The Use of Tainted Flows in Network Compression for Distributed Network Intrusion Detection

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 detection, it is necessary to transmit data from remote sensors to a central analysis system. Transmitting all the data captured by a 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. Lossless compression techniques alone are not sufficient to address the bandwidth demand. This paper explores a concept called tainted flows employed in a lossy compression technique. The tainted flows technique uses a Bloom filter of malicious N-gram hashes to taint or mark flows as malicious. Network traffic is compressed by stopping the transmission of untainted flows after a user-defined threshold, but keeping all of the tainted flows. This tainted flows method was used to compress synthetic and competition data sets from 1998 until 2017 to about 30% of their original size with less than 1% loss of Snort alerts.

Keywords: network intrusion detection, lossy compression, N-grams, Bloom filters, Snort, Tcpdump.

1. INTRODUCTION

Distributed Network Intrusion Detection Systems (NIDS) allow 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.

The contribution of this research is to explore one method to compress network traffic without unacceptably impacting the ability of the NIDS to detect and analyze malicious activity. Based upon Smith and Hammell's findings that malicious network flows will manifest their maliciousness early (2019), this research adds the ability to

determine if a flow is malicious by collecting malicious N-grams and hashing them into a Bloom filter. Packets are then broken into N-grams, which are hashed and compared to the Bloom filter. If the hash of an N-gram matches one in the malicious Bloom filter, the flow is tainted and all packets from that flow will be transmitted to the CAS. If no N-grams match, transmission of the flow to the CAS will stop after a user-defined threshold.

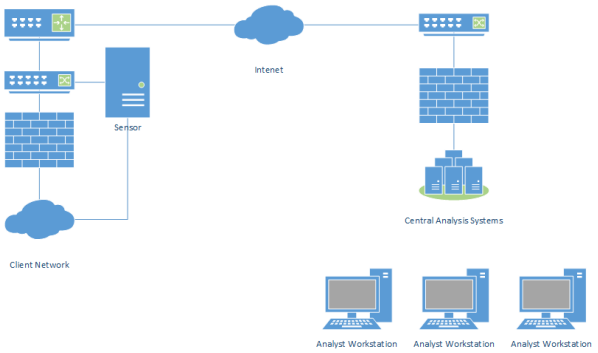

Figure 1 Distributed network intrusion detection

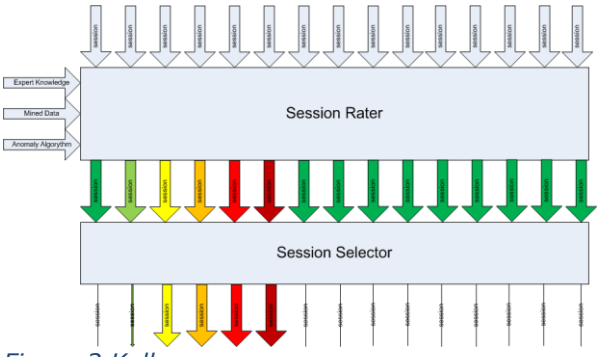

Figure 2 Kelly compressor

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. (Kemmerer & Vigna, 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 data set 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).

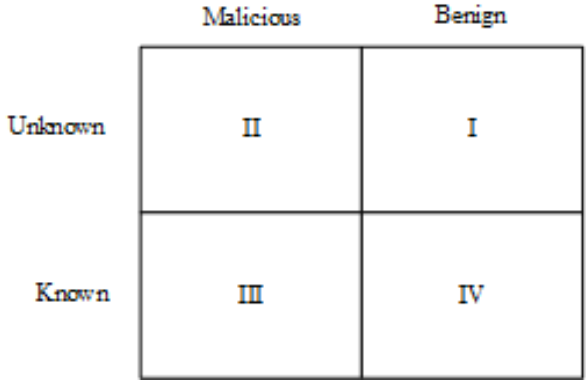

Figure 3 Network traffic composition

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).

Smith and Hammell, in their work on compressing network traffic based upon flow features (2019), discovered that malicious flows manifested themselves early; however, some malicious flows remained malicious deep into the flow. In this work the flow features are combined with a network intrusion detection algorithm to taint flows. This network intrusion detection algorithm needs to be very light-weight. Anagram (Wang, Parekh, & Stolfo, 2006) used N-grams and Bloom filters (Bloom, 1970) to differentiate benign and malicious traffic. Its approach was to load one Bloom filter with hashes of N-grams of known benign network traffic payloads and another Bloom filter with hashes of N-grams of known malicious network traffic payloads. ELIDe used N-grams but a different hashing method for detecting malicious traffic (Chang, Harang, & Payer, 2013). FAST-D (Yu & Leslie, 2017) inspired by Anagram and ELIDe also makes use of N-grams and Bloom filters.

3. APPROACH

This approach to the problem builds on Smith and Hammell's work (2019) with flow features by using N-grams and a Bloom filter to discover if a flow is malicious inside a user-defined threshold. Six data sets were selected to test the technique. Prototype tools were written to create the Bloom filters, compress the traffic, and control the experiments.

Data Sets

These data sets were chosen because they provide different types of data sets spanning from 1998 to 2017. The data sets provide a mix of synthetic and competition with 4 being synthetic and 2 competition.

DARPA Data Sets

As part of their evaluation of intrusion detection systems conducted under a grant from the Defense Advanced Research Projects Agency (DARPA), Lippman *et al.* created a data set of synthetic network traffic (2000). The 2 weeks of tcpdump testing data from 1998 have been combined into one file for this research. The DARPA data set from 1999 contains "inside" and "outside" network capture files. The 2 weeks for tcpdump testing data from the "outside" have been combined into one file for this research.

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 their efforts to generate labeled data sets (2009). Two network traffic sensors were employed in the exercise: gator-usama010 and gator-usama020. The raw network data captured from both sensors has been condensed into 2 files for use in this research.

University of New Brunswick

In 2012, the Information Security Centre of Excellence (ISCX) at the University of New Brunswick created a data set for intrusion detection research (Intrusion detection evaluation dataset (ISCXIDS2012), 2012) (Shiravi, Shiravi, Tavallaee, & Ghorbani, 2012). This synthetic labeled data set contains full network capture files. In 2017, the Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick created another data set for intrusion detection research (Intrusion detection evaluation dataset (CICIDS2017), 2017). The ISCX 2012 data set consists of 7 days of traffic. The first day is reported to contain no malicious activity; therefore, the network capture data from the remaining 6 days was concatenated into one file for this research. The CIC 2017 data set consists of 5 days of traffic. The first day is free from malicious traffic and designed to be used for training. The remaining 4 days of network traffic was concatenated into one file for this research.

Rule Sets

The initial rule set was the registered rule set downloaded from www.snort.org in August 2013. This rule set was effective with the traffic from the CDX 2009 data set, but worked poorly with the DARPA 1998 data set. The more rules that are tested by Snort, the more computing resources Snort requires to complete the analysis. These resources may climb to the point where Snort is unable to keep up with the network traffic causing packet loss. Therefore, the registered rule set from 2013 has most of the rules commented out. In order to analyze older data sets, it was necessary to tailor the rule set to ensure that rules appropriate to the time period are active. Four rule sets were used in this research: Circa2000 is the registered rule set from 2013

with rules appropriate for 2000 activated, Circa2009 is the registered rule set from 2013 with rules appropriate for 2009 activated, RegAug2013 is the registered rules as downloaded from the Snort website in August 2013, and RegSep2018 is the registered rule set as downloaded from the Snort web-site in September 2018.

Software
This research included the development of several pieces of prototype software. These include tools to create and manipulate files containing Bloom filters, software to collect network traffic, track flows, and test packets against the Bloom filter, and software to control the experiments.

Bloom Filters
Bloom filters are stored in binary files for speed of ingest and reduction in size. The Bloom filter files are stored in network byte order to facilitate use of different architectures. Each file contains a magic number with a version number, the size of the Bloom filter, the number of unique elements in the Bloom filter, the size of the N-grams, length of the Bloom filter name, the name of the Bloom filter, the number of hashes, hash type and seed pairs, and the filter itself. The program `mkbflt` is used to create Bloom filters, the program `dispbflt` is used to display the contents of a Bloom filter, and the program `bfltmerge` is used to merge 2 or more Bloom filters.

PCAP File Manipulators
The `pcappurge` program was written to remove malicious traffic from a file in PCAP (Jacobson, Leres, & McCanne, 2015) format based upon the alerts found by Snort (Roesch, 1999). The `pappurge` program is sensitive to time warps where a packet is read that is younger than packets previously processed. The programs `pcaptsplit` and `pcaptmerge` were developed to address this problem. The `pcaptsplit` program takes a file in PCAP format and creates one or more files that do not contain any time warps. The `pcaptmerge` program takes the PCAP files created by `pcaptsplit` and merges them into a single PCAP file without any time warps.

Network Traffic Compressor
The `netcomp` program was written to collect traffic using Libpcap (Jacobson, Leres, & McCanne, 2015) and compress that traffic using lossy compression based upon tainted flows. The `netcomp` program tracks flows terminating transmission after a user-provided threshold similar to the tool developed by Smith and Hammell (2019). In addition, it decomposes the payload into N-grams, hashes these N-grams, and compares the hash against a malicious Bloom filter. Flows that contain matching N-grams are tainted and traffic associated with tainted flows will continue to be transmitted to the CAS.

Experimental Control
The data set database was developed to support this research. The data set database contains information about the data sets, rule sets, and the Snort alerts discovered when data sets are analyzed by rule sets. It consists of 3 tables implemented as colon-separated text files. This choice was made because many of these experiments will be conducted on hardware that simulates a network intrusion detection sensor, which would not support a relational database system. The data is accessed through the `dsdb` command.

The experiments involved running the `netcomp` command at different thresholds in order to plot the alert loss rate (ALR) and the compression. The `netcompit` shell script that controlled the experiments employed the data set database to obtain the required information about each data set and rule set. It also employed the Sequence program (Smith & Hammell II, 2018) to set the thresholds.

Preparation
Before the experiments can begin it is necessary to prepare a malicious Bloom filter for each data set. The `pcappurge`, which is used to remove malicious traffic based upon Snort alerts, is very sensitive to time warps. A time warp occurs when packets are out of order. More specifically, it occurs when the date on the current packet is further in the past than the date of the previously read packet. Time warps must first be removed from the data set using `pcaptsplit` and `pcaptmerge`. The unabridged data set is then analyzed with Snort applying the appropriate rule set. Once there is a PCAP file for the data set free of time warps and a Snort alert file, the `pcappurge` program can split the file into a malicious PCAP file and a benign PCAP file. First the `mkbflt` program uses the benign PCAP file to create a Bloom filter of benign N-grams. The `mkbflt` program then uses the malicious PCAP file and the benign Bloom filter to create a Bloom filter of malicious N-grams. Malicious flows contain a considerable amount of benign N-grams. If these were not filtered, the false positive rate would be unacceptable.

4. RESULTS

In the following tests the ALR is plotted in circles and the compression, expressed as a percentage of the original size of the data set, is plotted in triangles. The threshold has been normalized using (1) where t is the threshold, p is the current packet count, and m is a configurable maximum packet count. Using this formula it is possible for the computed threshold to be negative if the packet count exceeds the maximum packet count. This also explains why at a threshold of zero the compression is not always 100%. In each of these tests m was set to 200.

$$t = 1 - \frac{p}{m}. \quad (1)$$

DARPA 1998 Testing
Figure 4 shows the effect of compressing the DARPA 1998 testing data set using the tainted flows technique and analyzing that data with Snort using the Circa2000 rule set. When the threshold is set to 0.75, the data set is compressed to 77.00% of its original size, and 0.00% of the alerts have been lost. When the threshold is set to 0.95, the data set is compressed to 66.00% of the original size and 0.27% of the alerts have been lost.

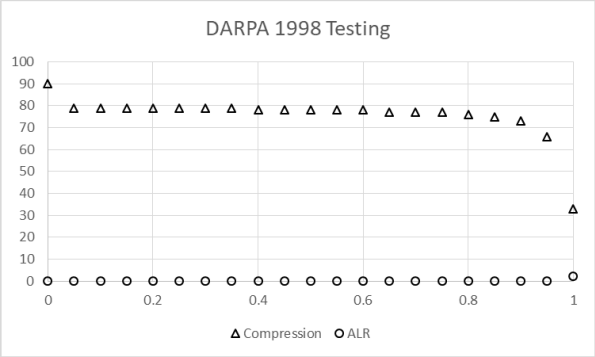

Figure 4 Tainted flow-based compression using the DARPA 1998 testing data set and the Circa2000 rule set

DARPA 1999 Testing Outside
Figure 5 shows the effect of compressing the DARPA 1999 testing outside data set using the tainted flows technique and analyzing that data with Snort using the Circa2000 rule set. When the threshold is set to 0.00, the data set is compressed to 78.00% of its original size, and 0.01% of the alerts have been lost. When the threshold is set to 0.96, the data set is compressed to 33.00% of the original size and 0.87% of the alerts have been lost.

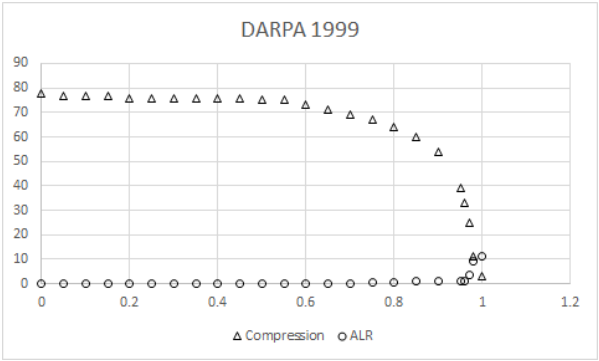

Figure 5 Tainted flow-based compression using the DARPA 1999 outside testing data set and the Circa2000 rule set

CDX 2009 gator-usama010
Figure 6 shows the effect of compressing the CDX 2009 gator-usama010 data set using the tainted flows technique and analyzing that data with Snort using the Circa2009 rule set. When the threshold is set to 0.50, the data set is compressed to 84.00% of its original size, and 0.00% of the alerts have been lost. When the threshold is set to 0.96, the data set is compressed to 61.00% of the original size and 0.58% of the alerts have been lost.

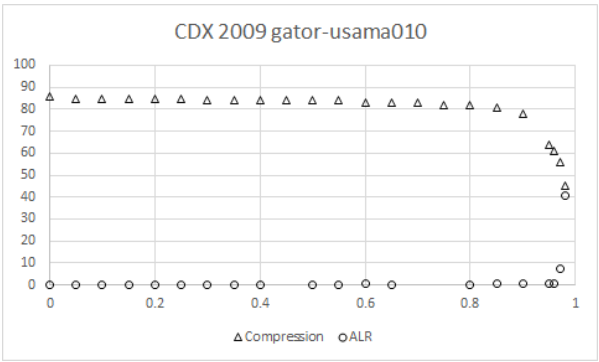

Figure 6 Tainted flow-based compression using the CDX 2009 gator-usama010 data set and the Circa2009 rule set

CDX 2009 gator-usama020
Figure 7 shows the effect of compressing the CDX 2009 gator-usama020 data set using the tainted flows technique and analyzing that data with Snort using the Circa2009 rule set. When the threshold is set to 0.00, the data set is compressed to 30.00% of its original size, and 0.00% of the alerts have been lost. When the threshold is set to 0.95, the data set is compressed to 23.00% of the original size and 0.57% of the alerts have been lost.

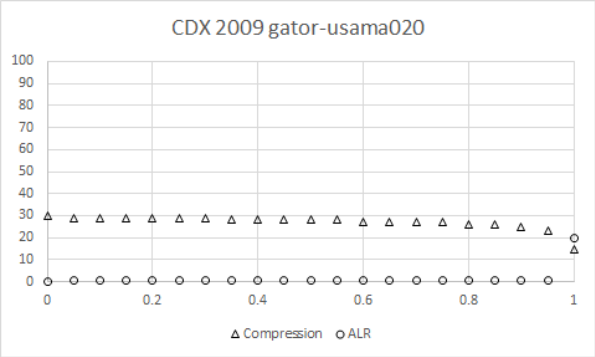

Figure 7 Tainted flow-based compression using the CDX 2009 gator-usama020 data set and the Circa2009 rule set

ISCX 15 Jun 2012

Figure 8 shows the effect of compressing the ISCX 15 Jun 2012 data set using the tainted flows technique and analyzing that data with Snort using the RegAug2013 rule set. When the threshold is set to 0.00, the data set is compressed to 88.00% of its original size, and 0.04% of the alerts have been lost. When the threshold is set to 0.99, the data set is compressed to 31.00% of the original size and 0.89% of the alerts have been lost.

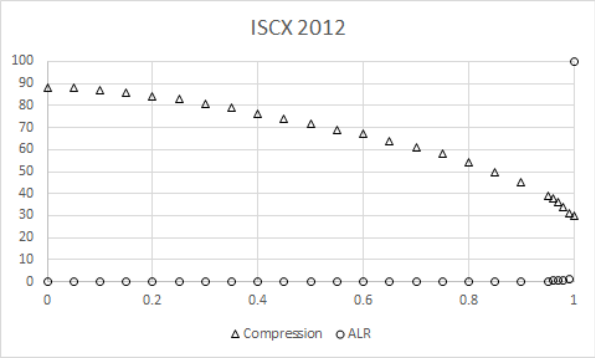

Figure 8 Tainted flow-based compression using the ISCX 15 Jun 2012 data set and the RegAug2013 rule set

CIC 5 Jul 2017

Figure 9 shows the effect of compressing the CIC 5 Jul 2017 data set using the tainted flows technique and analyzing that data with Snort using the RegSep2018 rule set. When the threshold is set to 0.80, the data set is compressed to 36.00% of its original size, and 0.00% of the alerts have been lost. When the threshold is set to 0.90, the data set is compressed to 34.00% of the original size and 0.94% of the alerts have been lost.

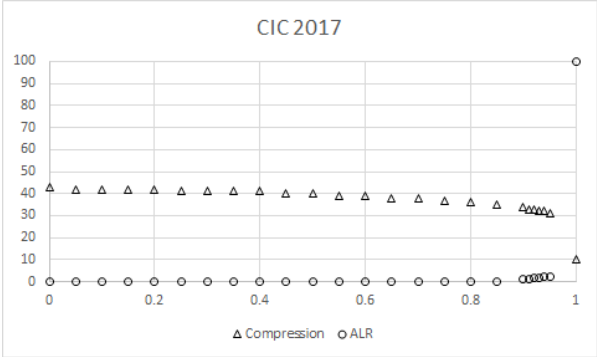

Figure 9 Tainted flow-based compression using the CIC 5 Jul 2017 data set and the RegSep2018 rule set

Summary

Figure 9 summarizes the results listing the data set, threshold, ALR, and size. Assuming that less than 1% ALR is acceptable, this table shows that for 4 out of 6 of the data sets the tainted flow compression technique was able to reduce the size of the data set to close to 30% or less. Smith et al. discovered that this level of compression was enough to allow standard lossless compression techniques to bring the data to within 10% of the original size (2017).

The DARPA 1998 testing data set includes a large amount of TELNET traffic. A large number of the alerts contained in that data set were TELNET failed login attempts. TELNET traffic tends to consist of a very large number of very small packets. The thresholds range from 0.90 to 0.96 percent. This equates to packet counts from 20 to 8. It is clear 8 packets would seldom be enough to capture a TELNET failed login. Further, the TELNET has fallen into disuse. The preponderance of Transmission Control Protocol alerts in the CDX 2009 gator-usama010 data set are from the Hyper Text Transport Protocol. A threshold of 0.96 means that we are stopping transmission of untainted flows only 8 packets into each flow. Considering that this is a competition data set, it is likely that there is simply that much traffic associated with each tainted flow.

Table 1 thresholds, ALR, and size for each data set with ALR less than 1%

Data set	Threshold	ALR	Size
D98TE	0.95	0.27%	66.00%
D99TEO	0.96	0.87%	33.00%
CDX09u010	0.96	0.58%	61.00%
CDX09u020	0.95	0.57%	23.00%
ISCX 2012	0.99	0.89%	31.00%
CIC 2017	0.90	0.94%	34.00%

5. CONCLUSIONS

The tainted flow lossy network compression technique was able to compress 4 out of 6 of the tested data sets to about 30% of the original size. This level of compression is good enough to allow standard lossless technique to complete compressing to less than 10%. There are reasonable explanations as to why the other 2 data sets did not compress well. This research demonstrates that the tainted flow technique is a promising approach for compressing network traffic in distributed network intrusion detection applications.

There are 3 major directions in future work. One is to optimize the code to run as efficiently as possible. Another is to test the thresholds discovered here on different data sets. Lastly, the program should be tested in a realistic environment.

The prototype is written in C++. The flow tracking code relies on the standard map data structure, which should execute in $0log_2(n)$ time. The N-gram matching relies on a Bloom filter, which is highly efficient. Since the program must run as efficiently as possible to prevent over-burdening the sensor and introducing packet loss, it would benefit from further optimization.

It is one thing to adjust the thresholds until the optimal threshold is discovered for a particular data set; however, this will not be possible in the real world. In the future, these findings may be tested by selecting a threshold before the data is compressed and analyzed.

All of these tests were conducted reading a PCAP file that had already been captured. In the future this prototype should be installed on hardware designed to simulate a sensor and traffic should be replayed on a network interface to be captured, compressed, and analyzed. This would allow the prototype's performance to be compared to other tools like Snort in a relevant environment.

6. REFERENCES

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 422-426.

Chang, R. J., Harang, R. E., & Payer, G. S. (2013). *Extremely lightweight intrusion detection (ELIDe) (No. ARL-CR-0730).* Adelphi, MD: US Army Research Laboratory.

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

Intrusion detection evaluation dataset (CICIDS2017). (2017). Retrieved September 2018, from University of New Brunswick: http://www.unb.ca/cic/datasets/ids-2017.html

Intrusion detection evaluation dataset (ISCXIDS2012). (2012). Retrieved September 2018, from University of New Brunswick: http://www.unb.ca/cic/datasets/ids.html

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

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

Kemmerer, R. A., & Vigna, G. (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.

Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security, 31*(3), 357-374.

Smith, S. C., & Hammell II, R. J. (2018). *Controlling Experiments Using Mathematical Sequences.* US Army Research Laboratory. Aberdeen Proving Ground, MD: US Army Research Laboratory.

Smith, S. C., & Hammell II, R. J. (2019). The use of Flow Features in Lossy Network Traffic Compression for Network Intrusion Detection Applications. *The 10th International Multi-Conference on Complexity, Informatics and Cybernetics: IMCIC 2019* (pp. 181-186). Orlando, FL: International Institute of Informatics and Systemics.

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. (2016a). 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. (2016b). 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.

Wang, K., Parekh, J., & Stolfo, S. (2006). Anagram: A content anomaly detector resistant to mimicry attack. *International Workshop on Recent Advances in Intrusion Detection* (pp. 226-248). Hamburg, Germany: Springer.

Yu, K., & Leslie, N. O. (2017). FAST-D: malware and intrusion detection for mobile ad hoc networks (MANETs). *NATO Specialist Meeting IST-145 on Predictive Analytics and Analysis in the Cyber Domain* (pp. 10-11). Sibiu, Romania: NATO IST.