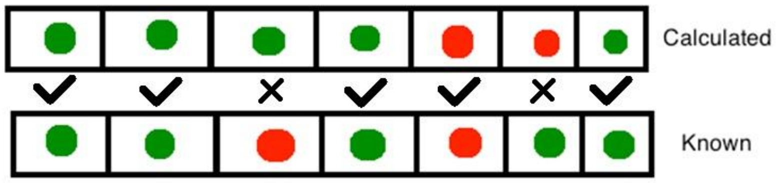


Figure 5.2: Hit and miss count illustration

5.2.2.2 Hit and Miss Score

The hit and miss score is a credit scheme where different amounts of credits are awarded and taken away based on the situation. There is some uncertainty with the information provided by the Kyoto2006+ detectors. The programs are primarily signature based detection systems and may miss some anomalies [4]. The credit score aims to accommodate for this uncertainty (See Table 5.3 for an outline of the credit scoring key). Similar to the hit and miss count, the number of anomalous and normal flows for each time slices was preserved. After applying the algorithms, a list of presumed abnormal and normal time bins was produced. Each time bin flagged as abnormal was compared with the known counts of normal and abnormal flows. In the most obvious case, where there are all infections and no normal flows, two credits are awarded. If there are no flows at all, two credits are taken away. No reward or penalty was awarded if only normal flows were present. This may be a scenario in which the authors of the Kyoto2006+

could have incorrectly classified the bin. If a bin with normal and anomalous flows is classified, then one credit is awarded. These scores provided an indication of how precisely the algorithm performed. A higher score indicates a more precise analysis.

Anomaly Count	Normal Count	Credits
0	> 0	0
0	0	-2
> 0	> 0	+1
> 0	0	+2

Table 5.3: Credit Score Breakdown

5.3 Experimental Procedure

For each method, a number of experiments were conducted. The parameters considered for each method include the combination of the infection rate, the four scenarios described in section 5.2.1, and the time ranges. Table 5.4 summarizes the tunable parameters and default values adopted for the experiments.   The following summarizes some experimental conditions:

- Infection rate: 0%, 5%, 10%, 20% and 75%, 100%, 120%
- Length of attack time: short (45 minutes) and long (150 minutes)
- Intensity of attach: low (0%, 5%, 10%, 20%) and high (75%, 100%, 120%)
- Number of time ranges: 16

Algorithm	Parameter	Values	Default Value
Modified PCA	Component Selection Criteria	Subspace method, Top k=1, Variance > 1 method	Subspace method
	Number of Time Shifts	1,2,3	1
Haar Wavelet Filtering	Input data size / Boundaries	Input size / filtering levels 128/ (2,3,2),(3,2,2) 256 / (3,2,3),(3,3,2) 512 / (3,3,3),(4,3,2)	128 / (2,3,2)
	Local Variability Weights	(5, 1, 0), (3, 2, 0)	(5, 1, 0)
BOTH	Threshold method	Moving Average without outliers, Moving Average, Static Average, Weighted Moving Average	Moving Average without outliers

Table 5.4: Outline of parameters used in experimentation

5.3.1 Test Set Creation

Analytics were performed on each the available data for each month. The number of anomalous and normal flows every 15 minutes was counted. For each of the scenarios, listed in section 5.2.1, four time ranges were selected. A total of two days from each of the 8 months in 2009 were examined. When examining for appropriate dates, short was defined as three 15 minute time bins, and long was considered as ten 15 minutes time bins. A low amount of flows were considered to be able to create at most a 20% infection rate. A high amount of flows was considered to be able to create at most a 20% infection rate. The Modified Principle Component Analysis and Haar Wavelet Filtering algorithms each have their own set of tunable parameters (See Table 5.4 for an outline of parameters used in experimentation).

5.3.2 Modified Principle Component Analysis Experiments

A number of experiments with the Modified PCA were conducted. In addition, the selection criteria used were top K, subspace and variance. SPE scores were calculated as evaluation measurements. Specific experiments conducted for the thesis are described as follows.

Experiment #1: Pure data across all time ranges

- Apply the profiling on 0% infection rate to each test sets.
- Run the modified PCA algorithm using a number of time shifts and selected component selection criteria. The number of time shifts varies from 1, 2 and 3. The selection criteria options are top K, subspace and variance.
- Average out the SPE scores from the ten test sets.
- Plot out average SPE scores versus time.
- Run for all combinations of time shifts and selection criteria.
- Repeat for all 16 time ranges.

Experiment #2: For each of the four dates in the long and low scenario

- Apply the PCA profiling on the ten 20% infection rate test sets.
- Filter out any observations which are 15 times more than the mean on one date.
- Run the modified PCA algorithm using a number of time shifts and selected component selection criteria. The number of time shifts varies from 1, 2 and 3. The selection criteria options are top K, subspace and variance.
- Average out the SPE scores from the ten test sets.
- Apply each of the threshold methods to the average output.
- Apply the two evaluation techniques to each of the threshold methods.
- Run for all nine combinations of time shifts and selection criteria.
- Repeat using the ten 10%, and the ten 5% infection rate test sets.

Experiment #3 used the dates of the short and low scenario and repeated Experiment #2.

Experiment #4 used the dates of the long and high scenario, infection rates of 75%, 100% and 120% and repeated Experiment #2. Experiment #5 used the dates of the short and high scenario, infection rates of 75%, 100% and 120% and repeated Experiment #2.

5.3.3 Haar Wavelet Filtering Analysis Experiments

A number of experiments with the Haar Wavelet Filtering were conducted. Three different data input sizes of 128, 256 and 512 and their respective filtering boundaries were used. In addition, the local variability weights of 5,1,0 and 3,2,0 where used. Local variability scores were calculated as evaluation measurements. Specific experiments conducted for the thesis are described as follows.

Experiment #1: Pure data across all dates

- Apply the profiling using the different number of bins / bin width on 0% infection rate test sets. The number of bins per day is 128, 256 and 512.
- Run the Haar Wavelet algorithm on the three different profiles using the respective boundaries and weight combinations. See Table 5.4 for a list of sizes and boundaries. The weights (high pass, mid pass, low pass) used were (5,1,0) and (3,2,0)
- Average out the Local Variability scores from the ten test sets.
- Plot out average Local Variability scores versus time.
- Run for all combinations of size/boundary and weightings.

Experiment #2: For each of the four time ranges in the long and low scenario

- Apply the profiling using the different number of bins / bin width on 0% infection rate test sets. The number of bins per day is 128, 256 and 512.
- Run the Haar Wavelet algorithm on the three different profiles using the respective boundaries and weight combinations. Refer to Table 5.4.
- Average out the Local Variability scores from the ten test sets.
- Apply each of the threshold methods to the average output.
- Apply the two evaluation techniques to each of the threshold methods.
- Run for all 12 combinations of boundaries and weightings.

Experiment #3 used the dates of the short and low scenario and repeated Experiment #2.

Experiment #4 used the dates of the long and high scenario, infection rates of 75%, 100% and

120% and repeated Experiment #2. Experiment #5 used the dates of the short and high scenario, infection rates of 75%, 100% and 120% and repeated Experiment #2.

5.3.4 Hybrid PCA - Haar Wavelet Filtering Analysis Experiments

Experiment #1: Pure data across all dates

- Apply the PCA profiling on 0% infection rate test sets.
- Run the Hybrid algorithm on the created profile.
- Plot out average delta scores versus time.
- Repeat for each of the 16 time ranges.

Experiment #2: For each of the four dates in the long and low scenario

- Apply the profiling to 20% infection rate test sets.
- Run the Hybrid algorithm on the created profile.
- Average out the delta scores from the ten test sets.
- Apply each of the threshold methods to the average output.
- Apply the two evaluation techniques to each of the threshold methods.
- Run for all 12 combinations of boundaries and weightings.

Experiment #3 used the dates of the short and low scenario and repeated Experiment #2. Experiment #4 used the dates of the long and high scenario, infection rates of 75%, 100% and 120% and repeated Experiment #2. Experiment #5 used the dates of the short and high scenario, infection rates of 75%, 100% and 120% and repeated Experiment #2.

6.0 Experimental Results and Discussion

This chapter presents and discusses the results of the anomaly detection experiments conducted using the Kyoto2006+ data source. Experimental results and discussions for three different methods – Modified Principle Component Analysis, Wavelet Analysis, and the Hybrid approach – are presented in this chapter. Those three methods and the experimental procedure has been outlined in Chapter 5.

6.1 Evaluation of Modified Principle Component Analysis

This section outlines the results obtained from running experiments with the Modified Principle Component Analysis (PCA) approach outlined in section 5.3.2 and concludes with a brief discussion.

6.1.1 Modified PCA and Pure Data

This section examines the results of the Modified PCA approach applied to a dataset with no infections. Figure 6.1 shows the SPE scores of January 6, 2009 on all ten test sets. This date is representative of the 16 selected dates. Even though there are ten test sets, there seem to be test sets which produce an extremely high score compared to the other observations. The value is so high that the other plots visually flatten to zero. Upon removing these test sets, as shown in Figure 6.2, all the remaining test sets behave alike. Examining the observations for the time bins with extremely high scores, it was observed that there were values 15 times larger than the particular feature mean. For the remaining experiments, the PCA data profiles were filtered prior to analysis. This filter removes time slices that have a feature value 15 times greater than the feature mean and flags them as potentially anomalous. Figure 6.3 shows the output of the

Modified PCA approach applied to the filtered January 6, 2009 data. After applying the filter, all ten sets have a similar pattern. As a result, it is possible to create an average curve of the ten test sets which is representative of the entire date. This average is referred to as a footprint. Applying the filtering to the 16 dates, it is observed that each date has a distinct footprint. As shown in Figures 6.1, 6.2, and 6.3, regardless of the threshold technique, the algorithm suggests that there are anomalies. An anomaly is present if the score is a visually high value.

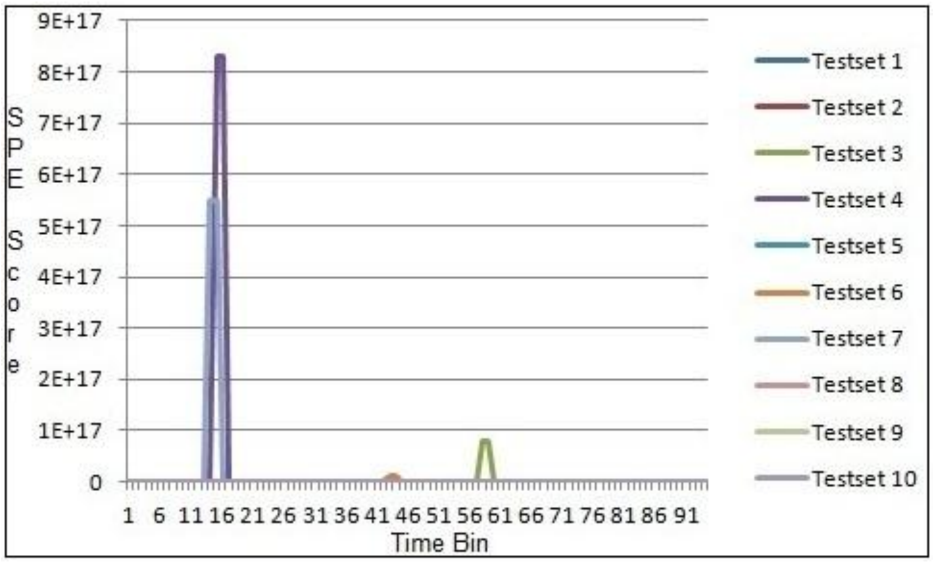


Figure 6.1: SPE scores from Unfiltered Modified PCA analysis on January 6, 2009 data

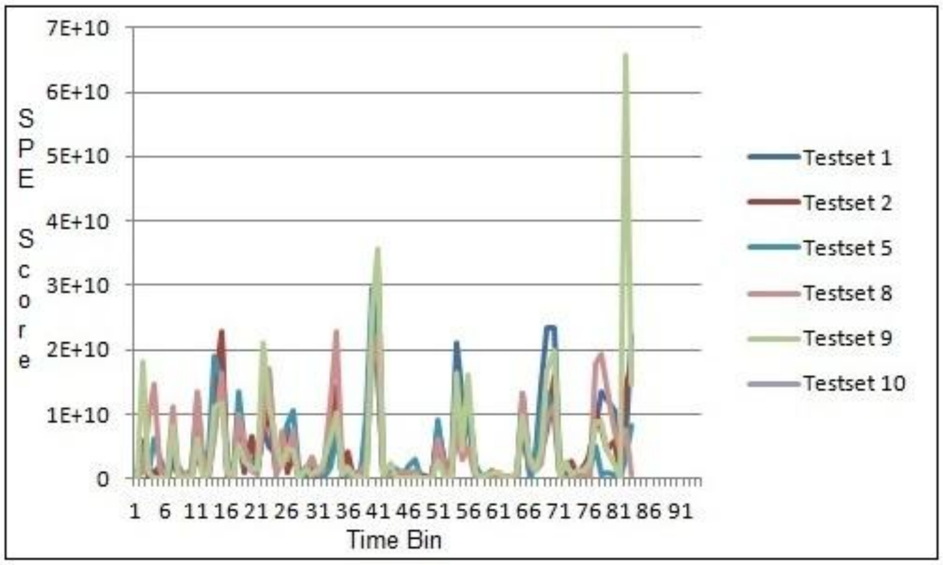


Figure 6.2: Subset of SPE scores from Modified PCA analysis on January 6, 2009 data

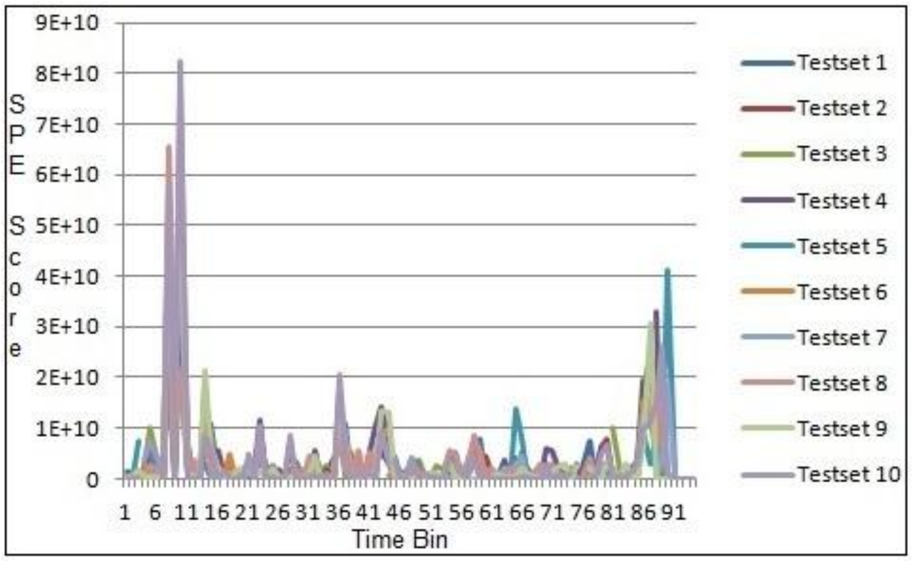


Figure 6.3: SPE scores from Filtered Modified PCA analysis on January 6, 2009 data

6.1.2 Short and Low Infection

This section examines the results of the Modified PCA experiments performed on a low infection rate in a short period of time. The four dates for this particular scenario, outlined in Table 5.2, all had consistent behaviour. As a result, for simplicity, only March 8, 2009 is discussed. Figure 6.4 illustrates the footprints observed with default parameters for the 5%, 10% and 20% infection ratio test sets (See Table 5.4 for default parameters). The infected time bins were bins 88, 89 and 90. None of these plots have a spike during the infected time bins, as these bins were filtered out, using the 15 times the column mean approach. Table 6.1 lists the hit/miss scores, and the hit/miss counts. The default threshold method generates a miss rate of about 15%. For other threshold methods, the results obtained for miss count range from 25% to 35%, see Section 6.1.8 for details.

Infection Rate	Score	Hit Count (% Hit)	Miss Count (% Miss)
5%	-2	83 (86%)	13 (14%)
10%	-2	83 (86%)	13 (14%)
20%	-2	84 (87%)	12 (13%)

Table 6.1: March 8, 2009 - PCA experiment results

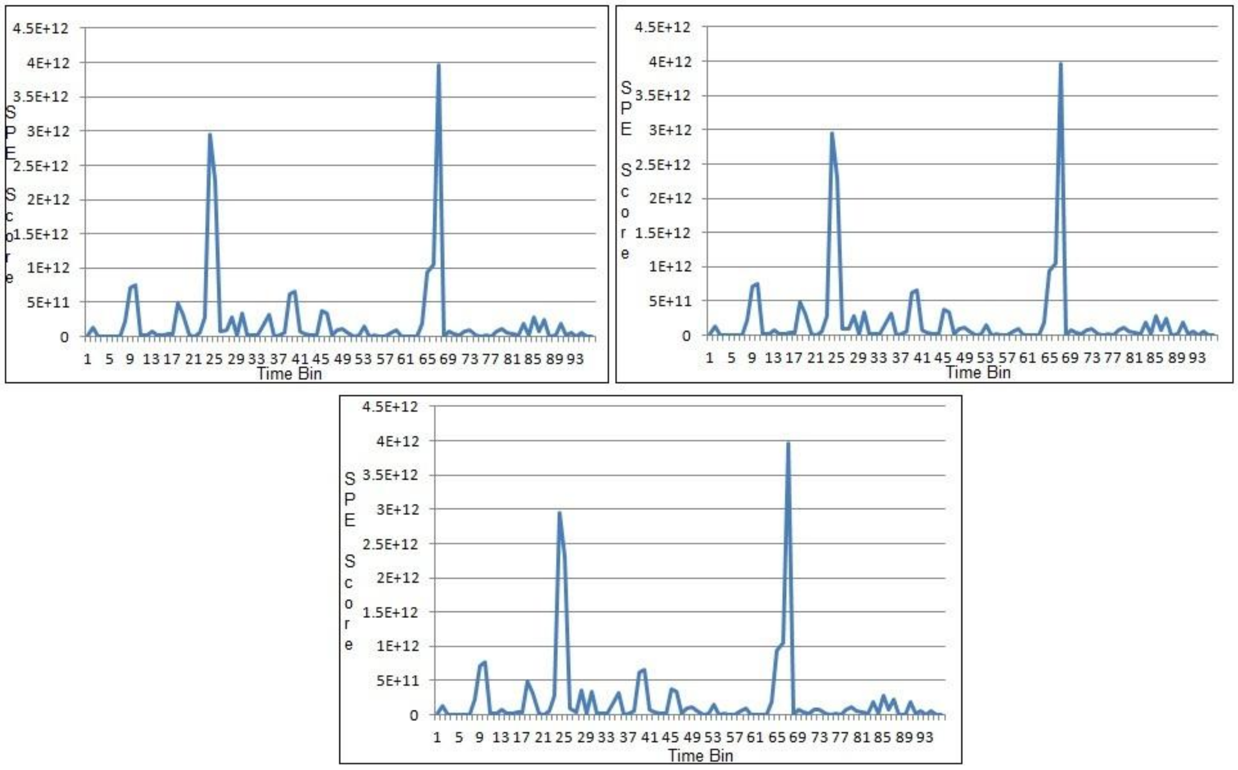


Figure 6.4: March 8, 2009 - PCA SPE scores 5% (top left), 10% (top right), 20% (bottom) results

6.1.3 Long and Low Infection

This section examines the results of the Modified PCA approach experiments performed on a low infection rate in a long period of time. The four dates for this particular scenario, outlined in Table 5.2, all had consistent behaviour. As a result, for simplicity, only May 28, 2009 is discussed. Figure 6.5 illustrates the footprints observed under default parameters on the 5%, 10% and 20% infection ratio test sets. The infected time bins were bins 61 - 71. As before, none of these plots had a spike during the infected time bins. A number of the time bins were filtered out. Table 6.2 lists the hit/miss scores and the hit/miss counts. The default threshold method generates a miss rate of about 20%. The other threshold methods range from 25% to 40%, see Section 6.1.8 for details.

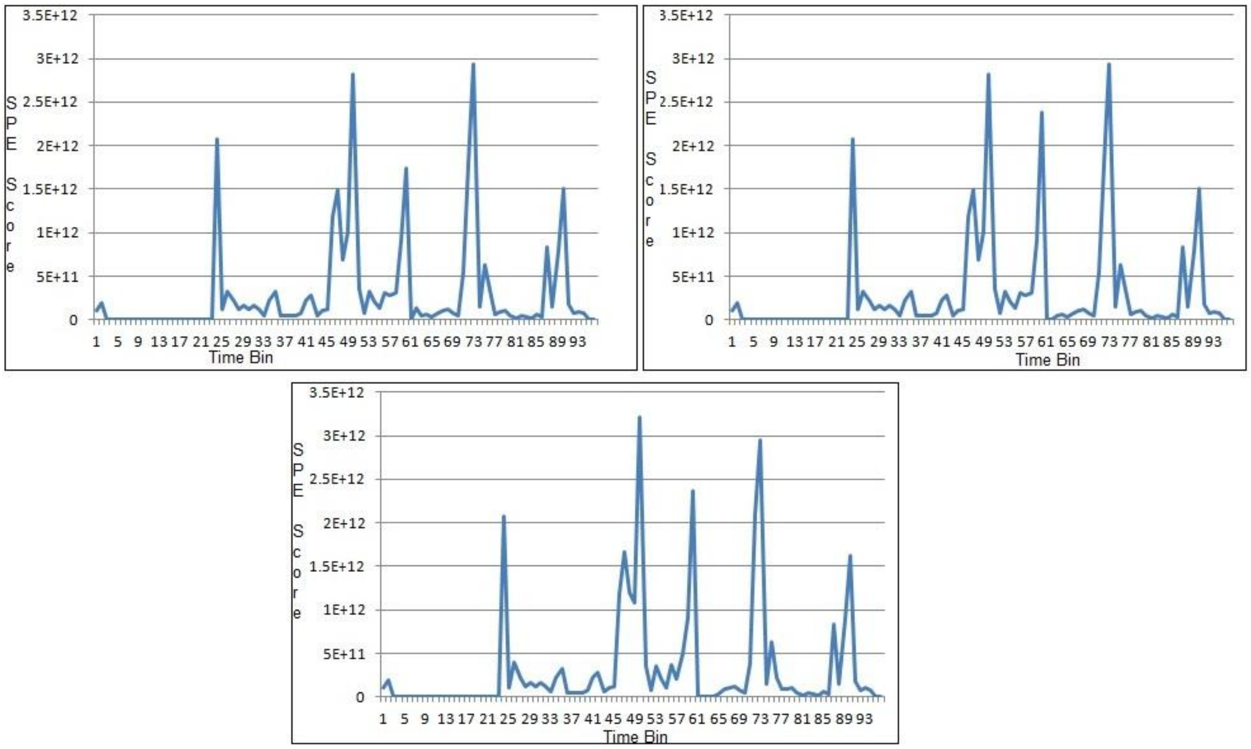


Figure 6.5: May 28, 2009 - PCA SPE scores 5% (top left), 10% (top right), 20% (bottom) results

Infection Rate	Score	Hit Count (% Hit)	Miss Count (% Miss)
5%	-2	75 (78%)	21 (22%)
10%	-2	76 (79%)	20 (21%)
20%	-2	77 (80%)	19 (20%)

Table 6.2: May 28, 2009 - PCA experiment results

6.1.4 Short and High Infection

This section examines the results of the Modified PCA approach experiments performed on a high infection rate in a short period of time. The four dates for this particular scenario, outlined in Table 5.2, all had consistent behaviour. As a result, for simplicity, only January 15, 2009 is discussed. Figure 6.6 illustrates the footprints observed under default parameters on the 75%, 100% and 120% infection ratio test sets. The infected time bins were bins 83 - 86. None of these plots had a spike during the infected time bins. These infected time bins were filtered out. Table 6.3 lists the hit/miss score and the hit/miss counts. The default threshold method generates a miss rate of about 30%. The other threshold methods range from 35% to 40%, see Section 6.1.8 for details.

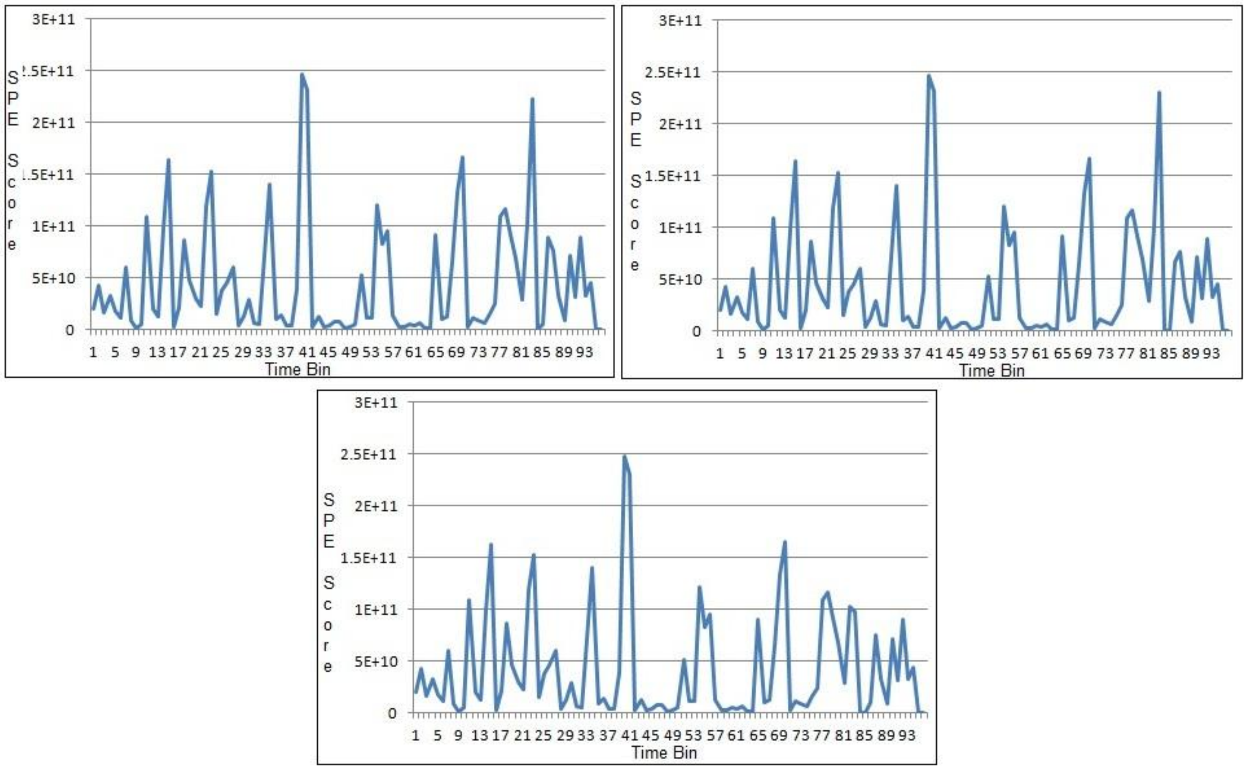


Figure 6.6: January 15, 2009 – PCA SPE scores 75% (top left), 100% (top right), 120% (bottom) results

Infection Rate	Score	Hit Count (% Hit)	Miss Count (% Miss)
75%	0	66 (69%)	30 (31%)
100%	0	67 (70%)	29 (30%)
120%	0	66 (69%)	30 (31%)

Table 6.3: January 15, 2009 - PCA experiment results

6.1.5 Long and High Infection

This section examines the results of the modified PCA algorithm experiments performed on a high infection rate in a long period of time. The four dates for this particular scenario, outlined in Table 5.2, all had consistent behaviour. As a result, for simplicity, only June 10, 2009 is discussed. Figure 6.7 illustrates the footprints observed under default parameters on the 75%, 100% and 120% infection ratio test sets. Unlike previous scenarios, the algorithm was able to determine the anomalous regions. Time bins 22 - 32 are the infected time bins. Some of the time bins were filtered out, and a major spike highlighted the remainder. Table 6.4 outlines the hit/miss scores and the hit/miss counts measured using the default threshold. The threshold method generates a miss rate of about 20%. The other threshold methods range from 15% to 35%, see Section 6.1.8 for details.

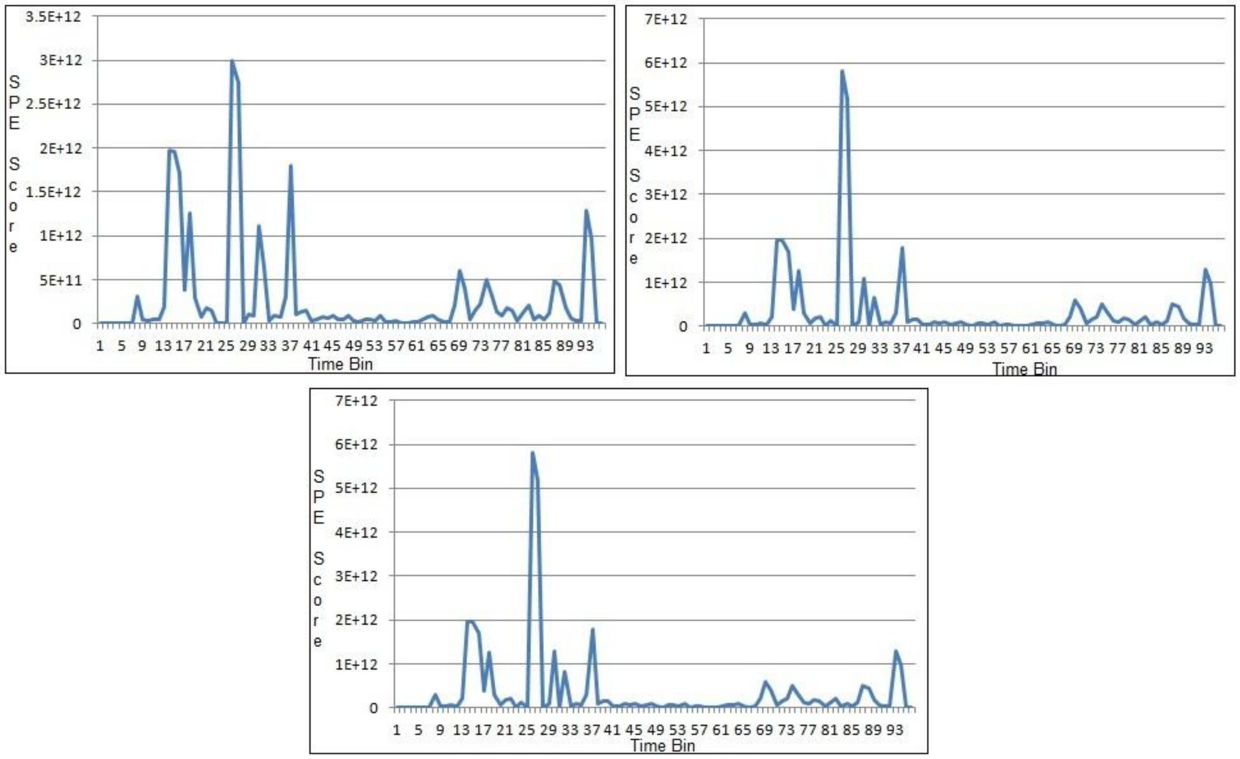


Figure 6.7: June 10, 2009 - PCA SPE scores 75% (top left), 100% (top right), 120% (bottom) results

Infection Rate	Score	Hit Count (% Hit)	Miss Count (% Miss)
75%	0	76 (79%)	20 (21%)
100%	0	80 (83%)	16 (17%)
120%	0	78 (81%)	18 (19%)

Table 6.4: June 10, 2009 - PCA experiment results

6.1.6 Varying number of Time Shifts

The time shifts were introduced by [6] in order to account for temporal change from [16]. Experiments were performed varying the number of time shifts and the results listed in tables 6.5

– 6.8 summarize the measurements for each scenario on the specified dates. Increasing the number of time shifts never improved the score. The hit/miss scores either remained the same or decreased. Increasing the time shifts increased the miss count. This direct relationship showed that, using the Kyoto2006+ dataset, one time shift was optimal.

Infection Rate	Time Shifts	Score	Hit Count	Miss Count
5%	1	-2	83	13
	2	-2	78	18
	3	-6	73	23
10%	1	-2	83	13
	2	-2	78	18
	3	-6	74	22
20%	1	-2	84	12
	2	-2	78	18
	3	-6	71	25

Table 6.5: March 8, 2009 - PCA Short and Low infection varying time shift results

Infection Rate	Time Shifts	Score	Hit Count	Miss Count
5%	1	-2	75	21
	2	-4	74	22
	3	-6	71	25
10%	1	-2	76	20
	2	-4	73	23
	3	-6	71	25
20%	1	-2	77	19
	2	-4	75	21
	3	-6	73	23

Table 6.6: May 28, 2009 - PCA Long and Low infection varying time shift results

Infection Rate	Time Shifts	Score	Hit Count	Miss Count
75%	1	0	66	30
	2	0	63	33
	3	0	60	36
100%	1	0	67	29
	2	0	64	32
	3	0	63	33
120%	1	0	66	30
	2	0	67	29
	3	0	61	35

Table 6.7: January 15, 2009 - PCA Short and High infection varying time shift results

Infection Rate	Time Shifts	Score	Hit Count	Miss Count
75%	1	0	76	20
	2	-1	65	31
	3	-3	60	36
100%	1	0	80	16
	2	-1	81	15
	3	-3	66	33
120%	1	0	78	18
	2	-1	79	17
	3	-3	63	33

Table 6.8: June 10, 2009 - PCA Long and High infection varying time shift results

6.1.7 Varying Selection Criteria

The selection criterion determines what and how many components are used in analysis. Experiments varying the selection criterion between three options were performed. As outlined in section 2.4, the subspace approach, the variance approach and the top one approach were used. The results listed in tables 6.9 – 6.12 summarize the evaluation measurements for selection criteria on the specified dates. Selecting any component with a variance of one or greater performed the worst. It was observed that this approach consistently had the worse hit/miss score and hit/miss count. Selecting only the top component performed better than the variance approach in most cases; however, the subspace approach performed the best in all cases.

Infection Rate	Measurement	Subspace selection	Variance > 1 selection	Top K = 1 components
5%	Score	-2	-2	-2
	Hit Count	83	59	83
	Miss Count	13	37	13
10%	Score	-2	-2	-2
	Hit Count	83	58	83
	Miss Count	13	38	13
20%	Score	-2	-2	-2
	Hit Count	84	57	84
	Miss Count	12	39	12

Table 6.9: March 8, 2009 – PCA Short and Low infection varying selection criteria results

Infection Rate	Measurement	Subspace selection	Variance > 1 selection	Top K = 1 components
5%	Score	-2	-2	-2
	Hit Count	75	61	75
	Miss Count	21	35	21
10%	Score	-2	-2	-2
	Hit Count	76	63	76
	Miss Count	20	33	20
20%	Score	-2	-2	-2
	Hit Count	77	60	77
	Miss Count	19	36	19

Table 6.10: May 28, 2009 – PCA Long and Low infection varying selection criteria results

Infection Rate	Measurement	Subspace selection	Variance > 1 selection	Top K = 1 components
75%	Score	0	0	0
	Hit Count	66	55	67
	Miss Count	30	41	29
100%	Score	0	1	0
	Hit Count	67	55	68
	Miss Count	29	41	28
120%	Score	0	0	0
	Hit Count	66	57	67
	Miss Count	30	39	29

Table 6.11: January 15, 2009 – PCA Short and High infection varying selection criteria results

Infection Rate	Measurement	Subspace selection	Variance > 1 selection	Top K = 1 components
75%	Score	0	2	0
	Hit Count	76	64	76
	Miss Count	20	32	20
100%	Score	0	3	0
	Hit Count	80	67	79
	Miss Count	16	29	17
120%	Score	0	4	0
	Hit Count	78	68	78
	Miss Count	18	28	18

Table 6.12: June 10, 2009 – PCA Long and High infection varying selection criteria results

6.1.8 Threshold Techniques and Modified PCA

Each of the experiments performed used four different threshold techniques to automatically identify anomalous time bins. The results listed in tables 6.13 – 6.16 summarize the evaluation measurements for threshold techniques on the specified dates. The weighted average and moving average consistently had a lower SPE score and higher miss count. The moving average without outliers performed the best in scenarios with low infection rates. In scenarios with higher infection rates, the static average performed slightly better than the moving average without outliers.

Infection Rate	Measurement	Static Average	Moving Average	Moving Average without outliers	Weighted Average
5%	Score	-2	-2	-2	-2
	Hit Count	76	68	83	74
	Miss Count	20	28	13	22
10%	Score	-2	-2	-2	-2
	Hit Count	76	68	83	74
	Miss Count	20	28	13	22
20%	Score	-2	-2	-2	-2
	Hit Count	76	68	84	74
	Miss Count	20	28	12	22

Table 6.13: March, 8, 2009 - PCA Short and Low threshold results

Infection Rate	Measurement	Static Average	Moving Average	Moving Average without outliers	Weighted Average
5%	Score	-2	-26	-2	-2
	Hit Count	74	62	75	75
	Miss Count	22	34	21	21
10%	Score	-2	-26	-2	-2
	Hit Count	74	62	76	75
	Miss Count	22	34	20	21
20%	Score	-2	-26	-2	-2
	Hit Count	75	61	77	76
	Miss Count	21	35	19	20

Table 6.14: May 28, 2009 - PCA Long and Low threshold results

Infection Rate	**Measurement**	**Static Average**	**Moving Average**	**Moving Average without outliers**	**Weighted Average**
75%	Score	0	0	0	0
	Hit Count	64	63	66	61
	Miss Count	32	33	30	35
100%	Score	1	0	0	1
	Hit Count	64	64	67	62
	Miss Count	32	32	29	34
120%	Score	0	0	0	0
	Hit Count	63	62	66	63
	Miss Count	33	34	30	33

Table 6.15: January 15, 2009 - PCA Short and High threshold results

Infection Rate	**Measurement**	**Static Average**	**Moving Average**	**Moving Average without outliers**	**Weighted Average**
75%	Score	1	1	0	1
	Hit Count	76	66	76	71
	Miss Count	20	30	20	25
100%	Score	3	0	0	1
	Hit Count	80	66	80	72
	Miss Count	16	30	16	24
120%	Score	4	1	0	1
	Hit Count	81	66	78	72
	Miss Count	15	30	18	24

Table 6.16: June 10, 2009 - PCA Long and High threshold results

6.1.9 Summary and Discussion

The major advantage of the Modified Principle Component Analysis (PCA) approach was that it considered all of the features at once. Regardless of how many features, all were considered in analysis and a single score was calculated for each time bin. There were two user defined parameters: the number of time shifts and the component selection criteria.

The number of time shifts. In [6], the authors showed that four or more time shifts do not have any significant impact. Our research examined one, two and three time shifts. The number of time shifts affects how much the algorithm considers changes in time. For instance, a large number time shifts will show a great deal of change in a feature, and changes will be very common. The algorithm will become less sensitive to change [6]. Conversely, no time shifts fail to take into account any the temporal change. There is a need for modest consideration of how the data changes over the next few observations. Our analysis concluded that one time shift performed the best. This suggests that there is a great deal of change between time bins in the Kyoto2006+ and re-enforces the need for considering temporal changes. As more time shifts were applied, the algorithm was less sensitive to the change and performed worse.

Component Selection Criteria. The Modified PCA approach determines a set of ranked components and only a subset of are used in anomaly detection. The number of selected components affects subspace creation and the SPE score calculation. A selection of a large number of components converges close to the original feature space. It would not unlock and new patterns or provide an optimizations. Overly simple components would miss out on important information and not be very meaningful. Ideally, a handful of independent components are needed to form the anomalous space. In our analysis, it was observed that in most cases, the Top K=1 approach and the subspace approach had the same results. This suggests the subspace approach used one, or close to one, components to build the subspace. It can be concluded that there are a number of redundant features in the Kyoto2006+ dataset. A few components describe the entire dataset. Considering all features with variance greater than one builds involves a larger number of components, and explains why this approach performed the worst.

A number of test sets were built using different days in the week, and different infection ratios. Regardless of the infection rate or day in week, the output SPE scores of the modified PCA algorithm highlighted time bins. The output SPE score was the Euclidean Magnitude, and is directly related to the size of the projected. The SPE score were always large and is due to the nature of the observations, and the measurement. This can be illustrated in the 0% infection rate experiments. The algorithm considered all features at once and was able to modestly pick out the infected time bins. It performed best in a situation where there are a lot of malicious flows over a long period of time.

6.2 Evaluation of Haar Wavelet Analysis

This section outlines the results obtained from running the experiments with the Haar Wavelet Analysis approach outlined in section 5.3.3 and concludes with a brief discussion.

6.2.1 Haar Wavelet Analysis and Pure data

This section examines the results of the Haar wavelet filtering analysis applied to a dataset with no infections. The experiments performed in this section examine all infection scenarios. Figure 6.8 shows the local variability scores for each TCP feature and Figure 6.9 shows local variability scores for each OTHER feature of all test sets on January 6, 2009. This date is representative of the 16 selected dates. It can be observed that, for a few features, there exists a test set with much higher local variability values than the rest of the test sets. Similar to the Modified PCA approach, it is determined that an observation 15 times larger than the mean of that feature. Upon removing the test sets, a footprint was noticed across the remaining test sets. Unlike the Modified PCA approach, the local variability score did not have an extreme difference in values and did not flatten the rest of the plot to zero. Also, the average calculations are not affected. The Haar Wavelet Filtering analysis does not use initial filtering. The average local variability is used in analysis and is referred to as the footprint. As shown in Figure 6.10

and Figure 6.11, regardless of the threshold technique, the algorithm will suggest that there are anomalies present.

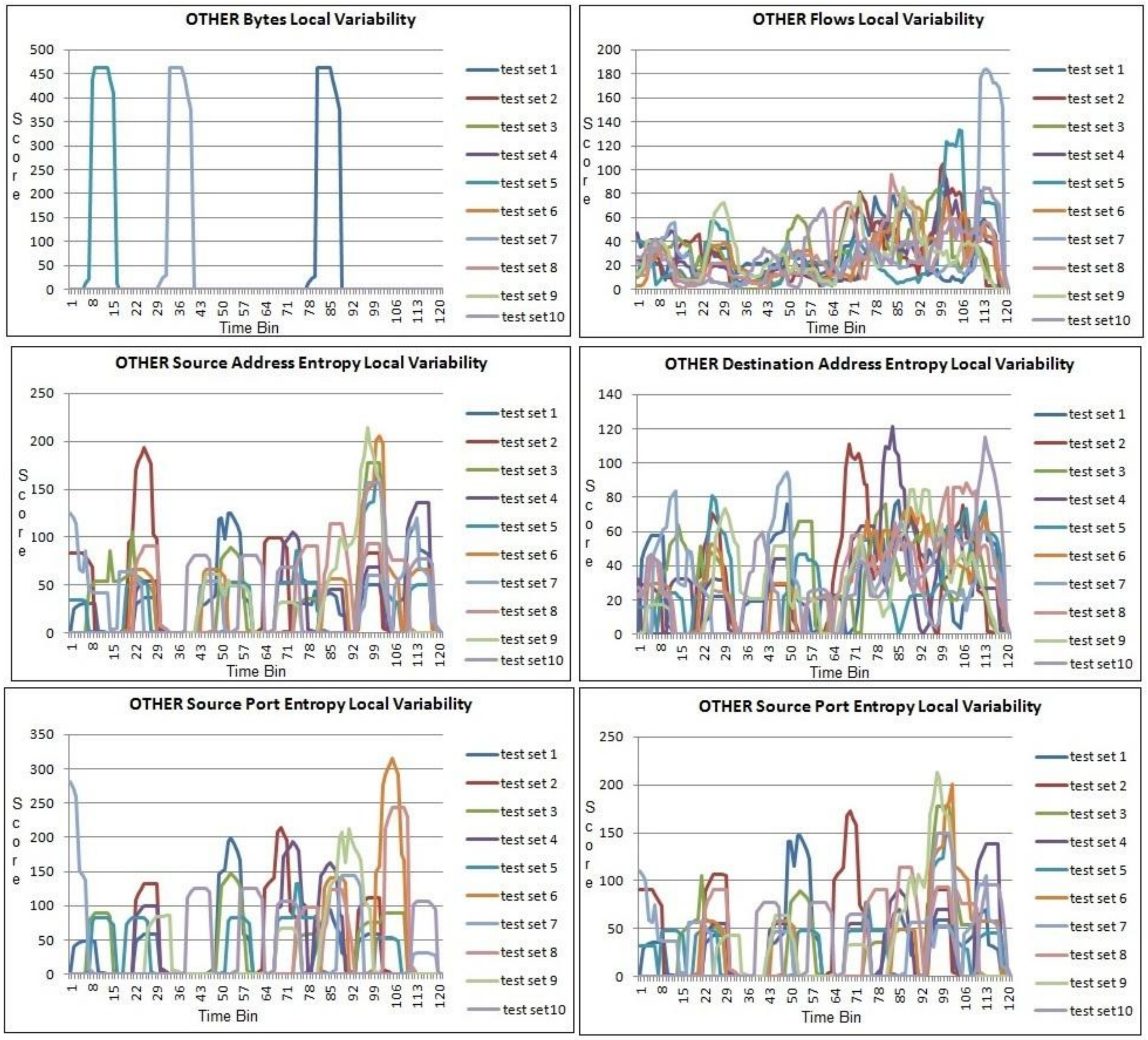


Figure 6.8: January 6, 2009 - OTHER Wavelet flows Local Variability Scores

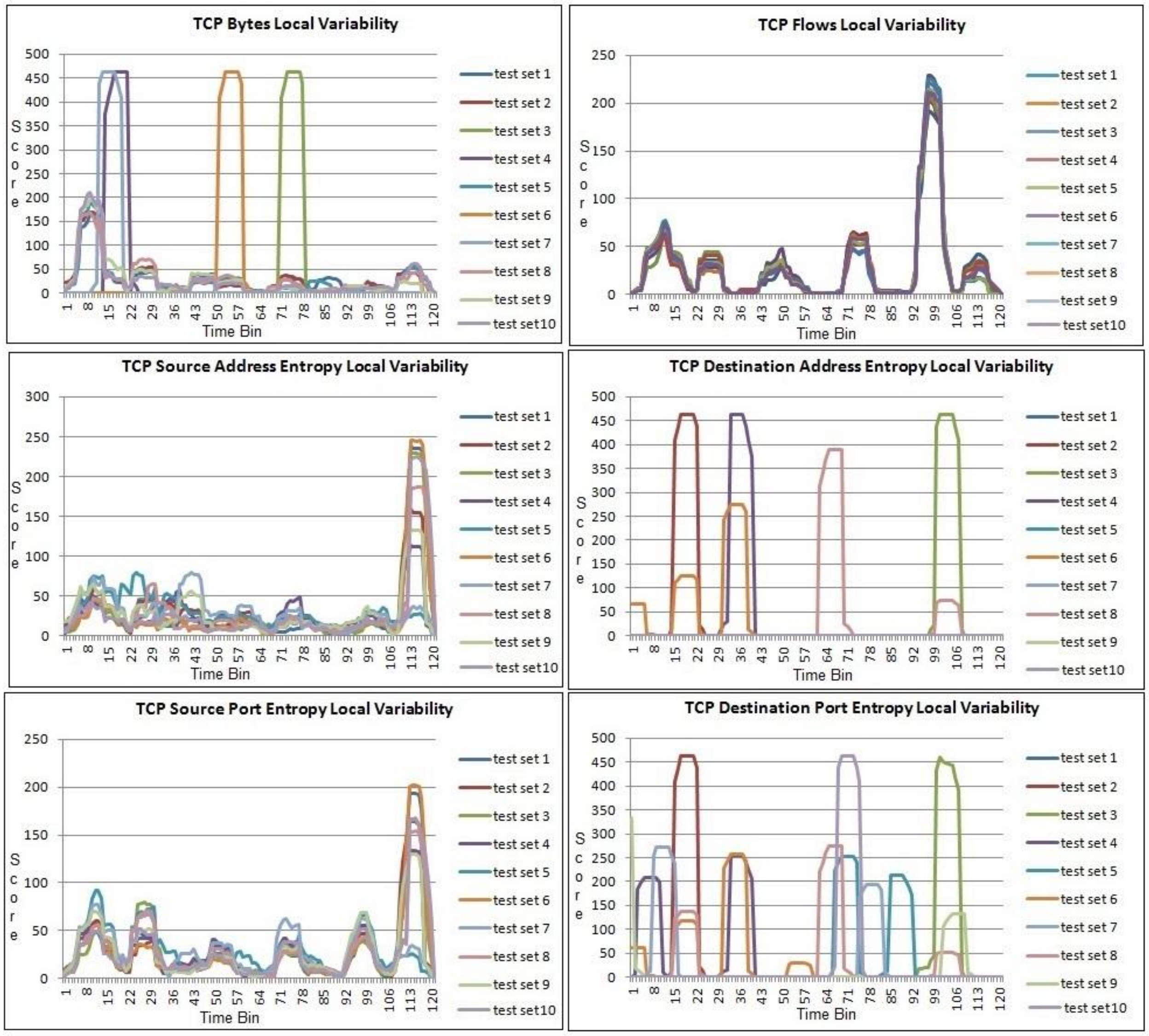


Figure 6.9: January 6, 2009 - TCP Wavelet flows Local Variability Scores

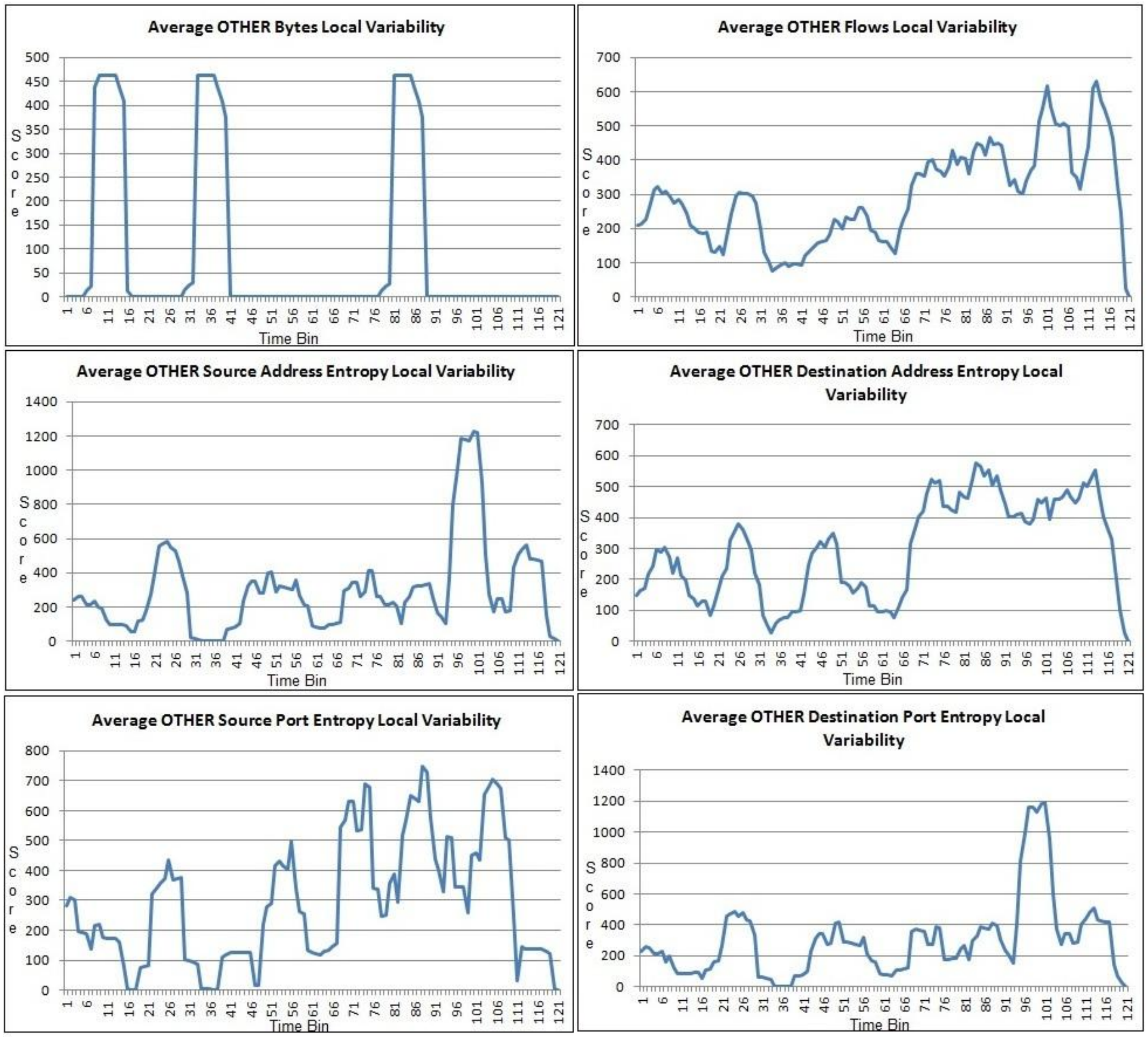


Figure 6.10: January 6, 2009 - Average OTHER Wavelet flows Local Variability Scores

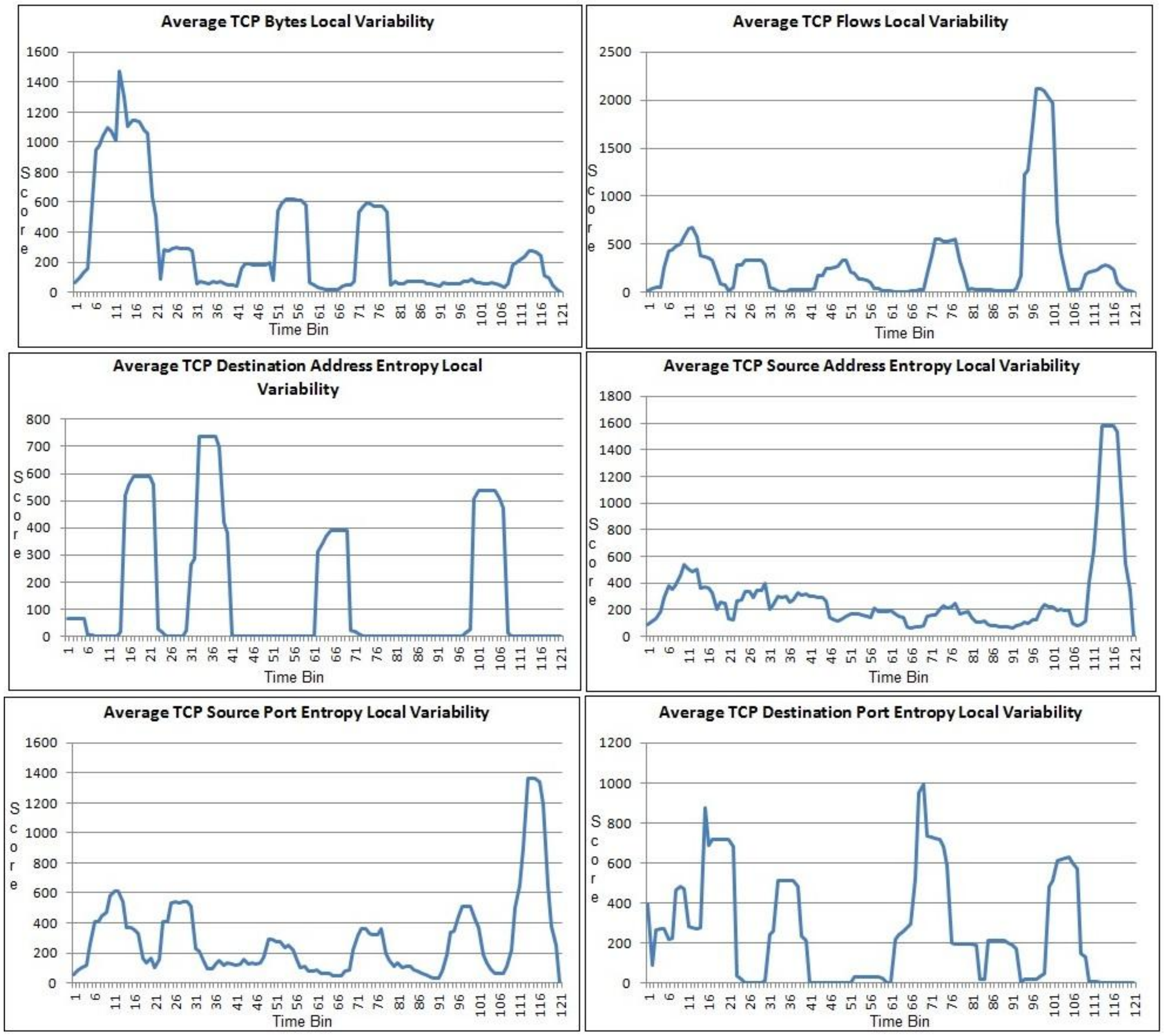


Figure 6.11: January 6, 2009 - Average TCP Wavelet flows Local Variability Scores

6.2.2 Short and Low Infection

This section examines the results of the Haar Wavelet experiments performed on a low infection rate in a short period of time. The four dates for this particular scenario, outlined in Table 5.2, all had consistent behaviour. As a result, for simplicity, only March 8, 2009 is discussed. Figure 6.12 and Figure 6.13 illustrate the footprints observed with default parameters for the TCP and OTHER features for the 5% infection ratio (See Table 5.4 for a list of default conditions). The experiments were conducted on test sets of 5%, 10% and 20% infection ratios. The infected time bins were bins 116, 117 and 118. A number of the feature plots depicted spikes during the infected times. Table 6.17 lists the hit/miss scores and the hit/miss counts measured for each feature. The default threshold method determines a 19% miss rate in the best feature and about a 64% miss rate in the worst performing feature. The other threshold methods range from 5% to 65%, see Section 6.2.8 for details.

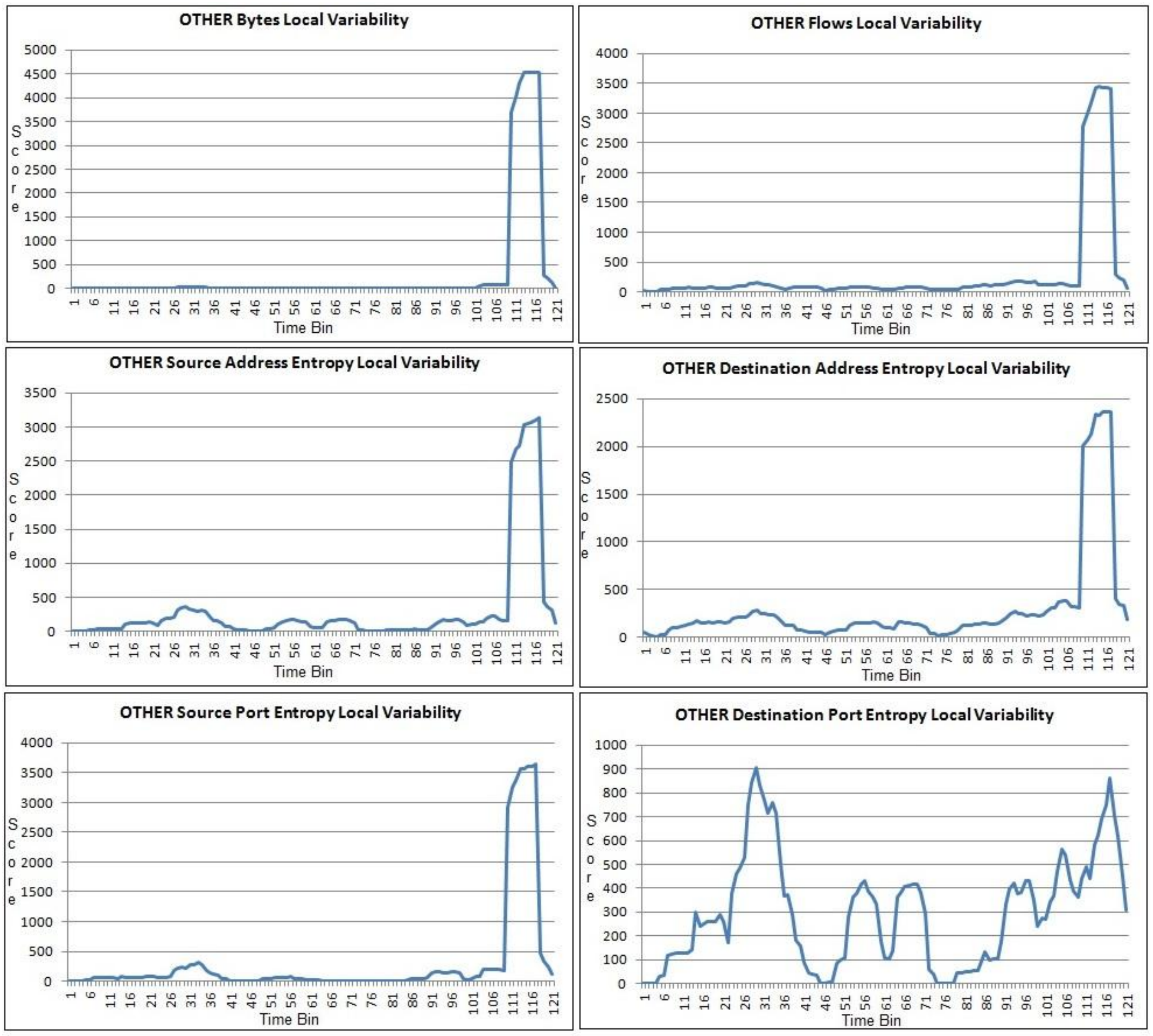


Figure 6.12: March 8, 2009 - Average OTHER Wavelet Local Variability Scores with 5% infection rate

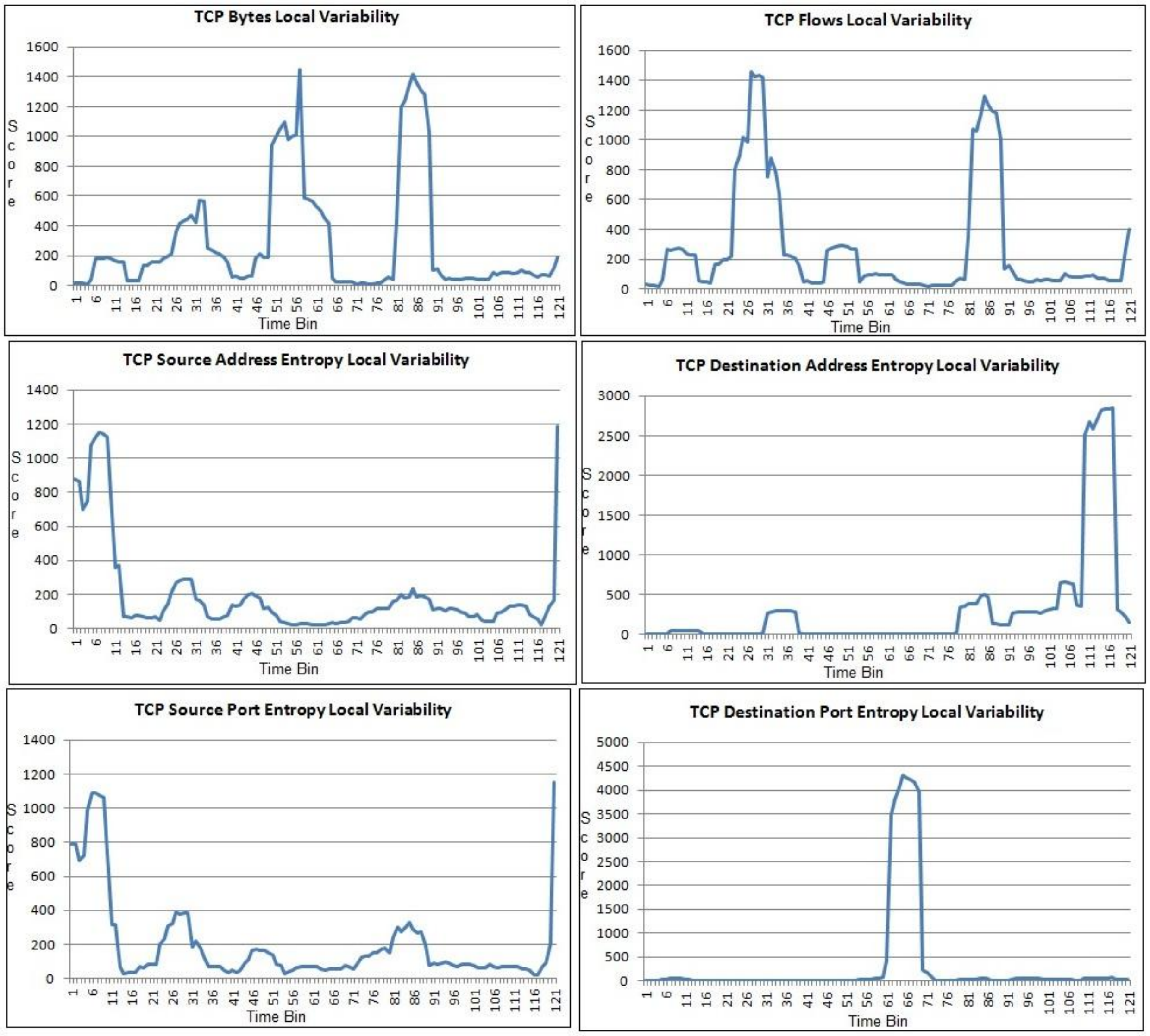


Figure 6.13: March 8, 2009 - Average TCP Wavelet Local Variability Scores with 5% infection rate

Infection Rate	**Feature**	**Score**	**Hit Count (% Hit)**	**Miss Count (% Miss)**
5%	TCP Bytes	-12	51 (53%)	45 (47%)
	TCP Flows	-12	54 (56%)	42 (44%)
	TCP Source Address Entropy	-10	46 (48%)	50 (52%)
	TCP Destination Address Entropy	-9	49 (51%)	47 (49%)
	TCP Source Port Entropy	-10	51 (53%)	45 (47%)
	TCP Destination Port Entropy	-14	68 (71%)	28 (29%)
	OTHER Bytes	1	72 (75%)	24 (25%)
	OTHER Flows	-11	27 (28%)	69 (62%)
	OTHER Source Address Entropy	-13	24 (25%)	72 (75%)
	OTHER Destination Address Entropy	-9	15 (16%)	81 (84%)
	OTHER Source Port Entropy	-13	39 (41%)	57 (43%)
	OTHER Destination Port Entropy	-13	21 (22%)	75 (78%)
10%	TCP Bytes	-12	50 (52%)	46 (48%)
	TCP Flows	-12	53 (55%)	43 (45%)
	TCP Source Address Entropy	-10	45 (47%)	51 (53%)
	TCP Destination Address Entropy	-8	51 (53%)	45 (47%)
	TCP Source Port Entropy	-9	53 (55%)	43 (45%)
	TCP Destination Port Entropy	-14	67 (70%)	29 (30%)
	OTHER Bytes	1	71 (74%)	25 (26%)
	OTHER Flows	-11	25 (26%)	71 (74%)
	OTHER Source Address Entropy	-13	25 (26%)	71 (74%)
	OTHER Destination Address Entropy	-9	16 (17%)	80 (83%)
	OTHER Source Port Entropy	-13	38 (40%)	58 (60%)
	OTHER Destination Port Entropy	-12	24 (25%)	72 (75%)
20%	TCP Bytes	-12	49 (51%)	47 (49%)
	TCP Flows	-12	52 (54%)	44 (46%)
	TCP Source Address Entropy	-9	46 (48%)	50 (52%)
	TCP Destination Address Entropy	-8	52 (54%)	44 (46%)
	TCP Source Port Entropy	-8	55 (57%)	41 (43%)
	TCP Destination Port Entropy	-14	66 (69%)	30 (31%)
	OTHER Bytes	2	71 (74%)	25 (26%)
	OTHER Flows	-11	23 (24%)	73 (76%)
	OTHER Source Address Entropy	-12	24 (25%)	72 (75%)
	OTHER Destination Address Entropy	-8	14 (15%)	82 (85%)
	OTHER Source Port Entropy	-12	36 (38%)	60 (62%)
	OTHER Destination Port Entropy	-11	25 (26%)	71 (74%)

Table 6.17: March 8, 2009 - Wavelet Experiment results

6.2.3 Long and Low Infection

This section examines the results of the Haar Wavelet experiments performed on a low infection rate in a long period of time. The four dates for this particular scenario, outlined in Table 5.2, all had consistent behaviour. As a result, for simplicity, only May 28, 2009 is discussed. Figure 6.14 and Figure 6.15 illustrate the footprints observed with default parameters for the TCP and OTHER features for the 5% infection ratio (See Table 5.4 for a list of default conditions). The experiments were conducted on test sets of 5%, 10% and 20% infection ratios. The infected time bins were 81 - 96. A number of the feature plots depicted spikes during the infected times. Table 6.18 lists the hit/miss scores and the hit/miss counts measured for each feature. The default threshold method determines a 24% miss rate in the best feature and about a 54% miss rate in the worst performing feature. The other threshold methods range from 3% to 55%, see Section 6.2.8 for details.

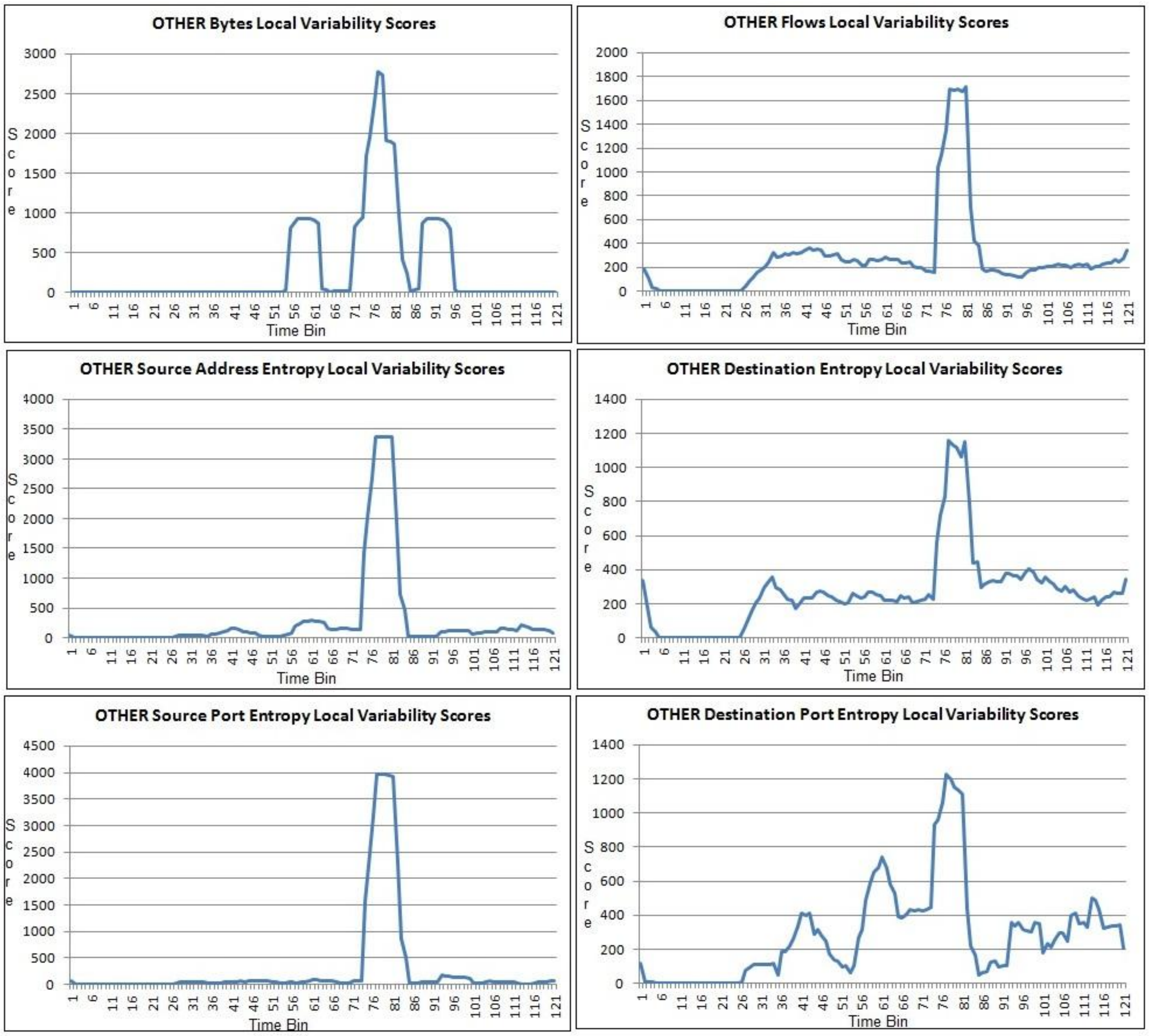


Figure 6.144: May 28, 2009 - Average OTHER Wavelet Local Variability Scores with 5% infection rate

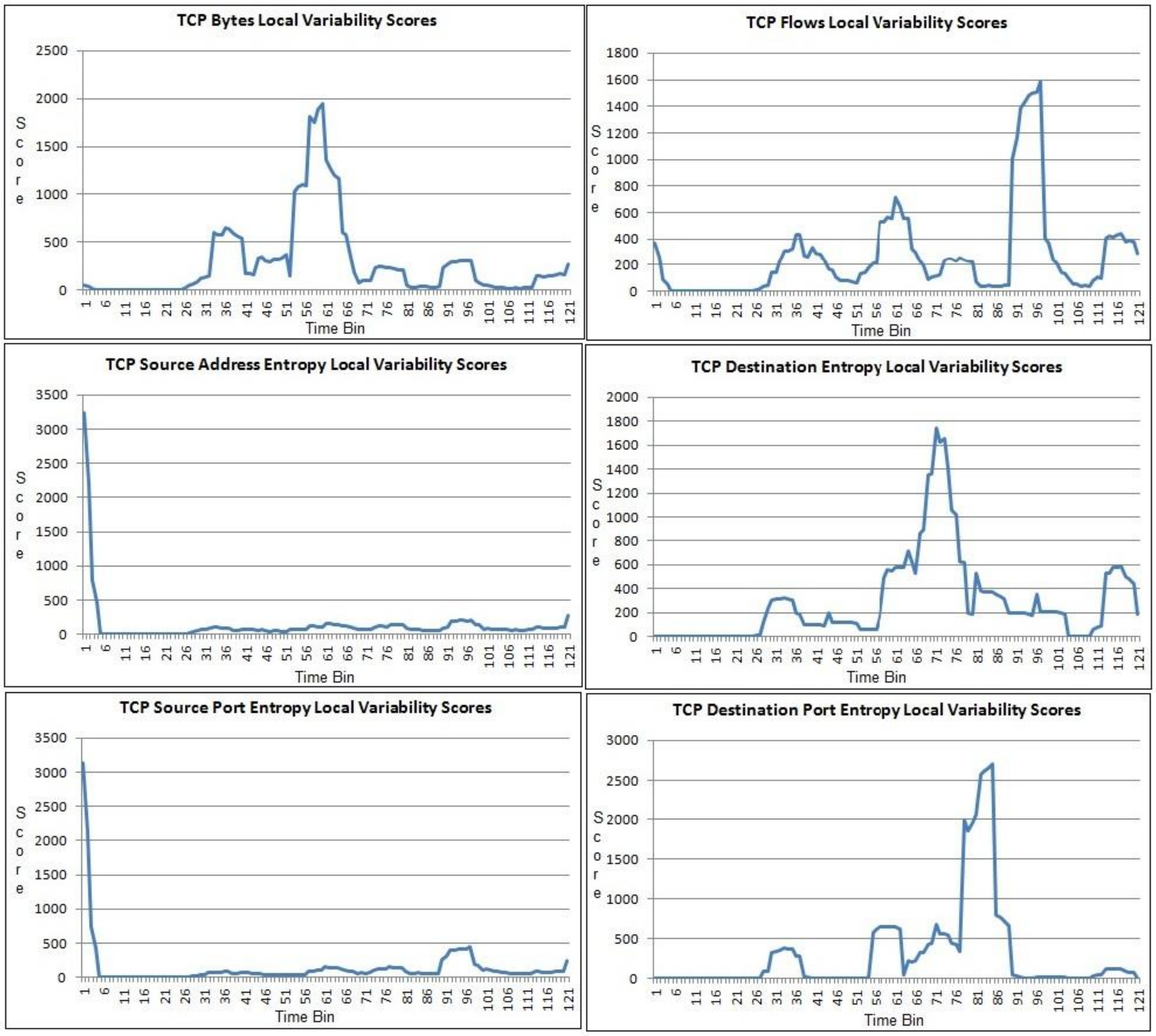


Figure 6.155: May 28, 2009 - Average TCP Wavelet Local Variability Scores with 5% infection rate

Infection Rate	**Feature**	**Score**	**Hit Count (% Miss)**	**Miss Count (% Miss)**
5%	TCP Bytes	-14	49 (51%)	47 (49%)
	TCP Flows	-10	46 (48%)	50 (52%)
	TCP Source Address Entropy	0	50 (52%)	46 (48%)
	TCP Destination Address Entropy	-14	53 (55%)	43 (45%)
	TCP Source Port Entropy	0	49 (51%)	47 (49%)
	TCP Destination Port Entropy	-12	64 (67%)	32 (33%)
	OTHER Bytes	-11	70 (73%)	26 (27%)
	OTHER Flows	-13	39 (41%)	57 (59%)
	OTHER Source Address Entropy	-11	71 (74%)	25 (76%)
	OTHER Destination Address Entropy	-11	52 (54%)	44 (46%)
	OTHER Source Port Entropy	-11	58 (60%)	38 (40%)
	OTHER Destination Port Entropy	-11	44 (46%)	52 (54%)
10%	TCP Bytes	-14	48 (50%)	48 (50%)
	TCP Flows	-10	45 (47%)	51 (53%)
	TCP Source Address Entropy	0	49 (51%)	47 (49%)
	TCP Destination Address Entropy	-14	52 (54%)	44 (46%)
	TCP Source Port Entropy	0	48 (50%)	48 (50%)
	TCP Destination Port Entropy	-11	65 (68%)	31 (32%)
	OTHER Bytes	-10	70 (73%)	26 (27%)
	OTHER Flows	-12	54 (56%)	42 (44%)
	OTHER Source Address Entropy	-10	61 (64%)	35 (36%)
	OTHER Destination Address Entropy	-10	52 (54%)	44 (46%)
	OTHER Source Port Entropy	-10	59 (61%)	37 (39%)
	OTHER Destination Port Entropy	-11	42 (44%)	54 (56%)
20%	TCP Bytes	-14	46 (48%)	50 (52%)
	TCP Flows	-10	44 (46%)	52 (54%)
	TCP Source Address Entropy	0	47 (49%)	49 (51%)
	TCP Destination Address Entropy	-10	54 (56%)	42 (44%)
	TCP Source Port Entropy	0	46 (48%)	50 (52%)
	TCP Destination Port Entropy	-10	62 (65%)	34 (35%)
	OTHER Bytes	-11	67 (70%)	29 (30%)
	OTHER Flows	-10	40 (42%)	56 (58%)
	OTHER Source Address Entropy	-7	56 (58%)	40 (42%)
	OTHER Destination Address Entropy	-7	59 (61%)	37 (39%)
	OTHER Source Port Entropy	-7	54 (56%)	42 (44%)
	OTHER Destination Port Entropy	-7	51 (53%)	45 (47%)

Table 6.18: May 28, 2009 - Wavelet Experiment results

6.2.4 Short and High Infection

This section examines the results of the Haar Wavelet experiments performed on a high infection rate in a short period of time. The four dates for this particular scenario, outlined in Table 5.2, all had consistent behaviour. As a result, for simplicity, only January 15, 2009 is discussed. Figure 6.16 and Figure 6.17 illustrate the footprints observed with default parameters for the TCP and OTHER features for the 5% infection ratio (See Table 5.4 for a list of default conditions). The experiments were conducted on test sets of 75%, 100% and 120% infection ratios. The infected time bins were during bins 111, 112, 113 and 114. A number of the feature plots depicted spikes during the infected times. Table 6.19 lists the hit/miss scores and the hit/miss counts measured for each feature. The default threshold method determines an 18% miss rate in the best feature and about a 52% miss rate in the worst performing feature. The other threshold methods range from 4% to 57%, see Section 6.2.8 for details.

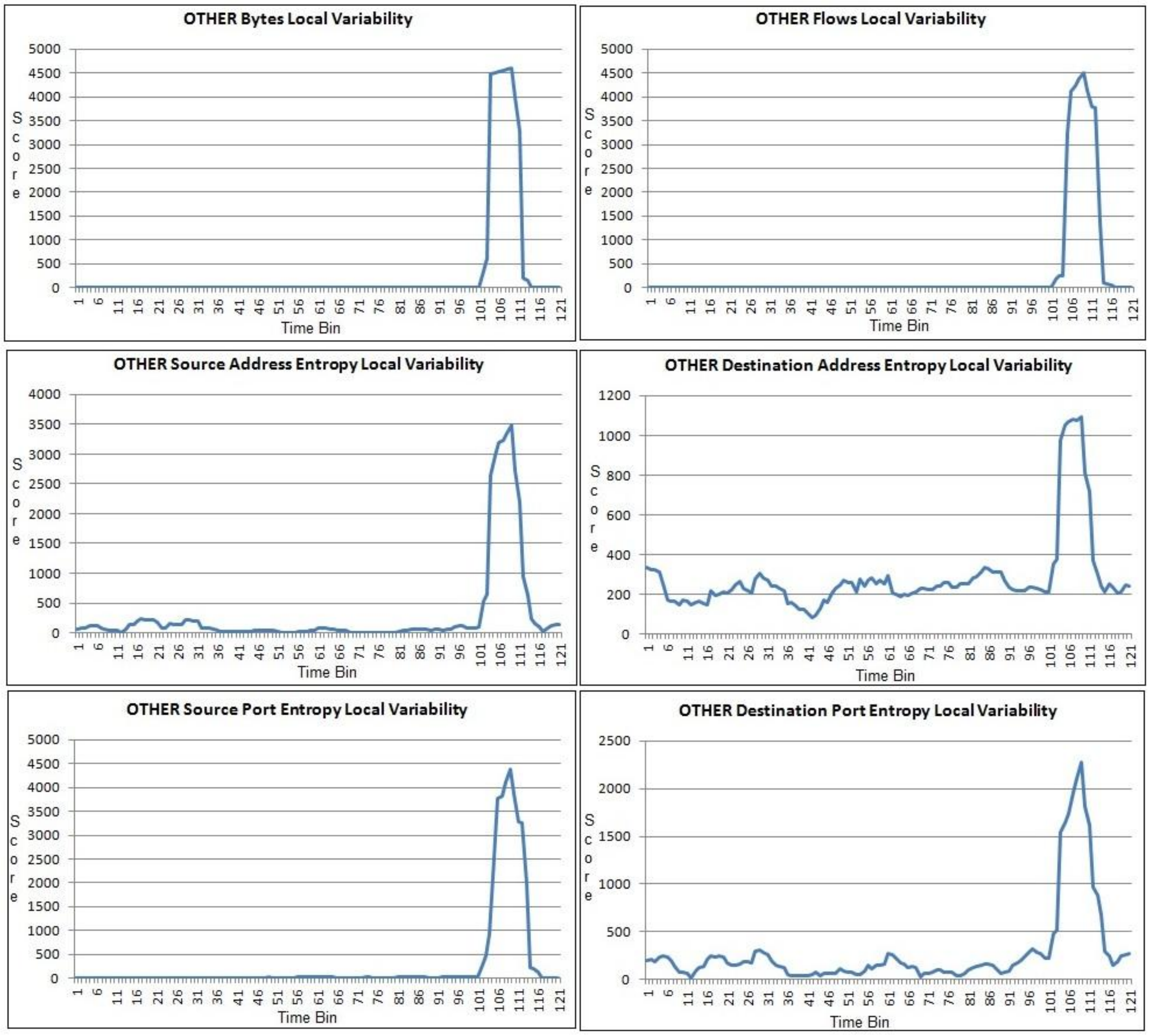


Figure 6.166: January 15, 2009 - Average OTHER Wavelet Local Variability Scores with 5% infection rate

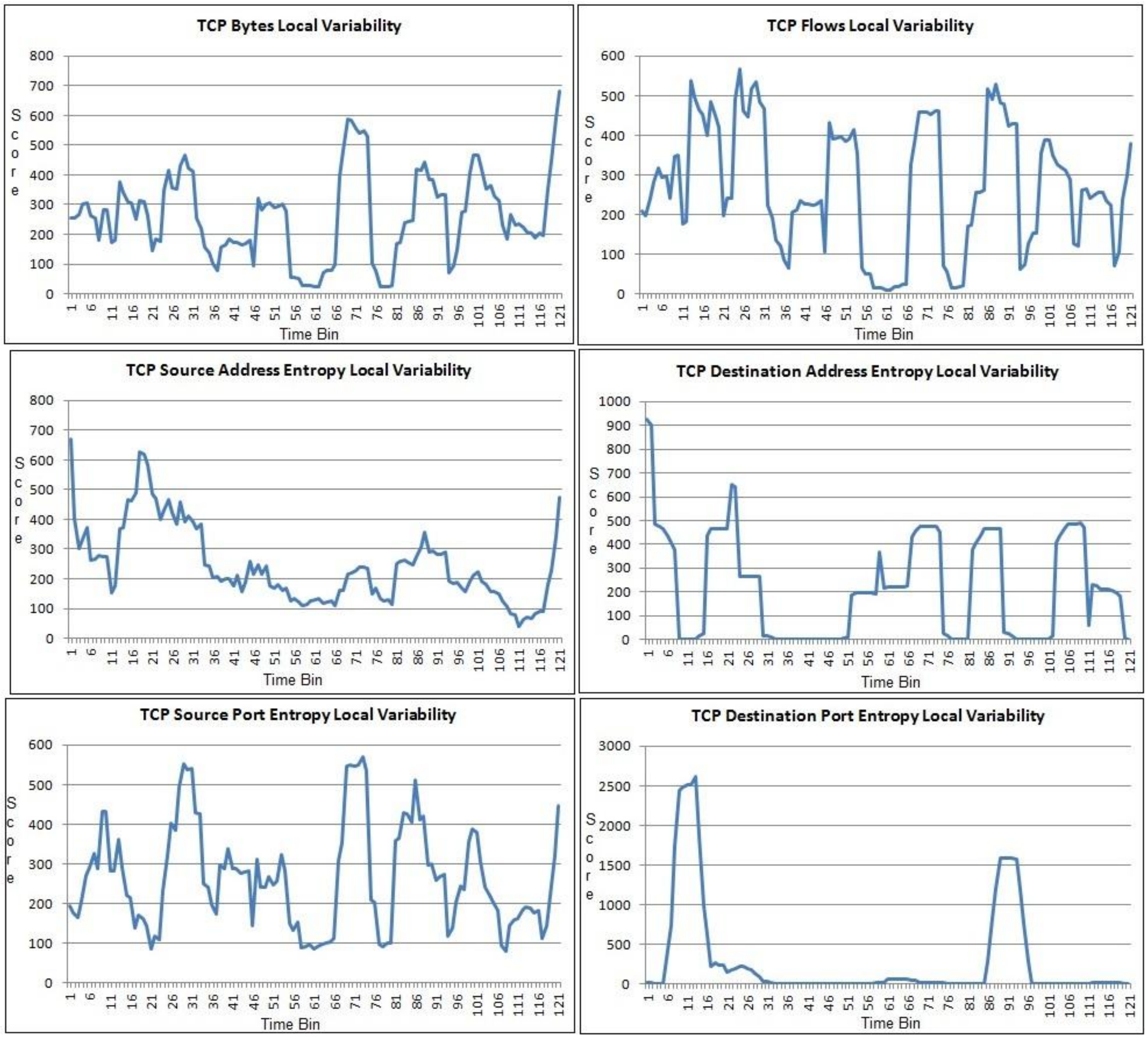


Figure 6.177: January 15, 2009 - Average TCP Wavelet Local Variability Scores with 5% infection rate

Infection Rate	Feature	Score	Hit Count (% Miss)	Miss Count (% Miss)
75%	TCP Bytes	0	33 (34%)	63 (66%)
	TCP Flows	1	29 (30%)	67 (70%)
	TCP Source Address Entropy	0	50 (52%)	46 (48%)
	TCP Destination Address Entropy	0	64 (67%)	32 (33%)
	TCP Source Port Entropy	2	43 (45%)	53 (55%)
	TCP Destination Port Entropy	0	70 (73%)	26 (27%)
	OTHER Bytes	1	42 (44%)	54 (56%)
	OTHER Flows	2	35 (36%)	61 (64%)
	OTHER Source Address Entropy	0	37 (39%)	59 (61%)
	OTHER Destination Address Entropy	0	35 (36%)	61 (64%)
	OTHER Source Port Entropy	2	40 (42%)	56 (58%)
	OTHER Destination Port Entropy	1	36 (38%)	60 (62%)
100%	TCP Bytes	0	32 (33%)	64 (67%)
	TCP Flows	2	30 (31%)	66 (69%)
	TCP Source Address Entropy	0	49 (51%)	47 (49%)
	TCP Destination Address Entropy	3	68 (71%)	28 (29%)
	TCP Source Port Entropy	3	44 (46%)	52 (54%)
	TCP Destination Port Entropy	0	69 (72%)	27 (28%)
	OTHER Bytes	1	45 (47%)	51 (53%)
	OTHER Flows	2	34 (35%)	62 (65%)
	OTHER Source Address Entropy	1	42 (44%)	54 (56%)
	OTHER Destination Address Entropy	0	32 (33%)	64 (67%)
	OTHER Source Port Entropy	2	40 (42%)	56 (58%)
	OTHER Destination Port Entropy	1	35 (36%)	61 (64%)
120%	TCP Bytes	0	32 (33%)	64 (67%)
	TCP Flows	2	30 (31%)	66 (69%)
	TCP Source Address Entropy	0	49 (51%)	47 (49%)
	TCP Destination Address Entropy	3	74 (77%)	22 (23%)
	TCP Source Port Entropy	3	44 (46%)	52 (54%)
	TCP Destination Port Entropy	0	69 (72%)	27 (28%)
	OTHER Bytes	1	45 (47%)	51 (53%)
	OTHER Flows	2	34 (35%)	62 (65%)
	OTHER Source Address Entropy	1	40 (42%)	56 (58%)
	OTHER Destination Address Entropy	1	33 (34%)	63 (66%)
	OTHER Source Port Entropy	2	39 (41%)	57 (59%)
	OTHER Destination Port Entropy	4	39 (41%)	57 (59%)

Table 6.19: January 15, 2009 - Wavelet Experiment results

6.2.5 Long and High Infection

This section examines the results of the Haar Wavelet experiments performed on a high infection rate in a long period of time. The four dates for this particular scenario, outlined in Table 5.2, all had consistent behaviour. As a result, for simplicity, only June 10, 2009 is discussed. Figure 6.18 and Figure 6.19 illustrate the footprints observed with default parameters for the TCP and OTHER features for the 5% infection ratio (See Table 5.4 for a list of default conditions). The experiments were conducted on test sets of 75%, 100% and 120% infection ratios. The infected time bins were during bins 30 - 40. A number of the feature plots depicted spikes during the infected times. Table 6.20 lists the hit/miss scores and the hit/miss counts measured for each feature. The default threshold method determines a 12% miss rate in the best feature and about a 57% miss rate in the worst performing feature. The other threshold methods range from 4% to 63%, see Section 6.2.8 for details.

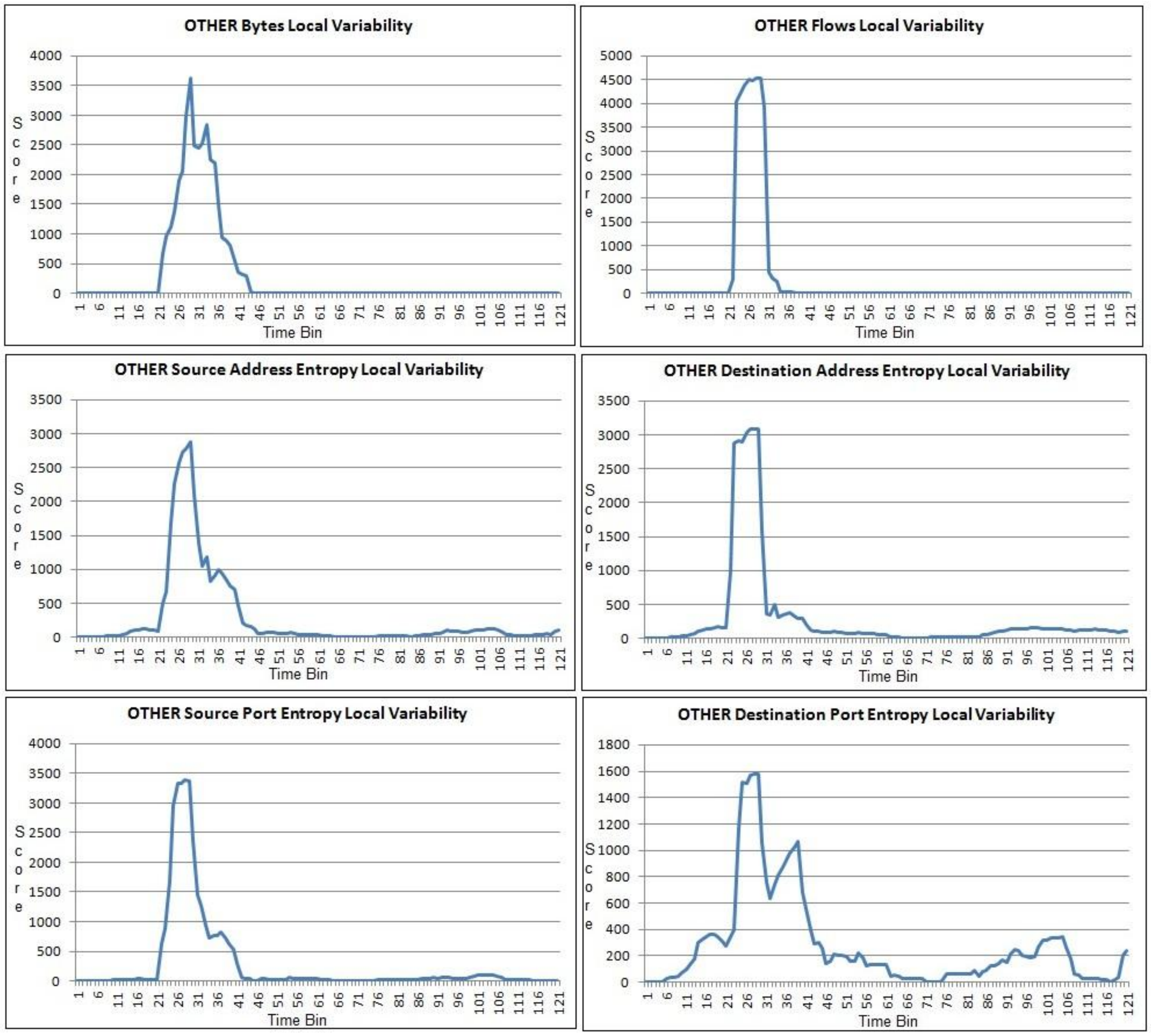


Figure 6.188: June 10, 2009 - Average OTHER Wavelet Local Variability Scores with 5% infection rate

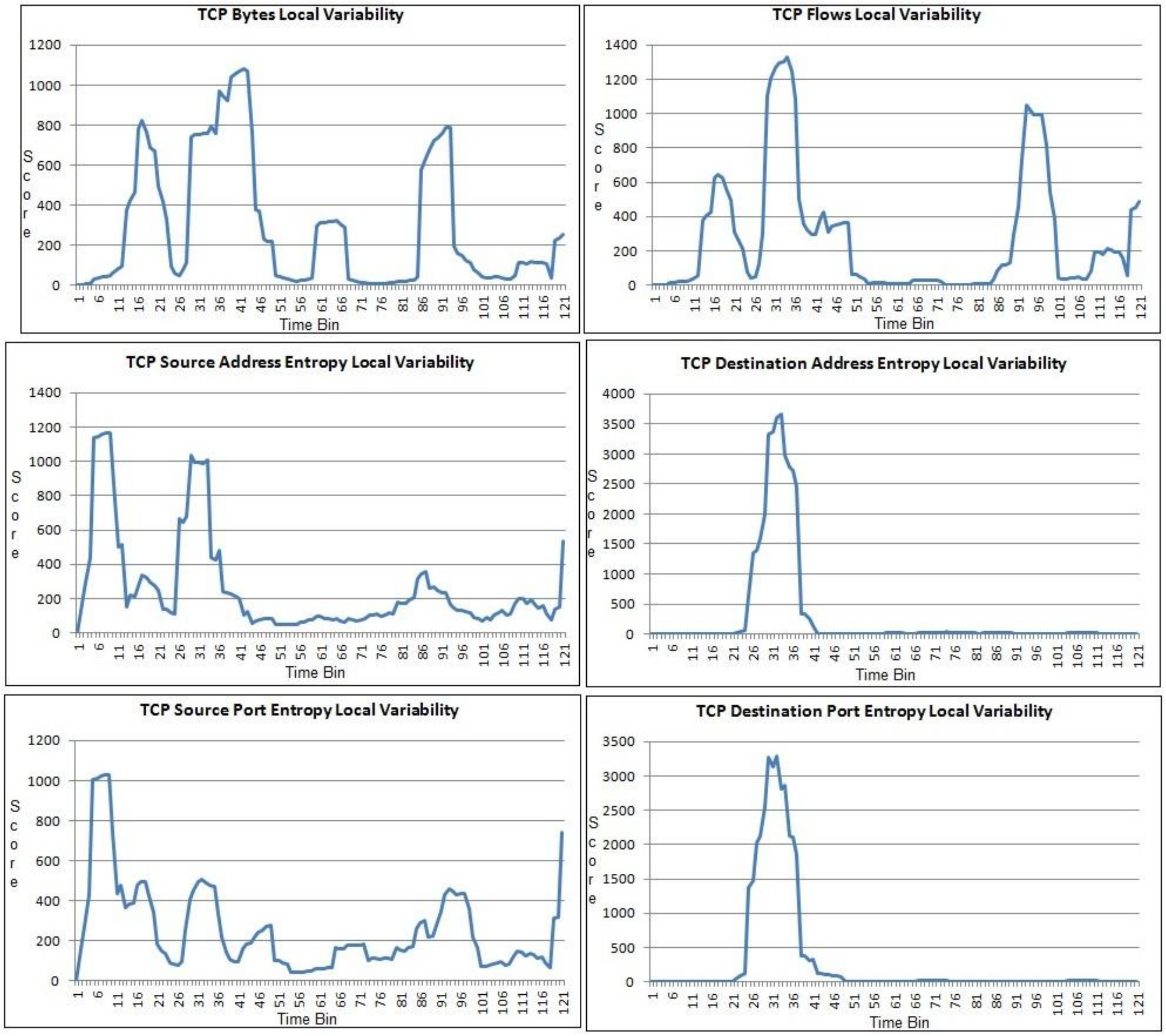


Figure 6.19: June 10, 2009 - Average TCP Wavelet Local Variability Scores with 5% infection rate

Infection Rate	Feature	Score	Hit Count (% Miss)	Miss Count (% Miss)
75%	TCP Bytes	-6	51 (53%)	45 (47%)
	TCP Flows	-11	48 (50%)	48 (50%)
	TCP Source Address Entropy	-14	36 (38%)	60 (62%)
	TCP Destination Address Entropy	-3	71 (74%)	25 (26%)
	TCP Source Port Entropy	-12	35 (36%)	61 (64%)
	TCP Destination Port Entropy	3	72 (75%)	24 (25%)
	OTHER Bytes	4	80 (83%)	16 (17%)
	OTHER Flows	-17	57 (59%)	39 (41%)
	OTHER Source Address Entropy	-9	27 (28%)	69 (72%)
	OTHER Destination Address Entropy	-18	24 (25%)	72 (75%)
	OTHER Source Port Entropy	-9	65 (68%)	31 (32%)
	OTHER Destination Port Entropy	-7	30 (31%)	66 (69%)
100%	TCP Bytes	-5	52 (54%)	44 (46%)
	TCP Flows	-11	49 (51%)	47 (49%)
	TCP Source Address Entropy	-7	60 (63%)	36 (27%)
	TCP Destination Address Entropy	-2	71 (74%)	25 (26%)
	TCP Source Port Entropy	-12	34 (35%)	62 (65%)
	TCP Destination Port Entropy	6	74 (77%)	22 (23%)
	OTHER Bytes	6	81 (84%)	15 (16%)
	OTHER Flows	-17	56 (58%)	40 (42%)
	OTHER Source Address Entropy	-9	26 (27%)	70 (73%)
	OTHER Destination Address Entropy	-18	23 (24%)	73 (76%)
	OTHER Source Port Entropy	-1	72 (75%)	24 (25%)
	OTHER Destination Port Entropy	-6	34 (35%)	62 (65%)
120%	TCP Bytes	-4	53 (55%)	43 (45%)
	TCP Flows	-12	47 (49%)	49 (51%)
	TCP Source Address Entropy	-6	51 (53%)	45 (47%)
	TCP Destination Address Entropy	-2	70 (73%)	26 (27%)
	TCP Source Port Entropy	-12	32 (33%)	64 (67%)
	TCP Destination Port Entropy	6	73 (76%)	23 (24%)
	OTHER Bytes	6	80 (83%)	16 (17%)
	OTHER Flows	-17	55 (57%)	41 (43%)
	OTHER Source Address Entropy	-7	55 (57%)	41 (43%)
	OTHER Destination Address Entropy	-17	24 (25%)	72 (75%)
	OTHER Source Port Entropy	-4	68 (71%)	28 (29%)
	OTHER Destination Port Entropy	-3	31 (32%)	65 (68%)

Table 6.20: June 10, 2009 - Wavelet Experiment results

6.2.6 Varying Dataset size

A number of experiments varying input dataset size and corresponding filtering boundaries were performed. The results in tables 6.21 – 6.24 summarize the evaluation measurements for each scenario on the previously specified dates. It was observed that increasing the dataset size did not improve the hit/miss scores or hit/miss counts.

Size	**Bounds (High pass, Mid pass, Low pass)**	**Feature**	**Miss count**		
			5%	**10%**	**20%**
128	2,3,2	TCP Bytes	45	46	47
		TCP Flows	42	43	44
		TCP Source Address Entropy	50	51	50
		TCP Destination Address Entropy	47	45	44
		TCP Source Port Entropy	45	43	41
		TCP Destination Port Entropy	28	29	30
		OTHER Bytes	24	25	25
		OTHER Flows	69	71	73
		OTHER Source Address Entropy	72	71	72
		OTHER Destination Address Entropy	81	80	82
		OTHER Source Port Entropy	57	58	60
		OTHER Destination Port Entropy	75	72	71
	3,2,2	TCP Bytes	43	44	45
		TCP Flows	45	46	47
		TCP Source Address Entropy	59	60	61
		TCP Destination Address Entropy	50	49	52
		TCP Source Port Entropy	52	53	53
		TCP Destination Port Entropy	28	29	30
		OTHER Bytes	24	25	30
		OTHER Flows	72	76	77
		OTHER Source Address Entropy	72	73	74
		OTHER Destination Address Entropy	78	79	80
		OTHER Source Port Entropy	55	57	60
		OTHER Destination Port Entropy	75	75	75
256	3,2,3	TCP Bytes	65	66	68
		TCP Flows	86	87	89
		TCP Source Address Entropy	111	112	114
		TCP Destination Address Entropy	51	50	51
		TCP Source Port Entropy	100	101	103
		TCP Destination Port Entropy	42	43	45

		OTHER Bytes	25	27	26
		OTHER Flows	138	140	136
		OTHER Source Address Entropy	84	85	85
		OTHER Destination Address Entropy	103	106	103
		OTHER Source Port Entropy	60	65	65
		OTHER Destination Port Entropy	82	80	80
	3,3,2	TCP Bytes	65	66	68
		TCP Flows	85	86	88
		TCP Source Address Entropy	109	110	112
		TCP Destination Address Entropy	56	50	58
		TCP Source Port Entropy	97	98	100
		TCP Destination Port Entropy	40	41	43
		OTHER Bytes	26	27	26
		OTHER Flows	138	139	133
		OTHER Source Address Entropy	84	85	85
		OTHER Destination Address Entropy	104	105	103
		OTHER Source Port Entropy	60	65	65
		OTHER Destination Port Entropy	82	80	80
512	3,3,3	TCP Bytes	109	112	117
		TCP Flows	139	142	141
		TCP Source Address Entropy	233	236	237
		TCP Destination Address Entropy	53	45	52
		TCP Source Port Entropy	223	226	227
		TCP Destination Port Entropy	49	53	56
		OTHER Bytes	29	30	31
		OTHER Flows	238	238	240
		OTHER Source Address Entropy	80	84	81
		OTHER Destination Address Entropy	128	135	133
		OTHER Source Port Entropy	71	73	70
		OTHER Destination Port Entropy	79	82	82
	4,3,2	TCP Bytes	107	109	114
		TCP Flows	56	158	157
		TCP Source Address Entropy	246	248	252
		TCP Destination Address Entropy	55	51	64
		TCP Source Port Entropy	238	239	240
		TCP Destination Port Entropy	45	47	52
		OTHER Bytes	30	30	31
		OTHER Flows	239	239	236
		OTHER Source Address Entropy	71	72	69
		OTHER Destination Address Entropy	124	132	129
		OTHER Source Port Entropy	60	62	66
		OTHER Destination Port Entropy	72	71	73

Table 6.21: March 8, 2009 Varying input data / boundary size results

Size	Bounds (High pass, Mid pass, Low pass)	Feature	Miss count		
			5%	10%	20%
128	2,3,2	TCP Bytes	47	48	50
		TCP Flows	50	51	52
		TCP Source Address Entropy	46	47	49
		TCP Destination Address Entropy	43	44	42
		TCP Source Port Entropy	47	48	50
		TCP Destination Port Entropy	32	31	34
		OTHER Bytes	27	26	29
		OTHER Flows	58	42	56
		OTHER Source Address Entropy	36	35	40
		OTHER Destination Address Entropy	45	44	37
		OTHER Source Port Entropy	39	37	42
		OTHER Destination Port Entropy	53	54	45
	3,2,2	TCP Bytes	47	48	50
		TCP Flows	50	51	52
		TCP Source Address Entropy	44	45	47
		TCP Destination Address Entropy	42	43	42
		TCP Source Port Entropy	39	40	42
		TCP Destination Port Entropy	33	32	33
		OTHER Bytes	30	30	37
		OTHER Flows	63	42	40
		OTHER Source Address Entropy	36	35	36
		OTHER Destination Address Entropy	45	44	44
		OTHER Source Port Entropy	39	39	40
		OTHER Destination Port Entropy	46	45	46
256	3,2,3	TCP Bytes	89	92	95
		TCP Flows	91	94	97
		TCP Source Address Entropy	76	79	82
		TCP Destination Address Entropy	67	70	48
		TCP Source Port Entropy	73	76	78
		TCP Destination Port Entropy	40	37	38
		OTHER Bytes	49	53	54
		OTHER Flows	93	92	93
		OTHER Source Address Entropy	53	55	69
		OTHER Destination Address Entropy	102	103	98
		OTHER Source Port Entropy	55	57	66
		OTHER Destination Port Entropy	73	81	71
	3,3,2	TCP Bytes	87	90	93
		TCP Flows	91	94	98
		TCP Source Address Entropy	75	78	81
		TCP Destination Address Entropy	67	70	48
		TCP Source Port Entropy	72	75	77

		TCP Destination Port Entropy	40	37	42
		OTHER Bytes	31	35	36
		OTHER Flows	89	89	92
		OTHER Source Address Entropy	53	55	69
		OTHER Destination Address Entropy	102	103	100
		OTHER Source Port Entropy	55	57	66
		OTHER Destination Port Entropy	72	81	70
	3,3,3	TCP Bytes	170	174	180
		TCP Flows	206	212	217
		TCP Source Address Entropy	196	190	199
		TCP Destination Address Entropy	75	81	62
		TCP Source Port Entropy	192	190	198
		TCP Destination Port Entropy	52	52	42
		OTHER Bytes	54	53	64
		OTHER Flows	193	185	190
		OTHER Source Address Entropy	91	97	104
		OTHER Destination Address Entropy	208	205	208
		OTHER Source Port Entropy	52	59	57
		OTHER Destination Port Entropy	130	134	126
	4,3,2	TCP Bytes	167	17	177
		TCP Flows	197	203	209
		TCP Source Address Entropy	193	189	189
		TCP Destination Address Entropy	73	79	66
		TCP Source Port Entropy	186	182	184
		TCP Destination Port Entropy	50	50	44
		OTHER Bytes	26	42	38
		OTHER Flows	196	192	201
		OTHER Source Address Entropy	86	99	103
		OTHER Destination Address Entropy	213	207	210
		OTHER Source Port Entropy	52	55	57
		OTHER Destination Port Entropy	125	133	113

Table 6.22: May 28, 2009 Varying input data / boundary size results

Size	**Bounds (High pass, Mid pass, Low pass)**	**Feature**	**Miss count**		
			75%	**100%**	**120%**
128	2,3,2	TCP Bytes	63	64	64
		TCP Flows	67	66	66
		TCP Source Address Entropy	46	47	47
		TCP Destination Address Entropy	32	28	22
		TCP Source Port Entropy	53	52	52
		TCP Destination Port Entropy	26	27	27
		OTHER Bytes	54	51	51
		OTHER Flows	61	62	62
		OTHER Source Address Entropy	59	51	56
		OTHER Destination Address Entropy	61	64	63
		OTHER Source Port Entropy	56	56	57
		OTHER Destination Port Entropy	60	61	57
	3,2,2	TCP Bytes	62	63	63
		TCP Flows	65	64	64
		TCP Source Address Entropy	57	58	58
		TCP Destination Address Entropy	35	29	22
		TCP Source Port Entropy	53	52	52
		TCP Destination Port Entropy	28	29	30
		OTHER Bytes	54	55	55
		OTHER Flows	62	63	62
		OTHER Source Address Entropy	55	57	55
		OTHER Destination Address Entropy	54	55	54
		OTHER Source Port Entropy	61	62	61
		OTHER Destination Port Entropy	61	63	59
	3,2,3	TCP Bytes	109	107	108
		TCP Flows	104	102	102
		TCP Source Address Entropy	124	121	121
		TCP Destination Address Entropy	42	37	31
		TCP Source Port Entropy	107	105	105
		TCP Destination Port Entropy	47	49	50
		OTHER Bytes	98	99	100
		OTHER Flows	105	108	109
		OTHER Source Address Entropy	69	70	65
		OTHER Destination Address Entropy	124	125	114
		OTHER Source Port Entropy	77	79	80
		OTHER Destination Port Entropy	73	77	75
	3,3,2	TCP Bytes	110	109	108
		TCP Flows	105	104	103
		TCP Source Address Entropy	123	122	123
		TCP Destination Address Entropy	43	36	31
		TCP Source Port Entropy	106	105	104

		TCP Destination Port Entropy	47	48	50
		OTHER Bytes	105	105	107
		OTHER Flows	104	107	109
		OTHER Source Address Entropy	68	69	65
		OTHER Destination Address Entropy	123	124	114
		OTHER Source Port Entropy	78	78	79
		OTHER Destination Port Entropy	73	75	75
	3,3,3	TCP Bytes	246	248	249
		TCP Flows	173	173	174
		TCP Source Address Entropy	246	244	245
		TCP Destination Address Entropy	50	39	39
		TCP Source Port Entropy	241	240	241
		TCP Destination Port Entropy	85	88	86
		OTHER Bytes	104	111	108
		OTHER Flows	221	224	230
		OTHER Source Address Entropy	71	76	74
		OTHER Destination Address Entropy	217	219	219
		OTHER Source Port Entropy	77	77	87
		OTHER Destination Port Entropy	82	97	82
	4,3,2	TCP Bytes	236	239	239
		TCP Flows	194	197	197
		TCP Source Address Entropy	243	240	240
		TCP Destination Address Entropy	48	41	41
		TCP Source Port Entropy	237	238	238
		TCP Destination Port Entropy	76	81	86
		OTHER Bytes	118	121	123
		OTHER Flows	221	228	226
		OTHER Source Address Entropy	73	72	63
		OTHER Destination Address Entropy	207	213	206
		OTHER Source Port Entropy	84	89	91
		OTHER Destination Port Entropy	84	98	81

Table 6.23: January 15, 2009 Varying input data / boundary size results

Size	Bounds (High pass, Mid pass, Low pass)	Feature	Miss count		
			75%	100%	120%
128	2,3,2	TCP Bytes	45	44	43
		TCP Flows	48	47	49
		TCP Source Address Entropy	60	36	45
		TCP Destination Address Entropy	25	25	26
		TCP Source Port Entropy	61	62	64
		TCP Destination Port Entropy	24	22	23
		OTHER Bytes	16	15	16
		OTHER Flows	39	40	41
		OTHER Source Address Entropy	69	70	41
		OTHER Destination Address Entropy	72	73	72
		OTHER Source Port Entropy	31	24	28
		OTHER Destination Port Entropy	66	62	65
	3,2,2	TCP Bytes	44	43	42
		TCP Flows	42	37	38
		TCP Source Address Entropy	63	35	47
		TCP Destination Address Entropy	32	32	30
		TCP Source Port Entropy	52	54	55
		TCP Destination Port Entropy	27	25	26
		OTHER Bytes	20	22	20
		OTHER Flows	39	40	41
		OTHER Source Address Entropy	75	75	41
		OTHER Destination Address Entropy	75	41	77
		OTHER Source Port Entropy	39	32	34
		OTHER Destination Port Entropy	69	70	65
	3,2,3	TCP Bytes	95	94	91
		TCP Flows	81	78	80
		TCP Source Address Entropy	121	112	114
		TCP Destination Address Entropy	31	28	30
		TCP Source Port Entropy	114	115	117
		TCP Destination Port Entropy	52	33	53
		OTHER Bytes	35	25	28
		OTHER Flows	57	58	61
		OTHER Source Address Entropy	96	95	57
		OTHER Destination Address Entropy	119	119	122
		OTHER Source Port Entropy	46	41	46
		OTHER Destination Port Entropy	84	79	80
	3,3,2	TCP Bytes	93	92	89
		TCP Flows	83	82	80
		TCP Source Address Entropy	122	113	115
		TCP Destination Address Entropy	31	29	31
		TCP Source Port Entropy	116	117	118

		TCP Destination Port Entropy	51	32	31
		OTHER Bytes	26	19	22
		OTHER Flows	60	61	64
		OTHER Source Address Entropy	98	99	45
		OTHER Destination Address Entropy	124	124	126
		OTHER Source Port Entropy	48	41	45
		OTHER Destination Port Entropy	85	81	82
	3,3,3	TCP Bytes	189	190	184
		TCP Flows	130	131	132
		TCP Source Address Entropy	235	232	236
		TCP Destination Address Entropy	50	49	51
		TCP Source Port Entropy	234	228	231
		TCP Destination Port Entropy	59	49	55
		OTHER Bytes	54	40	43
		OTHER Flows	97	100	105
		OTHER Source Address Entropy	107	110	112
		OTHER Destination Address Entropy	207	205	203
		OTHER Source Port Entropy	74	61	64
		OTHER Destination Port Entropy	103	99	98
	4,3,2	TCP Bytes	192	190	84
		TCP Flows	121	122	125
		TCP Source Address Entropy	233	231	237
		TCP Destination Address Entropy	52	47	55
		TCP Source Port Entropy	234	232	238
		TCP Destination Port Entropy	63	45	49
		OTHER Bytes	45	33	38
		OTHER Flows	99	101	107
		OTHER Source Address Entropy	111	105	102
		OTHER Destination Address Entropy	200	208	205
		OTHER Source Port Entropy	47	57	59
		OTHER Destination Port Entropy	106	99	98

Table 6.24: June 10, 2009 Varying input data / boundary size results

6.2.7 Varying Local Variability Weights

A number of experiments were performed using the default parameters, and two different local variability weights. The results in tables 6.25 – 6.28 summarize the evaluation measurements for each scenario on the previously specified dates. It was observed that the weights, which applied an emphasis to high pass values, had the lowest miss counts best. The miss counts were never worse using weights focusing on the mid pass values.

Bounds (High pass, Mid pass, Low pass)	Weights (High pass, Mid pass, Low pass)	Features	Miss Count		
			5%	10%	20%
2,3,2	5,1,0	TCP Bytes	45	46	47
		TCP Flows	42	43	44
		TCP Source Address Entropy	50	51	50
		TCP Destination Address Entropy	47	45	44
		TCP Source Port Entropy	45	43	41
		TCP Destination Port Entropy	28	29	30
		OTHER Bytes	24	25	25
		OTHER Flows	69	71	73
		OTHER Source Address Entropy	72	71	72
		OTHER Destination Address Entropy	81	80	82
		OTHER Source Port Entropy	57	58	60
		OTHER Destination Port Entropy	75	72	71
	3,2,0	TCP Bytes	44	45	46
		TCP Flows	41	42	43
		TCP Source Address Entropy	51	52	53
		TCP Destination Address Entropy	46	45	46
		TCP Source Port Entropy	51	52	53
		TCP Destination Port Entropy	28	29	30
		OTHER Bytes	25	31	32
		OTHER Flows	68	67	70
		OTHER Source Address Entropy	71	72	73
		OTHER Destination Address Entropy	77	80	80
		OTHER Source Port Entropy	53	57	59
		OTHER Destination Port Entropy	75	74	74
3,2,2	5,1,0	TCP Bytes	43	44	45
		TCP Flows	45	46	47
		TCP Source Address Entropy	59	60	61
		TCP Destination Address Entropy	50	49	52
		TCP Source Port Entropy	52	53	53
		TCP Destination Port Entropy	28	29	30
		OTHER Bytes	24	25	30
		OTHER Flows	72	76	77
		OTHER Source Address Entropy	72	73	74
		OTHER Destination Address Entropy	78	79	80
		OTHER Source Port Entropy	55	57	60
		OTHER Destination Port Entropy	75	75	75

	3,2,0	TCP Bytes	43	44	45
		TCP Flows	42	43	44
		TCP Source Address Entropy	47	48	49
		TCP Destination Address Entropy	52	50	50
		TCP Source Port Entropy	49	50	50
		TCP Destination Port Entropy	28	29	30
		OTHER Bytes	30	28	24
		OTHER Flows	64	66	67
		OTHER Source Address Entropy	72	73	70
		OTHER Destination Address Entropy	82	82	78
		OTHER Source Port Entropy	54	54	57
		OTHER Destination Port Entropy	75	74	74

Table 6.25: March 8, 2009 Varying Local Variability weights results

Bounds (High pass, Mid pass, Low pass)	**Weights (High pass, Mid pass, Low pass)**	**Features**	**Miss Count**		
			5%	**10%**	**20%**
2,3,2	5,1,0	TCP Bytes	47	48	50
		TCP Flows	50	51	52
		TCP Source Address Entropy	46	47	49
		TCP Destination Address Entropy	43	44	42
		TCP Source Port Entropy	47	48	50
		TCP Destination Port Entropy	32	31	34
		OTHER Bytes	27	26	29
		OTHER Flows	58	42	56
		OTHER Source Address Entropy	36	35	40
		OTHER Destination Address Entropy	45	44	37
		OTHER Source Port Entropy	39	37	42
		OTHER Destination Port Entropy	53	54	45
	3,2,0	TCP Bytes	48	49	51
		TCP Flows	49	50	52
		TCP Source Address Entropy	47	48	50
		TCP Destination Address Entropy	42	43	37
		TCP Source Port Entropy	47	48	50
		TCP Destination Port Entropy	33	32	29
		OTHER Bytes	27	26	28
		OTHER Flows	62	36	60
		OTHER Source Address Entropy	36	36	43
		OTHER Destination Address Entropy	49	47	45
		OTHER Source Port Entropy	39	39	41
		OTHER Destination Port Entropy	46	45	43
3,2,2	5,1,0	TCP Bytes	47	48	50
		TCP Flows	50	51	52
		TCP Source Address Entropy	44	45	47
		TCP Destination Address Entropy	42	43	42
		TCP Source Port Entropy	39	40	42
		TCP Destination Port Entropy	33	32	33
		OTHER Bytes	30	30	37
		OTHER Flows	63	42	40
		OTHER Source Address Entropy	36	35	36
		OTHER Destination Address Entropy	45	44	44
		OTHER Source Port Entropy	39	39	40
		OTHER Destination Port Entropy	46	45	46
	3,2,0	TCP Bytes	46	45	48
		TCP Flows	50	49	52
		TCP Source Address Entropy	46	45	48
		TCP Destination Address Entropy	44	43	37

		TCP Source Port Entropy	46	45	48
		TCP Destination Port Entropy	33	34	30
		OTHER Bytes	31	28	31
		OTHER Flows	36	62	38
		OTHER Source Address Entropy	36	36	37
		OTHER Destination Address Entropy	46	46	45
		OTHER Source Port Entropy	37	39	38
		OTHER Destination Port Entropy	48	56	45

Table 6.26: May 28, 2009 Varying Local Variability weights results

Bounds (High pass, Mid pass, Low pass)	Weights (High pass, Mid pass, Low pass)	Features	Miss Count		
			75 %	100%	120%
2,3,2	5,1,0	TCP Bytes	63	64	64
		TCP Flows	67	66	66
		TCP Source Address Entropy	46	47	47
		TCP Destination Address Entropy	32	28	22
		TCP Source Port Entropy	53	52	52
		TCP Destination Port Entropy	26	27	27
		OTHER Bytes	54	51	51
		OTHER Flows	61	62	62
		OTHER Source Address Entropy	59	51	56
		OTHER Destination Address Entropy	61	64	63
		OTHER Source Port Entropy	56	56	57
		OTHER Destination Port Entropy	60	61	57
	3,2,0	TCP Bytes	58	59	59
		TCP Flows	64	63	63
		TCP Source Address Entropy	50	51	51
		TCP Destination Address Entropy	29	29	22
		TCP Source Port Entropy	58	57	57
		TCP Destination Port Entropy	25	26	26
		OTHER Bytes	49	49	49
		OTHER Flows	58	59	56
		OTHER Source Address Entropy	53	54	55
		OTHER Destination Address Entropy	55	56	55
		OTHER Source Port Entropy	54	56	57
		OTHER Destination Port Entropy	63	63	57
3,2,2	5,1,0	TCP Bytes	62	63	63
		TCP Flows	65	64	64
		TCP Source Address Entropy	57	58	58
		TCP Destination Address Entropy	35	29	22
		TCP Source Port Entropy	53	52	52
		TCP Destination Port Entropy	28	29	30
		OTHER Bytes	54	55	55
		OTHER Flows	62	63	62
		OTHER Source Address Entropy	55	57	55
		OTHER Destination Address Entropy	54	55	54
		OTHER Source Port Entropy	61	62	61
		OTHER Destination Port Entropy	61	63	59
	3,2,0	TCP Bytes	60	61	61
		TCP Flows	64	63	63
		TCP Source Address Entropy	55	56	56
		TCP Destination Address Entropy	28	31	25

		TCP Source Port Entropy	61	60	60
		TCP Destination Port Entropy	28	29	29
		OTHER Bytes	59	60	60
		OTHER Flows	60	60	56
		OTHER Source Address Entropy	60	56	55
		OTHER Destination Address Entropy	57	60	59
		OTHER Source Port Entropy	62	61	58
		OTHER Destination Port Entropy	66	65	61

Table 6.27: January 15, 2009 Varying Local Variability weights results

Bounds (High pass, Mid pass, Low pass)	**Weights (High pass, Mid pass, Low pass)**	**Features**	**Miss Count**		
			75%	**100%**	**120%**
2,3,2	5,1,0	TCP Bytes	45	44	43
		TCP Flows	48	47	49
		TCP Source Address Entropy	60	36	45
		TCP Destination Address Entropy	25	25	26
		TCP Source Port Entropy	61	62	64
		TCP Destination Port Entropy	24	22	23
		OTHER Bytes	16	15	16
		OTHER Flows	39	40	41
		OTHER Source Address Entropy	69	70	41
		OTHER Destination Address Entropy	72	73	72
		OTHER Source Port Entropy	31	24	28
		OTHER Destination Port Entropy	66	62	65
	3,2,0	TCP Bytes	46	45	44
		TCP Flows	49	45	47
		TCP Source Address Entropy	61	47	57
		TCP Destination Address Entropy	24	24	22
		TCP Source Port Entropy	64	65	66
		TCP Destination Port Entropy	24	22	23
		OTHER Bytes	16	15	15
		OTHER Flows	39	40	41
		OTHER Source Address Entropy	70	74	38
		OTHER Destination Address Entropy	66	75	70
		OTHER Source Port Entropy	31	25	28
		OTHER Destination Port Entropy	63	63	63
3,2,2	5,1,0	TCP Bytes	44	43	42
		TCP Flows	42	37	38
		TCP Source Address Entropy	63	35	47
		TCP Destination Address Entropy	32	32	30
		TCP Source Port Entropy	52	54	55
		TCP Destination Port Entropy	27	25	26
		OTHER Bytes	20	22	20
		OTHER Flows	39	40	41
		OTHER Source Address Entropy	75	75	41
		OTHER Destination Address Entropy	75	41	77
		OTHER Source Port Entropy	39	32	34
		OTHER Destination Port Entropy	69	70	65
	3,2,0	TCP Bytes	46	45	44
		TCP Flows	44	42	43
		TCP Source Address Entropy	62	56	58
		TCP Destination Address Entropy	28	31	28

		TCP Source Port Entropy	57	57	59
		TCP Destination Port Entropy	24	22	23
		OTHER Bytes	21	22	20
		OTHER Flows	39	40	41
		OTHER Source Address Entropy	69	72	37
		OTHER Destination Address Entropy	69	70	72
		OTHER Source Port Entropy	40	35	37
		OTHER Destination Port Entropy	65	70	70

Table 6.28: June 10, 2009 - Varying Local Variability weights results

6.2.8 Threshold Techniques and Haar Wavelet Filtering analysis

Each of the experiments performed used four different threshold techniques to automatically identify anomalous time bins. The results listed in tables 6.29 – 6.32 summarize the evaluation measurements for threshold techniques on the specified dates. In each scenario, at least one feature was able to detect the infected time. It was observed that all the dynamic averages performed worse than the static average.

Infection Rate	**Feature**	**Miss Count**			
		Static Average	**Moving Average without Outliers**	**Moving Average**	**Weighted Moving Average**
5%	TCP Bytes	33	45	61	49
	TCP Flows	29	42	56	38
	TCP Source Address Entropy	26	50	58	44
	TCP Destination Address Entropy	30	47	71	50
	TCP Source Port Entropy	32	45	53	39
	TCP Destination Port Entropy	10	28	65	38
	OTHER Bytes	7	24	50	38
	OTHER Flows	8	69	70	70
	OTHER Source Address Entropy	16	72	79	74
	OTHER Destination Address Entropy	18	81	79	81
	OTHER Source Port Entropy	10	57	76	68
	OTHER Destination Port Entropy	57	75	79	76
10%	TCP Bytes	34	46	62	50
	TCP Flows	30	43	57	39
	TCP Source Address Entropy	27	51	59	45
	TCP Destination Address Entropy	17	45	70	50
	TCP Source Port Entropy	33	43	52	38
	TCP Destination Port Entropy	11	29	64	41
	OTHER Bytes	8	25	52	40
	OTHER Flows	8	71	73	70
	OTHER Source Address Entropy	12	71	79	74
	OTHER Destination Address Entropy	13	80	80	82
	OTHER Source Port Entropy	9	58	77	69
	OTHER Destination Port Entropy	55	72	79	72
20%	TCP Bytes	35	47	63	51
	TCP Flows	31	44	58	40
	TCP Source Address Entropy	28	50	58	44

	TCP Destination Address Entropy	7	44	69	51
	TCP Source Port Entropy	34	41	51	37
	TCP Destination Port Entropy	16	30	65	42
	OTHER Bytes	8	25	52	40
	OTHER Flows	8	73	73	69
	OTHER Source Address Entropy	11	72	80	72
	OTHER Destination Address Entropy	11	82	80	81
	OTHER Source Port Entropy	8	60	78	67
	OTHER Destination Port Entropy	39	71	77	71

Table 6.29: March 8, 2009 – Wavelet Short and Low threshold results

Infection Rate	**Feature**	**Miss Count**			
		Static Average	**Moving Average without Outliers**	**Moving Average**	**Weighted Moving Average**
5%	TCP Bytes	47	65	52	39
	TCP Flows	50	65	46	42
	TCP Source Address Entropy	46	64	44	23
	TCP Destination Address Entropy	43	43	43	42
	TCP Source Port Entropy	47	62	32	24
	TCP Destination Port Entropy	32	47	36	35
	OTHER Bytes	26	43	33	27
	OTHER Flows	57	70	44	28
	OTHER Source Address Entropy	25	65	39	10
	OTHER Destination Address Entropy	44	58	43	41
	OTHER Source Port Entropy	38	63	40	9
	OTHER Destination Port Entropy	52	67	56	54
10%	TCP Bytes	40	48	68	53
	TCP Flows	43	51	68	47
	TCP Source Address Entropy	24	47	67	45
	TCP Destination Address Entropy	41	44	46	44
	TCP Source Port Entropy	25	48	65	33
	TCP Destination Port Entropy	34	31	44	35
	OTHER Bytes	27	26	44	27
	OTHER Flows	13	42	70	40
	OTHER Source Address Entropy	9	35	64	39
	OTHER Destination Address Entropy	35	44	57	42
	OTHER Source Port Entropy	8	37	62	40
	OTHER Destination Port Entropy	54	54	68	53
20%	TCP Bytes	42	50	70	55
	TCP Flows	46	52	68	49
	TCP Source Address Entropy	26	49	69	46
	TCP Destination Address Entropy	4	42	52	43
	TCP Source Port Entropy	26	50	67	35
	TCP Destination Port Entropy	4	34	46	36
	OTHER Bytes	29	29	48	35
	OTHER Flows	57	56	71	49
	OTHER Source Address Entropy	23	40	58	39
	OTHER Destination Address Entropy	64	37	51	49
	OTHER Source Port Entropy	16	42	60	42
	OTHER Destination Port Entropy	37	45	63	46

Table 6.30: May 28, 2009 - Wavelet Long and Low threshold results

Infection Rate	Feature	Miss Count			
		Static Average	Moving Average without Outliers	Moving Average	Weighted Moving Average
75%	TCP Bytes	63	63	67	62
	TCP Flows	59	67	73	64
	TCP Source Address Entropy	53	46	45	41
	TCP Destination Address Entropy	54	32	57	50
	TCP Source Port Entropy	60	53	56	51
	TCP Destination Port Entropy	23	26	55	29
	OTHER Bytes	11	54	55	55
	OTHER Flows	6	61	62	51
	OTHER Source Address Entropy	9	59	65	60
	OTHER Destination Address Entropy	25	61	60	61
	OTHER Source Port Entropy	8	56	68	64
	OTHER Destination Port Entropy	17	60	61	56
100%	TCP Bytes	64	64	68	63
	TCP Flows	60	66	72	65
	TCP Source Address Entropy	54	47	46	42
	TCP Destination Address Entropy	13	28	50	40
	TCP Source Port Entropy	61	52	55	52
	TCP Destination Port Entropy	24	27	52	30
	OTHER Bytes	10	51	52	56
	OTHER Flows	7	62	62	53
	OTHER Source Address Entropy	9	54	64	56
	OTHER Destination Address Entropy	22	64	62	61
	OTHER Source Port Entropy	9	56	69	63
	OTHER Destination Port Entropy	13	61	62	58
120%	TCP Bytes	64	64	68	63
	TCP Flows	60	66	72	65
	TCP Source Address Entropy	54	47	46	42
	TCP Destination Address Entropy	12	22	44	39
	TCP Source Port Entropy	61	52	55	52
	TCP Destination Port Entropy	25	27	51	30
	OTHER Bytes	10	51	52	56
	OTHER Flows	7	62	61	53
	OTHER Source Address Entropy	10	56	63	58
	OTHER Destination Address Entropy	15	63	60	60
	OTHER Source Port Entropy	10	57	69	63
	OTHER Destination Port Entropy	9	57	56	55

Table 6.31: January 15, 2009 - Wavelet Short and High threshold results

Infection Rate	**Feature**	**Miss Count**			
		Static Average	**Moving Average without Outliers**	**Moving Average**	**Weighted Moving Average**
75%	TCP Bytes	33	45	59	55
	TCP Flows	33	48	65	50
	TCP Source Address Entropy	30	60	60	54
	TCP Destination Address Entropy	7	25	63	44
	TCP Source Port Entropy	43	61	62	55
	TCP Destination Port Entropy	5	24	53	37
	OTHER Bytes	10	16	46	20
	OTHER Flows	17	39	80	41
	OTHER Source Address Entropy	8	69	68	58
	OTHER Destination Address Entropy	9	72	71	67
	OTHER Source Port Entropy	8	31	67	54
	OTHER Destination Port Entropy	24	66	67	61
100%	TCP Bytes	32	46	58	42
	TCP Flows	32	47	64	44
	TCP Source Address Entropy	19	36	55	39
	TCP Destination Address Entropy	6	25	60	33
	TCP Source Port Entropy	44	62	64	50
	TCP Destination Port Entropy	6	22	53	30
	OTHER Bytes	9	15	45	11
	OTHER Flows	10	40	79	40
	OTHER Source Address Entropy	10	70	69	56
	OTHER Destination Address Entropy	17	73	71	69
	OTHER Source Port Entropy	9	24	57	40
	OTHER Destination Port Entropy	24	67	67	57
120%	TCP Bytes	31	43	57	41
	TCP Flows	31	49	65	15
	TCP Source Address Entropy	17	45	55	39
	TCP Destination Address Entropy	5	26	62	34
	TCP Source Port Entropy	45	64	65	51
	TCP Destination Port Entropy	5	25	53	29
	OTHER Bytes	9	16	45	10
	OTHER Flows	11	41	80	41
	OTHER Source Address Entropy	11	41	66	50
	OTHER Destination Address Entropy	11	72	70	70
	OTHER Source Port Entropy	10	28	62	43
	OTHER Destination Port Entropy	22	65	65	55

Table 6.32: June 10, 2009 - Wavelet Long and High threshold results

6.2.9 Summary and Discussion

The Haar Wavelet Filtering Analysis approach examines each feature and provides an indication of which features might have anomalies. There were three user-defined parameters: the size of the input data, filtering boundaries and the local variability weights for score calculation. The dataset size affects the bin width and the resolution of the filtering. A typical attack is short and 98% of the flows are less than a minute. A larger bin will have more flows, overall higher values, and will less likely to reflect quick spikes in values. Conversely, if the bin width is too small then the feature values will be small, and analysis becomes overly granular.

Input data size and filter boundaries. In the Haar Wavelet Filtering algorithm, the boundary sizes are related to input size. These boundaries define the amount of the high pass, mid pass and low pass values in filtering. Larger dataset size allows more levels to be divided into the high, medium and low regions. The experiments varying the data set sizes and filtering boundaries showed that an input size of 128 elements and emphasis on mid pass values performed the best. In this case, the bin width was 11.25 minutes and similar to what was used in literature. Larger input data sizes result in smaller bin sizes and performed worse. This can be attributed to smaller observation values and less change between time bins.

Local Variability Weights. The local variability weights are used in the deviation score calculation. The weights apply emphasis to each of the filtered values at a point of time. The high pass weights affect the sensitivity to high frequency changes and the mid pass weights affect the mid pass frequency changes. The experiments conducted examined a strong bias and a small bias towards high pass values. Each feature has its own independent analysis and the results from all of the features were examined. The weights with a strong bias generally had lower miss counts. The feature, which isolated the injected malicious flows, consistently had smaller miss counts. A possible reason for this is because the mid pass values are not as meaningful as the high pass values. Most of the anomalous flows were less than one minute long, in 11.25 min bins, and quick changes in features would be more evident.

The Haar Wavelet Filtering analysis was able to isolate all of the infected bins in at least one of the feature analysis. As the infection rate and duration increased, the approach performed better. The major drawback of this approach is that every feature is a separate independent analysis and can result in several independent, unrelated results. In practice, it would be very difficult to determine which feature(s) detecting the actual intrusions, and which are missing them.

6.3 Evaluation of Hybrid PCA – Haar Wavelet Filtering Analysis

This section outlines the results obtained from running the experiments with the Hybrid PCA- Haar Wavelet Filtering analysis approach outlined in section 5.3.4 and concludes with a brief discussion.

6.3.1 Hybrid Analysis and Pure data

This section examines the results of the Hybrid PCA – Haar Wavelet Filtering analysis applied to a dataset with no infections. The results presented are for January 6, 2009, which is representative of all 16 dates. Figure 6.20 shows the delta scores across all ten test sets. Unlike the Modified PCA and Haar Wavelet Filtering approaches, the Hybrid PCA – Haar Wavelet Filtering approach does not have extreme values to influence the average or flatten the visual plot. As a result, pre-algorithm filtering is not needed in the Hybrid PCA - Haar Wavelet Filtering approach. The threshold analysis was applied to the average of the test sets or footprint.

Similar to previous approaches, as shown in Figure 6.20, regardless of the threshold technique, the algorithm will suggest anomalous time bins.

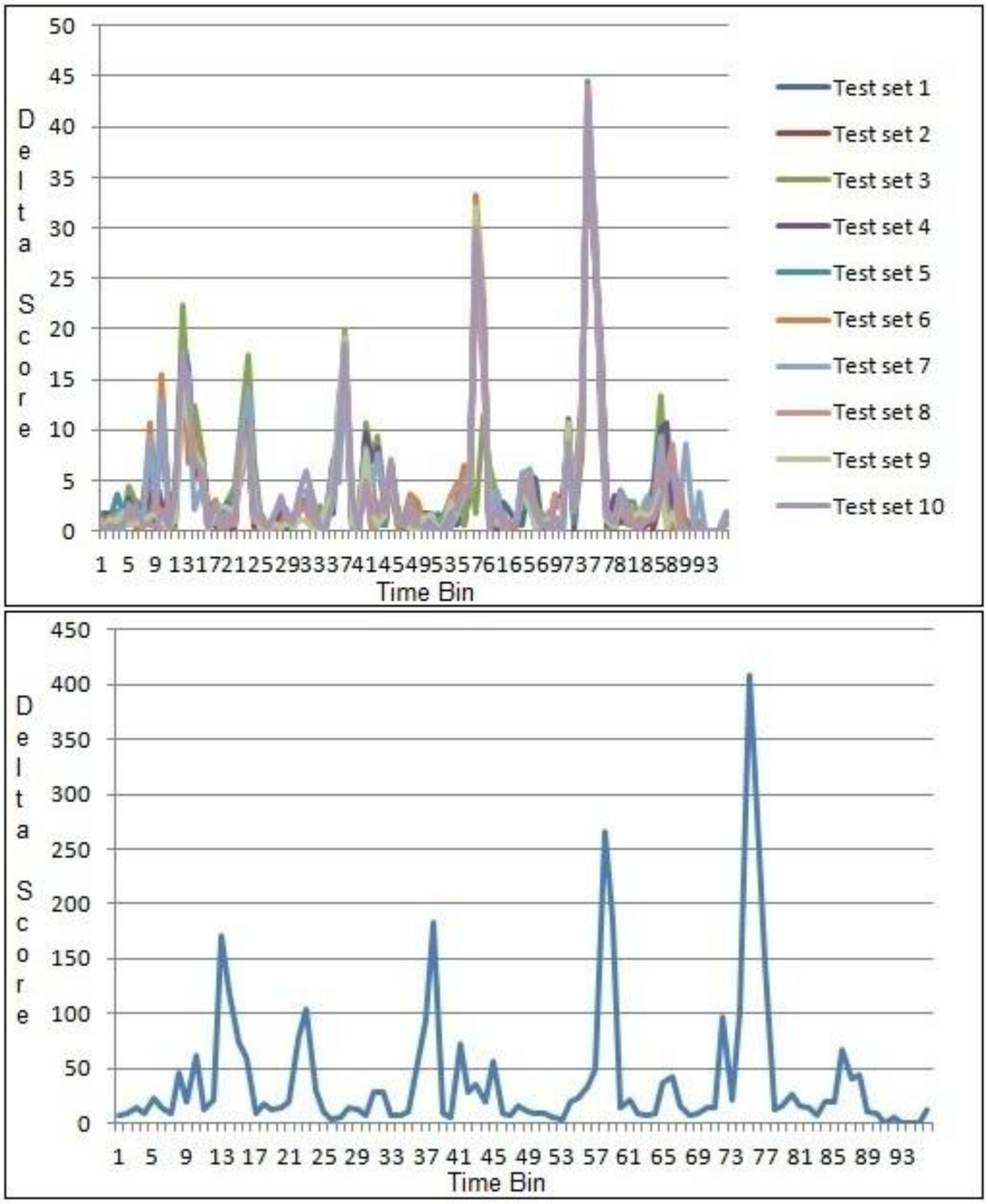


Figure 6.200: January 6, 2009 – Hybrid Delta scores (top) and Average Delta scores (bottom)

6.3.2 Short and Low Infection

This section examines the results of the Hybrid PCA – Haar Wavelet filtering analysis experiments performed on a low infection rate in a short period of time. The four dates for this particular scenario, outlined in Table 5.2, all had consistent behaviour. As a result, for simplicity, only March 8, 2009 is discussed. Figures 6.21 illustrates the footprints observed with default parameters conducted on test sets of 5%, 10% and 20% infection ratios respectively (See Table 5.4 for a list of default conditions). The infected time bins were during bins 88, 89 and 90 and the plots clearly picked up the infected times. Table 6.33 lists the hit/miss scores and the hit/miss counts measured for each feature. The default threshold method determines a miss rate of about 30%. The other threshold methods range from 4% to 39%, see Section 6.3.6 for details

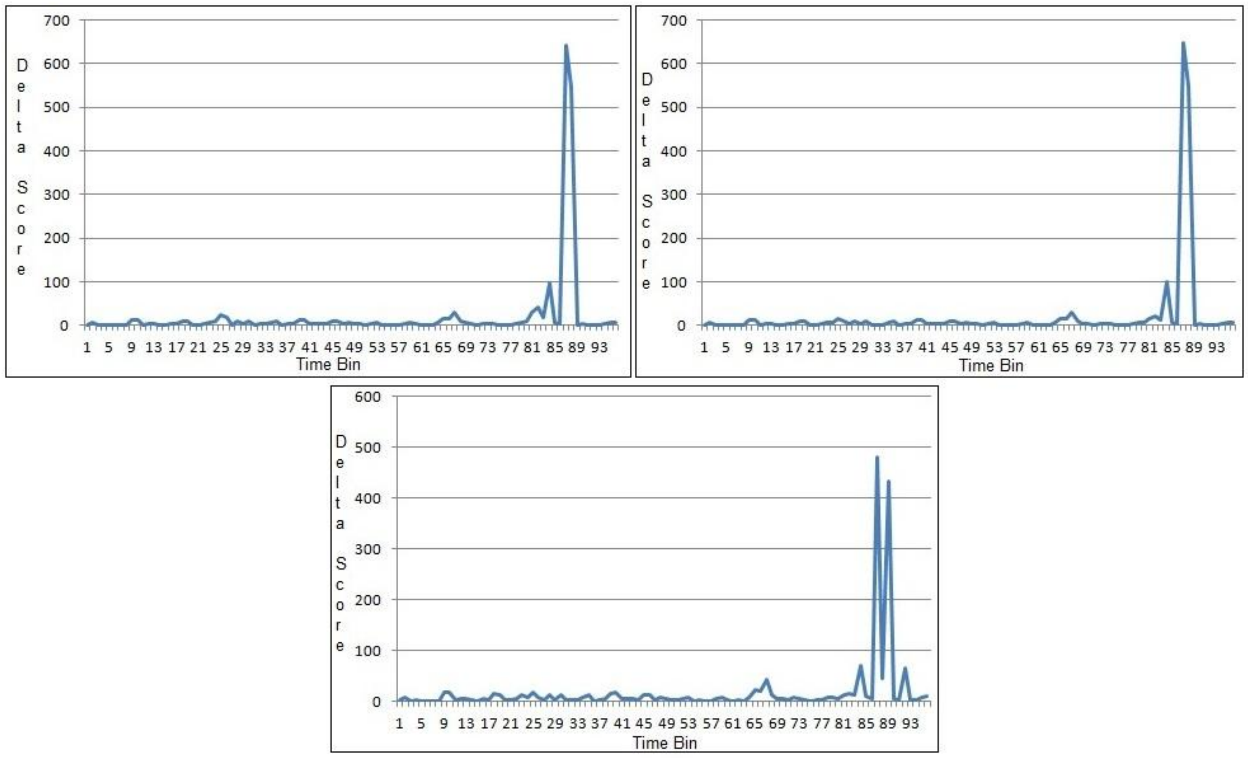


Figure 6.211: March 8, 2009 - Hybrid Delta Scores of 5% (top left), 10% (top right), 20%(bottom) infection rates

Infection Rate	Score	Hit Count (% Hit)	Miss Count (% Miss)
5%	1	62 (65%)	34 (35%)
10%	1	62 (65%)	34 (35%)
20%	1	63 (66%)	36 (34%)

Table 6.33: March 8, 2009 - Hybrid experiment results

6.3.3 Long and Low Infection

This section examines the results of the Hybrid PCA – Haar Wavelet filtering analysis experiments performed on a low infection rate in a short period of time. The four dates for this particular scenario, outlined in Table 5.2, all had consistent behaviour. As a result, for simplicity, only May 28, 2009 is discussed. Figures 6.22 illustrates the footprints observed with default parameters conducted on test sets of 5%, 10% and 20% infection ratios respectively (See Table 5.4 for a list of default conditions). The infected time bins were during bins 61 – 71 and the plots clearly picked up the infected times. Table 6.34 lists the hit/miss scores and the hit/miss counts measured for each feature. The default threshold method determines a miss rate of about 20%. The other threshold methods range from 11% to 40%, see Section 6.3.6 for details

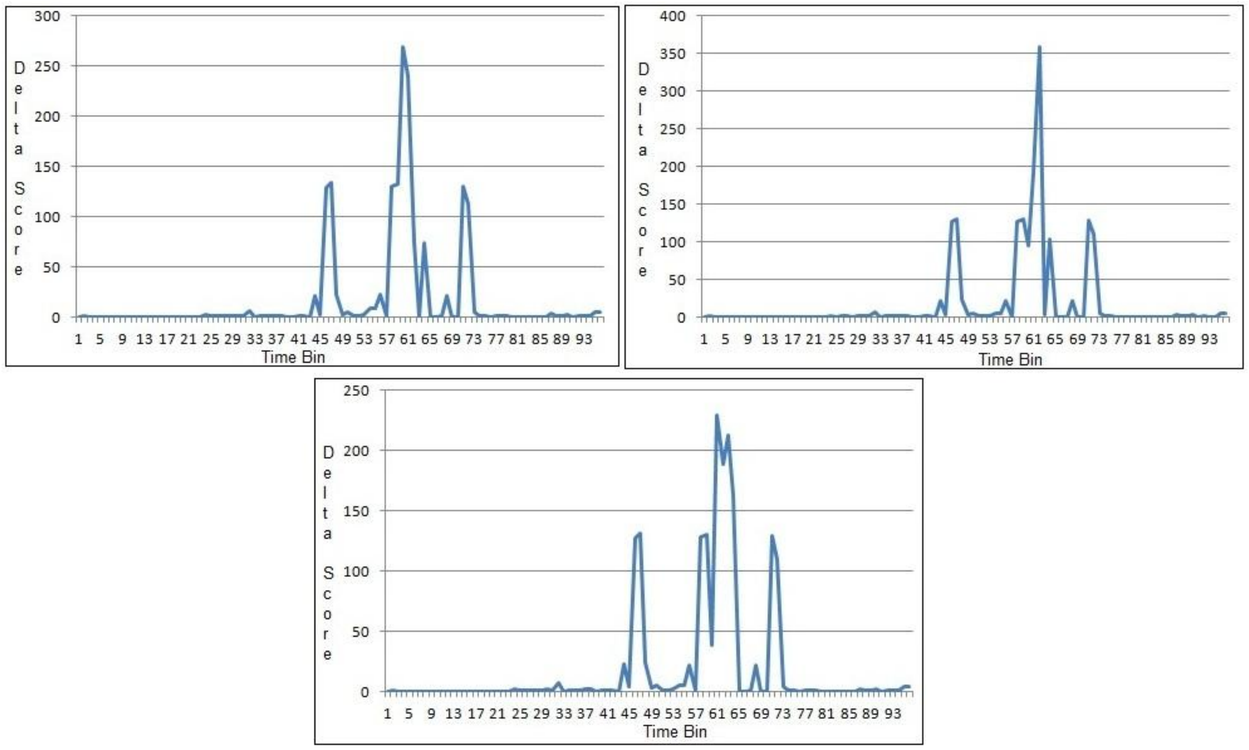


Figure 6.222: May 28, 2009 - Hybrid Delta Scores of 5% (top left), 10% (top right), 20% (bottom) infection rates

Infection Rate	Score	Hit Count (% Hit)	Miss Count (% Miss)
5%	1	71 (74%)	25 (26%)
10%	0	77 (80%)	19 (20%)
20%	2	77 (80%)	19 (20%)

Table 6.34: May 28, 2009 - Hybrid experiment results

6.3.4 Short and High Infection

This section examines the results of the Hybrid PCA – Haar Wavelet filtering analysis experiments performed on a low infection rate in a short period of time. The four dates for this particular scenario, outlined in Table 5.2, all had consistent behaviour. As a result, for simplicity, only January 15, 2009 is discussed. Figures 6.23 illustrates the footprints observed with default parameters conducted on test sets of 75%, 100% and 120% infection ratios respectively (See Table 5.4 for a list of default conditions). The infected time bins were during bins 83, 84, 85, 86 and the plots clearly picked up the infected times. Table 6.35 lists the hit/miss scores and the hit/miss counts measured for each feature. The default threshold method determines a miss rate of about 35%. The other threshold methods range from 4% to 38%, see Section 6.3.6 for details

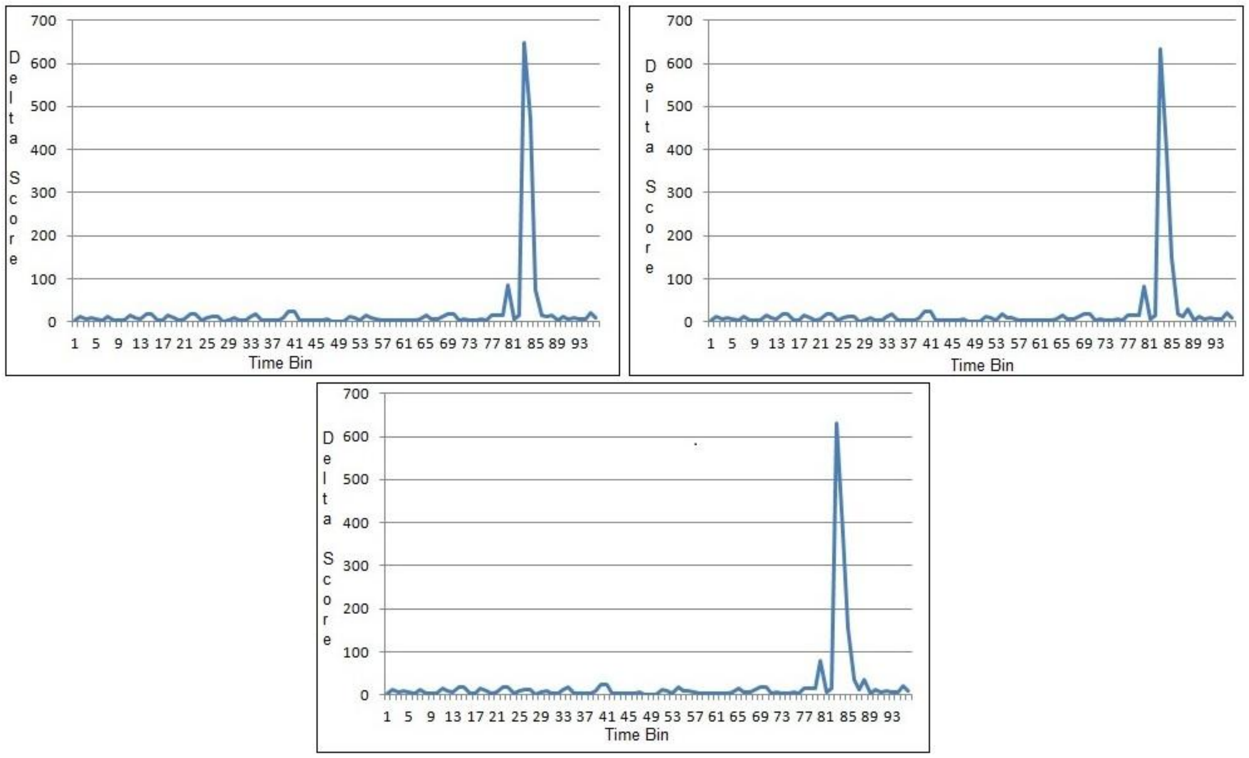


Figure 6.233: January 15, 2009 - Hybrid Delta Scores of 75% (top left), 100% (top right), 120%(bottom) infection rates

Infection Rate	Score	Hit Count (% Hit)	Miss Count (% Miss)
75%	1	61 (64%)	35 (36%)
100%	2	62 (65%)	34 (35%)
120%	3	62 (65%)	34 (35%)

Table 6.35: January 15, 2009 - Hybrid experiment results

6.3.5 Long and High Infection

This section examines the results of the Hybrid PCA – Haar Wavelet filtering analysis experiments performed on a low infection rate in a short period of time. The four dates for this particular scenario, outlined in Table 5.2, all had consistent behaviour. As a result, for simplicity, only June 10, 2009 is discussed. Figures 6.24 illustrates the footprints observed with default parameters conducted on test sets of 75%, 100% and 120% infection ratios respectively (See Table 5.4 for a list of default conditions). The infected time bins were during bins 22 - 32 and the plots clearly picked up the infected times. Table 6.36 lists the hit/miss scores and the hit/miss counts measured for each feature. The default threshold method determines a miss rate of about 25%. The other threshold methods range from 6% to 41%, see Section 6.3.6 for details

Infection Rate	Score	Hit Count (% Hit)	Miss Count (% Miss)
75%	3	73 (76%)	23 (24%)
100%	3	76 (79%)	20 (21%)
120%	2	70 (73%)	26 (27%)

Table 6.36: June 10, 2009 - Hybrid experiment results

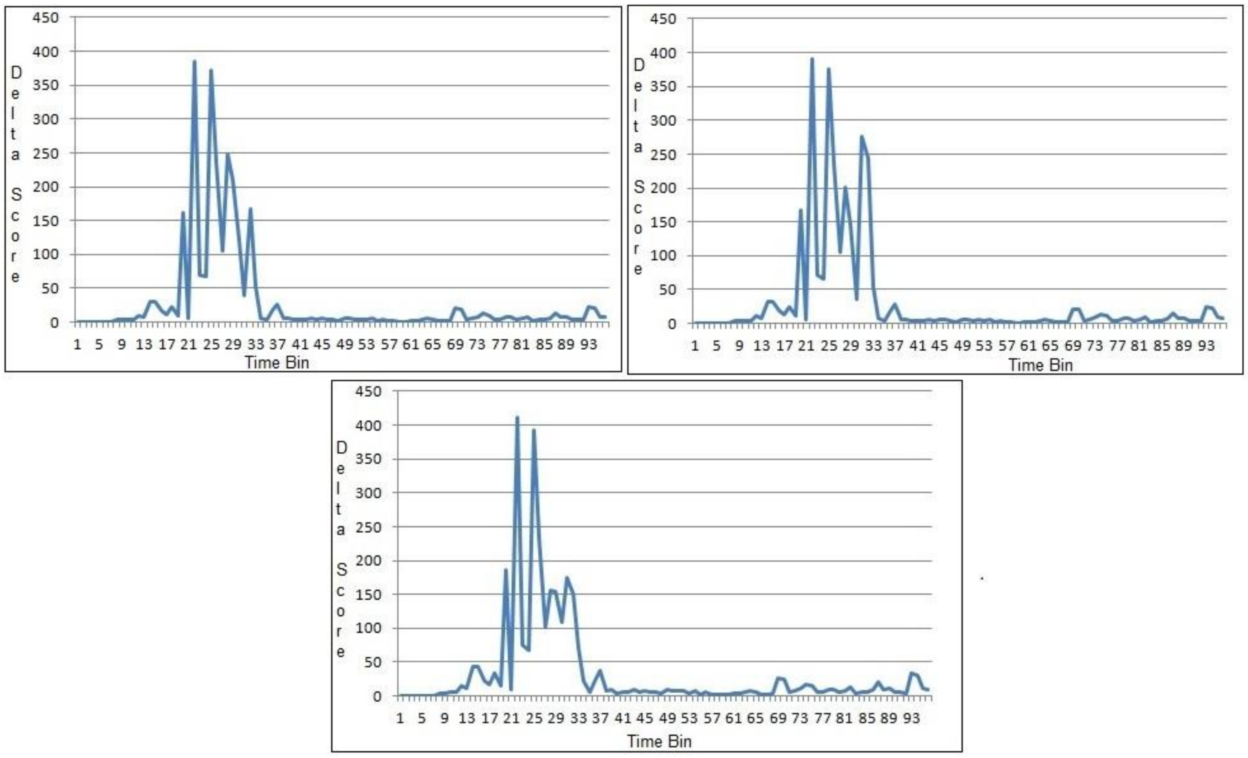


Figure 6.244: June 10, 2009 - Hybrid Delta Scores of 75% (top left), 100% (top right), 120% (bottom) infection rates

6.3.6 Threshold Techniques and Hybrid PCA – Haar Wavelet filtering Analysis

Each of the experiments performed used four different threshold techniques. The results listed in tables 6.37 – 6.40 summarize the evaluation measurements for threshold techniques on the specified dates. The delta scores contain sets of distinctly high scores and are represented by spikes when plotted. The static average was found to perform the best.

Infection Rate	Measurement	Average	Moving Average without outliers	Moving Average	Weighted Average
5%	Score	1	1	1	1
	Hit Count	90	62	63	60
	Miss Count	6	34	33	36
10%	Score	1	1	1	1
	Hit Count	92	62	63	59
	Miss Count	4	34	33	37
20%	Score	2	1	1	1
	Hit Count	89	63	66	63
	Miss Count	7	36	33	36

Table 6.37: March 8, 2009 - Hybrid threshold results

Infection Rate	Measurement	Average	Moving Average without outliers	Moving Average	Weighted Average
5%	Score	2	-1	-21	-1
	Hit Count	84	77	56	69
	Miss Count	12	19	40	27
10%	Score	2	0	-20	0
	Hit Count	84	77	56	74
	Miss Count	12	19	40	24
20%	Score	4	2	-18	2
	Hit Count	85	77	57	76
	Miss Count	11	19	39	23

Table 6.38: May 28, 2009 - Hybrid threshold results

Infection Rate	**Measurement**	**Average**	**Moving Average without outliers**	**Moving Average**	**Weighted Average**
75%	Score	4	1	1	1
	Hit Count	92	61	58	59
	Miss Count	4	35	38	37
100%	Score	2	2	2	2
	Hit Count	91	62	59	60
	Miss Count	5	34	37	36
120%	Score	3	2	2	2
	Hit Count	91	62	59	59
	Miss Count	5	34	37	37

Table 6.39: January 15, 2009 - Hybrid threshold results

Infection Rate	**Measurement**	**Average**	**Moving Average without outliers**	**Moving Average**	**Weighted Average**
75%	Score	9	3	2	4
	Hit Count	90	73	60	66
	Miss Count	6	23	36	30
100%	Score	9	3	2	5
	Hit Count	90	76	60	68
	Miss Count	6	20	36	28
120%	Score	10	2	0	5
	Hit Count	88	73	55	68
	Miss Count	8	26	41	28

Table 6.40: June 10, 2009 - Hybrid threshold results

6.3.7 Summary and Discussion

The Hybrid PCA - Haar Wavelet Filtering analysis is an approach that considers all features and has few input parameters. This method draws upon the strengths of the Modified PCA approach and the Haar Wavelet Filtering analysis. It examines all features using Principle Component Analysis and selects the single best principle component for anomalous subspace creation. It uses the one top component to build the anomalous space. As shown in Modified PCA experimentation, this approach performs best with this dataset. Next, the Haar Wavelet Filtering analysis is applied with input data size of 128. As shown in Haar Wavelet Filtering analysis, applying filter boundaries, and local variability weightings, which have a bias towards high frequency values, performs best. In experimentation, it was able to constantly detect the injections. This approach was able to detect all infected time bins; regardless of the how infected a time bins was or how many time bins were infected (See Table 6.41 for a summary of the differences between the approaches). As the intensity or duration of anomalous behaviour increased, the algorithm performed better.

Characteristic	Modified PCA Approach	Haar Wavelet Approach	Hybrid Approach
Number of input dimensions	All	1	All
Number of output dimensions	1	All (1 for each dimension)	1
Number of tunable user parameters	2	3	0
Overall detection performance	Fair.	Ranges from Good to poor, depending on the dimension.	Good.

Table 6.41: Summary of differences between algorithms

6.4 Threshold Techniques Discussion

In order to automate the anomaly detection analysis, four different threshold techniques were examined. One threshold technique considered the entire dataset and the three remaining threshold techniques considered sliding window. The static average calculates a single cutoff line, while the dynamic techniques form a cutoff function. Depending on the algorithm and the nature of the scores, the techniques performed differently. The Modified PCA algorithm creates scores that have a lot of highs and lows. Also, the extreme highs skew the average value. In this approach, the static average performed the worst. The dynamic average approaches, which considered only a small subset of consecutive values, had much better miss counts. The Haar wavelet approach and the Hybrid approach have distinct regions of high values. Using a sliding window through regions of low values is meaningless. Dynamic thresholds are overly granular in these approaches. The static average was able to calculate an appropriate cutoff to isolate these distinct spikes. By examining the output plots, it is evident that the threshold techniques choice is directly dependant on the algorithm applied.

7.0 Conclusion

This chapter summarizes the thesis and presents conclusions. Future work and directions for further research are also discussed.

7.1 Conclusions and Summary

This thesis introduces a hybrid intrusion detection approach based on anomaly detection methods. The hybrid approach involves using well known statistical analysis and spectral analysis techniques to provide the network administrator time slices of potential network traffic intrusions. Unlike existing Intrusion Detection Systems, which primarily use signature based investigation, a novel technique which used as adaptive model based on flow characteristics was presented. The statistical analysis technique, Modified Principle Component Analysis, considers data trends and creates a new set of axis to better describe the dataset. It projects data onto the anomalous subspace and provides an anomalous score for each time bin. Two parameters investigated in this research were the number of time shifts and how the components were selected for the anomalous subspace. It was concluded that the features in the Kyoto2006+ dataset are highly correlated. Building a subspace of components that contains 99% of the normal space performed the best in experimentation. The difference of observation values between time bins changes a fair amount and it was determined that considering more than one time shift degraded performance. For anomaly detection, the algorithm calculated a Square Prediction Error (SPE) score for an indication of how anomalous each time bin was. Experiments were conducted with various time bin infection rates and over a short and long number of infected time bins. The PCA approach was able to produce higher SPE score for infected time slices; however, these values were not large enough to stand out. The scores were not one of the dominant spikes when plotted.

Secondly, the spectral analysis technique used Haar Wavelet Filtering analysis to determine abrupt changes in the data over time. It provides a score to show how anomalous each time bin may be. Three parameters investigated in this research were the number of time bins fed into the algorithm, the filtering boundaries and the frequency component weightings for score calculation. Due to the nature of the Haar Wavelet Filtering, the input size had to be a power of two. Considering the Kyoto2006+ dataset volume of observations, the optimal number of input time bins, through experimentation, was determined to be 128. The time bins were large enough to be meaningful. The spectral analysis algorithm modeled high pass values to represent the start /end time and the mid pass values to represent the duration of malicious behaviour. Through experimentation, boundaries where the high pass region consisted of the first two levels, the mid pass region consisted of the next three levels and the low pass regions consisted of the remaining levels performed best. These specific boundaries resulted create a preference on high pass frequency values. The local variability weights applied a further bias to each of the filtered values to determine a single scalar score. Once again, the weights heavier on high pass performed best. Considering this trend, it can be concluded that in the Kyoto2006+ dataset has significant fluctuation of feature values and are best isolated by high frequency analysis. The Haar Wavelet Filtering approach performed independent feature analysis and produces several independent scores. The experiments examined different days in week, different times, various infections ratios and various amount of infected time bins. Under all circumstances tested, the analysis was able to isolate the infected time bins in at least one of the feature analysis. When examining individual experiment results, it was observed that at least one of the feature results identified the infected time bins ideally; yet other features were completely wrong. Without knowing where the anomalies are, it would be very difficult to determine which features accurately picked up the anomalies. The Haar Wavelet Filtering approach performed much better than the Modified PCA approach; however, it is very difficult to determine which of the feature plots to consider.

Both the Modified PCA and the Haar Wavelet Filtering approach detected the infected time bins. Both approaches had their strengths and drawbacks. The Hybrid analysis draws upon the strengths of each approach and combines them in a complementary fashion. The Hybrid analysis considers the entire feature set, as in modified PCA, and performs an in depth analysis, as in Haar Wavelet Filtering. Similar to the previous two algorithms, the hybrid approach was tested on several dates, using various infection rates and different number of consecutive infected time bins. In all circumstances, the Hybrid approach detected all the infected times bins. It produced a single output score that distinctly identified regions of anomalous behaviour.

An automated detection approach was investigated and various threshold techniques were applied to the output scores of each of the algorithms. In the modified PCA approach, where the plotted output had many regions of high spikes, it was observed that moving thresholds performed best. Applying threshold techniques to smaller portions to determine local spikes performed much better than considering the entire dataset. Conversely, the hybrid and wavelet analysis had a distinct and localized region of high values. The remaining scores were all very low. In the experiments using these algorithms, the static average performed the best. One threshold approach did not always perform the best. Depending on which algorithm is applied, one threshold technique would perform better the others.

In conclusion, applying the modified PCA approach on the Kyoto2006+ dataset performed modestly well. Applying Haar Wavelet Filtering on the Kyoto2006+ better detected the injected anomalous flows; however, due to the amount of independent feature analysis, it is not very practical. The Hybrid approach used the strengths of both previous approaches. This algorithm was applied on the Kyoto2006+ dataset and it was always able to consistently identify the anomalous time bins. It can be concluded that the Hybrid PCA – Haar Wavelet Filtering approach performed the best.

7.2 Future Work

Directions for future research are discussed next.

- Investigate techniques to isolate specific hosts or sources within time bins. Our approach suggests a specific infected time bin, and it would be very useful to provide an indication of which hosts within this time behaved anomalously.
- Devise techniques to isolate specific characteristics, which contributed to the anomaly. The major benefit of the Haar Wavelet approach was that it specifically listed the feature interest. Due to the nature of the Modified PCA approach and the Hybrid approach, semantic meaning is lost.
- Investigate multiple infected time bins regions. Our analysis examined the base cases of a single infected region of a short or long time. It would be useful to examine how the algorithms perform one datasets with multiple infected regions.
- Apply anomaly detection algorithms on various datasets. There are a number of different types of dataset available and each have their limitations. Some datasets have higher traffic volume and different are characterized by different features.
- Apply the profiling phase to involve to TCP and UDP network protocols. Due to availability of data, this research involved dividing between TCP and OTHER, all remaining, protocols. When considering a different dataset, the profiling could be divided between TCP and UDP network protocols.

This research was built upon existing research. It starts with known statistical and spectral analysis techniques and builds upon them. The result of these algorithms provides an indication of which time bins may have malicious behaviour, and it would be beneficial to provide the network administrator more information as to which sources are infected, or to which features are creating the anomaly. Furthermore, the experiments showed promising result under base cases with the novel labeled dataset. It would be interesting to apply the anomaly detection algorithms on different dataset with higher data volume, different features, and different protocol partitions. This research provides was a good starting point in comparing the performance of the different anomaly detection algorithms in network traffic data.

References

[1] J. M. Estevez-Tapiador, P. Garcia-Teodoro, and J. E. Diaz-Verdejo, “Anomaly detection methods in wired networks: a survey and taxonomy,” *Computer Commun.*, vol. 27, no. 16, pp. 1569–1584, 2004.

[2] *Introduction to Cisco ISO NetFlow - A Technical Overview.* [Online]. Available at: http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_white_paper0900aecd80406232.html [Last Accessed: March 2012].

[3] *Netflow version 9.* [Online]. Available at: http://www.cisco.com/en/US/products/ps6645/products_ios_protocol_option_home.html [Last Accessed: March 2012]

[4] *The Bro Security Network Monitor.* [Online]. Available at: http://www.bro-ids.org [Last Accessed: March 2012]

[5] M. Campos and B. Milenova, “Creation and deployment of data mining-based intrusion detection systems in oracle database l0g,” in *Proc. 4th Int. Conf. Mach. Learning and Applicat.,* pp. 8–15, 2005.

[6] D. Brauckhoff, K. Salamatian, and M. May, “Applying PCA for traffic anomaly detection: Problems and solutions,” in *Proc. of the IEEE INFOCOM*, pp. 2866–2870, 2009.

[7] P. Barford, J. Kline, D. Plonka, and A. Ron, “A signal analysis of network traffic anomalies,” in *Proc. of the 2nd ACM SIGCOMM Workshop on Internet measurement*, pp. 71–82, 2002.

[8] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, “Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation,” in *Proc. of the 1st Workshop on Building Anal. Datasets and Gathering Experience Returns for Security,* pp. 29–36, 2011.

[9] J. Zhang and M. Zulkernine, “A hybrid network intrusion detection technique using random forests,” in *Proc. of the 1st Int. Conf. Availability, Reliability and Security*, pp. 8–15, 2006.

[10] Y.T. Chan, *Wavelet Basics.* Boston, MA: Kluwer Academic Publishers, 1995.

[11] B. Vidakovic and P. Mueller. (1994). *Wavelets for kids* [Online]. Available at: www2.isye.gatech.edu/~brani/wp/kidsA.pdf [Last Accessed: March 2012]

[12] J. Goswami and A. Chan, *Fundamental of Wavelets – Theory, Algorithms and Applications*. Toronto, Canada: Wiley, 1999.

[13] A. Webb, *Statistical Pattern Recognition, 2nd ed*. West Sussex, England: Wiley, 2002.

[14] W. Stallings, *Data and Computer Communications, 9th ed.* Toronto, Canada: Prentice Hall, 2011.

[15] H. Ringberg, A. Soule, J. Rexford, and C. Diot, “Sensitivity of PCA for traffic anomaly detection,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, no. 1, pp. 109–120, 2007.

[16] A. Lakhina, M. Crovella, and C. Diot, “Diagnosing network-wide traffic anomalies,” in *ACM SIGCOMM Computer Commun. Review*, vol. 34, pp. 219–230, 2004.

[17] *KDD 1999 Cup Data*. [Online]. Available at: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [Last Accessed: March 2012]

[18] F. Wanner, “Anomaly Analysis using Host-behavior Clustering,” M.S. thesis, ETH–Swiss Federal Institute of Technology Zurich, 2007.

[19] K. Nyalkalkar, S. Sinhay, M. Bailey, and F. Jahanian, “A comparative study of two network-based anomaly detection methods,” in *Proc. of the IEEE INFOCOM*, pp. 176–180, 2011.

[20] L. Smith, *A Tutorial on Principal Components Analysis* [Online]. Available at: http://www.sccg.sk/~haladova/principal_components.pdf [Last Accessed: March 2012]

[21] R. Dunia and S. J. Qin, “Multi-dimensional fault diagnosis using a subspace approach,” in *American Control Conf.*, 1997.

[22] S. Jungsuk, H. Takakura, Y. Okabe, D. Inoue, E. T. O. Masashi, and K. Nakao, “A Comparative Study of Unsupervised Anomaly Detection Techniques Using Honeypot Data,” in *IEICE Trans. Inform. and Syst.*, vol. 93, no. 9, pp. 2544–2554, 2010.

[23] J. Song, H. Takakura, and Y. Kwon, “A Generalized Feature Extraction Scheme to Detect 0-Day Attacks via IDS Alerts,” in *Int. Symp. Applicat. and the Internet*, pp. 55–61, 2008.

[24] *SNORT*. [Online]. Available at: http://www.snort.org/ [Last Accessed: March 2012]

[25] L. Zheng, P. Zou, Y. Jia, and W. Han, “Traffic Anomaly Detection and Containment Using Filter-Ary-Sketch,” in *Procedia Engineering*, vol. 29, pp. 4297–4306, 2012.

[26] J. Song, H. Takakura, Y. Okabe, and K. Nakao. (2011, Aug.). “Toward a more practical unsupervised anomaly detection system.” *Inform. Sciences*. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025511004245 [Last Accessed: March 2012]

[27] A. Dainotti, A. Pescapé, and G. Ventre, “Wavelet-based detection of DoS attacks,” in *Proc. of the IEEE Global Telecommun. Conf. (GLOBECOM)*, pp. 1–6, 2006.

[28] M. Salagean, “Real network traffic anomaly detection based on Analytical Discrete Wavelet Transform,” in *Proc. of the 12th Int. Conf. Optimization of Elect. and Electron. Equipment (OPTIM),* pp. 926–931, 2010.

[29] M. Salagean and I. Firoiu, “Anomaly detection of network traffic based on Analytical Discrete Wavelet Transform,” in *Proc. of the 8th Int. Conference Commun.*, pp. 49–52, 2010.

[30] A. Lakhina, M. Crovella, and C. Diot, “Characterization of network-wide anomalies in traffic flows,” in *Proc. of the 4th ACM SIGCOMM conf. on Internet measurement,* pp. 201–206, 2004.

[31] B. Whitehead, C. H. Lung, and P. Rabinovitch, “Tracking per-flow state—Binned Duration Flow Tracking,” in *Proc. of the Int. Symp. Performance Evaluation of Comput. and Telecommun. Syst. (SPECTS)*, pp. 73–80, 2010.

[32] C. Callegari, S. Giordano, and M. Pagano, “Application of Wavelet Packet Transform to Network Anomaly Detection,” in *Next Generation Teletraffic and Wired/Wireless Advanced Networking*, pp. 246–257, 2008.

[33] A. Soule, K. Salamatian, and N. Taft, “Combining filtering and statistical methods for anomaly detection,” in *Proc. of the 5th ACM SIGCOMM conf. on Internet Measurement*, pp. 31–31, 2005.

[34] *Flex.* [Online]. Available at: http://www.adobe.com/products/flex.html [Last Accessed: March 2012]

[35] *Apache Tomcat.* [Online]. Available at: http://tomcat.apache.org/ [Last Accessed: March 2012]