INL/EXT-13-28273

SCADA Protocol Anomaly Detection Utilizing Compression (SPADUC) 2013

Gordon Rueff
Lyle Roybal
Denis Vollmer

January 2013

The INL is a U.S. Department of Energy National Laboratory
operated by Battelle Energy Alliance

INL/EXT-13-28273

SCADA Protocol Anomaly Detection Utilizing Compression (SPADUC) 2013

Gordon Rueff
Lyle Roybal
Denis Volmer

January 2013

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Electricity Delivery and Energy Reliability
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

ABSTRACT

There is a significant need to protect the nation's energy infrastructures from malicious actors using cyber methods. Supervisory, Control, and Data Acquisition (SCADA) systems may be vulnerable due to the insufficient security implemented during the design and deployment of these control systems. This is particularly true in older legacy SCADA systems that are still commonly in use. The purpose of INL's research on the SCADA Protocol Anomaly Detection Utilizing Compression (SPADUC) project was to determine if and how data compression techniques could be used to identify and protect SCADA systems from cyber attacks. Initially, the concept was centered on how to train a compression algorithm to recognize normal control system traffic versus hostile network traffic. Because large portions of the TCP/IP message traffic (called packets) are repetitive, the concept of using compression techniques to differentiate "non-normal" traffic was proposed. In this manner, malicious SCADA traffic could be identified at the packet level prior to completing its payload. Previous research has shown that SCADA network traffic has traits desirable for compression analysis. This work investigated three different approaches to identify malicious SCADA network traffic using compression techniques. The preliminary analyses and results presented herein are clearly able to differentiate normal from malicious network traffic at the packet level at a very high confidence level for the conditions tested. Additionally, the master dictionary approach used in this research appears to initially provide a meaningful way to categorize and compare packets within a communication channel.

EXECUTIVE SUMMARY

Securing the country's energy sector infrastructure from cyber-attack is critical to the well-being of the American people and is a central focus to the Department of Energy's (DOE) Office of Electricity Delivery and Energy Reliability (OE) Cybersecurity for Energy Delivery Systems (CEDS) program. The DOE program aims to enhance the reliability and resilience of the nation's energy infrastructure by reducing the risk of energy disruptions due to cyber-attacks.

The purpose of INL's research on the SCADA (Supervisory Control and Data Acquisition) Protocol Anomaly Detection Utilizing Compression (SPADUC) project was to investigate how data compression algorithms could be used to identify cyber attacks on SCADA networks. The initial premise was to specifically see if a compression algorithm could be trained to differentiate normal control system traffic versus hostile network traffic. Because large portions of the TCP/IP message traffic (called packets) are repetitive, the concept of using compression techniques to differentiate traffic that is "non-normal" was proposed as a way to identify and quarantine malicious traffic at the packet level on a SCADA network before its payload is completed.

An open source implementation of a Lempel-Ziv-Welch (LZW) lossless data compression algorithm was used to analyze surrogate but typical SCADA network traffic. LZW compression algorithms are a well known and mature technology used to compress data of any sort (binary or ASCII). This is the technique used by commercial product WinZip in the Personal Computer world. LZW techniques compress data by using a dictionary technique to store repetitive sequences of bytes as they occur in the data stream. The dictionary is initialized with a default set of entities that usually consists of the 256 ASCII character set as a starting block. When a file or a TCP/IP packet (consisting of tens to hundreds of bytes) is compressed, this dictionary is added to every time a new sequence of characters is encountered.

Preliminary research shows that SCADA network traffic has traits desirable for using compression analysis as a method to identify abnormal network traffic. Primarily, SCADA network traffic is often very repetitive with only minor differences between packets. This results in highly compressible data streams that generally show improved compression ratios in subsequent packets. Two difficulties in using traditional compression techniques were identified. The first is that inter-dependence in the compression ratio metric between packets violates the use of statistical analysis needed to differentiate good from bad traffic. The second is that not all of the packets in a communication channel compress well together due to different packet contents. The use of compression techniques to identify malicious traffic on SCADA networks in real time appears to have some significant merit for infrastructure protection. The

preliminary analyses and results presented herein are clearly able to identify malicious network traffic at the packet level at a very high confidence level for the conditions tested. However, the conditions tested are rather limited in scope and should be expanded into more realistic simulations of hacking events using techniques and approaches that are representative of a real-world attack on a SCADA system.

The master dictionary approach used in this research looks like it will provide a meaningful way to categorize and compare packets within a communication channel. The analysis is subjective for now so future research will be necessary to quantify the packet groupings and then introduce malicious packet comparisons to get false positive and false negative rates.

CONTENTS

FIGURES

ACRONYMS

ASCII	American Standard Code for Information Interchange
CEDS	Cybersecurity for Energy Delivery Systems
DNP	Distributed Network Protocol
DOE-OE	Department of Energy Office of Electricity Delivery and Energy Reliability
DoS	Denial of Service
ICS	Industrial Control Systems
ILAO	Input Length Adjusted Output
IP	Internet Protocol
LZW	Lempel-Ziv-Welch
MSSQL	Microsoft Structured Query Language
PCAP	Packet Capture
PN	Packet Number
PTD	Packet Type Designator
R&D	Research and Development
SCADA	Supervisory Control and Data Acquisition
SPADUC	SCADA Protocol Anomaly Detection Utilizing Compression
TCP	Transmission Control Protocol

SCADA Protocol Anomaly Detection Using Compression Techniques (SCADA Protocol Anomaly Detection Utilizing Compression (SPADUC) 2013

1. SCADA PROTOCOL ANOMALY DETECTION PROJECT

The INL SCADA Protocol Anomaly Detection Utilizing Compression (SPADUC) Project is funded through the Department of Energy's (DOE) Office of Electricity Delivery and Energy Reliability (OE) Cybersecurity for Energy Delivery Systems (CEDS) Research and Development (R&D) Program. The DOE program aims to enhance the reliability and resilience of the nation's energy infrastructure by reducing the risk of energy disruptions due to cyber-attacks. The project is a one year effort, started in FY 2012, awarded as part of the INL NSTB Core Frontier R&D tasks.

2. INTRODUCTION

The purpose of INL's research on the SCADA Protocol Anomaly Detection Utilizing Compression (SPADUC) project is to investigate whether a compression algorithm can be trained to recognize proper control system traffic versus hostile network traffic going toward the same service. Supervisory Control and Data Acquisition (SCADA) systems operating over TCP/IP protocols in network hardware typically contain several static layers of headers. The ratio of header size to control or response data is often very high. Moreover, the types of headers are often limited because of the repetitive nature of SCADA networks. Control of hardware and status of systems and sensors often occur along regularly timed sequences. Therefore, network traffic on dedicated SCADA systems and at the boundaries of SCADA systems where data transfer takes place tend to be of a monotonous nature. Because large portions of the TCP/IP message traffic (called packets) are repetitive, the concept of using compression techniques to differentiate traffic that is "non-normal" was proposed as a way to identify and quarantine malicious traffic at the packet level on a SCADA network before its payload is completed. An open source implementation of a Lempel-Ziv-Welch (LZW) lossless data compression algorithm was also proposed to analyze surrogate but typical SCADA network traffic to see if malicious traffic could be differentiated from normal SCADA traffic.

LZW compression algorithms are a well known and mature technology used to compress data of any sort (binary or ASCII). It is the technique used by commercial product WinZip in the Personal Computer world. LZW techniques compress data by using a dictionary technique to store repetitive sequences of bytes as they occur in the data stream. The dictionary is initialized with a default set of entities that usually consists of the 256 ASCII character set as a starting block. When a file or a TCP/IP packet (consisting of tens to hundreds of bytes) is compressed, this dictionary is added to every time a new sequence of characters is encountered. So for example, in a word document that has a lot of repeated words, a dictionary for the word "tomorrow" could be added and every time "tomorrow" is encountered in the document, it can be represented by a single dictionary entry that may be between 8 and 20 bits long depending on where it sits in the dictionary and how large the dictionary is allowed to grow. In this manner, the word "tomorrow" is 8 bytes or 64 bits long and can be represented by 10 bits, for example. Therefore the compression of this word is 10/64 and is 16% the size of the original entity in this crude example.

2.1 Data Collection and Surrogate Data

2.1.1 Normal Traffic Data Set

The normal traffic used for this research was collected from a production SCADA network using a mirror port on the networking equipment. The traffic was sorted into channels by service and client as defined by the server IP address, server port, protocol, and client IP address. The five most verbose of these channels were then selected for further analysis.

2.1.2 Malicious Data Set

In order to simulate malicious traffic, the researchers decided to concentrate on identifying a particular type of exploit. Based on the hypotheses being tested, exploits that result in code execution and require a payload to execute would be the easiest malicious traffic to identify. Other types of malicious traffic, such as spoofed values or denial of service (DoS) based on simple invalid fields, would probably not be identifiable from the compression ratio of the packet because the contents of the packet do not change much. The researchers inserted metasploit payloads into packets of the otherwise normal traffic.

2.2 Data Analysis Techniques Used

Several analysis techniques involving LZW compression of the network packet data were investigated. The proposed metric used to characterize the data streams on a packet-by-packet basis was the number of "encodes" generated when a packet is compressed. An encode is an entry into the LZW dictionary database; each encode represents a unique sequence of bytes in the packet. Therefore, a packet that generates a higher number of encodes can be considered less compressible than one that generates fewer encodes. It has been suggested by previous research[ab] that malware has a higher entropy value than normal data and is therefore less compressible using LZW techniques. This means that more "encodes" are generated in data containing malware or portions of malware.

3. PRELIMINARY RESEARCH

3.1.1 Compression Attribute Identification

The goal of this initial work was to look for patterns in the compression and data from a DNP3 network capture obtained from an INL assessment. If the packets contain similar content, then the compression ratio should improve as more packets are compressed and added to the dictionary.

This set of network traffic contains 2,176 DNP3 packets. These packets come in six different sizes of TCP data lengths: 120, 136, 184, 200, 584, and 736 bits. The analysis was performed using an unmodified LZW library that reports output length in bits instead of integer codes. The packets had each TCP data section passed to the LZW compression routine in sequence, with the dictionary retained between each packet. The results are shown in Figure 1.

---

[a] Zhou, Y., Inge, M., Malware Detection Using Adaptive Data Compression.

[b] Lyda, R., Hamrock, J., Using Entropy Analysis to Find Encrypted and Packed Malware, IEEE Computer Society, IEEE Security and Privacy, 2007.

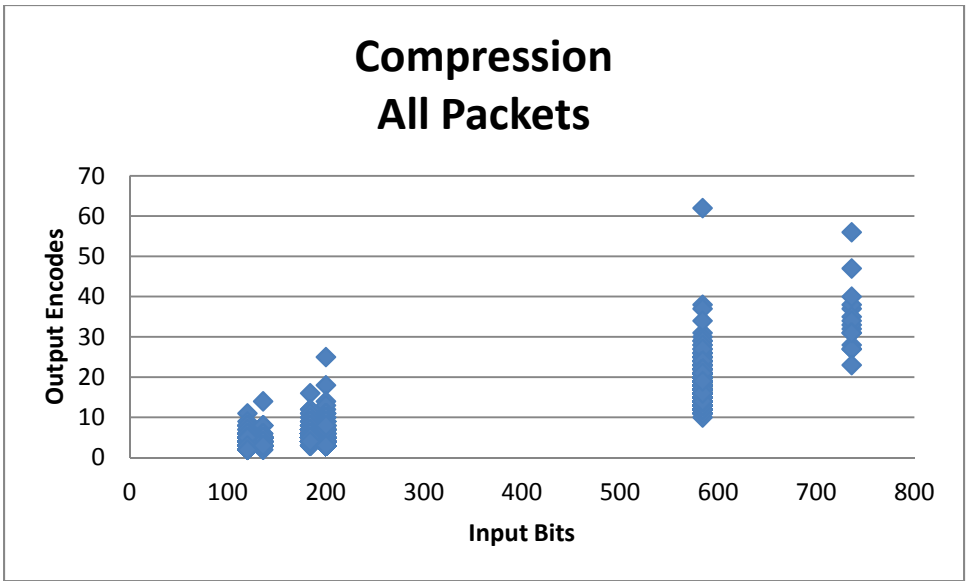

Figure 1. LZW Compression of all DNP3 packets.

The LZW algorithm allows an encode to grow the dictionary tree one node at a time. Over the first *n* packets the algorithm could contain encode sequences *n* bytes long if the sequence is observed in every single packet. If a pattern is observed multiple times in a single packet, the encoding could grow even faster. The first packet in the sequence had 200 input bits and 25 output encodes. This means that no sequences in this packet were observed multiple times.

The largest packet in this data set was 736 bits, or 92 bytes, so the research team estimated, roughly, that by the 100th packet, the LZW dictionary would include the most common sequences of bytes and the output encode sizes would be roughly stable. Figure 2 contains the compression results of all data after the first 100 packets. The absolute range of these results for each packet input size is significantly reduced but still undesirably large.

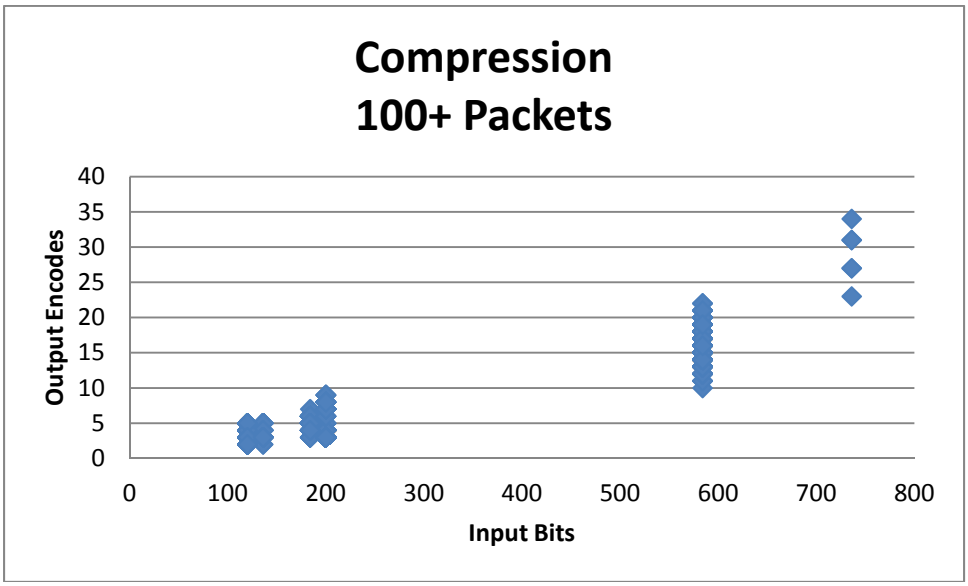

Figure 2. LZW Compression of packets after the first hundred.

Given the still rather large variance in the 100+ packets, compression sizes alone will probably not be sufficient to categorize normal and abnormal packets.

A common problem when using a learning algorithm to detect malicious activity is that a malicious actor could gradually introduce activity so that any algorithm would be incapable of identifying malicious activity. This compression algorithm will have the same the problem; however, if the dictionary can be locked after a set training time, the chances of this happening can be reduced.

The LZW algorithm has a direct relationship between the output encodes per packet and the dictionary size growth. Each output encode per packet except the last one grows the dictionary by 1 node. Therefore, a packet with 25 encodes will create 24 nodes in the dictionary tree.

Figure 3 shows the dictionary size in nodes after each packet is encoded. The long-term trend seems to be an approximately linear function with some non-linear behavior at the beginning.

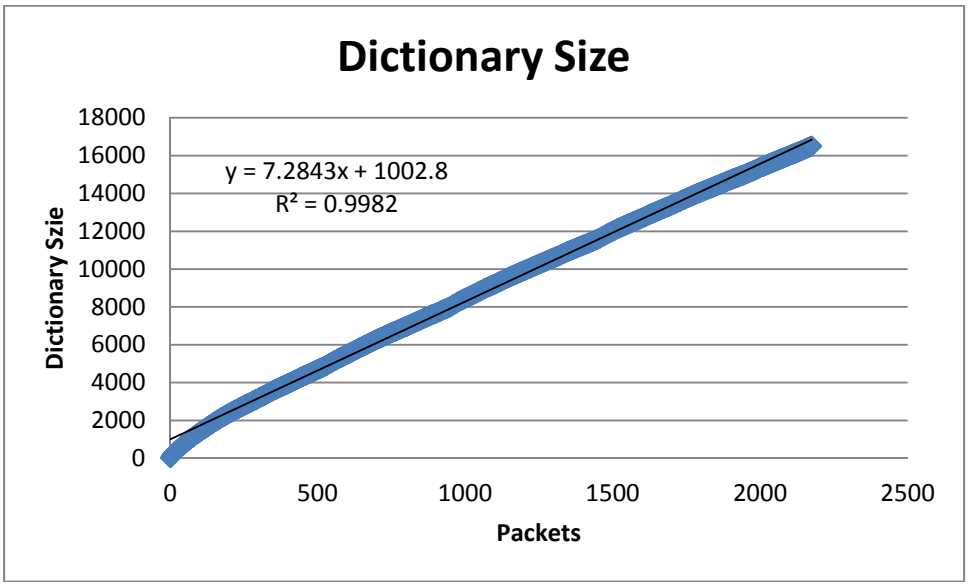

Figure 3. Dictionary size (nodes).

Figure 4 examines the dictionary size more closely during the first 57 packets in the data sequence. The slope of the trend line indicates that the first 57 packets are adding nearly twice as many nodes per packet as the average at the end of the data. This would suggest that the most useful nodes are added early in the dictionary construction.

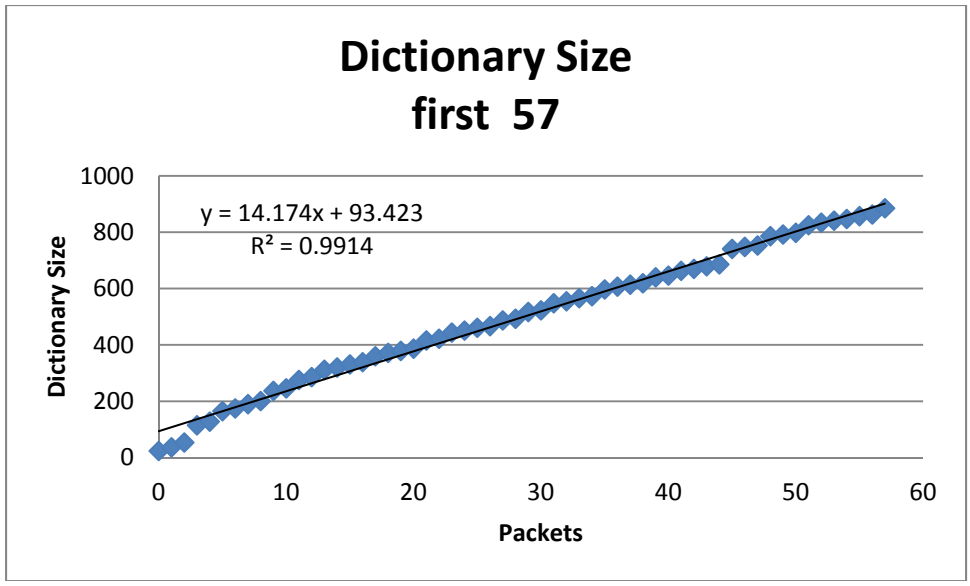

Figure 4. Dictionary size for the first 57 packets.

In order to highlight the change in dictionary growth early in the packet stream versus later, Figure 5 shows the dictionary size divided by number of encoded packets after encoding each packet. The average encoding size is decreasing continuously, which would hint that new nodes formed even by later packets are being used by subsequent packets; however, most of the highly useful nodes are probably formed early on. This implies that the dictionary can be locked early on in the process without significant impact on the compression results.

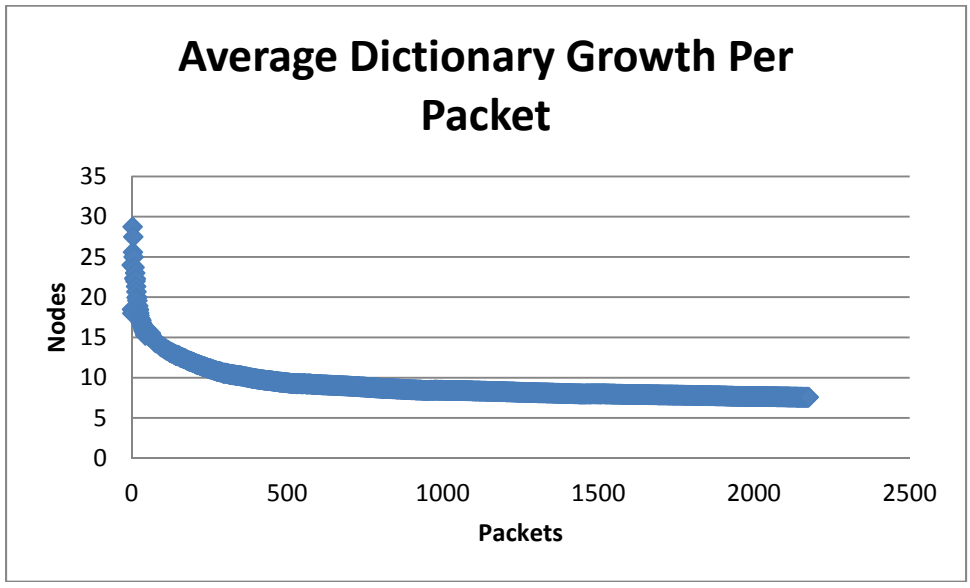

Figure 5. Average dictionary growth.

3.1.2 Production Data Set

It was decided that the initial assessment data was too small for statistical analysis and not representative of real data, so a number of SCADA operators were solicited to provide data for this research. One of them provided PCAP data on terms of anonymity.

3.1.2.1 TCP Port 1433

The first 100 packets that contained the TCP data are shown in Figure 6. They follow a very nice asymptotic curve with every data point falling into a nice distribution. The first 32,000 TCP data packets are shown in Figure 7. These data indicate that outliers begin appearing between packet 100 and 1,000.

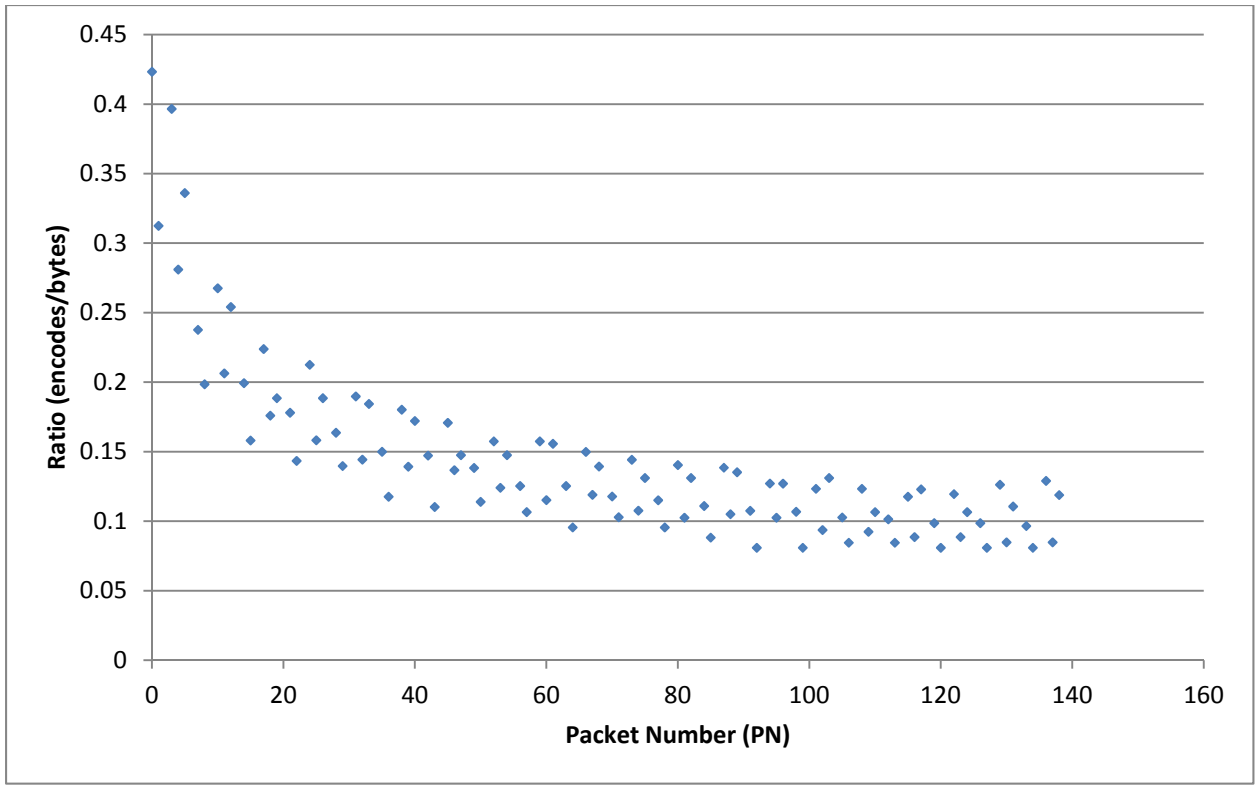

Figure 6. First 100 data packets on TCP port 1433.

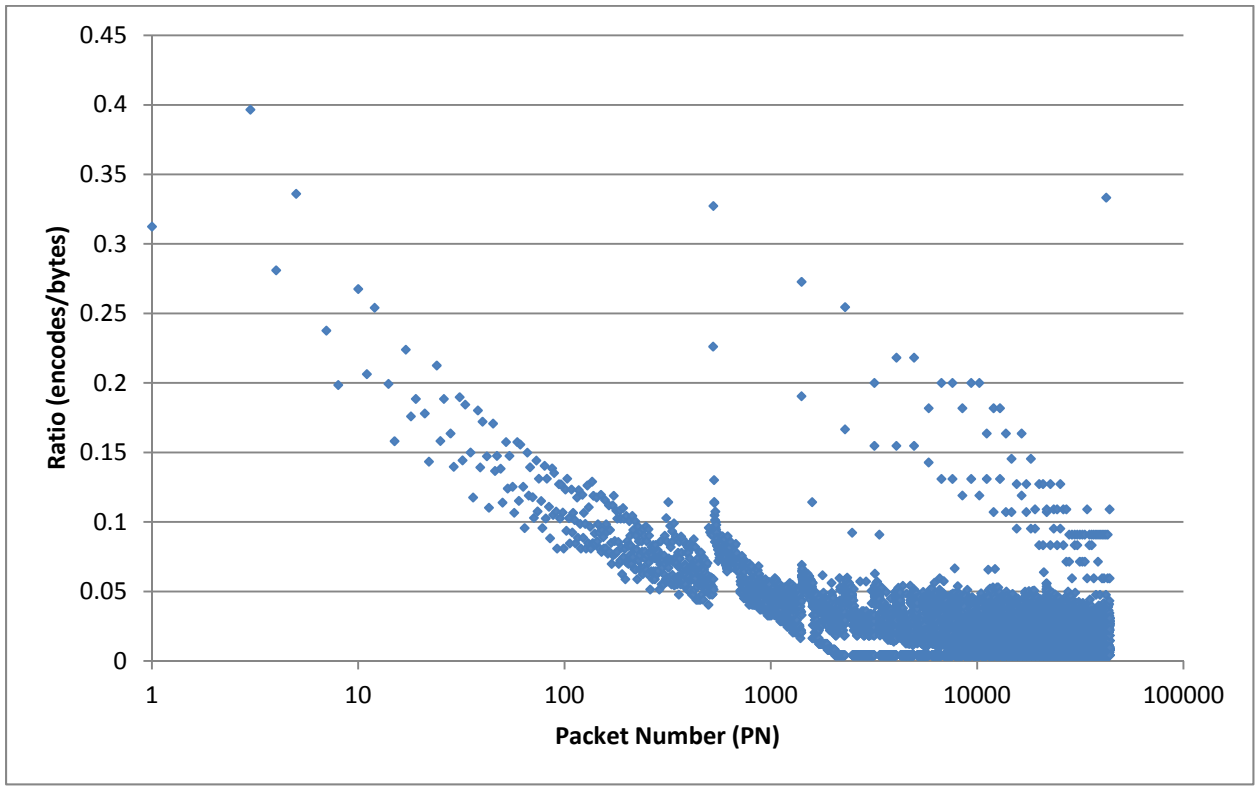

Figure 7. First 32,000 data packets on TCP port 1433.

3.1.2.2 TCP Port 5026

Data for TCP Port 5026 are shown in Figure 8 and Figure 9. A subjective analysis of the compression data resulted in several observations. The first 100 data packets of this protocol form two very nice but separate curves. It is possible that one curve is for the client destined packets and the other curve is the server. The first 32,000 data packets do not look like the first 100, but there appears to be a continuation of its curves. Three additional curves show up in this graph. The last two curves eventually merge into the first two, but the third curve creates a bar between the 0.2 and 0.25 compression ratios. There are actually four additional curves but the last two blend together due to the logarithmic nature of the graph.

Eventually some horizontal lines show up in the first curve, which could be attributed to over training or specialization. In this case, there is a 54 byte input packet that compresses to somewhere in between 2 and 8 encodes, depending on how much is an identical match to previous packets. The "bar" of points seems to match the particular packet length of 382 and actually occurs several times early on in the data but blends with packets of length 1,286 at this point.

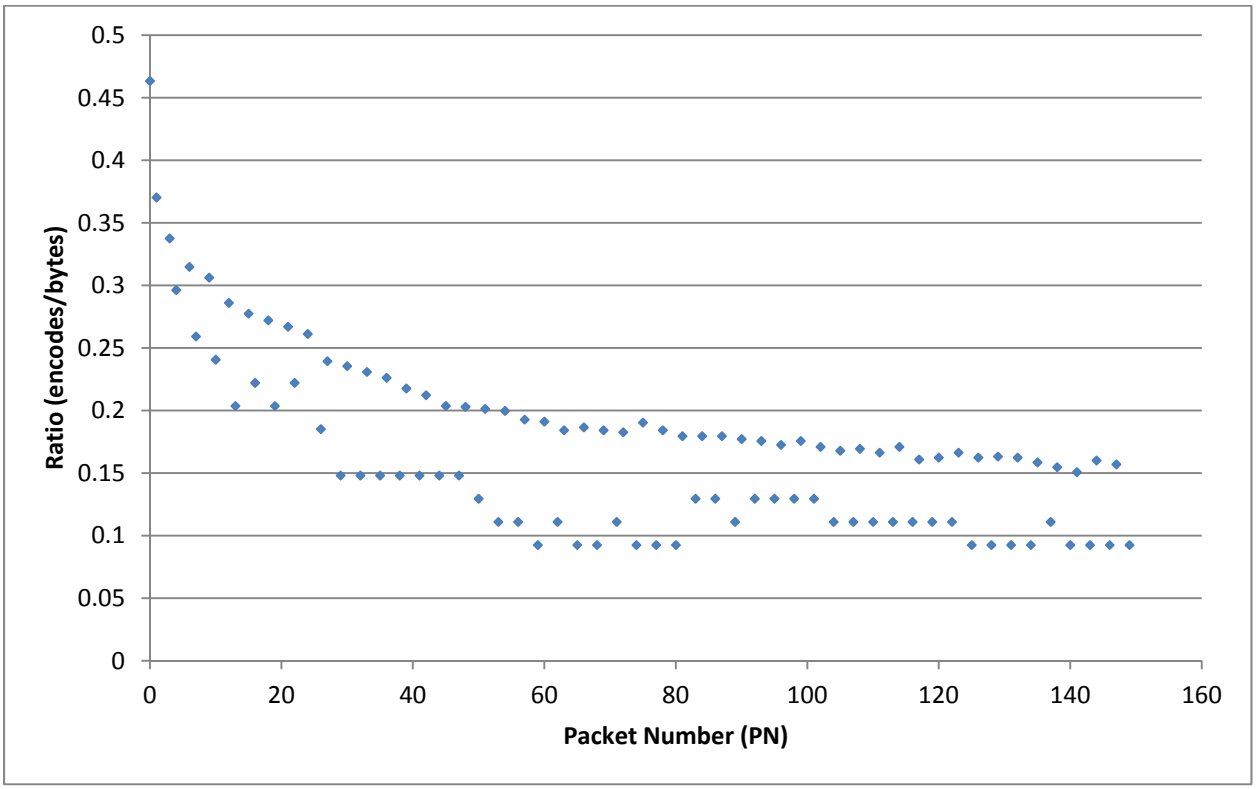

Figure 8. Compression ratio of first 100 data packets on TCP port 5026.

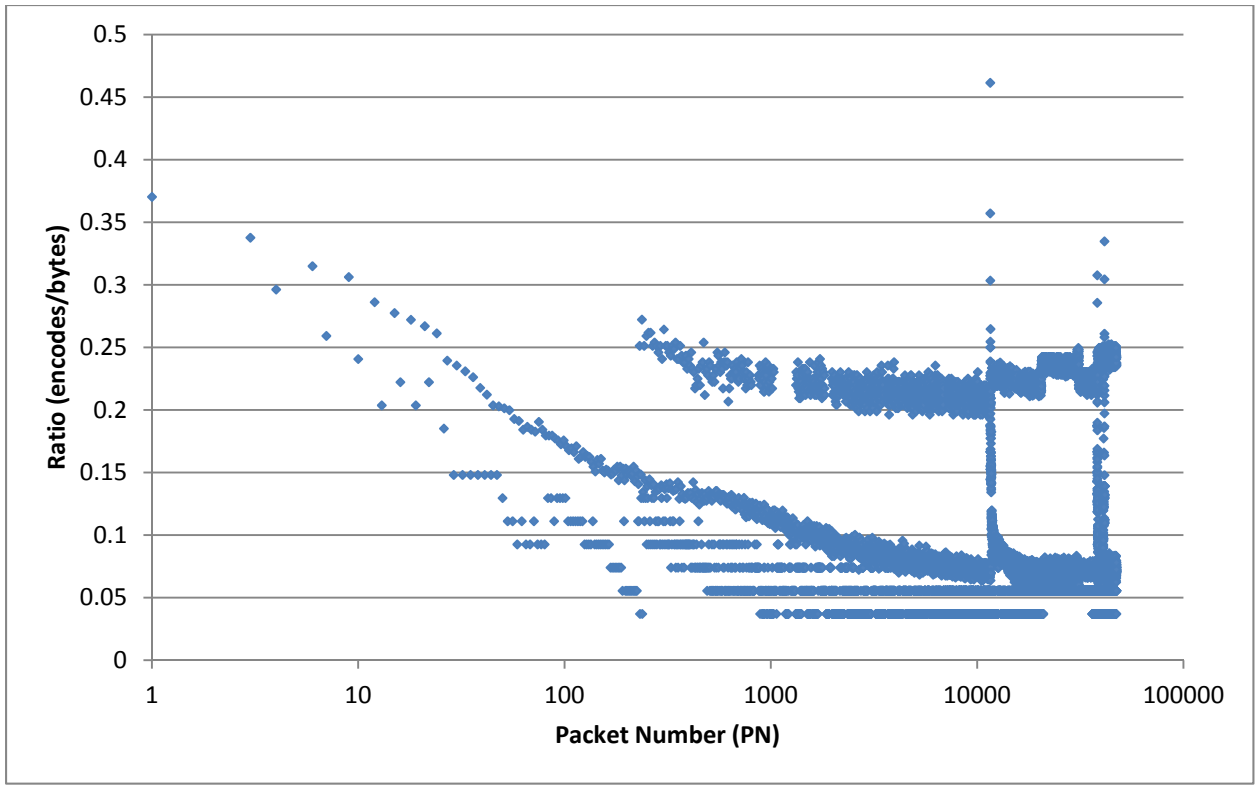

Figure 9. Compression ratio of TCP port 5026 data packets.

3.1.2.3 TCP Port 5413

Data for TCP Port 5413 are shown in Figure 10 and Figure 11. This protocol appears to have two curves in the first 100 data packets, although they are quite close and may not be worth separating. There appears to be one more curve starting at PN 13158.

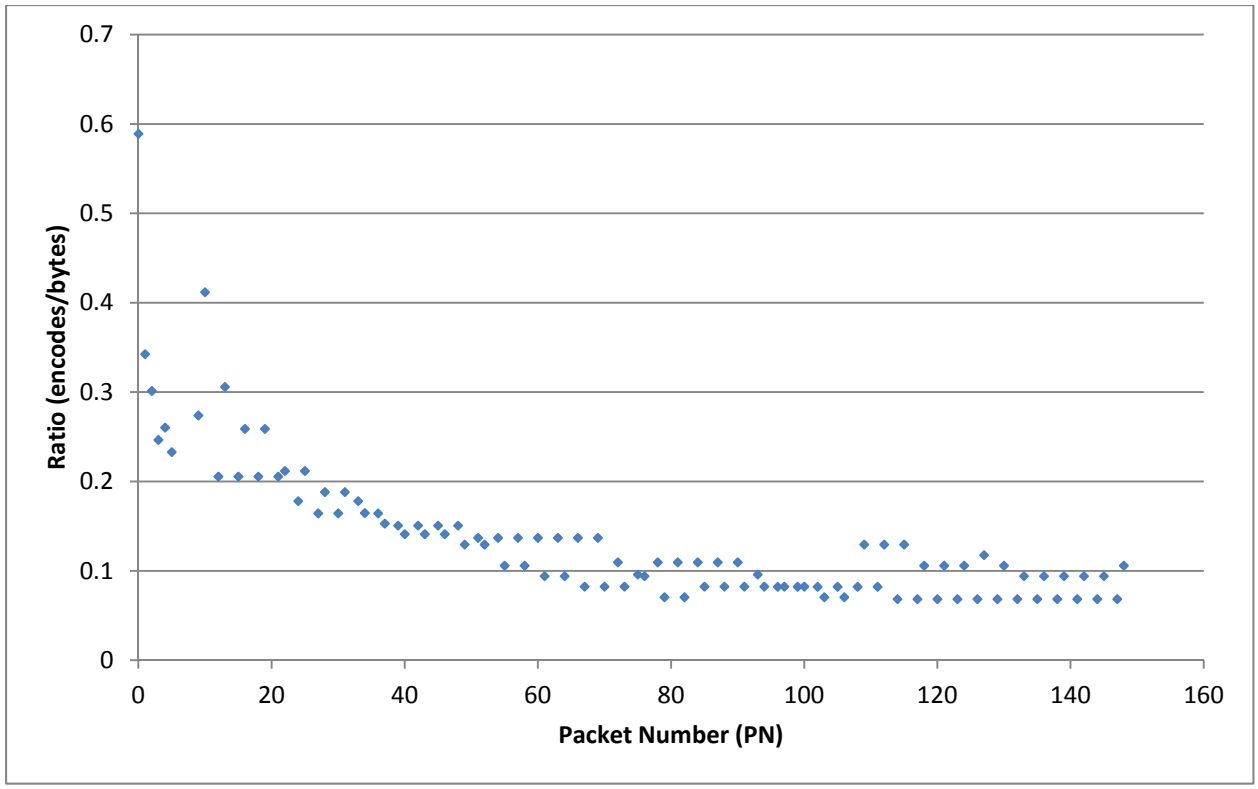

Figure 10. First 100 data packets on TCP port 5413.

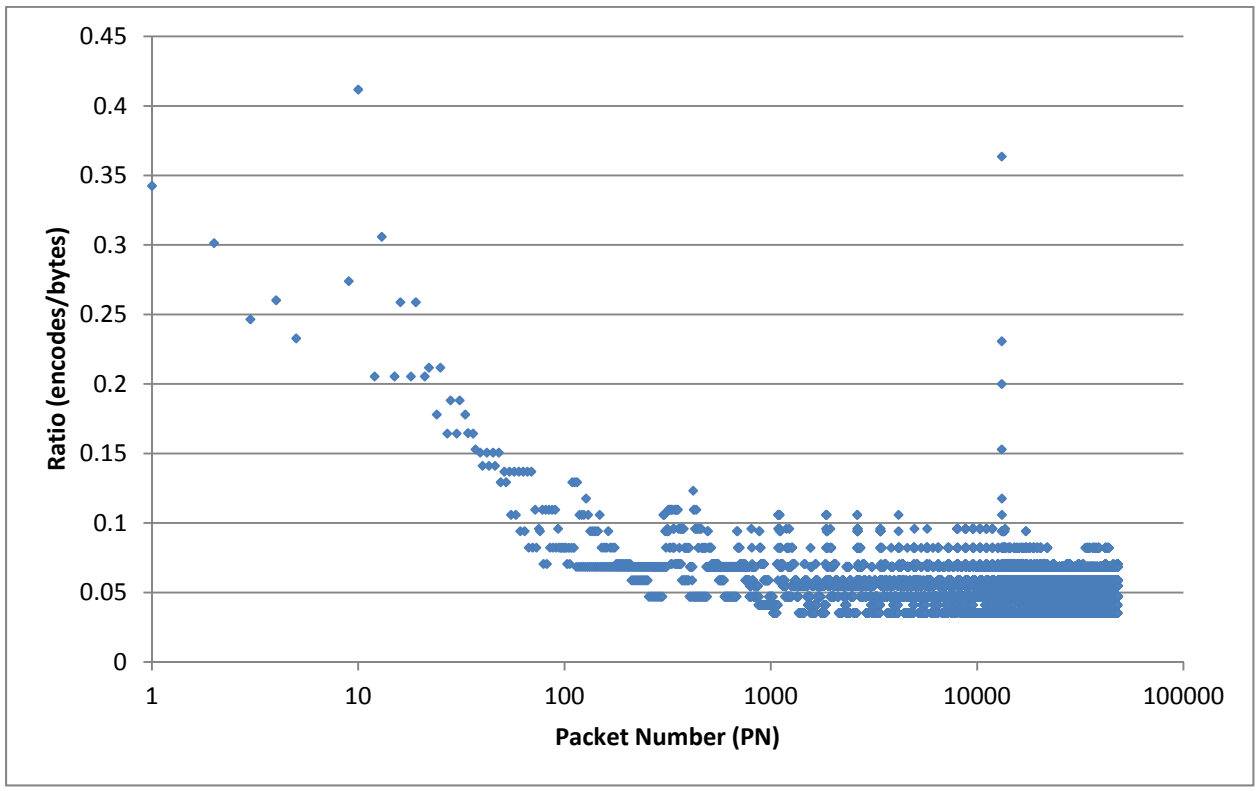

Figure 11. TCP data packets on port 5413.

3.1.2.4 TCP Port 18190

This protocol filled the dictionary at PN 4,316 and turns out to be encrypted. Figure 12 shows the compression as it approaches 0.4. This is actually the zero compression point since there are 8 bits in a byte and 20 bits in an encode.

The odd shape of the curve is attributable to a change in the size of the packets. Between PN 150 and PN 200, the size of the typical packet increases from 45 bytes to 1,460 bytes. This increases the number of nodes added to the dictionary per packet, therefore decreasing the compression ratio faster.

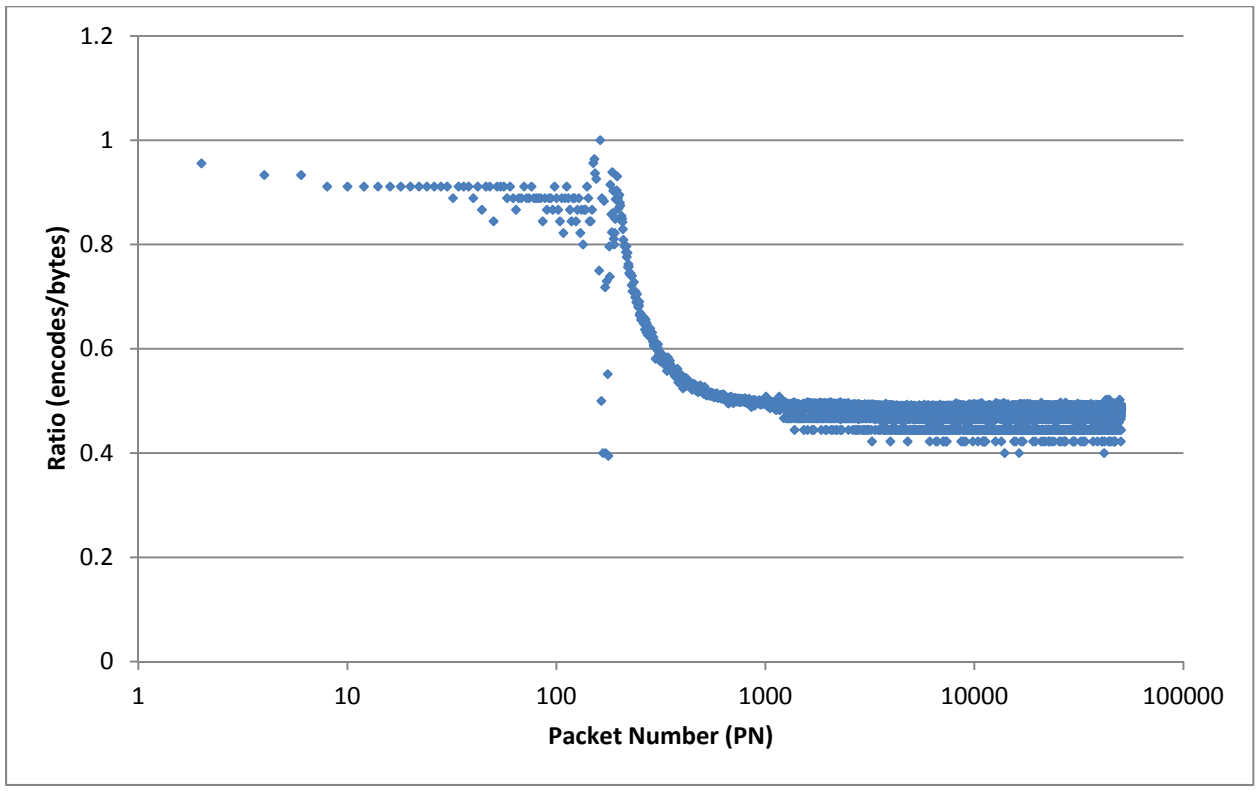

Figure 12. Compression ratio of TCP port 18190 data packets.

3.1.2.5 TCP Port 27000

Data for TCP Port 27000 are shown in Figure 13 and Figure 14. This channel seems to fit into the two curve model, with perhaps one curve for each direction of data. By PN 1000, horizontal lines are obviously forming in the upper curve, indicating specialization of a small input packet. The lower curve also eventually overspecializes and compresses the 147 byte input into a single encode, until something changes and it jumps back up to 7 encodes.

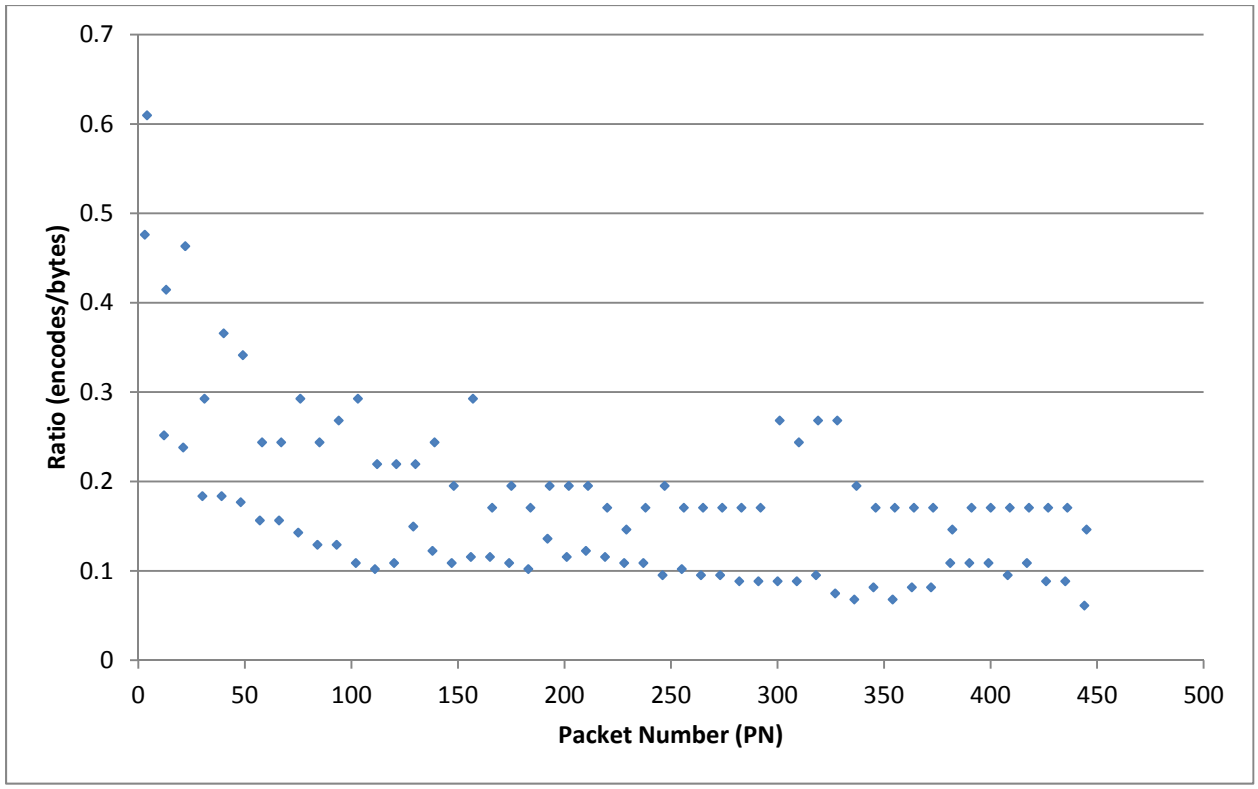

Figure 13. Compression ratio of first 100 data packets on TCP port 27000.

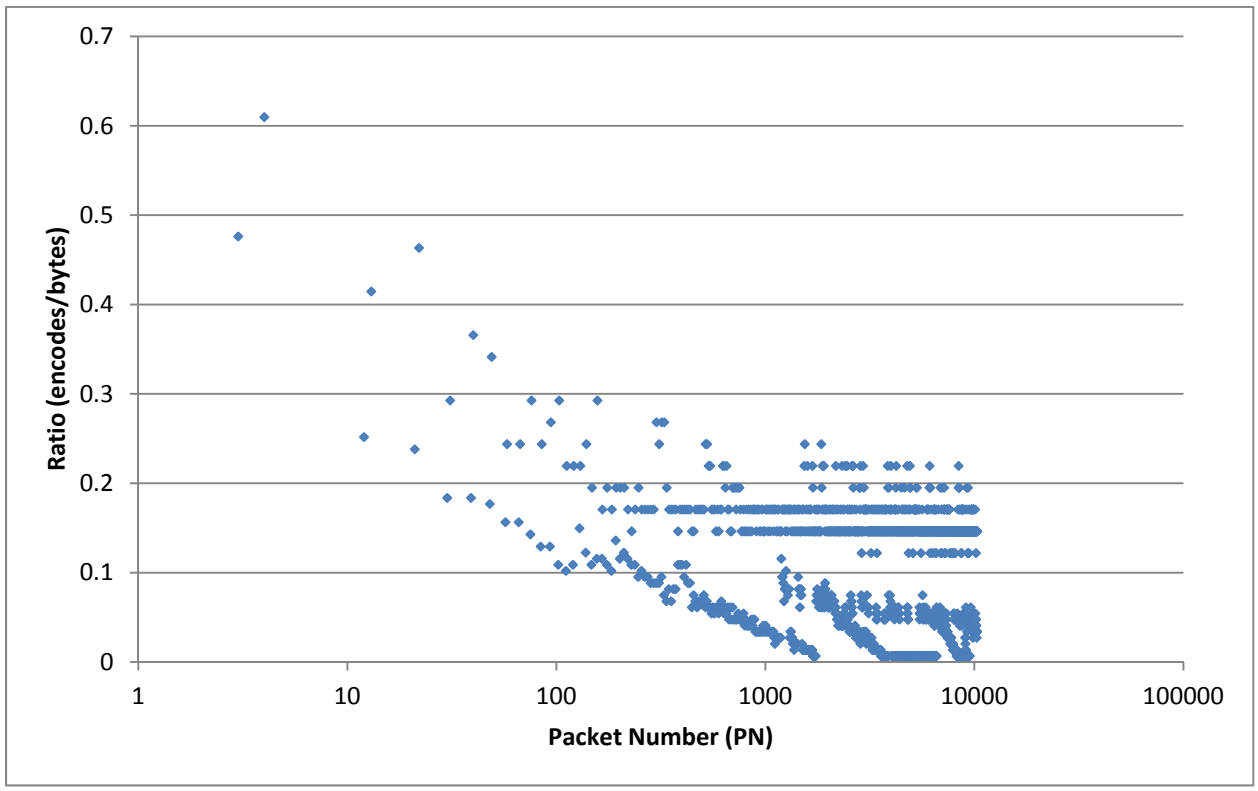

Figure 14. Compression ratio of data packets on TCP port 27000.

3.1.3 TCP Direction Filtering

Data for TCP Direction Filtering are shown in Figure 15 , Figure 16, and Figure 17. In an attempt to filter packets to a more consistent type, the five existing channels were filtered based on TCP direction. Each channel's data set becomes two data sets based on the packet's relation to the service port. For instance, the channel based on TCP port 27000 was divided into packets with a destination of TCP port 27000 and packets with a source of TCP port 27000.

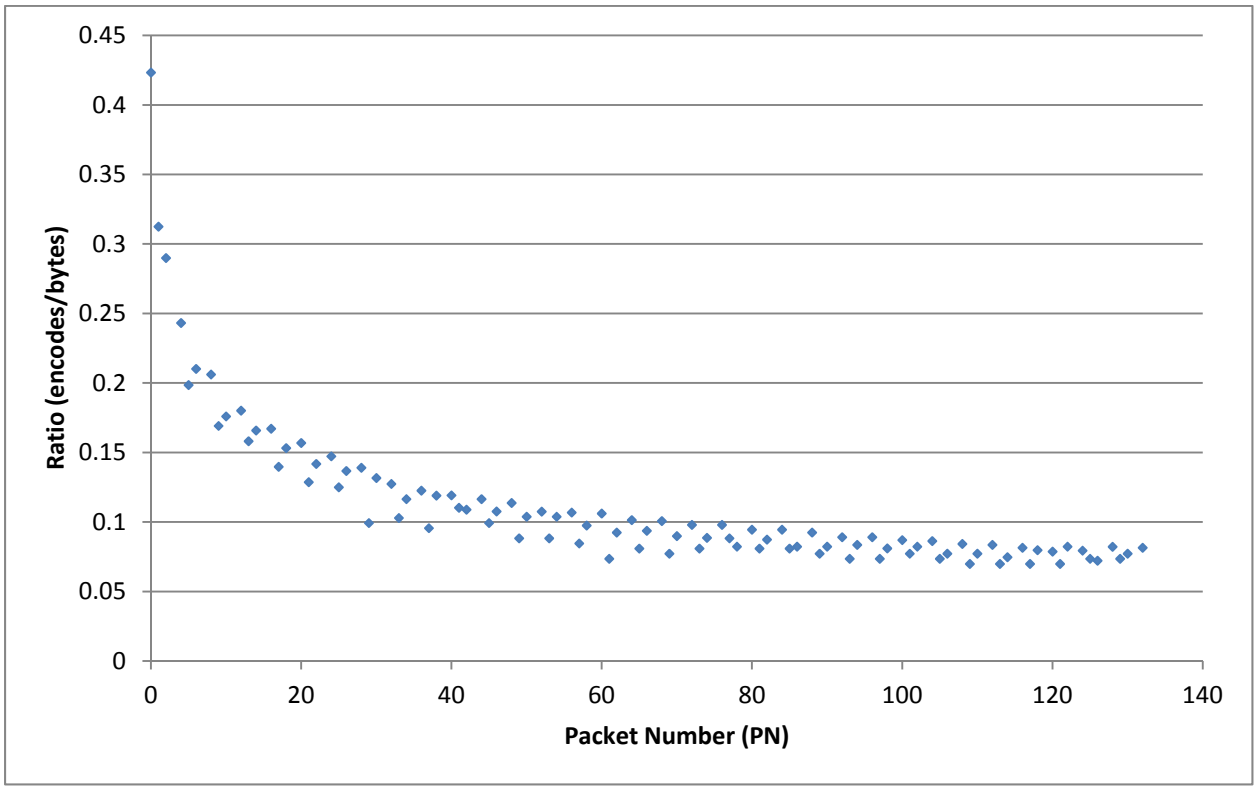

Figure 15 . Compression ratio of first 100 data packets for destination port 1433.

3.1.4 Input Length Adjusted Compression Ratio

One problem with using compression ratios is that a malicious actor could pad packets with highly compressible data to counter the less compressible malicious content. One option for handling this problem was to calculate the square of the encode input lengths. This would produce a value for each packet that is the sum of the squares of the encode input lengths. An encode input length is the number of bytes that that encode represents; thus, if you have encode ABC, representing (aaa)(bb)(cccc), the average bytes per encode is three and the new value is 29. A malicious actor who tries to use encodes DEF (DDDDDDD)(E)(F) stills hits an average of three but the input length adjusted output (ILAO) is now 51.

In order to test the idea of distinguishing packet types based on ILAO instead of compression ratio, the research team modified the code to give the ILAO values. Figure 18 shows the ILAO and average ILAO as a function of packet number. This approach looks promising and may help solve the compression ratio weakness.

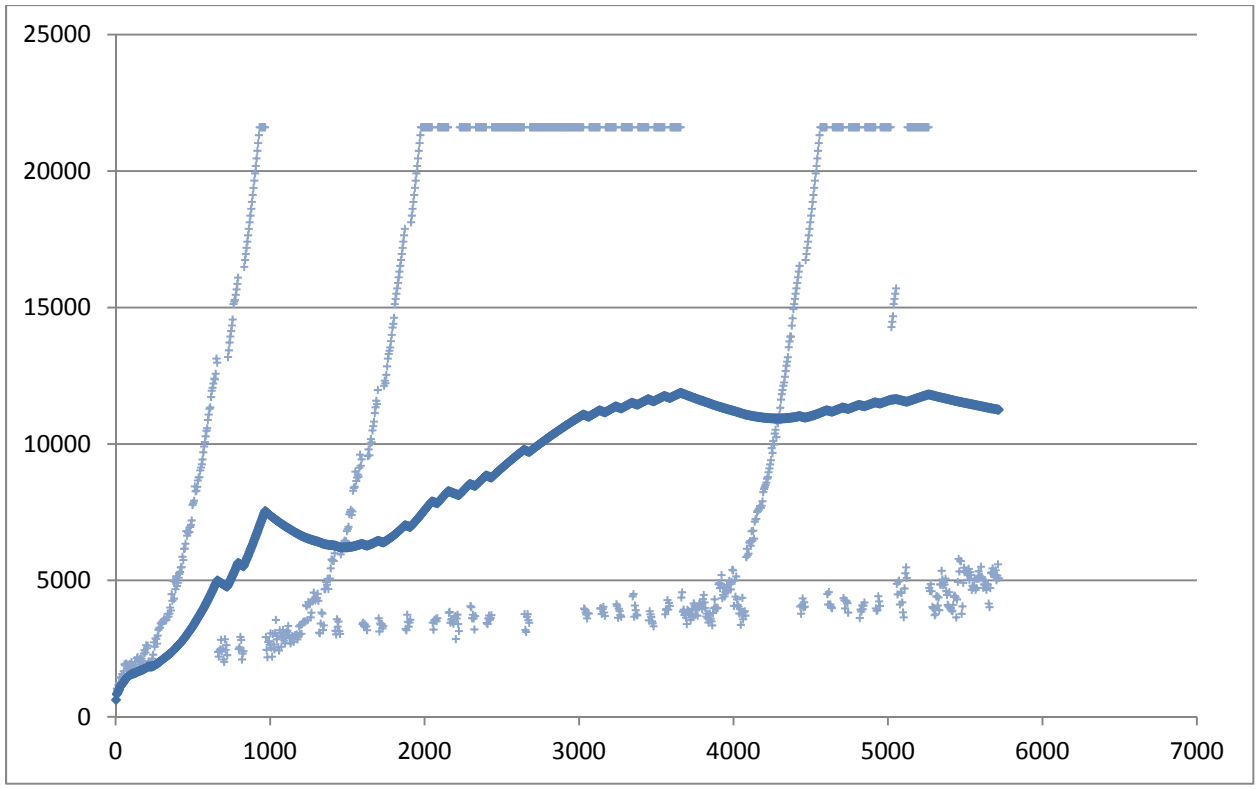

Figure 16. Adjusted compression ratio and average adjusted compression ratio of packets for destination port 27000.

3.1.5 Dictionary Chaining

At this point, the research team has identified the need for a way to classify new packets. Figure 18 does an excellent job of highlighting this need, and it zooms into the problematic area of Figure 17. There are two causes of problems for the current compression technique. There are three near vertical plots of dots in Figure 18. Each of these is the beginning of a new session inside this channel. Each time the MSSQL handshake occurs it compresses better than before but is still significantly less compressed than previous packets in the channel. Since there will any number of these seldom occurring packets, being able to group them together as a single occurrence will be necessary to prevent repeated false positive matches in an IDS system. The other potential problems are the presence of an encrypted packet in the session handshake and some invalid TCP keep-alive messages that contain a single data byte.

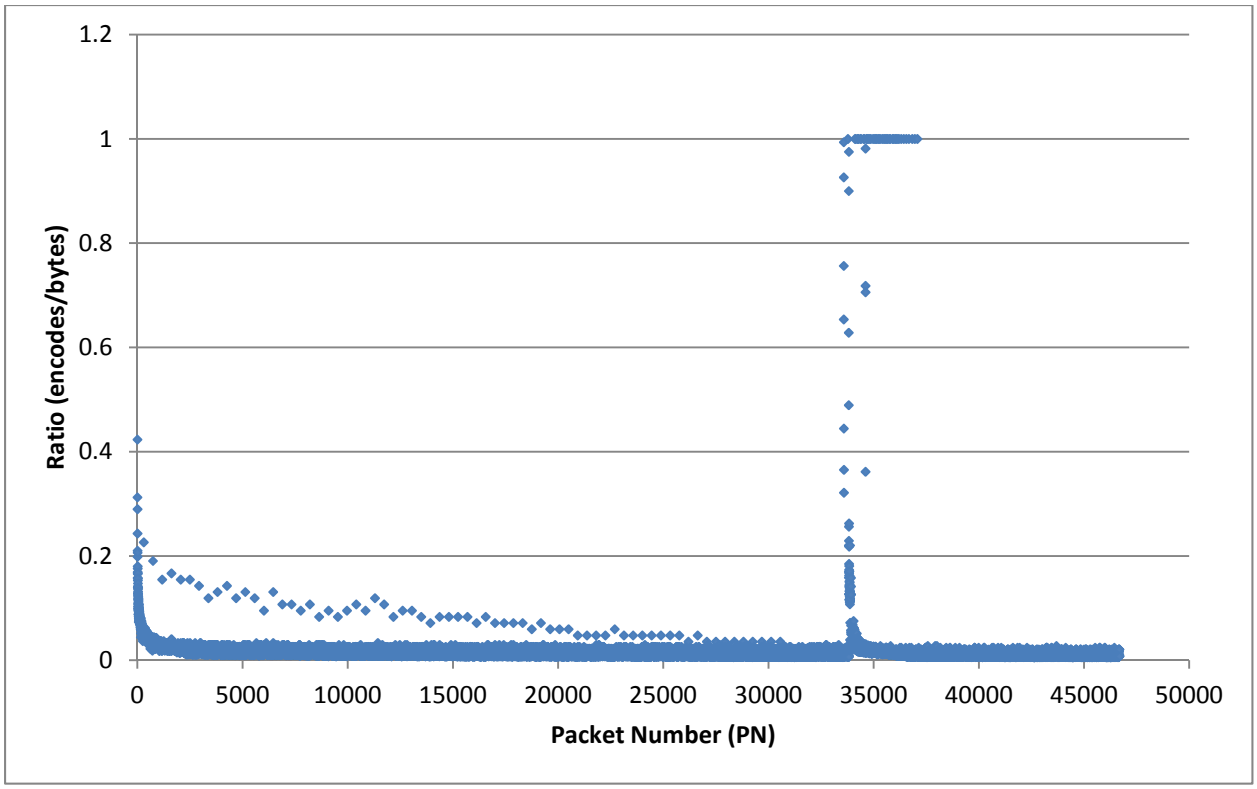

Figure 17. Compression ratio of first 32,000 data packets for destination port 1433.

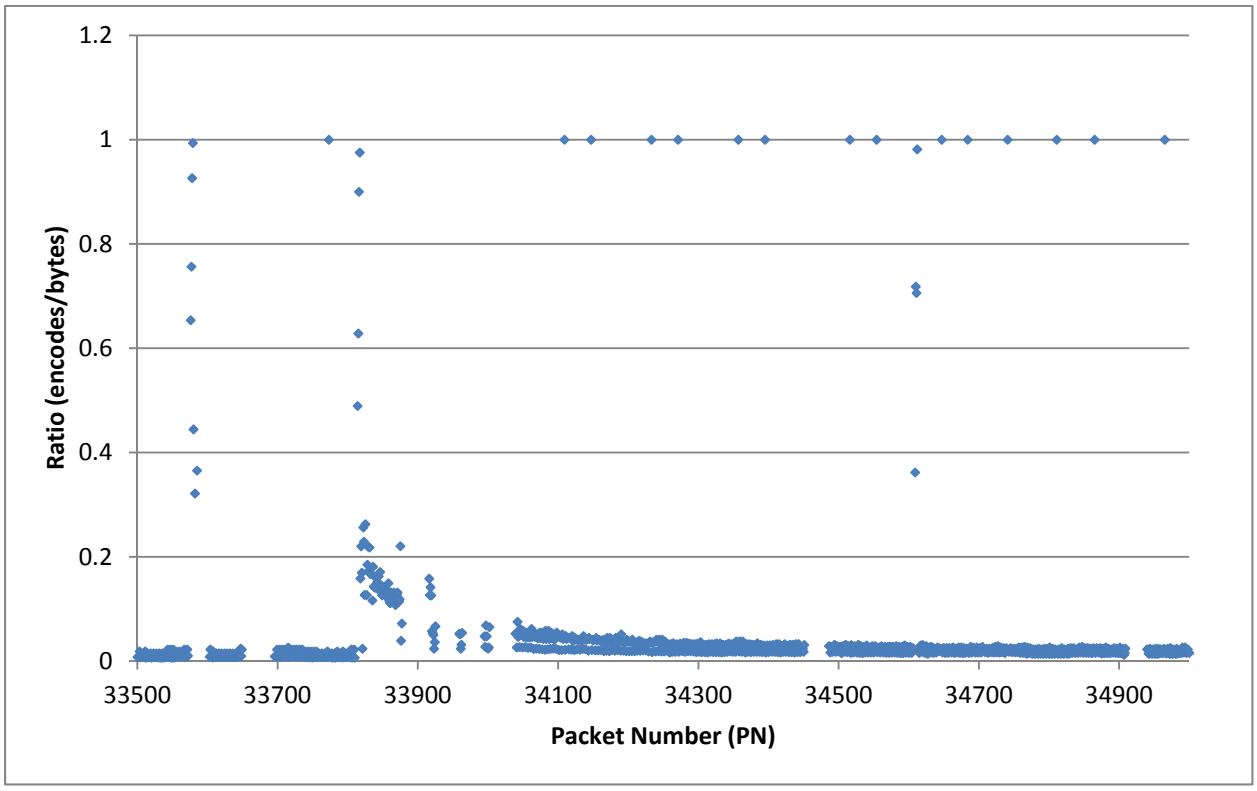

Figure 18. Compression ratio of packets at abnormality for destination port 1433.

One way of grouping packets is to identify them based on similar sequences of encodes. A better approach might be to use a series of dictionaries and try to get each one to specialize on a single packet type while rejecting others. To start, it could reject packets when the packet ratio is higher than the mean ratio for that dictionary.

Figure 19 is a protocol with identical 147 byte packets with the exception of a 4 byte string in the middle of the packet. The red points are the mean compression ratio and the blue dots are the per packet compression ratios. By using the mean compression ratio as a dividing point we can distinguish between well known packets and packet types with less known but related packets.

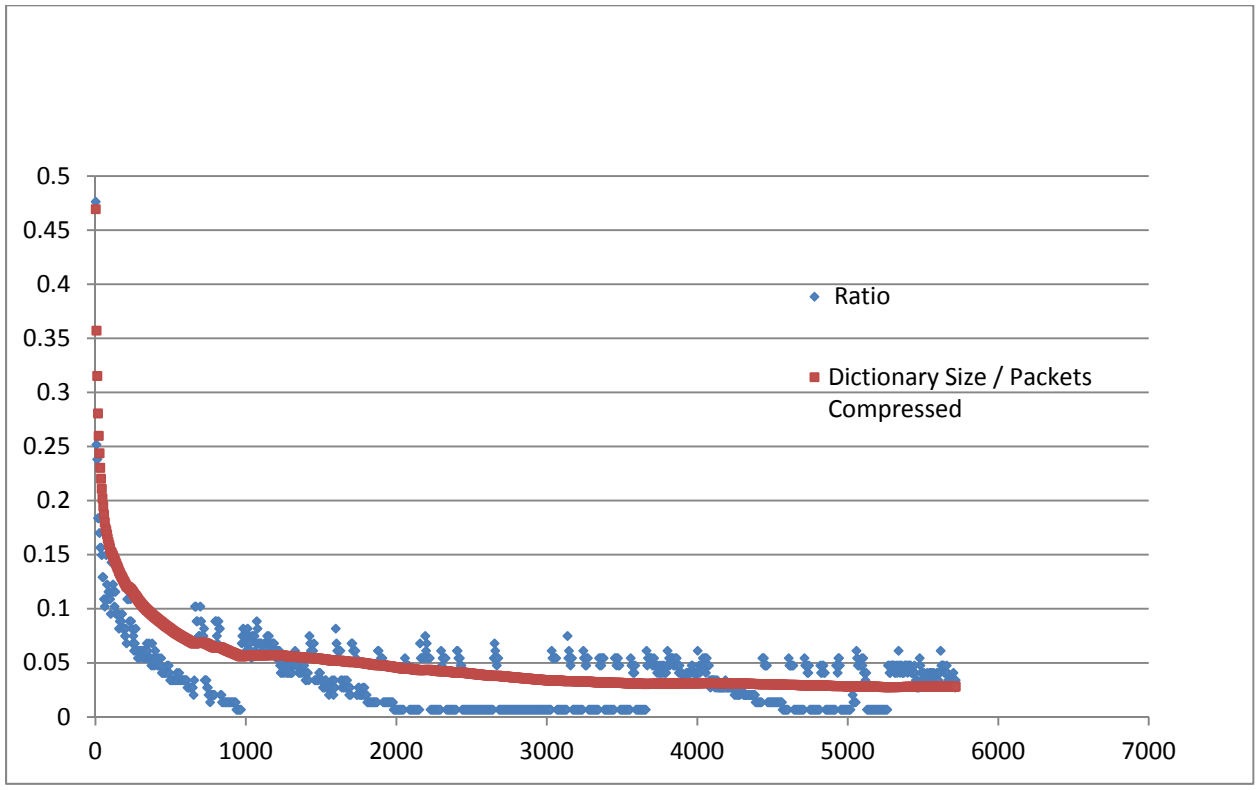

Figure 19: Compression ratio of packets for destination port 27000.

The code was modified to allow multiple dictionaries. When a packet failed to have a variation greater than the dictionary average, the research team created a new dictionary and populated it with the failed packet. Each packet was then tested against each dictionary until it found one it fits into. This technique worked very well for TCP destination 27000. It created 5 dictionaries, one for each of the packets that went above the average variation and one for the generic non-specialized packets. Unfortunately it failed on the other protocols tested for 2 reasons:

- It created hundreds of dictionaries within minutes.
- It was too slow to be practical

The multiple dictionaries are caused by a number of issues:

- TCP DST 5026 with tcp.len == 54 contains a little endian number that is increasing. The second byte changes less frequently than it takes the dictionary to specialize on it. The result is that every time the number increases by 256 a new dictionary is formed.
- TCP DST 5026 with tcp.len == 1284 contains two seemingly alternating messages. It takes a long time for the dictionary to specialize between these messages and when it does it still makes mistakes and includes the specialized packets in the generic dictionary.

Using a single dictionary results in a packet compression time of O(lg(m)*n) where *m* is the size of the dictionary and *n* is the number of bytes in the packets. Using a series of dictionaries, this time increases to O(lg(m)*n*p) where *p* is the number of dictionaries. With the dictionary count increasing rapidly, the computation time becomes excessive.

There is still the problem of not being able to group similar packets to reduce false positives in a timely manner. Overspecializing must be avoided where non-desirable, such as in a sequence number field. Speed could be improved by finding a way to use a single dictionary that incorporates attribution, which would allow association of different packet types based on their content. Using a look ahead should also be helpful because it would allow the sequencing of prefixes in addition to appendages and would thus reduce the impact of a single byte change corrupting the compression algorithm. The path to preventing overspecializing involves limiting dictionary size by applying some pruning, but the technique for doing so is uncertain. Pruning the dictionary is difficult because we need to account for 1 in 10,000 cases, but pruning is usually used to discard the 1 case in 10,000.

3.2 Preliminary Research Conclusion

Preliminary research shows that the compression ratio of normal SCADA network traffic has traits desirable for compression analysis. Primarily the network traffic is often similar between packets, as shown by improved compression ratios in subsequent packets. Two difficulties in using compression have been identified. The first is that inter-dependence in the ratio metric between packets violates principles of statistical analysis. The second is that not all of the packets in a communication channel compress well due to different packet contents. Further research in this area will focus on solving these problems.

4. INDEPENDENT RESEARCH

The original premise behind using compression techniques to identify malware in SCADA streams was to "train" a dictionary on a normal set of SCADA data such that all "normal" encodes will be embedded in the trained dictionary and that packets in the data stream should compress nearly equally. As it turns out, there were two problems with this proposed methodology. The first was a temporal issue with the data. It appears that TCP/IP sessions are re-initialized periodically either because network connections are lost or timed out and later re-established. When this occurs, the nature of the communications headers appears to change, altering the characteristics of the SCADA network stream significantly. This is illustrated in Figure 19, which shows "compression ratio" for each packet as it arrives in time (sequence) for all the normal SCADA traffic on destination port 5026. Compression ratio is defined as the number of encodes generated during packet compression divided by the packet size. Small compression ratios indicate a packet that was highly compressed by the LZW algorithm. These data were generated by allowing the dictionary used by the LZW compression algorithm to grow during this process to maximum value of 1,048,576 entries.

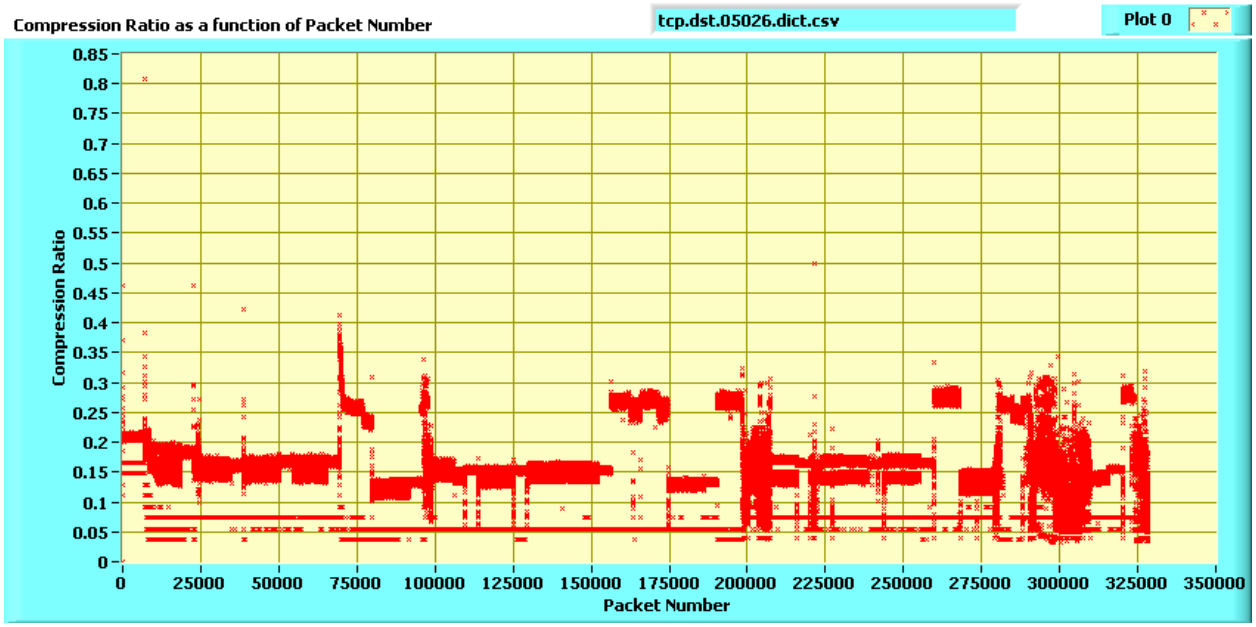

Figure 20. Time sequence plot of compression ratio as a function of packet number for port 5026.

There are several features that can be extracted from the data presentation in Figure 20. First, the amount of data analyzed is large at over 325,000 packets. Second, although difficult to see in this plot, there is a startup trend in compression ratio where the dictionary is growing with each packet analyzed until the system settles into a compression ratio of around 0.15. Third, the settling in process receives a shock periodically where the compression ratios increase dramatically and later settle in again. These shocks are caused by two factors we believe. The first factor is the arrival of a batch of packets that are of a new size and message content. The second factor appears to be associated with the establishment of a new session in the TCP client/server relationship. Fourth, it is apparent that the distribution of compression ratios is of a

binary nature and not continuous. This is not surprising because packet sizes are an integer value and the number of encodes generated is also an integer value. The binary nature of these distributions becomes more apparent when compressing smaller packets.

After analyzing the data from port 5026, it became apparent that a better way to look at the data was to plot the data as a function of packet size rather than by the sequence in which it arrived. Figure 20 shows the same data as Figure 21, but plotted as a function of packet byte size. The data group much more naturally when presented this way; however, the session start-up issues identified previously are apparent here by the straight line grouping along many of the packet sizes.

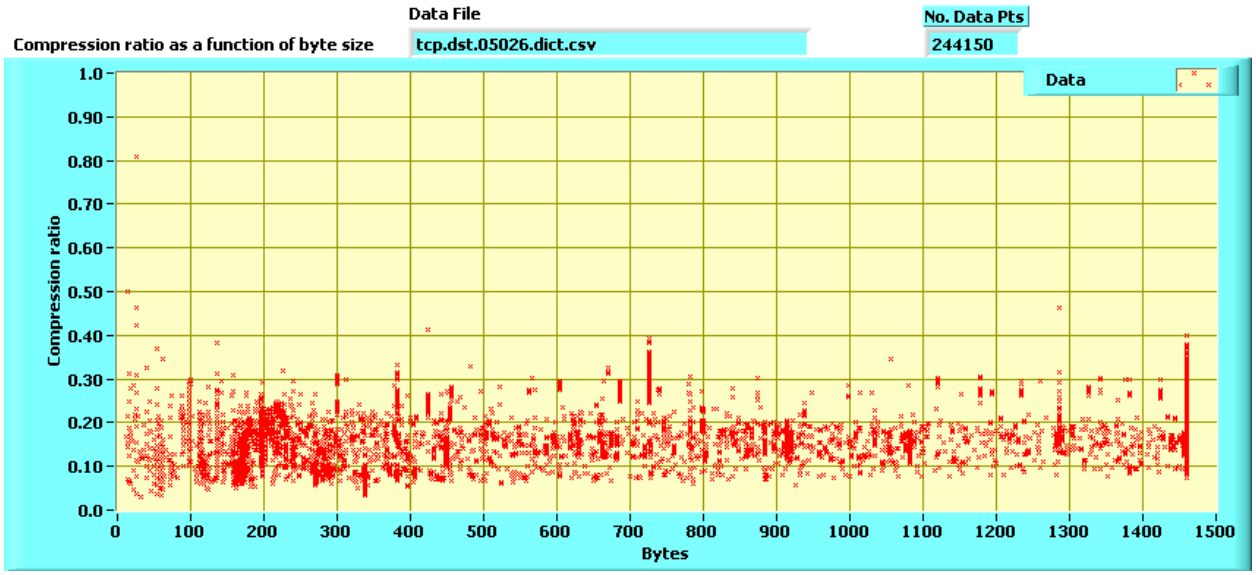

Figure 21. Port 05026 data plotted as a function of packet size.

One of the desires of this project was to investigate statistical means to classify normal data traffic against malicious traffic. However, training an LZW dictionary using either good or malicious data violates a fundamental assumption of statistical analysis when attempting to classify data at the packet level using compression ratio as a metric. Statistical analysis requires that each observation is independent unless there is a method to calculate the covariance between observations. In this case, the observation is the compression ratio (defined as the number of encodes divided by the packet size in bytes) and the number of encodes generated for a given packet is dependent on all previous packets as they have been used to create the dictionary used in calculating encodes.

4.1.1 Static Dictionary Analysis Method

This analysis technique examined the use of a fresh or static dictionary for each packet, and examined compression ratios obtained when starting packet compression from a default dictionary. This technique provides a basis for comparing and grouping compression ratios using statistical means because now each compression ratio calculation can be considered an independent calculation. The down side to this approach is that the concept of "training" the compression algorithm is abandoned. However, the hope was that there would still be enough

distinction between normal traffic and infected traffic to differentiate them using compression ratio of the packet as a metric.

The data plotted in Figure 22 were generated with each packet using the default start-up dictionary of the LZW compression algorithm. Additionally, since this provided a fairly tight grouping of data, a logarithmic model was fit to the data using a least squares algorithm to model the data. Compression ratio as a function of packet byte size is of the following form:

$$CR = A + B * ln(Bytes)$$

Where:

CR = compression ratio

A, B = Coefficients from the least squares fitting algorithm

Bytes = original packet size in bytes.

Figure 22 shows the data, fit, fit coefficients, and standard deviation of the fit with respect to the data for the destination TCP port 5026 traffic. It should be noted that data below 75 bytes in size are not shown, nor are they part of the model statistics because the compression ratios are very unpredictable at very small packet sizes. We believe this is because there are very few of the smaller packet sizes and these packets may have been incomplete or retransmitted for some reason. This is an issue that will have to be dealt with at some point in the future. Also shown in Figure 22 are the 95 and 99 percent confidence limits about the mean of the model. Confidence limits were calculated using standard statistical methods for determining correlation of model and data.[c]

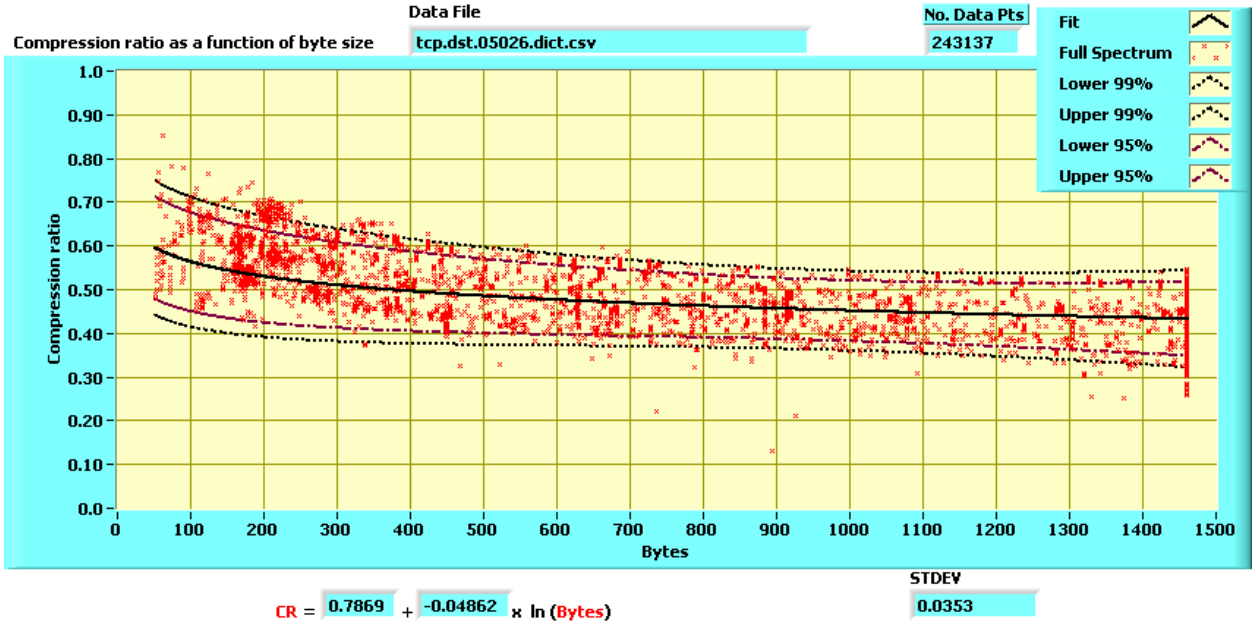

Figure 22 . Normal traffic data plotted as a function of byte size for TCP port 05026 destination data (default dictionary for each packet).

[c] Walpole, R. and Myers, R., Probability and Statistics for Engineers and Scientists, 2 Ed., pp. 292-293, Macmillan Publishing Co., Inc., 1978. ISBN 0-02-424110-5.

Figure 23 shows a plot of the source TCP port 5026 data. The characteristics of the source and destination data are similar to those shown in Figure 22 , as indicated by the fit coefficients and standard deviation calculated for each data set. The fit coefficients and data for this port are close enough statistically that they can be considered one data set/model.

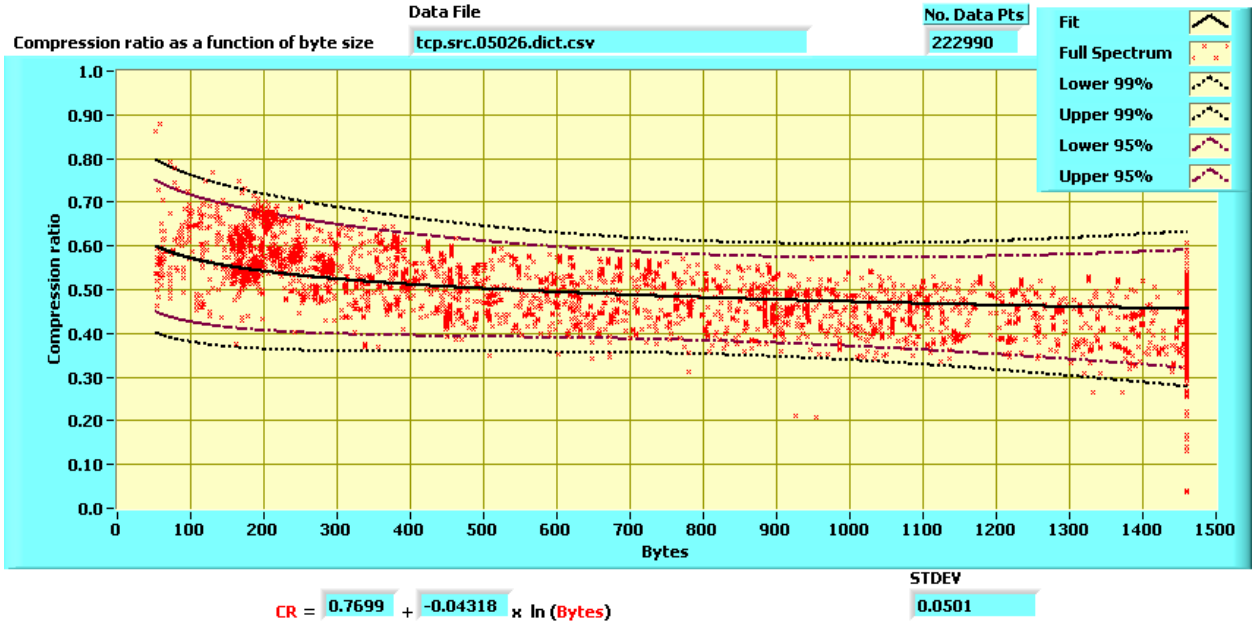

Figure 23. Normal traffic data plotted as a function of byte size for TCP port 5026 source data (default dictionary for each packet).

4.1.1.1 Port 1433 Data (Normal Traffic)

Network traffic going both directions over port 1433 were examined in the same manner done for port 5026 and are shown in Figure 24 and Figure 25 below. These two data sets can also be considered as one statistically, although it appears that they are different from the port 5026 data.

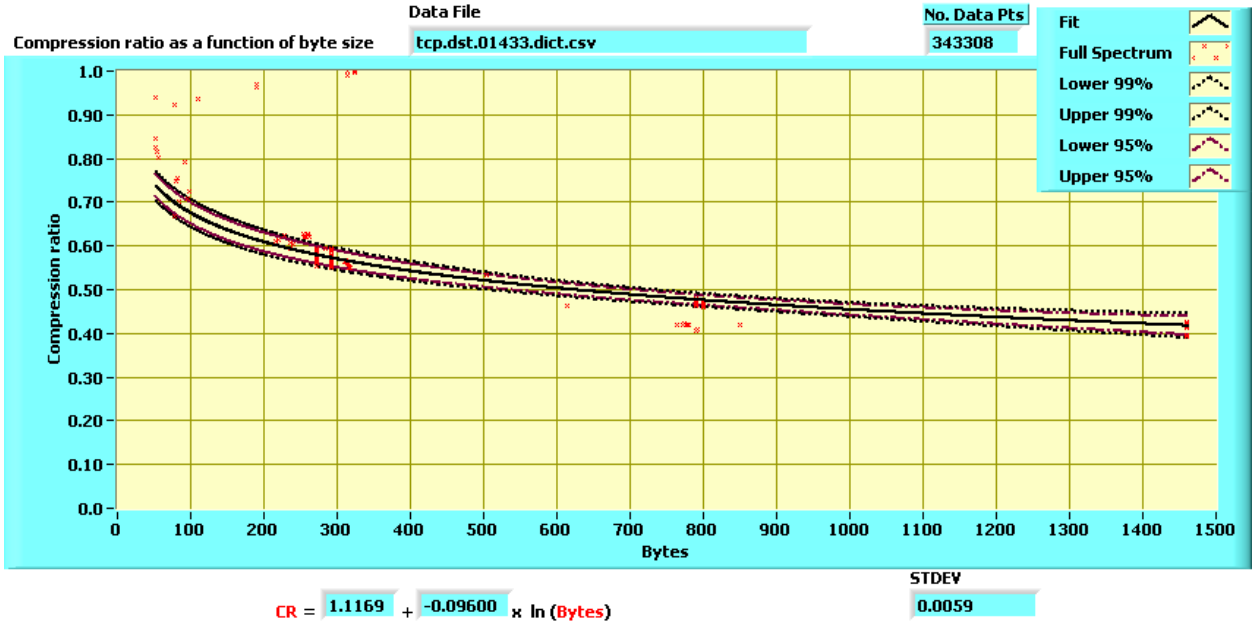

Figure 24. Normal traffic data plotted as a function of byte size for TCP port 1433 destination data (default dictionary for each packet).

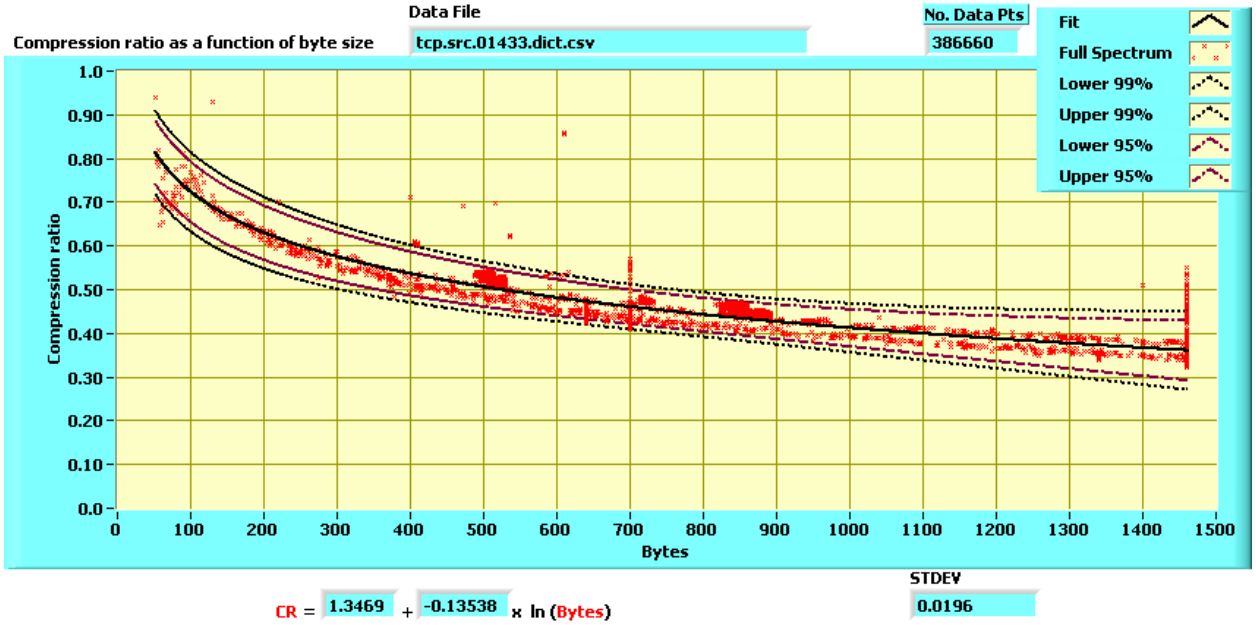

Figure 25. Normal traffic data plotted as a function of byte size for TCP port 1433 source data (default dictionary for each packet).

4.1.2 Static Dictionary Infected Data Analysis

Analysis of normal SCADA network traffic appears to group well enough that it will generate a statistical model that has reasonably small error bounds. The next step in the process of defining a way to discriminate malicious traffic from normal network traffic was to look at how infected data compresses and compare it to normal SCADA traffic. Figure 26 through Figure 35 shows compromised data sets for the destination and source TCP ports 5026 and 1433,

respectively, using a fresh dictionary for the compression of each packet in the network traffic stream. Observations and comparisons to the non-infected data streams are as follows:

1. Infected data sets are close enough statistically to be considered one model as evidenced by the fit coefficients calculated for each of the four data sets shown. This is consistent with the normal traffic data sets analyzed above.
2. The compression ratios for infected data sets compared to normal traffic data sets are significantly different for packet sizes less than about 1000 bytes. For larger packets sizes, the difference between the two data sets starts to overlap statistically. This will tend to create large numbers of false negatives and false positive when trying to distinguish malicious traffic from normal traffic.
3. Start-up or new sessions start issues that are more prevalent at larger packet sizes as seen by the straight lines in the plots of both the normal and compromised data sets.
4. Although this is not shown in the data plots presented here, the variability in compression ratio is much larger for small packet sizes of less than about 70 bytes.
5. The infected data is not statistically independent from the normal network traffic data in this study because of the way the compromised data was generated (i.e., by inserting malicious information into existing packets). This should be considered a preliminary analysis and drawing absolute conclusions based on these data sets needs to be investigated more thoroughly and compared with independent data sets.
6. Confidence levels grow more uncertain when the prediction moves away from the centroid of the data set as seen in all the figures below. This may cause prediction problems when trying to distinguish between normal and infected data.

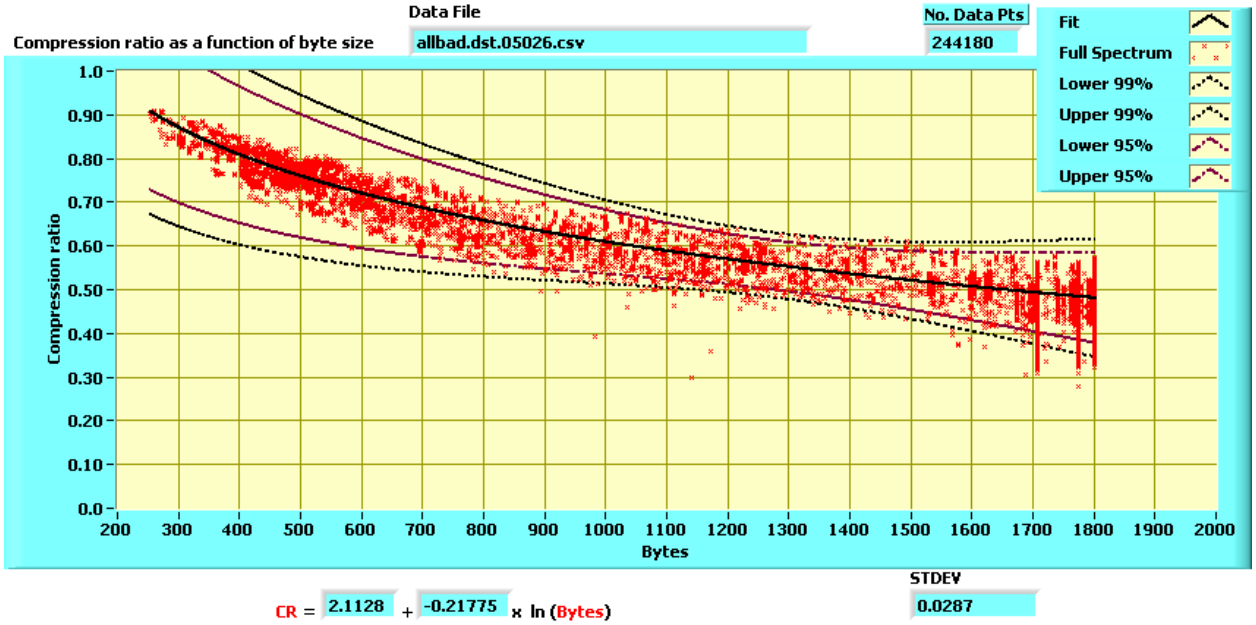

Figure 26. Data plotted as a function of Byte size for metasploit infected TCP port 5026 destination data (default dictionary for each packet).

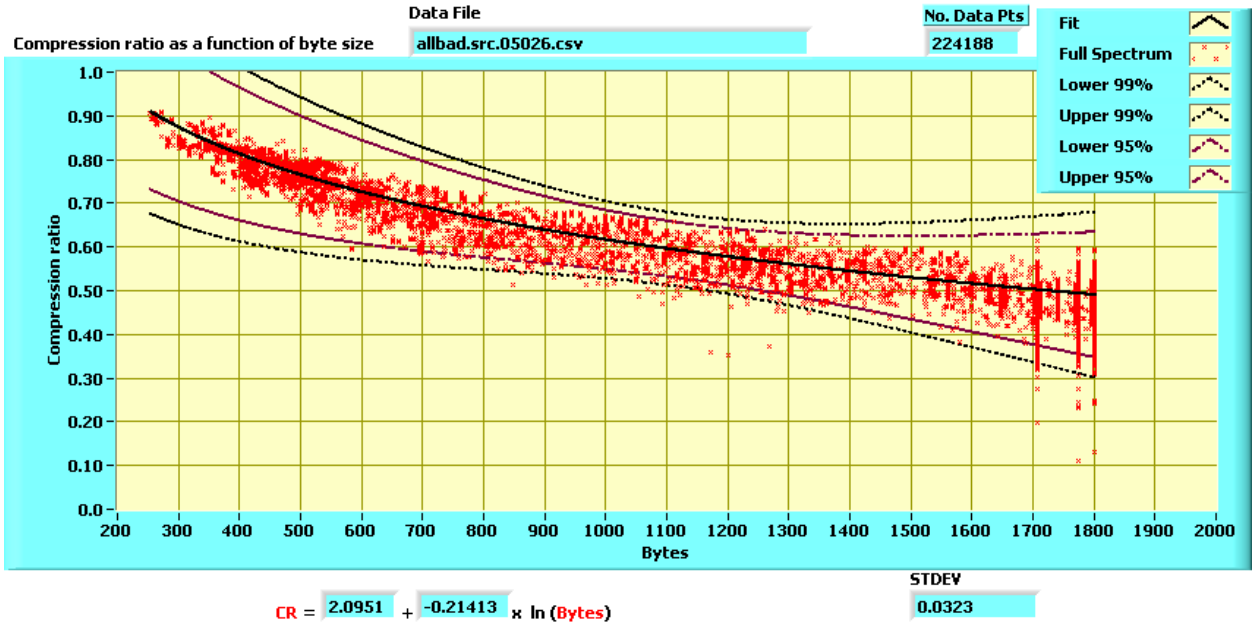

Figure 27. Data plotted as a function of Byte size for metasploit infected TCP port 5026 source data (default dictionary for each packet).

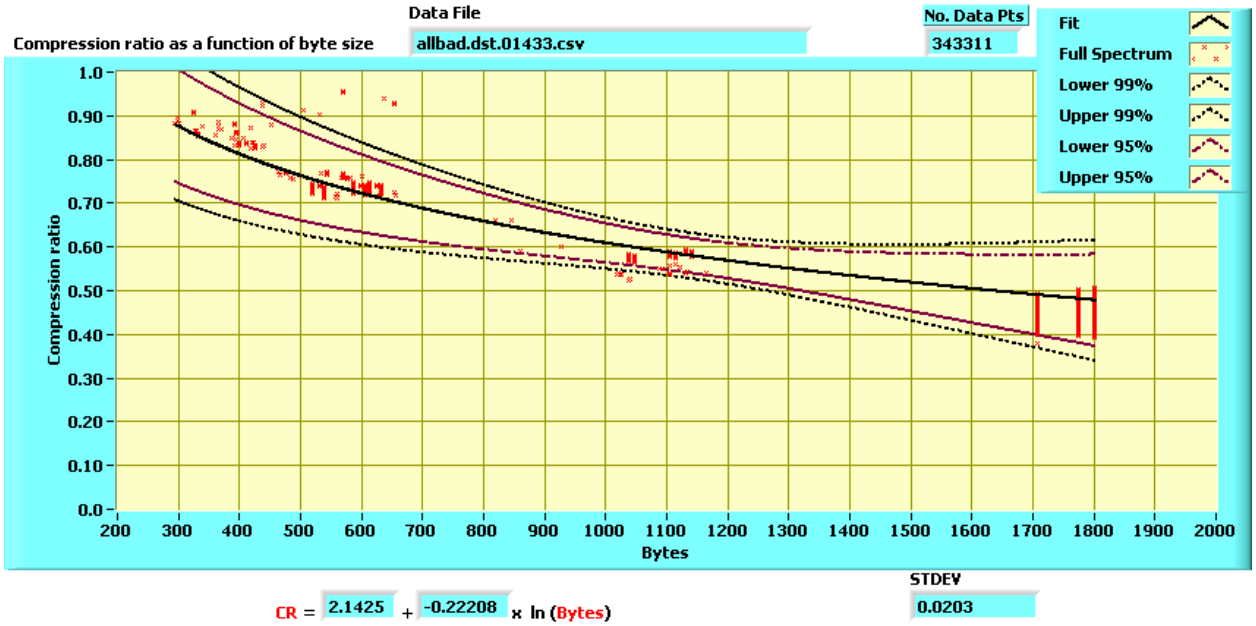

Figure 28. Data plotted as a function of Byte size for metasploit infected TCP port 1433 destination data (default dictionary for each packet).

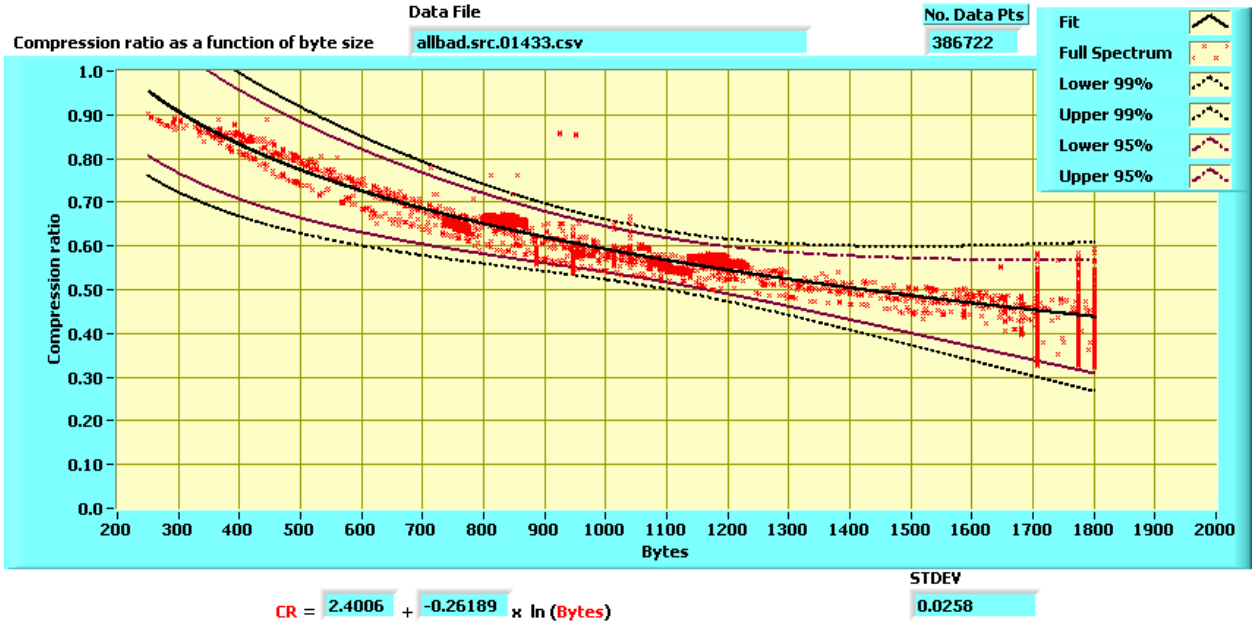

Figure 29. Data plotted as a function of Byte size for metasploit infected TCP port 1433 destination data (default dictionary for each packet).

4.1.3 Split Packet Analysis

Compression ratios for packet sizes of less than 300 or 400 bytes appear to be sufficiently different between normal network traffic and infected traffic to categorize data as good or likely suspect. However, the difference between the normal network traffic data and infected network data at larger packet sizes are not sufficient to statistically classify the packet as belonging to either class with the high confidence levels needed when dealing with these extremely large data sets. The primary reason for this is that the size of the injected simulated malicious software used for this study was 314, 247, and 341 bytes for the exploits used. So, for initially small packets of network traffic, the infected portion of the packet is large. While for large network packets (e.g., greater than 1000 bytes) the infected portion of it is less than 30 percent of the total packet size. Therefore, it becomes harder to distinguish malicious network traffic imbedded in larger network packets based on compression techniques. This is simply because they are a smaller portion of the total packet size and do not affect overall compression of the packet significantly enough to distinguish from normal traffic.

A scheme was devised to subdivide all network packets that were larger than 450 bytes and analyzed these sub-packets individually while attributing the analysis results to the entire packet. The algorithm used to break apart larger network data packets was to divide the total packet size by 300 and round the results to the nearest integer. The result was used to break the packet in the "sub-packets." Network packets that were smaller than 450 bytes were left intact by using this method. At 450 bytes, and larger multiples of 300 bytes, the network packet was subdivided into multiple sub-packets. Each sub-packet was compressed separately and the original packet was assigned a compression ratio from the sub-packet that provided the largest compression ratio.

This greatly improved the ability to distinguish malicious data embedded in the larger network data packets while minimally affecting the computed compression ratios for normal or

non-infected network traffic as shown in Figure 30 and Figure 31. Figure 30 shows a plot of compression ratio data for normal network traffic on the destination side of port 5026 using the split-packet analysis method described above. Compare this plot to Figure 22 and it is evident that the compression ratio stays relatively constant after 450 bytes which is where the split packet algorithm starts to modify the packet analysis. Figure 31 shows a plot of the compression ratio data for infected network data on the destination side of port 5026 and should be compared to Figure 26. These data clearly indicate the advantage of using the split packet analysis for larger network packets containing infected data. The compression ratio is nearly constant with a mean of about 0.82. This can be compared to Figure 26 where the mean of the compression ratio drops significantly as packet size increases. This allows a much easier distinction between normal and infected traffic at the larger packet sizes.

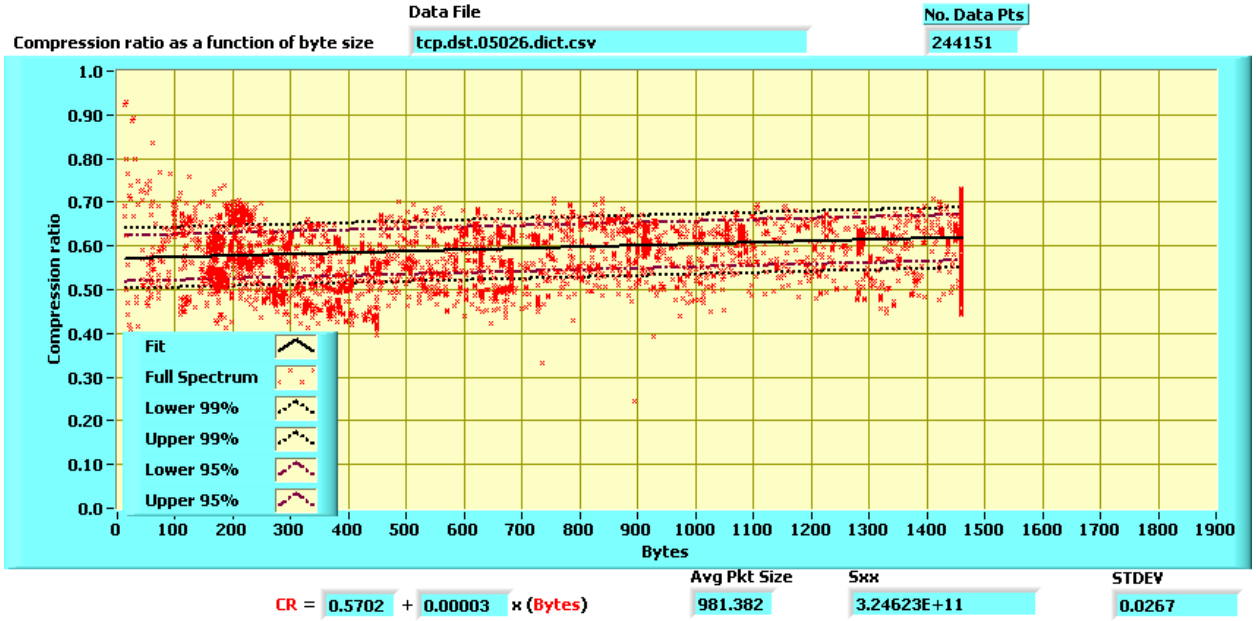

Figure 30. Normal traffic data plotted as a function of byte size using the split packet analysis for TCP port 5026 destination data (default dictionary for each packet).

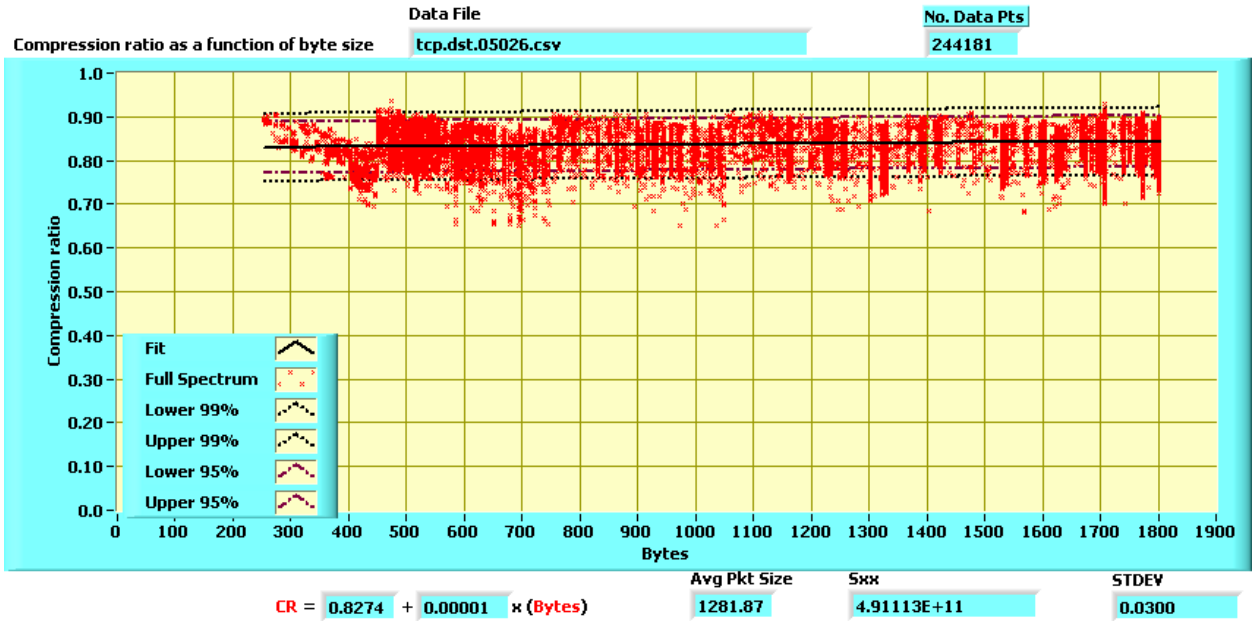

Figure 31. Data plotted as a function of Byte size for metasploit infected network traffic using the split packet analysis on TCP port 5026 destination data (default dictionary for each packet).

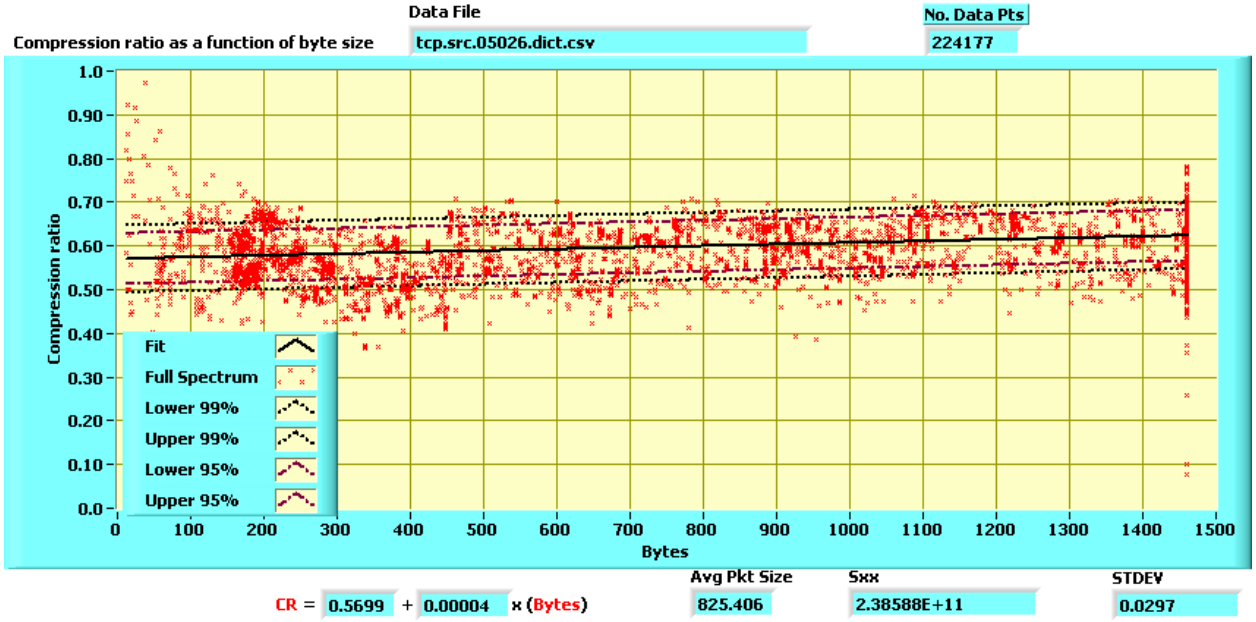

Figure 32. Normal traffic data plotted as a function of byte size using the split packet analysis for TCP port 5026 source data (default dictionary for each packet).

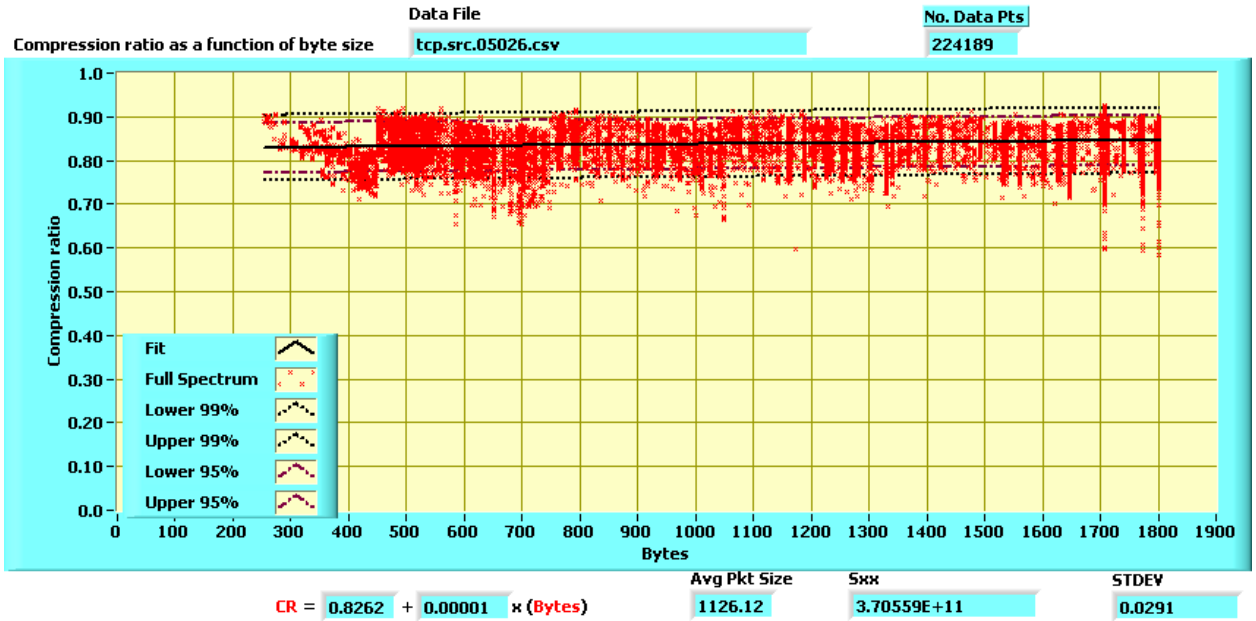

Figure 33. Data plotted as a function of Byte size for metasploit infected network traffic using the split packet analysis on TCP port 5026 source data (default dictionary for each packet).

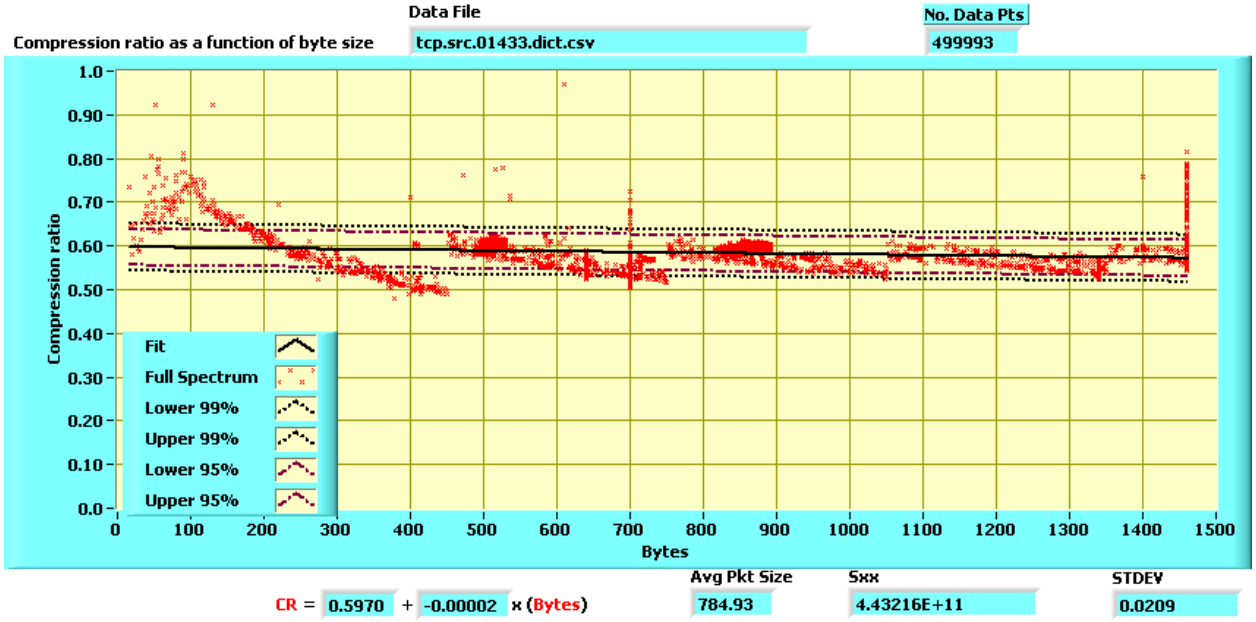

Figure 34. Normal traffic data plotted as a function of byte size using the split packet analysis for TCP port 1433 source data (default dictionary for each packet).

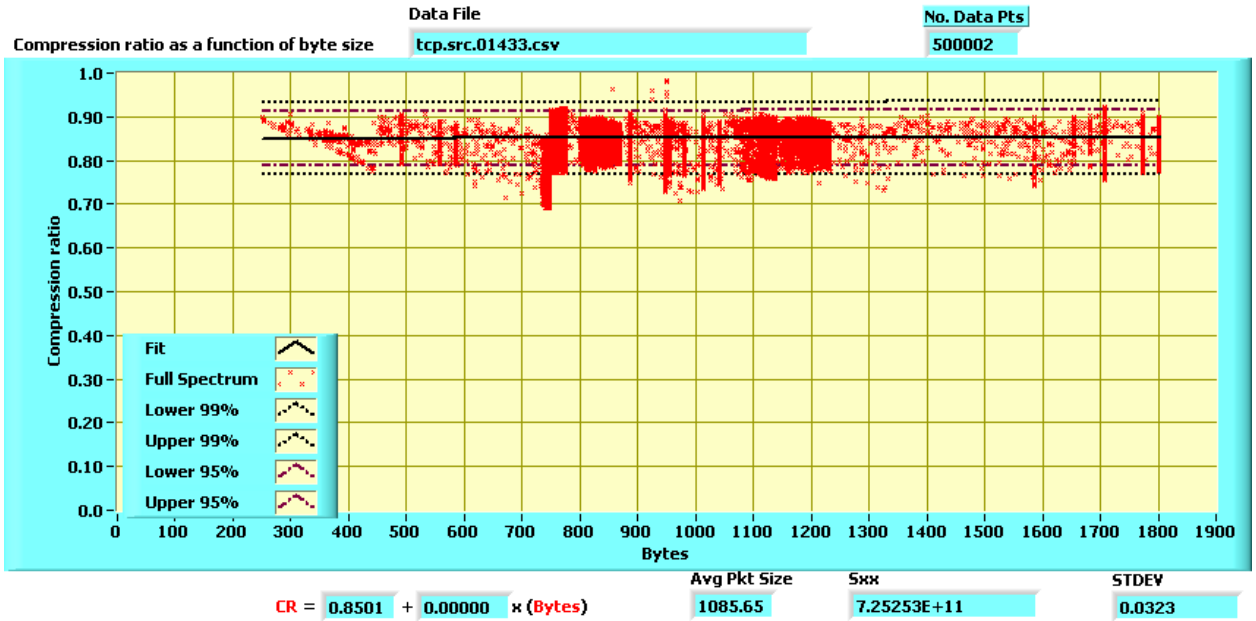

Figure 35. Data plotted as a function of Byte size for metasploit infected network traffic using the split packet analysis on TCP port 1433 source data (default dictionary for each packet).

4.1.3.1 Statistical Analysis on a per Byte Size basis

A variation of the split packet analysis is shown in Figure 36 through Figure 41, which displays the standard deviation for each packet size encountered during the network traffic analysis in which there were more than 20 occurrences of that packet size. For normal distribution assumptions and analysis generally more than 30 observations are required to derive valid statistics on an event. Twenty occurrences were chosen for this analysis. These figures provide the mean and three-sigma standard deviation about the mean for individual packet sizes. Three-sigma standard deviations include the 99th percentile of all occurrences within the data set. . The purpose of this analysis was to show a potential way of training an algorithm by creating statistical bounds for normal and malicious traffic at the packet size. Each distinct packet size would have an acceptable upper and lower boundary that is calculated in the training set. If a packet arrives that falls outside of the established 3-sigma boundaries it would be flagged as suspect. The main difference between this analysis and the previous split-packet analysis is that statistics are created at specific packet sizes rather than generating one model that spanned the spectrum of packet sizes.

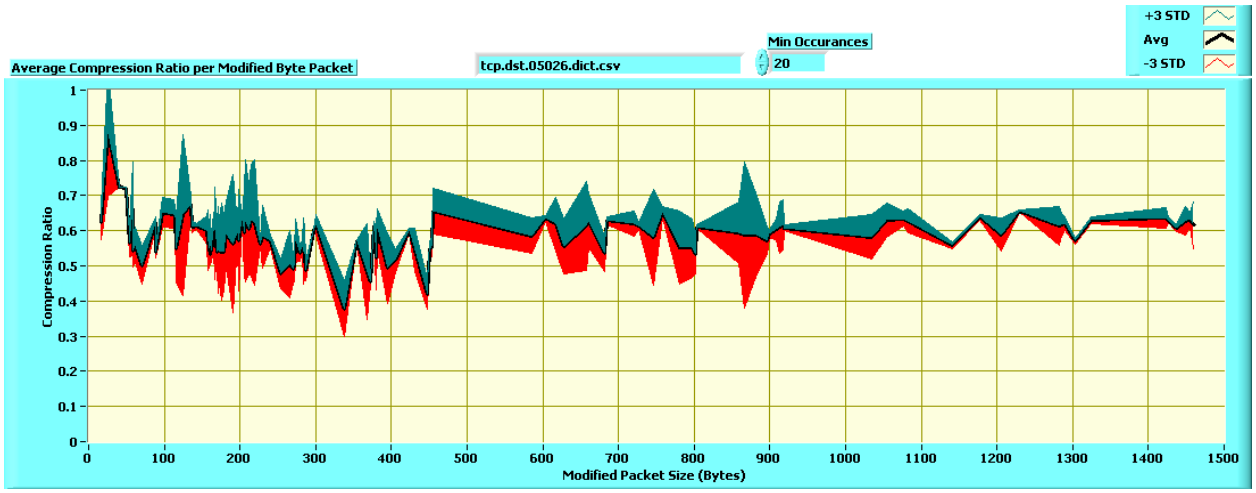

Figure 36. Normal traffic data plotted as a function of byte size using the split packet analysis for TCP port 5026 destination data with statistical analysis displayed for each byte with more than 20 data points (default dictionary for each packet).

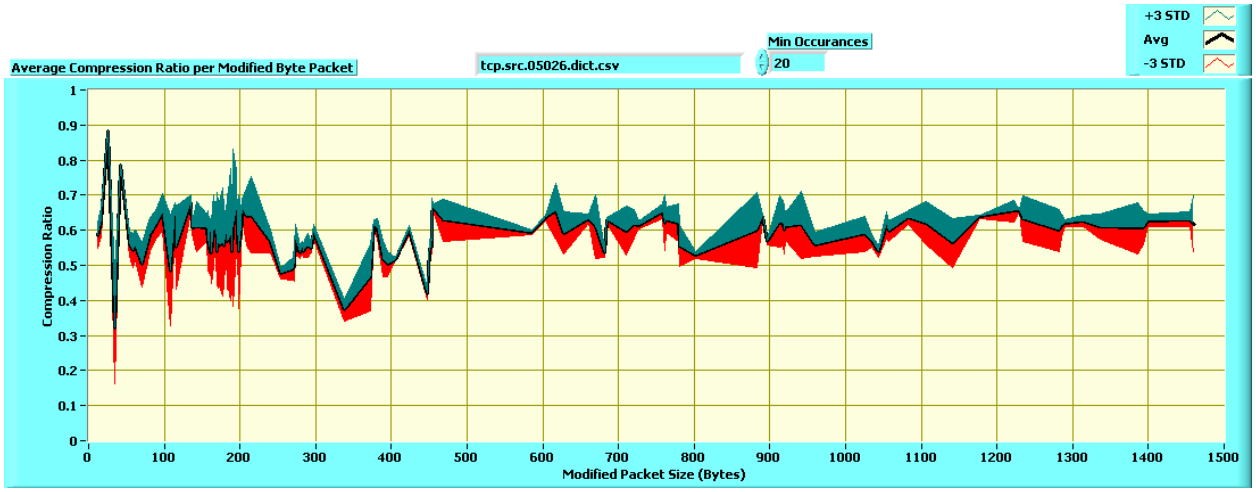

Figure 37. Normal traffic data plotted as a function of byte size using the split packet analysis for TCP port 5026 source data with statistical analysis displayed for each byte with more than 20 data points (default dictionary for each packet).

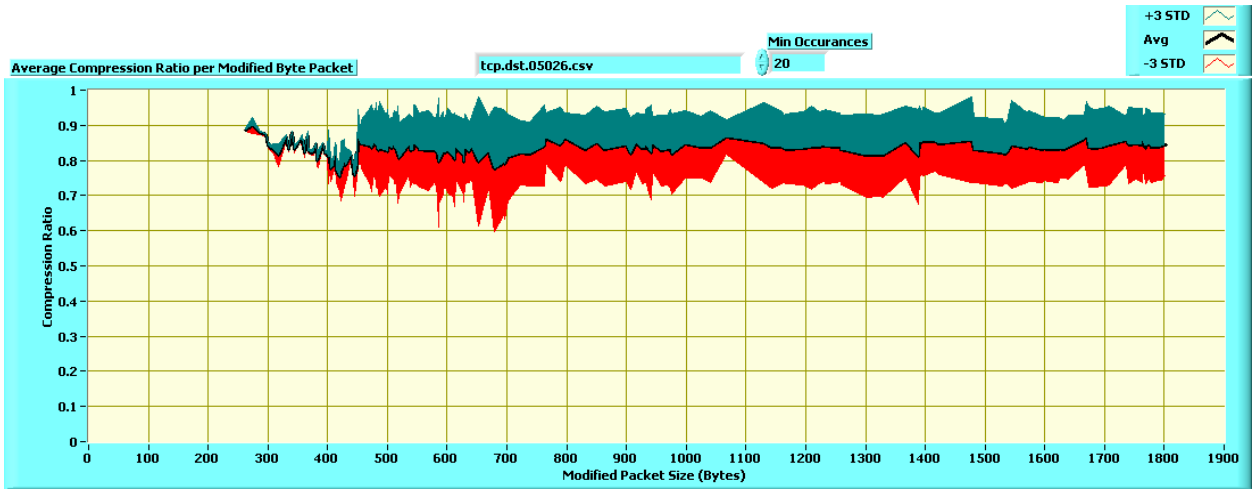

Figure 38. Data plotted as a function of Byte size for metasploit infected network traffic using the split packet analysis on TCP port 5026 destination data with statistical analysis displayed for each byte with more than 20 data points (default dictionary for each packet).

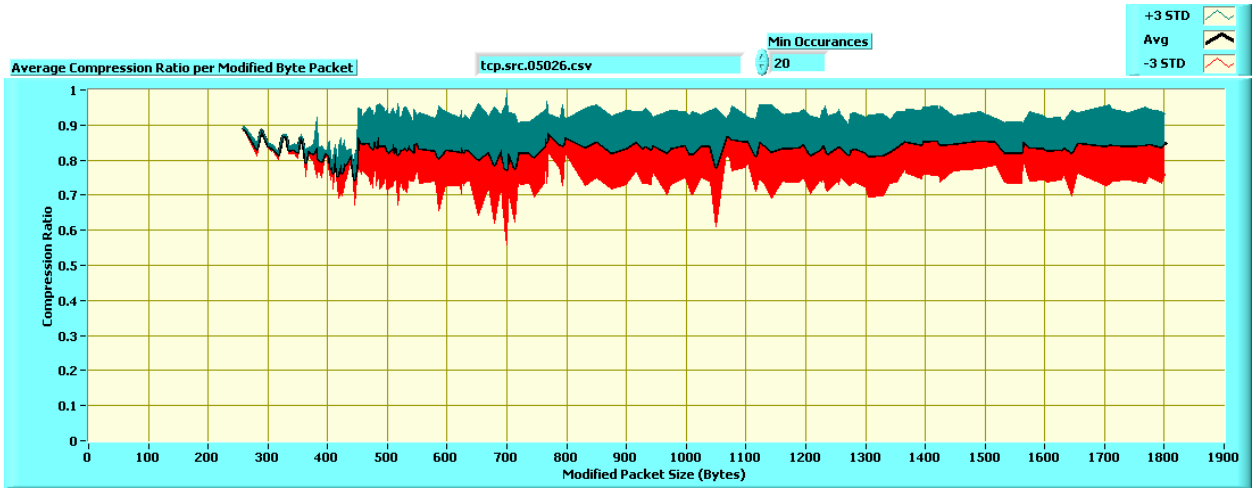

Figure 39. Data plotted as a function of Byte size for metasploit infected network traffic using the split packet analysis on TCP port 5026 source data with statistical analysis displayed for each byte with more than 20 data points (default dictionary for each packet).

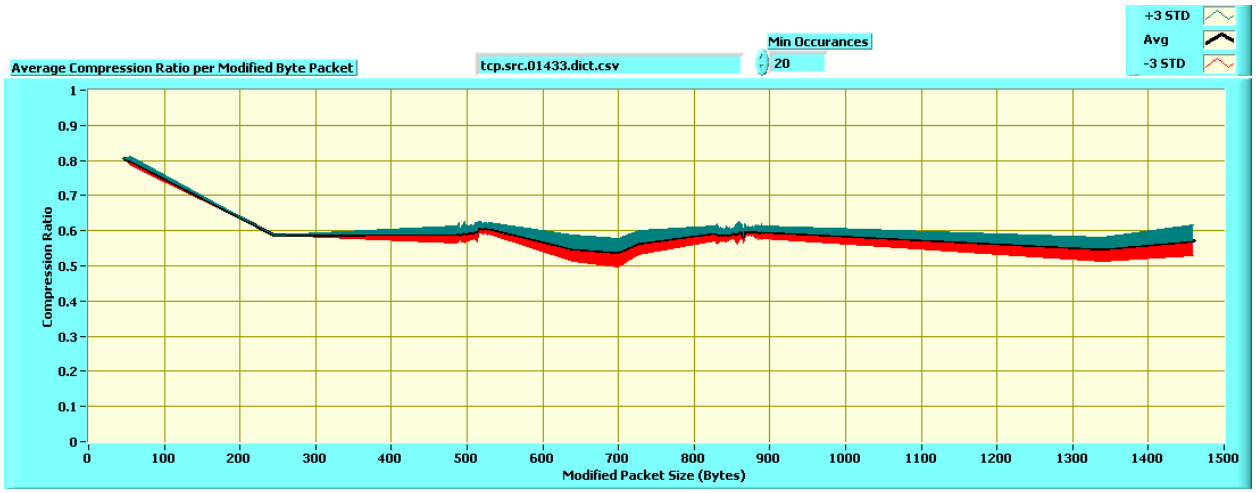

Figure 40. Normal traffic data plotted as a function of byte size using the split packet analysis for TCP port 1433 source data with statistical analysis displayed for each byte with more than 20 data points (default dictionary for each packet).

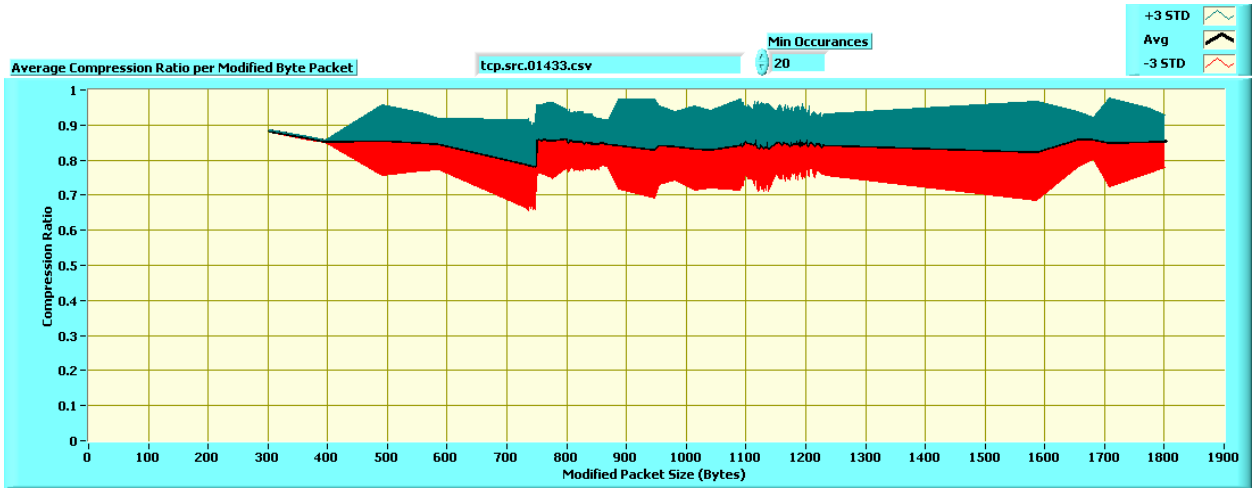

Figure 41. Data plotted as a function of Byte size for metasploit infected network traffic using the split packet analysis on TCP port 1433 source data with statistical analysis displayed for each byte with more than 20 data points (default dictionary for each packet).

4.2 Independent Section Conclusions and Recommendations

The use of compression techniques to identify malicious traffic on SCADA networks in real time appears to have some significant merit for infrastructure protection. The preliminary analyses and results presented herein are clearly able to identify malicious network traffic at the packet level at a very high confidence level for the conditions tested. However, the conditions tested are rather limited in scope and should be expanded into more realistic simulations of hacking events using techniques and approaches that are representative of a real-world attack on a SCADA system. Some specific recommendations are as follows:

1. Develop and implement a real-time software prototype capable of sitting on a real SCADA network and collect data based on known, normal SCADA traffic. This would allow a better characterization of the network traffic from both a volume and

statistical variability aspect. This activity should be done over several weeks of operations at a minimum.

2. Improve the attack simulation technique employed such that multiple attack types can be simulated and that the attack proceeds along more realistic time events rather than infecting every packet on the simulated network with mal-code as was done in the proof of principle study.
3. Implement a simulated attack on at SCADA system with real end goals of system compromise by using known techniques used by hackers to achieve these goals. The goals of this effort would be characterize real malicious traffic and to determine if this traffic can be discriminated against normal traffic for real-time protection against an attack.
4. Repeat items one and two above for multiple types of SCADA systems and attack implementations as time and money allow.

5. INTER-PACKET DEPENDENT RESEARCH

Independent packet analysis provides a solid mathematical basis for statistical analysis compressed data. Additional techniques and combinations of metrics were investigated to try and improve on this basic technique. The preliminary research showed that this was possible for very simple protocols with little variation. Complex protocols have multiple types of packets and additional semi-random or pseudo-random fields that cause problems for the master dictionary and simple compression ratio results. Here, modification of and multiple compression dictionaries were used in an attempt to address the statistical problems with using a single master dictionary while maintaining content comparisons between packets.

5.1.1 Packet Type Attribution Dictionary

The purpose of this experiment was to group similar packets based on the nodes in the compression dictionary that they use. This goal was similar to the dictionary chaining that was done earlier; however, it avoided the performance degradation that occurred during the earlier research. The combination of two metrics for each packet was meant to give more consistent results because similar packets that are grouped together should also compress at approximately the same ratio.

Each node in the dictionary was assigned a packet type designator (PTD) based on the type of packet that originally generated the node in the tree. Each new packet was assigned a packet type based on the composition of nodes used to compress the packet. There are two different metrics worth noting:

- The composition of the packet, such as 36% PTD-0 and 64% PTD-1.
- The percentage of each PTD's nodes that are used in this packet, such as 100% of PTD-0 nodes and 5% of PTD-1 nodes.

The concept behind this approach is that the percentage of a PTD's nodes used by a packet are the more accurate way of grouping similar packets. A PTD that is defined by a single unique characteristic may only have four node matches in the packet, but those 4 nodes should be enough to put that packet in that packet group.

After a packet's type was determined, its compression ratio was tested for fitness with that group's boundaries. If it did not fit, then a new PTD was generated for that packet and an anomaly event was generated. Finally the new nodes of the dictionary were be marked as part of that PTD.

A simple attribution model was run against data from ports 5413 and 27000. Ports 1433 and 5026 eventually cause a segmentation fault because the dictionary filled and additional PTDs were attempted to be added without any representative nodes.

The simple attribution model for our section 5 data was not viable for the larger data sets. This is a combination of two factors:

- Imprecise PTD test results in additional unwanted PTDs.
- First come first allocate for dictionary nodes results in early and frequent packets dominating the dictionary. This manifested itself later when new packets types could not create new nodes and each new packet resulting in a failed PTD match.

In our section 6 data, the research team limited the number of nodes associated with each PTD. Twenty bits are allocated for key size, so 8 bits were used for the PTD and 12 bits for nodes within that PTD. This effectively limited the dictionary to 256 PTDs with each PTD allocated up to 4,096 nodes in the dictionary.

For the complex protocols, this technique did not help. The dictionary filled and the PTDs exceeded its maximum value quickly with most of the PTDs containing only one to five packets during the entire communication.

5.1.2 Master Dictionary with Independent Packet Dictionaries

As seen in previous sections, the master dictionary for an arbitrary channel can grow to very large sizes, possibly filling the dictionary. The majority of the nodes that are added are single use nodes that are not relevant to later packets. This is undesirable because it slows down the compression algorithm, reduces dictionary space available for more useful nodes and increases the attacker's surface area by giving them more nodes to build a payload.

To solve this problem, a new approach to the master dictionary was taken. Each individual packet was compressed using a new empty dictionary. That dictionary was then merged into the master dictionary. Instead of concentrating on compression ratio, new metrics of master nodes, packet nodes, and shared nodes were recorded.

Shared nodes are nodes in the individual packet dictionary that are already contained in the master dictionary. The higher this number is, the more similar the current packet is to its preceding packets. Master nodes are nodes in the master dictionary that are not contained in the individual packet dictionary. The higher this number is, the less similar the current packet is to its preceding packets. Packet nodes are nodes that are in the packet dictionary but are not contained in the master dictionary. The higher this number is, the less similar the current packet is to its preceding packets.

Using individual packet dictionaries proved to provide much more stable data by removing the interdependence between packets in the stream. This also had the result of making it much easier for an attacker to craft a packet that can fit within the normal compression ratio. The attacker can craft the dead space in the exploit to adjust the compression ratio as needed. Two identical packets will create identical packet dictionaries and the master dictionary will not change when the second packet is merged. Similar but slightly different packets will add a few extra nodes to the master in comparison to the total number of nodes in the new packet. A radically different packet will add a substantial number of new nodes to the master dictionary. The two stage dictionary will allow the master to converge on its useful size much quicker. It should only take a single packet of any type to create all or most of the nodes that will be useful for identifying similar packets.

Results showed that three out of the eight data sets resulted in the master dictionary filling up. The dictionaries filled up because the variations in the packets still accumulated using this method. The only dictionary size reduction that occurs is due to the lack of longer dictionary chains that require multiple packets to build. Dictionary node creation due to checksums, sequence number, and value changes still accumulate and fill the dictionary.

5.1.3 Static Master Dictionary

The concept of comparing each packet dictionary to a master dictionary seems solid; however, care must be taken when adding to the master dictionary. This section steps back and

tries to find the statistical way to properly expand and shrink the master dictionary. The first packet was used to generate the master dictionary. All subsequent packets calculate several comparison metrics against the master dictionary but not alter it.

In Figure 42 and Figure 43, the node dispersal for the first 1,024 packets appears to result in three groups of packets. This type of grouping is very useful because each group could be considered a different type of packet contained in the communication channel. The statistical variation around each group can be computed and subsequent packets can be defined as a certain type or unmatched. In this particular instance, the group to the far left is the closest group to matching the original packet. The group at 250x250 is a close but slightly different group, and the group at 570x10 is a completely different packet group.

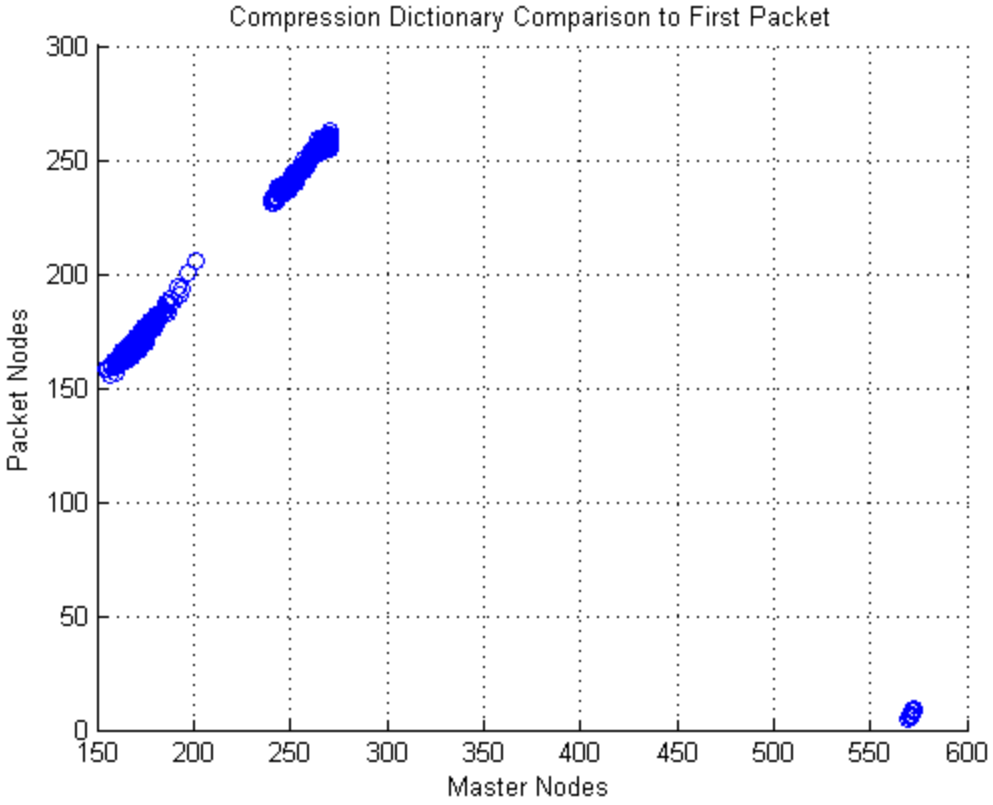

Figure 42. First 1,024 Packet Node Analysis for TCP destination port 5026.

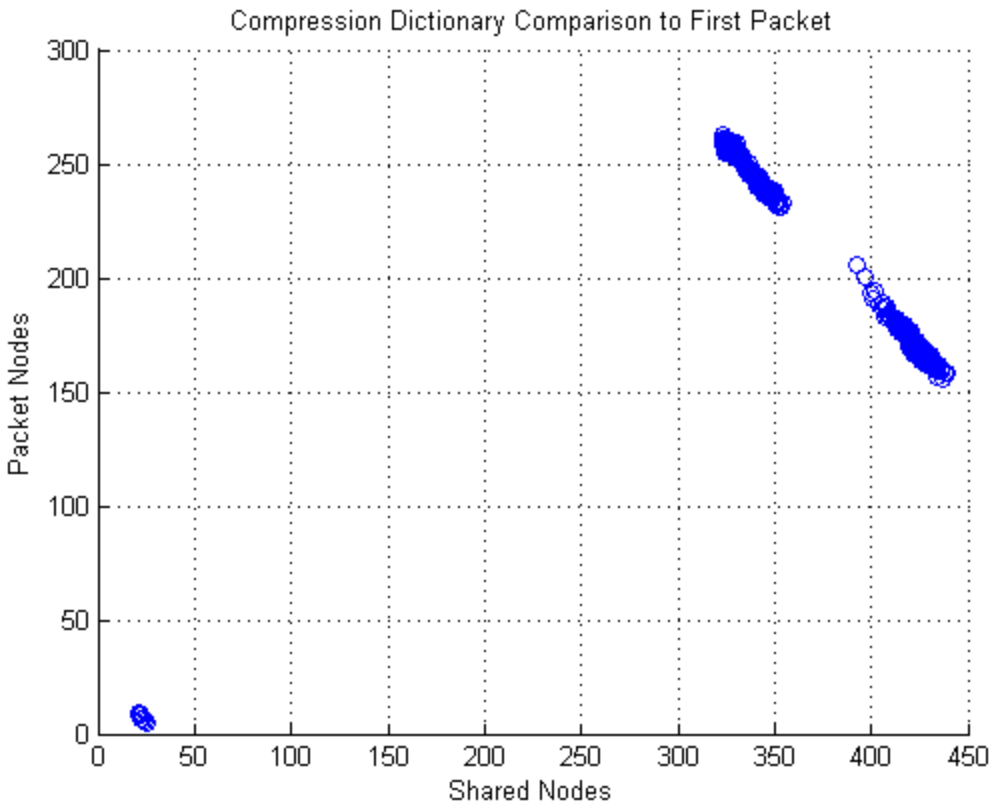

Figure 43. First 1,024 Packet Node Analysis for TCP destination port 5026.

Figure 44 contains the first 9,999 packets for TCP destination port 5026 with all 3 metrics graphed together. The packet grouping has continued but with several more groups getting resolved. There also appears to be some rogue packets on the far left.

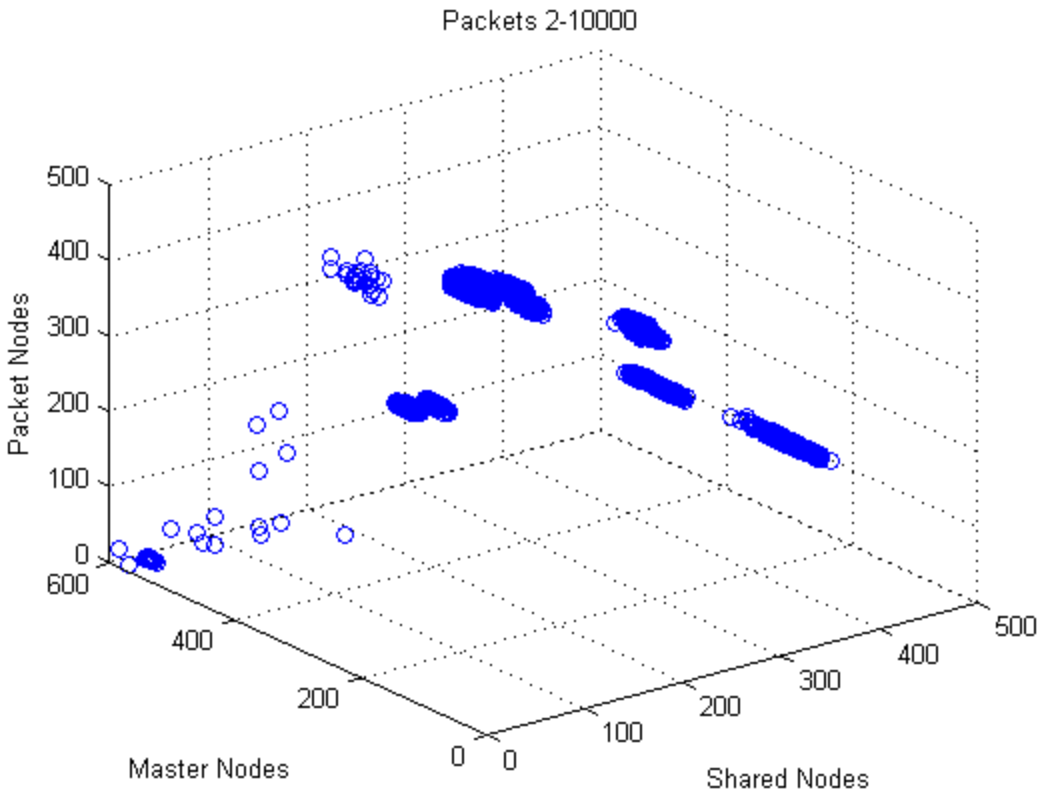

Figure 44. First 9,999 Packet Node Analysis for TCP destination port 5026.

5.2 Master Section Conclusion

The master dictionary approach appears to provide a meaningful way to categorize and compare packets within a communication channel. The analysis is subjective for now so future research will be necessary to quantify the packet groupings and then introduce malicious packet comparisons to get false positive and false negative rates.

There are always outliers in the normal traffic when using data analysis methods that include inter-packet dependency. This means that there will always be false positives given by this technique and they will need to be mitigated in some way.

6. Conclusion

Preliminary research into SPADUC showed that SCADA protocols have several features that will make them good candidates for compression analysis. Four out of the five protocols examined were compressible. When using an inter-packet dependent dictionary, most of the packets compressed at a better ratio than the previously compressed packets indicating that they shared similar content. In all but the simplest protocols, problems arose with simple inter-packet dictionary compressing. These problems such as seldom but valid handshakes resulted in unfavorable false positives when compared directly against the majority of the packets. Another problem with the inter-packet method was an interdependency of the packet compression results making it impossible to use statistical analysis to quantify false positive and negative rates.

After the initial inter-packet analysis, research proceeded along two lines to tackle the two separate issues. The first approach addressed the inter-packet dependency by removing it and looking at compression techniques using a static, default dictionary. The second approach focused on developing ways to modify the compression metrics so different packet types in a protocol would be grouped together into common packet types and compared based on packet type attributes.

The static, default dictionary approach was able to show that normal SCADA protocols and infected SCADA protocols compress at statistically significant different ratios, making discrimination on a statistical basis possible. Variations of this approach were investigated to enhance compression differences for very large message packets and were proved successful for the data sets investigated. Essentially larger packets were subdivided into smaller ones and analyzed independently with the results attributed back to the original packet. This concept improved the discrimination statistics and should also provide a measure of defense by introducing a way to modify how large packets are analyzed. Ultimately this will make it more difficult for an attacker to hide malicious data within larger packets and provides one more control mechanism for identifying a hypothetical attacker.

The packet type recognition line of research had difficulty finding universally useful ways to group packets by compression metrics. All but the final approach suffered dictionary drift caused by repeated one-off dictionary extensions that provided no value but still filled the dictionary. The final approach which just performed comparisons between the first packet and all subsequent packets showed that chosen metrics did appear to group into distinct types of packets.

Further research should concentrate on combining the two lines of research to use both the compression ratio and the dictionary description metrics. The dictionary description metrics need to be quantified and divided into groups that could then be used extend or prune the initial dictionary. Also, more data sets from additional SCADA systems should be examined and analyzed to increase the confidence that compression algorithms have an important place in SCADA cyber security methods.