The use of Snap Length in Lossy Network Traffic Compression for Network Intrusion Detection Applications

Sidney C. Smith
Sidney.c.smith24.civ@mail.mil
Computational Information Sciences Directorate
U.S. Army Research Laboratory
Aberdeen Proving Ground, MD 21005, U.S.A

Robert J. Hammell II
rhammell@towson.edu
Department of Computer and Information Sciences
Towson University
Towson, MD 21252, U.S.A

Abstract

In distributed network intrusion applications, it is necessary to transmit data from the remote sensors to the central analysis systems (CAS). Transmitting all the data captured by the sensor would place an unacceptable demand on the bandwidth available to the site. Most applications address this problem by sending only alerts or summaries; however, these alone do not always provide the analyst with enough information to truly understand what is happening on the network. Since lossless compression techniques alone are not sufficient to address the bandwidth demand, applications that send raw traffic to the CAS for analysis must employ some form of lossy compression. This lossy compression may take the form or dropping entire sessions, packets, or portions of packets. In this paper we explore impact of compressing network traffic by dropping portions of packets. This is accomplished by truncating packets through adjusting the snap length.

Keywords: compression, network intrusion detection, snap length, Snort, Tcpdump

1. INTRODUCTION

Distributed Network Intrusion Detection Systems (NIDS) allows a relatively small number of highly trained analysts to monitor a much larger number of sites; however, they require information to be transmitted from the remote sensor to the central analysis system (CAS) as pictured in Figure 1. Unless an expensive dedicated NIDS network is employed, this transmission must use the same channels that the site uses to conduct their daily business. This makes it important to reduce the amount of information transmitted back to the CAS to minimize the impact that the NIDS has on daily operations as much as practical.

Smith and Hammell (2017) proposed that it should be possible to create a lossy compression tool using anomaly detection techniques to rate each session and a modification of the Kelly criterion (Kelly, 1956) to select how much traffic from each session to return as seen in Figure 2.

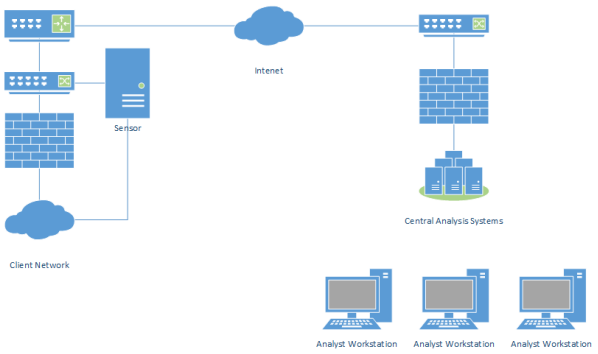

Figure 1. Distributed network intrusion detection

Once the determination of how much traffic to return is made, it is necessary to understand the best way to reduce that traffic. One could carve entire sessions out of the network traffic as Long and Morgan did. (2007) One could drop individual packets as Smith, Hammell, and Neyens did. (2017) Or one could truncate packets as Long did with the "snapper" tool. (2004) This research will consider the implications of the last method adjusting the snap length which truncates packets to achieve lossy compression.

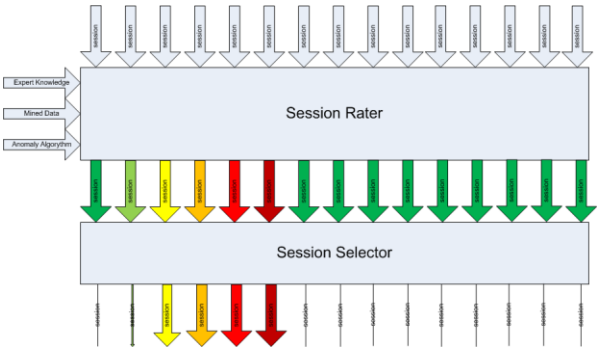

Figure 2. Kelly compressor

The remainder of this paper is organized into the following sections: Section 2 provides background, Section 3 outlines the approach chosen to address this problem, Section 4 presents our results, and Section 5 provides a conclusion and discussion of future work.

2. BACKGROUND

One popular strategy for implementing a distributed NIDS is to do all of the intrusion detection on the sensor and send only alerts or logs to the CAS. (Roesch, 1999) (Paxson, 1999) A second strategy might be to use lossless compression to reduce the size of the data returned to the CAS. A third strategy is to implement some form of lossy compression algorithm to send back relevant portions of traffic.

There are three problems with the first strategy. The first is that it has the potential to over burden the sensor's central processing unit (CPU) and introduce packet loss. Smith *et al.* discovered that the impact of packet loss can sometimes be quite severe for even small rates of packet loss. (2016a) (2016b) The second problem is that the alerts by themselves often do not contain enough information to determine whether the attack was successful. The third problem is that these systems are most often implemented with signature-based intrusion detection engines. Signature-based systems may be tuned to produce few false positives; however, they are ineffective at detecting zero-day and advanced persistent threats. (Kremmerer & Giovanni, 2002)

The problem with the second strategy is that lossless compression alone simply is not capable of reducing the amount of traffic enough. Using GNU Zip to compress the 2009 Cyber Defense Exercise dataset provides a compression ratio of 2:1. (Smith, Neyens, & Hammell, 2017) Compression ratios of better than 10:1 are required to minimize the impact of NIDS on day-to-day operations.

The third strategy is to use lossy compression to provide a solution. Network traffic may be considered to be composed of sessions that span spectrums from known to unknown and malicious to benign as illustrated in Figure 3. Quadrant III, the known malicious quadrant, is the domain of intrusion prevention systems as described by Ierace, Urrautia, and Bassett (Ierace, Urrutia, & Bassett, 2005). This research is most interested in quadrant II, the unknown malicious quadrant, because that is the quadrant where evidence of zero-day and advanced persistent threat attacks will be found. In 2004, Kerry Long described the Interrogator Intrusion Detection System Architecture (2004). In this architecture, remotely deployed sensors collect network traffic and transmit a subset of the traffic to the analysis level. Interrogator employs "a dynamic network traffic selection algorithm called Snapper'". (2004). Long and Morgan describe how they used data mining to discover known benign traffic that they excluded from the data transmitted back to the analysis servers (2007).

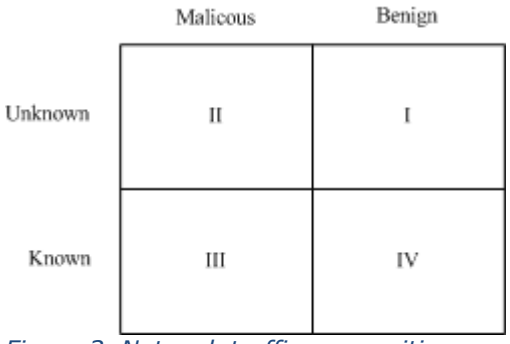

Figure 3. Network traffic composition

3. APPROACH

Tcpdump (Jacobson, Leres, & McCanne, 2017) is a very popular network capture tool. The data format used by tcpdump to store the captured network traffic has become the *de facto* standard format for network capture data. Snort (Roesch, 1999) is a very popular signature based network intrusion detection tool. Both tcpdump and snort support an option to set the snap length. This option is used to set the maximum size of any packet collected. Packets larger than the snap length will be truncated. It is primarily used to improve efficiency when the maximum transmission unit of the network is known. One might suspect that conducting several iterations of these experiments would be as simple as repeatedly executing one of the commands seen in **Error! Reference source not found.**. The authors of tcpdump pulled the packet capture routines out of tcpdump into a stand-alone library known as the packet capture library or libpcap (Jacobson, Leres, & McCanne, 2015). Today both tcpdump and snort leverage this library. It turns out that both tcpdump and snort implement snap length by passing the option to libpcap (Jacobson, Leres, & McCanne, 2015) which only implements this feature for live traffic capture.

```
$ tcpdump -r ${DATASET} -s ${SNAPLEN} \
> -w - |
> snort -N -c ${RULESET} -k none -r - -l .

$ snort -r ${DATASET} -k none \
> –c ${RULESET}\
> --snaplength ${SNAPLEN} -l .
```

Figure 4. Command line

To use the snap length features of either tcpdump or snort we needed to leverage an experimental environment similar to the one seen in Figure 5. Replaying a dataset several times at some multiple of the original speed small enough to ensure that packets are not lost in transmission requires a significant amount of time. We conducted this experiment only twice to gain a baseline. We developed a tool that will implement the snapping in software. We tested it against the baseline we established using the experimental environment. The validated tool was then used to quickly test the impact of snap length across multiple datasets.

Experimental Baseline

The experimental environment seen in Figure 5 consists of two workstation class systems with Gigabit Ethernet cards directly connected to each other. We did not configure the interfaces of these cards to prevent any extraneous traffic from appearing on this network. Albus is designated as the source, and tcpreplay (Turner & Bing, 2013) was used to replay the traffic. Severus was designated as the sink and tcpdump and snort were used to collect and analyze the traffic. Several iterations were conducted changing the snap length. The snap length used, the percentage for the original size of the data set, and the number of alerts are collected and plotted.

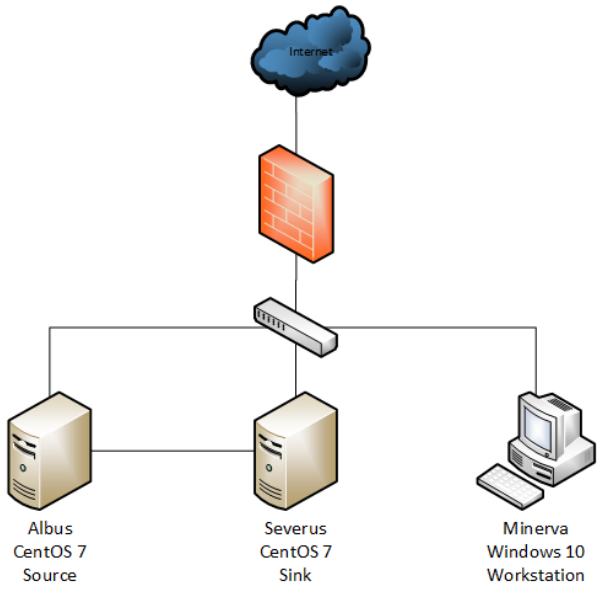

Figure 5. Experimental environment

Snapping Tool

There are three length fields in libpcap files. The first is a global length field. We set this field by passing the new snap length to the pcap_open_dead() function when we created the pcap_t structure which we passed to pcap_dump_open(). The other two length fields are contained in the pcap_pkthdr

structure. These are `caplen` and `len`. The `len` field is the original length of the packet, and the `caplen` field is the number of bytes actually stored in the libpcap file.

In previous research, we developed the `pcapcat` program (Smith S. C., 2013). This program simply takes the list of libpcap file names on the command line and reads each file writing it to standard output. This provided a necessary first step for any tools which will manipulate libpcap files, and a convenient method to join several libpcap files into one file. We took this program and added a snap length option. Implementing this option involved setting the global snap length when we created the output handle, and setting the `caplen` value of the `pcap_pkthdr`.

Datasets

In the following section we provide a brief summary of the various datasets that were used in our experiment. Table 1 provides a summary of the duration and packet count for each of these datasets.

DARPA Datasets

As part of their evaluation of intrusion detection systems, Lippman *et al.* created a dataset of synthetic network traffic (2000). We used the small sample dataset which was provided before the experiment to give the participants examples of the data that they would be provided in the evaluation. This dataset is about 10 min long and was used to validate that the tools were working correctly. They also created the four hour dataset. This dataset was used to evaluate the efficiency of the intrusion detection techniques. We used it because it is large enough to provide a good baseline but small enough to allow us to conduct our experiment in a reasonable amount of time. We used it to compare the results of using the snap length options of `tcpdump` and `snort` to our snapping tool. Finally we used the testing data from Wednesday and Friday of week 2. We selected these two days because Wednesday contains the smallest number of alerts and Friday contains the largest number of alerts.

Cyber Defense Exercise 2009

In 2009 the National Security Agency/Central Security Service (NSA/CSS) conducted an exercise pitting teams from the military academies of the United States and Canada against teams of professional network specialists to see who best defended their network. Data from this exercise was captured and used by Sangster *et al.* in his efforts to generate labeled datasets (2009). Two network traffic sensors were employed in the exercise: gator-usama010 and gator-usama020. We used the `pcapcat` program to consolidate the individual hours of for network traffic collected by each sensor into two libpcap files.

Mid-Atlantic Collegiate Cyber Defense Competition

Based upon the pattern of the Cyber Defense Exercises, a group of industry academics created the collegiate cyber defense competitions (Carlin, Manson, & Zhu, 2010). We used the network capture data for the Mid-Atlantic Collegiate Cyber Defense Competitions from 2010 and 2011 which is available from: https://www.netresec.com/?page=MACCDC.

Real World

We were able to collect real world network traffic from the top level architecture of one site of a research laboratory on the Defense Research Engineering Network.

Table 1. Datasets

Name	Seconds	Packets
DARPA98ss	624	14,523
DARPA984h	19,258	233,428
DARPATestW2Wed	86,400	2,026,473
DARPATestW2Fri	90,432	2,177,646
CDX09_usama010	378,000	5,218,144
CDX09_usama020	345,600	42,293,657
MACCDC2010	275,666	264,973,151
MACCDC2011	165,243	134,465,786
Real World	138,895	2,256,633,016

4. RESULTS

First we will review the results of our validation exercises. Then we will present the results of our validated snapping tool.

Validation in the Experimental Environment

The first step in the process is to ensure that our snapping tool provides the same results as we obtained using `tcpdump`. To do this we will take the DARPA98 Four Hour and DAPRA98 Small Sample datasets and replay them in the experimental environment seen in Figure 5. We automated 30 iterations of Albus replaying the traffic using the `tcpreplay` tool while Severus used `tcpdump` with using snap lengths ranging from 1542-42. These captured files were then analyzed with `snort`.

DARPA 98 Four Hour

To ensure that this experiment using the four hour dataset completed in a reasonable amount of time, we replayed the traffic at 10 times the original speed. In Figure 6, we have plotted the percentage of the original file size using triangles. We have plotted the alert loss rate (ALR) as a percentage in circles. We have also plotted the packet loss rate (PLR) as a percentage in crosses. In Figure 7, we have plotted the results of using the snapping tool on the same dataset. Comparing the graphs, we find that the relationship between the ALR and the snap length for the experimental environment and the snapping tool is very similar. The differences between the relationship between the compression and the snap length between the two experiments may be attributed to the PLR.

DARPA 98 Small Sample

To further assure that our snapping tool is performing correctly, we repeated the experiment with the DAPRA 98 Small Sample dataset. This dataset is about 10 min long allowing us to replay the traffic at the original speed and still complete the experiment in a reasonable amount of time. In Figure 8, we have plotted the percentage of the original file size using triangles.

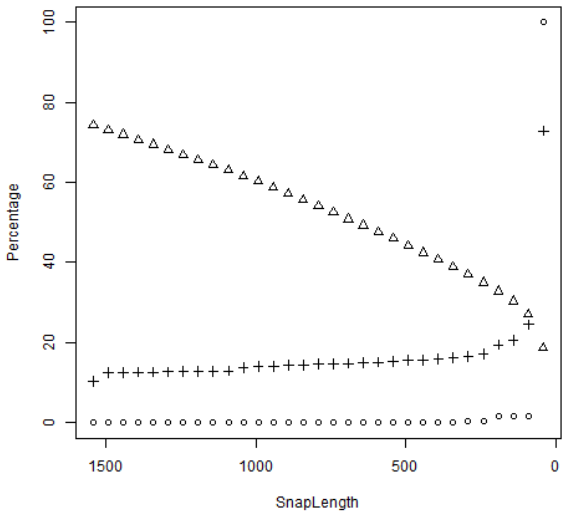

Figure 6. Results of using tcpdump to snap the packets of the DARPA 98 Four Hour dataset in the experimental environment

We have plotted the ALR as a percentage in circles. We have also plotted the PLR as a percentage in crosses. One thing of note is that packet loss is completely from packets that `snort` has discarded. Since these packets were discarded and not dropped, they are not subtracted from the size when the percentage of the original size is computed. In Figure 9, we have plotted the results of using the snapping tool on the same dataset. Again the results are very similar and from this we conclude that our snapping tool is truncating the packets in the same manner that they would be truncated using either `tcpdump` or `snort`.

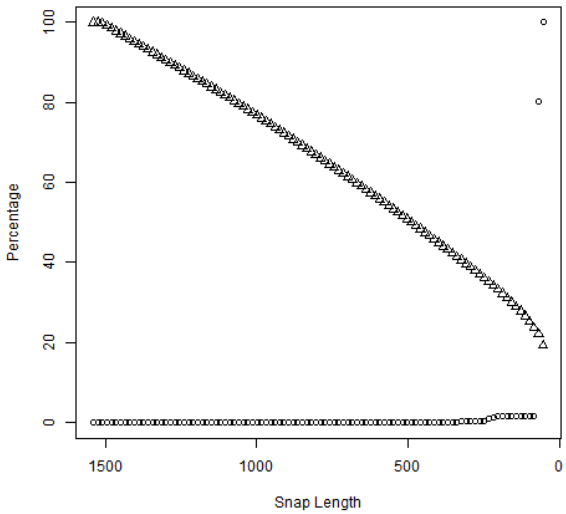

Figure 7. Snap length verses the ALR and Compression of the DARPA 98 Four Hour datasets using the snapping tool

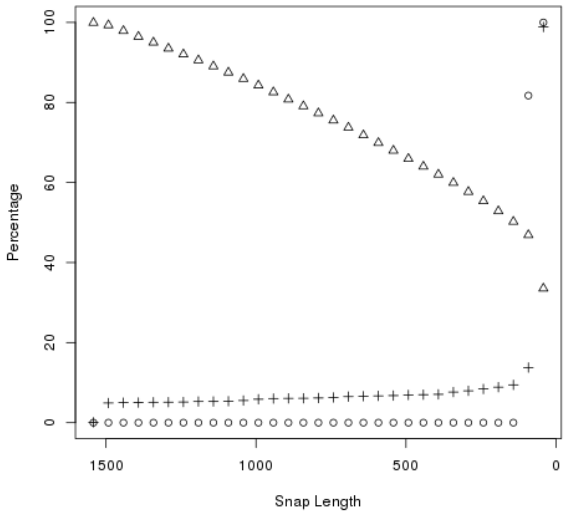

Figure 8. Results of using tcpdump to snap the packets of the DARPA 98 Small Sample dataset in the experimental environment

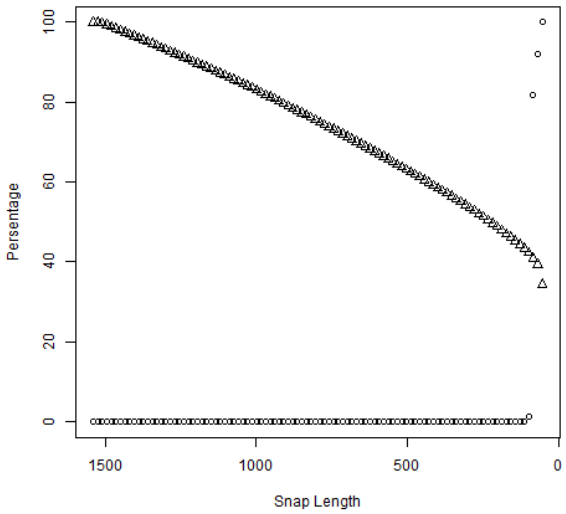

Figure 9. Snap Length verses the ALR and Compression of the DARPA 98 Small Sample dataset using the snapping tool

Experimentation with the Snapping Tool
Having validated that the snapping tool performs in the same manner as the snap length option to `tcpdump`, we may forgo further use of the experimental environment. We created a shell script to automate the snapping and analysis of the remaining datasets.

DARPA 98 Testing Week 2 Wednesday
In Figure 10 and Figure 11 we see the results of using our snapping tool on the 2 days we selected from the DARPA 98 Testing dataset. Notice that for each of these 2 datasets, we are able to gain a significant amount of compression by snapping packets with little or no increase in the ALR. The same may be said for the datasets that we used to validate the snapping tool.

Cyber Defense Exercise
In Figure 12 and Figure 13 we see the results of using our snapping tool on the Cyber Defense Exercise 2009 datasets. These graphs show a much earlier rise in ALR.

Mid-Atlantic Collegiate Cyber Defense Competition Datasets
In Figure 14 and Figure 15 we see the results of applying our snapping tool to the Mid-Atlantic Collegiate Cyber Defense Competitions of 2010 and 2011. With the 2010 data we see more dramatic rise in ALR, but not as dramatic as the rise we saw in the CDX data. With the 2011 data we see that it is possible for the snapping process to create alerts in the data that did not exist previously. The creation of false positive alerts was not one of the anticipated outcomes.

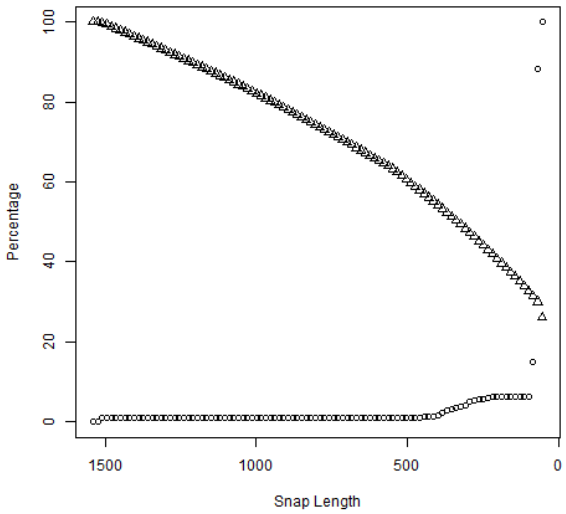

Figure 10. Snap length verses the ALR and Compression of the DARPA 98 testing week 2 day 3 datasets using the snapping tool

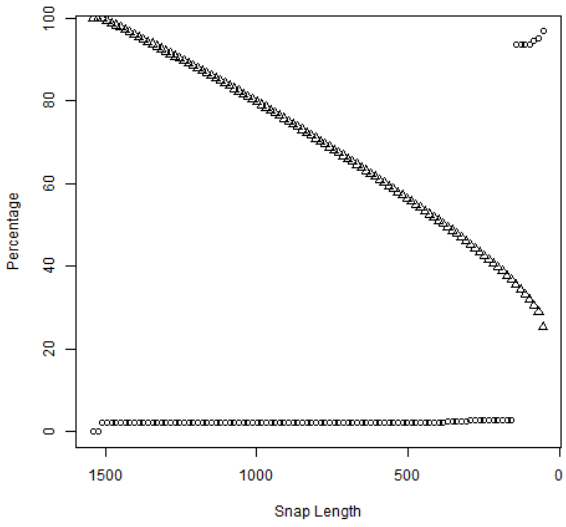

Figure 11. Snap length verses the ALR and Compression of the DARPA 98 testing week 2 day 6 datasets using the snapping tool

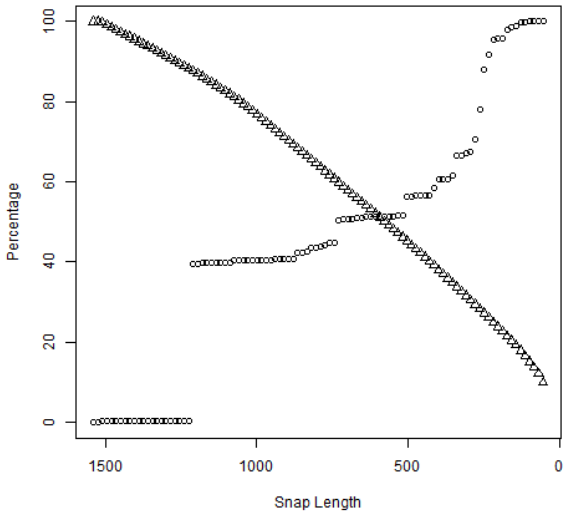

Figure 12. Snap Length verses the ALR and Compression for CDX2009 usama010 using the snapping tool

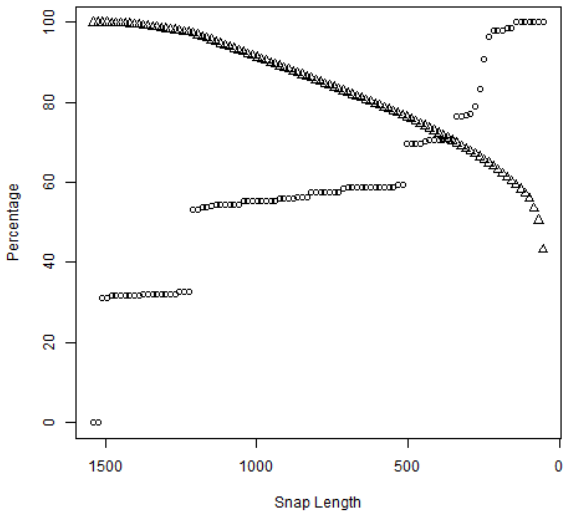

Figure 13. Snap Length verses the ALR and Compression for CDX2009 usama020 using the snapping tool

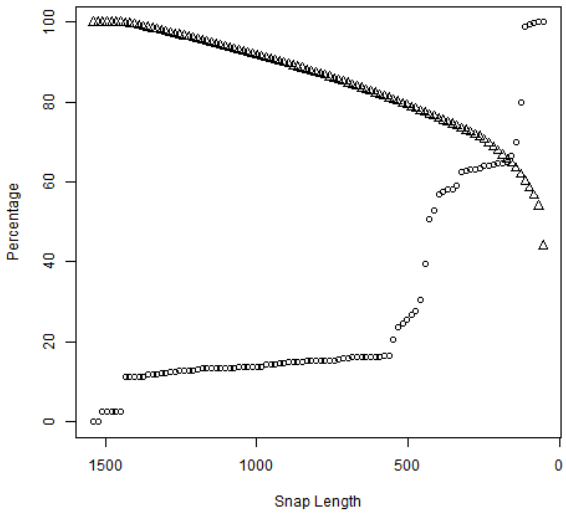

Figure 14. Snap length verses the ALR and Compression of the MACCDC 2010 dataset using the snapping tool

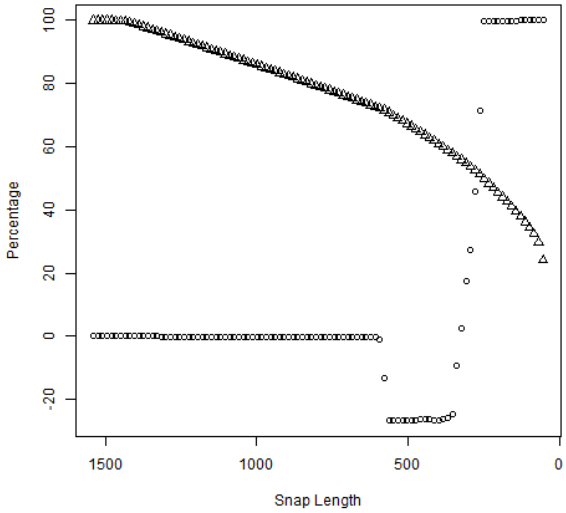

Figure 15. Snap length verses the ALR and Compression of the MACCDC 2011 dataset using the snapping tool

Real World

The results of the experiment using real world data may be seen in Figure 16. It would appear that that data set had a small number of very large packets, but once the snap length reached about 1500 the size started falling steadily, but the ALR raised quickly only to level off.

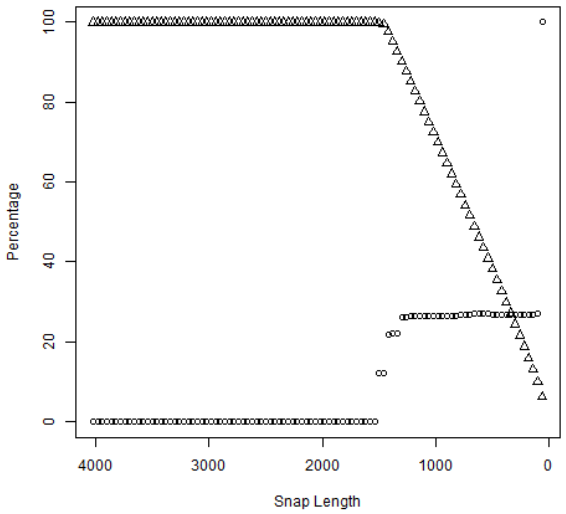

Figure 16. . Snap length verses the ALR and Compression for real world data using the snapping tool

5. CONCLUSIONS

Looking at our results from the DARPA datasets it would appear that employing snap length as a compression tool has the potential to reduce the size of the traffic that must be transmitted from the sensor to the CAS. Our results from the Cyber Defense Exercise data indicate that this might be a very dangerous technique as the ALR rises rapidly with the decrease in snap length. Our results from the Mid-Atlantic Collegiate Cyber Defense Competition and real world data seem to occupy the middle ground with the caveat that the technique may introduce false positive alerts.

It might appear that the malicious content in new traffic is deeper in the packet than malicious content in older traffic; however, an examination of the traffic reveals that this is not the case. We examined several packets which triggered an alert in the original data but did not trigger an alert in the abridged data. In each of these packets, the string in the rule existed in the abridged packet. The explanation for our results lies in the number of discarded packets observed in the experiment using the DARPA 98 Small Sample dataset in our experimental environment. Even though we used the option to instruct `snort` not to validate the checksums, it is discarding truncated packets. We are not seeing that the malicious nature is deeper in the packets in new traffic. We are seeing that packets in general including those with a malicious nature are larger in newer traffic. Also a large number of alerts in the DARPA datasets come from very small packets. A detection tool that does not discard truncated packets would have detected the malicious traffic. Also analysts reviewing the truncated traffic based upon alerts generated by `snort` seeing the unabridged traffic would be able to use the truncated traffic to conduct their analysis.

Although tools like `snort` are best run on the sensor where they may have a full view of the network traffic, there is value in running tools like this on the CAS where the size of the ruleset will not negatively impact of the amount of traffic which may be collected. In future work it will be necessary to explore other methods of lossy compression that might not have the same issues. Alternatively `snort` could be altered to accept truncated packets or a similar tool could be developed that would accept truncated packets.

6. REFERENCES

Carlin, A., Manson, D. P., & Zhu, J. (2010). Developing the Cyber Defenders of Tomorrow with Regional Collegiate Cyber Defense Competitions (CCDC). *Information Systems Education Journal*, 3-10.

Ierace, N., Urrutia, C., & Bassett, R. (2005). Intrusion Prevention Systems. *Ubiquity*, 2-2.

Jacobson, V., Leres, C., & McCanne, S. (2015, March 8). *PCAP -- packet capture library*. Retrieved from Tcpdump/Libpcap: http://www.tcpdump.org/manpages/pcap.3pcap.1.html

Jacobson, V., Leres, C., & McCanne, S. (2017, February 2). *tcpdump -- dump traffic on a network*. Retrieved from TCPDUMP & LIBPCAP: http://www.tcpdump.org/manpages/tcpdump.1.html

Kelly, J. L. (1956). A new interpretation of information rate. *Information Theory, IRE Transactiosn on*, 185-189.

Kremmerer, R. A., & Giovanni, V. (2002). Intrusion detection: a brief history and overview (supplement to Computer magazine). *Computer*, 27-30.

Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., . . .

Zissman, M. A. (2000). Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings* (pp. 12-26). Hilton Head, SC: IEEE.

Long, K. S. (2004). *Catching the Cyber Spy: ARL's Interrogator.* Aberdeen Proving Ground: Army Research Laboratory.

Long, K. S., & Morgan, J. B. (2007). *Using data mining to improve the efficiency of intrusion detection analsysis.* Army Research Laboratory. Aberdeen Proving Ground (MD): Army Research Laboratory.

Paxson, V. (1999). Bro: a system for detecting network intruders in real-time. *Computer Networks*, 2435-2463.

Roesch, M. (1999). Snort: lightweight intrusion detection for networks. *Proceedings of the 13th System Administration Conference (LISA '99)* (pp. 229-238). Seattle, WA: USENIX.

Sangster, B., O'Conner, T., Cook, T., Franelli, R., Dean, E., Adams, W. J., . . . Conti, G. (2009). Toward instrumenting network warfare competitions to generate labeled datasets. *Proc. of the 2nd Workshop on Cyber Security Experimentation and Test CSET09.* Montreal Canada.

Smith, S. C. (2013, May). The effect of packet loss on Network Intrusion Detection. *Towson University Institutional Repository*. Towson, MD, USA: Towson University.

Smith, S. C., & Hammell, R. J. (2017, aug). Proposal for Kelly Criterion-Inspired Lossy Network Compression for Network Intrusion Applications. *Journal of Information Systems Applied Research, 10*(2), 43-51.

Smith, S. C., Hammell, R. J., Wong, K. W., & Carlos, J. M. (2016). An Experimental Exploration of the Impact of Host-Level Packet Loss on Network Intrusion Detection. *Cybersecurity Symposium (CYBERSEC)* (pp. 13-19). IEEE.

Smith, S. C., Hammell, R. J., Wong, K. W., & Carlos, J. M. (2016). An experimental exploration of the impact of multi-level packet loss on network intrusion detection. *2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA)* (pp. 23-30). Towson, MD: IEEE.

Smith, S. C., Neyens, S. R., & Hammell, R. J. (2017). The use of Entropy in Lossy Network Traffic Compression for Network Intrusion Detection Applications. *Proceedings of the 12th International Conference on Cyber Warfare and Security ICCWS 2017* (pp. 352-360). Reading (UK): Academic Conferences and Publishing International Limited.

Smith, S. C., Neyens, S. R., & Hammell, R. J. (2017). The use of Entropy in Lossy Network Traffic Compression for Network Intrusion Detection Applications. *Proceedings of the 12th International Conference on Cyber Warfare and Security ICCWS* (pp. 352-360). Reading (UK): Academic Conferences and Publishing International Limited.

Turner, A., & Bing, M. (2013, December 14). *Tcpreplay: Pcap editing and replay tools for *nix*. Retrieved from Syn Fin dot Net: http://tcpreplay.synfin.net

Glossary

Real World Data: This is data collected from a network that is actually connected to the Internet and in use for real work by real people.

Signature-base IDS: These are intrusion detection systems which employ a ruleset of patterns or signatures that are then compared against packets in the data stream. Packets that match the patterns or signatures generate alerts.

Snap Length: This is also known as the snapshot length. Both tcpdump and snort provide options to limit the amount of data contained in each packet that captured from the network.

Top Level Architecture: This consists of the group of systems which are used to connect a local area network to the Internet. This is typically composed of a frontier router which is directly connected to the Internet Service Provider, a firewall, and a security router. It may also contain demilitarized zones for external facing resources.