A Theoretical Exploration of the Impact of Packet Loss on Network Intrusion Detection

Sidney C. Smith
Computational Information Sciences Directorate, ARL
APG, MD, USA
E-mail: sidney.c.smith24.civ@mail.mil
www.arl.army.mil

Robert J. Hammell II
Department of Computer and Information Sciences, Towson University
Towson, MD, USA
E-mail: rhammell@towson.edu
www.towson.edu

Travis W. Parker
ICF International
APG, MD, USA
E-mail: travis.w.parker16.ctr@mail.mil
www.icfi.com

Lisa M. Marvel
Computational Information Sciences Directorate, ARL
APG, MD, USA
E-mail: lisa.m.marvel.civ@mail.mil
www.arl.army.mil

Abstract

In this paper we review the problem of packet loss as it pertains to Network Intrusion Detection, seeking to answer two fundamental research questions which are stepping stones towards building a model that can be used to predict the rate of alert loss based upon the rate of packet loss. The first question deals with how the packet loss rate affects the sensor alert rate, and the second considers how the network traffic composition affects the results of the first question. Potential places where packet loss may occur are examined by dividing the problem into network, host, and sensor based packet loss. We posit theories about how packet loss may present itself and develop the Packet Dropper that induces packet loss into a dataset. Drop rates ranging from 0% to 100% are applied to four different datasets and the resulting abridged datasets are analyzed with Snort to collect alert loss rate. Conclusions are drawn about the importance of the distribution of packet loss and the effect of the network traffic composition.

Keywords: Network Intrusion Detection Packet Loss

1. Introduction

Network Intrusion Detection (NID) depends upon the sensor being able to see the traffic between the adversary and the target. This is often done in the network boundary by placing the sensor on a mirrored port configured to send the sensor a copy of all the packets flowing into and out of the network. Packet loss occurs when some of these packets fail to reach the sensor software for analysis. Intuitively, we understand that a sensor cannot detect what it cannot see; therefore, this packet loss has a negative impact on the sensor's ability to detect malicious activity. This fundamental truth is well known, and much work has been done to reduce or eliminate packet loss. Very little work has been done to understand, predict, and model packet loss or the impact on network intrusion detection. The focus of this research is to answer two fundamental research questions that are key stepping stones to building a model that can be used to predict packet loss and the impact on the performance of the network intrusion detection system.

Would the same rate of packet loss induce the same loss of sensor alerts regardless of the algorithm used to induce the packet loss? (e.g. would a random packet loss induction of 5% cause the same loss in sensor alerts no matter what seed was chosen, or would a sine wave based 5% packet loss strategy cause the same loss in sensor alerts as a 5% deterministic distribution strategy?) If the alert loss is highly dependent upon the packet loss strategy, this implies that the correctness of the packet loss model is very important. Conversely, if the alert loss is independent of the packet loss strategy, then the correctness of the packet loss model is not very important.

Are the results independent of the composition of the network traffic? (e.g. if packet loss were applied to capture the flag competition datasets that are exploit rich, would the same relationship hold; if we applied packet loss to a much older dataset, would we see the same relationship implying that our results are likely to hold over time?) Because the composition of network traffic varies significantly from site to site, if the results are highly dependent upon the composition of the network traffic, then we will be unable to generalize our results.

The remainder of this paper is organized into the following sections. Section two provides an overview of existing literature as it pertains to network, host and sensor based packet loss, reviews our understanding of packet loss, and we posit our theories as to how that packet loss may manifest itself. Section three describes the properties of the Packet Dropper application and describes the datasets that we used in this study. Section four discusses the results of applying the Snort NIDs tool with datasets abridged with the Packet Dropper. Section five summarizes our findings and describes future work.

2. Background

The focus of this research is not packet loss in general, but specifically any packet loss where packets from the adversary reach the target, but do not reach the NID sensor. We will divide this problem into three distinct areas: network, host and sensor packet loss which are illustrated in Figure 1. Previous research in this area has focused on eliminating packet loss. We will divide the literature along the same lines that we have divided our research. One must keep in mind however, that this categorization is of our own making and previous research may not fit well into our categories.

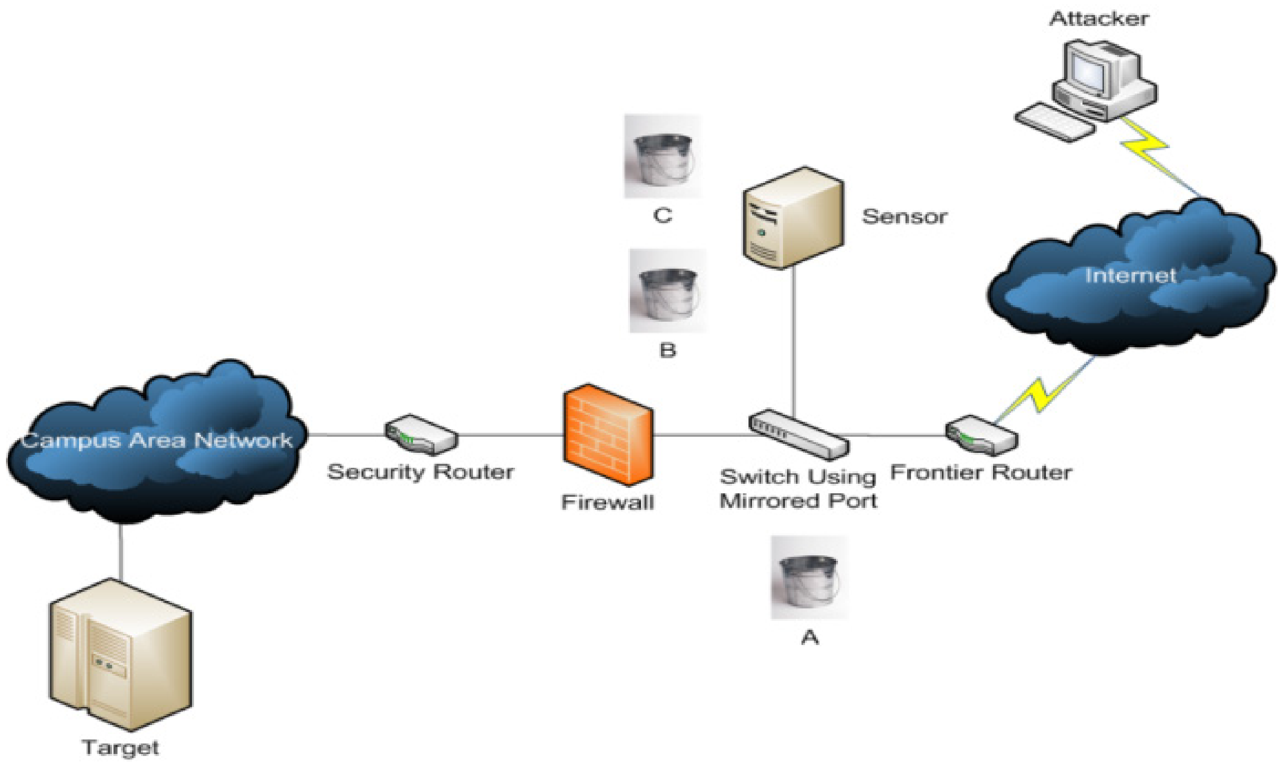

Fig. 1 Packet Loss Diagram

2.1. *Network*

Network based packet loss is defined as any circumstance that would prevent packets which reach the target from reaching the network interface of the sensor and is represented by bit bucket A in Figure 1. There is a significant amount of benign packet loss on the Internet, [1] but we are only interested in those packets that reach the target but fail to reach the sensor. Many sensors are connected to the network through a mirrored port on a switch. Since mirroring is the lowest priority task that switches perform, it is possible to create a situation where the malicious traffic would reach the target but fail to reach the sensor.[2]

Revisiting Figure 1 bit bucket A, we see a network switch with three network connections: the frontier router, the firewall and the NID sensor. Given that the routing and firewall functions are significantly more complicated than the switching function, one would expect the switch to easily keep pace with these two devices. Unlike network hubs which are half duplex, network switches are full duplex allowing traffic to flow between the firewall and the frontier router through the switch in both directions at the same time; however, there is still only one traffic path from the switch to the sensor. When packets arriving at the switch from the firewall and frontier router overlap, one must be buffered until the other has completed transmission to the sensor. In the event of heavy traffic in both directions, one could imagine this buffer filling and packets being dropped. Since TCP's congestion control algorithm tends to make network traffic bursty, we theorize that we can model this kind of packet loss using a Markov chain two state channel similar to the one that has been used since 1960 to simulate burst-noise.

2.2. *Host*

Host based packet loss is defined as any circumstance that would prevent packets which reach the network interface of the sensor from being presented to the analysis software and is represented by bit bucket B in Figure 1. The movement of packets from the Network Interface Card (NIC) through the kernel to user space where most sensor software is executed, with multiple potential points of failure, is illustrated in Figure 2.[3] Handling interrupts is significantly more costly than processing user or kernel code because interrupt processing cannot benefit from advances in processor performance. Interrupt handling may constitute 15% of the total processing time. Interrupt cost may be reduced 60% and packet loss rates by 46% by aggregating 32 interrupts.[4] Packet loss was greatly reduced by increasing the level-2 cache.[4] The Multi-Parallel

Intrusion Detection Architecture was able to achieve impressive capture rates on a 10GbE network by parallelizing both the kernel and user land detection processes by assigning each flow to a single core.[5] Yueai and Junjie[6] employed multiple sensors with load balancing to address the problem of host based packet loss. Chung, *et al.* discusses host based packet loss as packets that cannot be copied from kernel to user space quickly enough to prevent loss. They explore moving the NID engine into kernel space as one solution to this problem.[7]

Revisiting Figure 1 bit bucket B and Figure 2, we see buffers in the kernel used to hold packets that have been received by the NIC and have yet to be processed by the CPU. One could easily envision a general purpose computer system being unable to keep up with the firewall and the frontier router, both dedicated network devices tuned specifically to move network traffic. We theorize that we can simulate this effect by implementing a capped algorithm which drops packets when a threshold is exceeded.

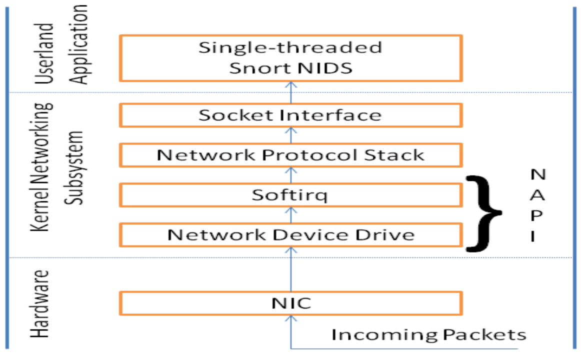

Fig. 2 Snort NIDS underlying kernel support architecture[3]

2.3. *Sensor*

Sensor based packet loss is defined as any circumstance that would prevent packets which are presented to the analysis software from being processed and is represented by bit bucket C in Figure 1. A prime candidate is resource exhaustion caused by the analysis software itself.

Some improvements in packet loss rates were obtained by increasing the level of optimization employed in the compiler.[4] Setting the netdev_budget, a kernel configuration parameter in the New Application Programming Interface (NAPI), to a low number like 2 has been shown to greatly decrease packet loss. This is primarily because both pulling packets from the network and analyzing the packets are bound by available CPU cycles and not buffer memory. By allocating the bulk of the CPU time to the sensor application, more packets may be processed and the packets that are dropped are dropped very early in the process where CPU cycles will not be wasted processing a packet which will only be dropped later.[3] Significant improvements in the packet loss rate could also be achieved by pruning the sensor rule set.[4] Multi-processing techniques have been employed to increase the throughput of the sensor and reduce packet loss.[8] In addition to this Song *et al.* point to the software crash as yet another cause of packet loss.[9] Wei *et al.* consider the problem of packet loss in an IPv6 environment. As a solution they propose breaking the detection task into three units: the Data Acquisition Unit, the Adapt Load Characteristic Analysis unit, and the Collaborative Analysis and Control Center.[10]

Revisiting Figure 1 bit bucket C, we expect that the resource consumption of the sensor application itself will contribute the greatest component of packet loss at this level. We theorize that we can simulate this effect through some cyclic function such as a sine wave which would model a process gradually gathering resources, performing its task, and gradually releasing the resources. Although, we believe these are good general assumptions, we expect that the exact function would be highly dependent upon the specific software in use and is therefore beyond the scope of this research.

2.4. *Combined Effect*

In their paper, "Characterizing Sources and Remedies for Packet Loss in Network Intrusion Detection Systems", Schaelicke and Freeland[4] observed a near linear relationship between Packet Loss Rate (PLR) and Alert Loss Rate (ALR). Grossly simplifying their results one could describe the relationship by the following equation:

$$ALR = PLR - 15 \tag{1}$$

Their network traffic, captured on the Internet connection of a major university, consisted of 13 seconds activity containing over 530,000 packets with 521 known attacks at a ratio of about 1000 packets per alert. Although sufficient for their purposes, there is insufficient robustness to extrapolate a general packet loss to alert loss relationship from their results.

The significant body of work addressing the reduction of packet loss demonstrates that it is generally considered a serious problem. As network bandwidth continues to outpace CPU clock speeds the problem will grow. Although much work has been done to minimize packet loss, very little work has been done to characterize packet loss and to quantify the impact of packet loss on NID. The purpose of our effort is to understand, predict and model both packet loss and its impact upon NID

3. Methodology

3.1. *Algorithms*

To model the effects of packet loss we developed the Packet Dropper application which reads files in PCAP format and writes to a file in PCAP format dropping packets according to a dropping algorithm and a set of configuration parameters. The implementation of five algorithms is presented here; these algorithms were chosen to illustrate our assumptions about the possible patterns of packet loss with the deterministic algorithm added as a control. Several other algorithms were developed during the research effort, but are not included here because their effects were virtually identical to the algorithms presented.

In all of the following equations *drop* is a Boolean value that is computed by the algorithm. The integer *count* is incremented every time a packet is read. The integer *randnum* is between 0 and 2,147,483,647 and generated using random(3) from the standard C library. The values of *interval* and *period* are provided by the user on the command line. The value of *dr* for drop rate is between 0 and 100 and provide by the user on the command line. For each packet read the value of *drop* is computed and if *drop* is *true* the packet is dropped else the packet is written.

3.1.1. The Deterministic Algorithm

First we implemented a very simple algorithm that evenly distributes packet loss across the dataset. We did this as a control to help discover how dependent alert rate loss is upon the manner in which the packets are dropped.

The deterministic algorithm drops packets evenly through the stream at a given rate expressed as a percentage. If the drop rate is set at 1%, Packet Dropper will drop every 100th packet. If the drop rate is set at 2%, it will drop every 50th packet. If the drop rate is set at 3%, it will drop every 33rd packet. This is further illustrated in equation number 2. Code in the algorithm handles the special case were the *droprate* is zero.

$$drop = \begin{cases} true,\ count \,\%\, \frac{100}{dr} = 0 \\ false,\ count \,\%\, \frac{100}{dr}\,! = 0 \end{cases} \qquad (2)$$

3.1.2. The Random Algorithm

As another control, we implemented the random algorithm which generates a random number for each packet read. We use modulo arithmetic to capture the least two significant digits of the random number. If this is lower than the drop rate we drop the packet, otherwise we keep the packet. This is further illustrated in equation number 3.

$$drop = \begin{cases} true,\ dr < randnum \,\%\, 100 \\ false,\ dr \geq randnum \,\%\, 100 \end{cases} \qquad (3)$$

3.1.3. The Sinusoidal Algorithm

To model sensor based packet loss using the sinusoidal algorithm we will divide the period into fixed blocks using the interval. For each interval we will compute the effective drop rate y by computing the sine at the center of the interval. The sine function has a natural period of 0 to 2π in radians; therefore, we will convert this natural period into the period we need by dividing our current packet count by the period and multiply it by the end of the period for:

$$y = A\sin\left(\frac{count+\frac{interval}{2}}{period}2\pi\right) \qquad (4)$$

Now the natural range of the sine function is from -1 to 1, but we need a range from 0 to 1 with a center at the drop rate. If we set the amplitude of the sine wave to the drop rate then add the adjusted sin value to the drop rate we get a function that moves from 0 to 2 times the drop rate centered around the drop rate. This is perfect for drop rates less than 50%. For drop rates greater than 50% the upper bound is outside of our range. To account for this, if the drop rate is greater than 50%, we use amplitude of 1 minus the drop rate. This is further illustrated in equations 5, 6.

$$A = \begin{cases} dr,\ x < 50 \\ 100 - dr,\ x \geq 50 \end{cases} \qquad (5)$$

$$drop = \begin{cases} true, & dr + y \geq randnum \,\%\, 100 \\ false, & dr + y < randnum \,\%\, 100 \end{cases} \qquad (6)$$

3.1.4. A Capped Algorithm

To model host based packet loss, we implemented the capped algorithm, which uses a first in first out (FIFO) data structure. When the algorithm is passed a new packet, the FIFO will be drained of all packets received in the prior interval; appropriately decrementing the packet and bit counts. Then the packet count of the current packet is compared against the cap, and if it is above the cap the packet is dropped.

We also made a modification to this algorithm to cap by bit count instead of simply using the packet count. If the NID process is CPU bound then the former algorithm tracking the packet count will provide the better simulation. If the process is buffer bound, then the second algorithm tracking the bit count will provide the better simulation.

3.1.5. A Two State Channel Model

To model network based packet loss, we implement an algorithm based upon a Markov chain with states used to generate bursts as described by Gilbert[11] and illustrated in Figure 3 where state *G* is a good state where no dropping take place and state *B* is a bad state where packet dropping takes place.

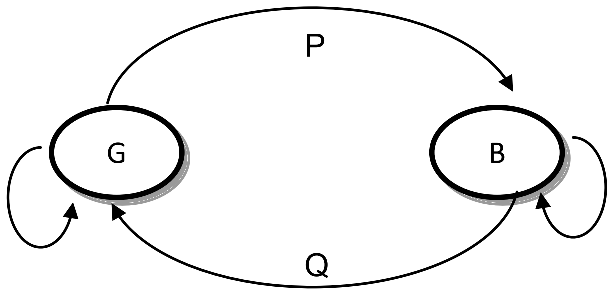

Fig. 3 Two State Channel Model

Gilbert[11] provides us with the following formula to calculate the error probability of the previous model letting d be the drop rate and P=P(Bad state|Good State) and Q = P(Good state|Bad state). When *h* represents the probability that the a packet will transmitted in state *B* we have:

$$d = \frac{(1-h)P}{Q+P} \qquad (7)$$

Solving for P we get:

$$P = \frac{dQ}{1-h-d} \qquad (8)$$

To allow for the full domain yet ensure that the value of P stays within the range for 0 to 1 we need to set h = 0, and Q = 0.05; The value of P is undefined when d = 1; therefore, we will make provisions for this condition.

3.1.6. Validation

To ensure that the effects of these different algorithms on network traffic were dissimilar enough to allow us to answer our research questions, they were evaluated by graphing the packets/second of about two and a half minutes of network traffic from the Cyber Defense Exercise in 2009.[12] This traffic was selected because it was consistent enough for the peaks and valleys not to obscure the results. Looking at Figure 4 the deterministic algorithm produces a line almost identical to the normal or unabridged line a little lower on the graph which is exactly the behavior that we would expect. The random algorithm provides a plot that looks very similar to the line we see using the deterministic algorithm. The line is so similar that it can be very hard to distinguish. This would indicate that although purely random dropping of packets will drop different packets, it will have a very similar impact to the network volume as the deterministic algorithm. The sine wave may be clearly seen in the plot of the sinusoidal algorithm. We can see the plateau in the traffic in the plotting of the capped algorithm. Although the difference between the two state and deterministic algorithms is not as dramatic as the sine or capped algorithms, it is quite different. The differences in the effects of each of these algorithms should be sufficient to allow us to answer our research questions.

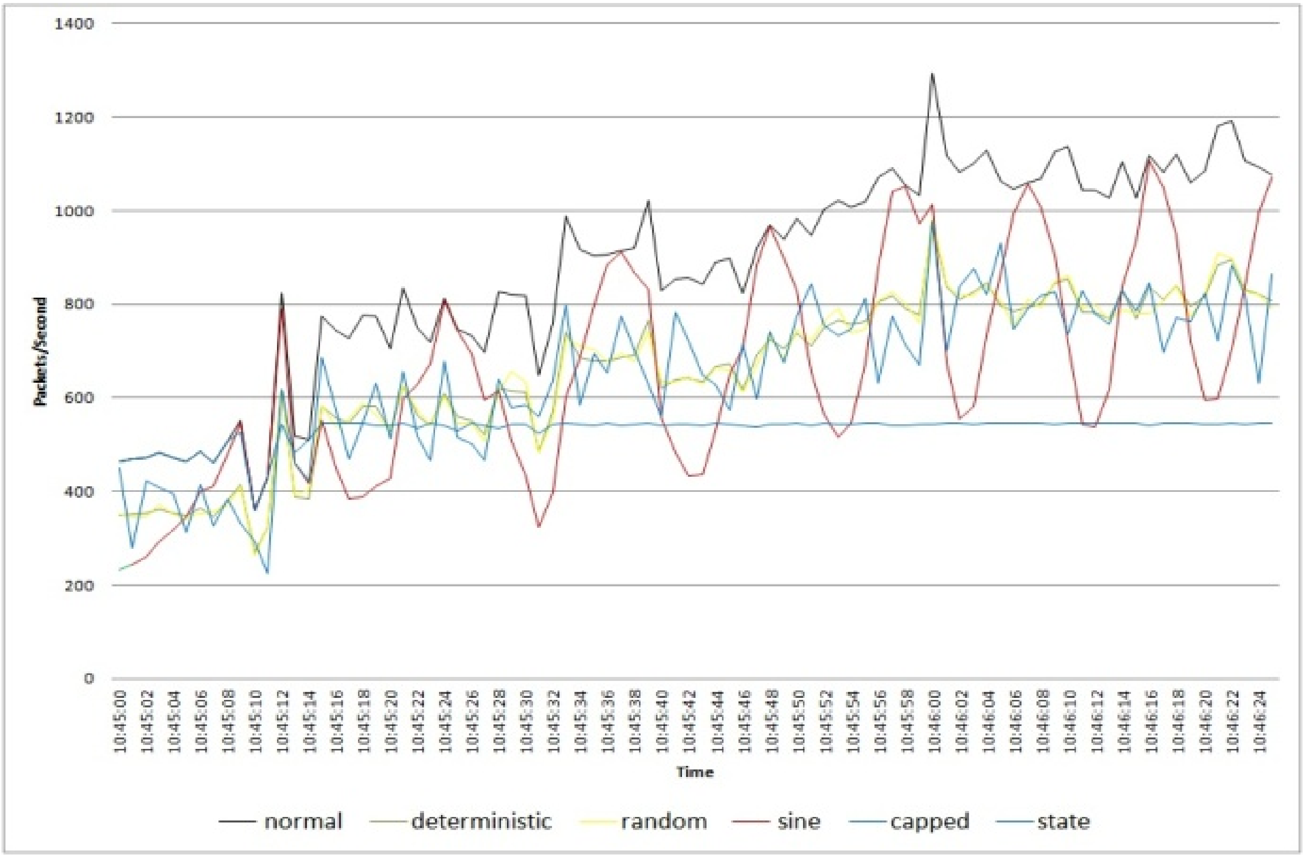

Fig. 4 Effects of 25% Packet Loss on Packets/Second with Various Algorithms

3.2. *Datasets*

3.2.1. DARPA 1998 Training Dataset

In 1995 Rome Laboratory and the Department of Defense Advanced Research Projects Agency (DARPA) contracted with Massachusetts Institute of Technology's Lincoln Laboratory to conduct a study on Evaluating Intrusion Detection Systems.[12] As part of their evaluation they created a training dataset that contained both malicious traffic and background traffic. Each session in the training data was labeled identifying whether it was part of an attack or not. The training dataset contains seven weeks of labeled data in PCAP format. The simulated network used to construct the 1998 training dataset was composed of UNIX workstations. Although this dataset is over twenty years old, it is still one of the best fabricated datasets available for research in intrusion detection. While the content of network traffic has changed significantly since 1998, we are including this dataset because it is widely used and accessible. This dataset contains 40,667,322 packets and about 16 Gigabytes of data. Snort detected 5,165 alerts for a ratio of about one alert for every 7900 packets.

3.2.2. DARPA 1999 Dataset

The next year Lincoln Labs released a dataset containing seven days of traffic from a simulated network with systems running Microsoft Windows operating systems.[13] This dataset contains 84,327,351 packets and about 18 Gigabytes of data. Snort detected 6,420 alerts for a ratio of about one alert for every 13,000 packets.

3.2.3. Cyber Defense Exercise 2009 (CDX 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 could best defend their network.[14] In their paper, "Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets" Sangster *et al.*[15] describe their efforts to collect and label traffic from this competition. We were able to obtain this data from https://www.itoc.usma.edu/research/dataset/. This dataset contains 47,511,801 packets and about 23 Gigabytes of data. Snort detected 2,900 alerts for a ratio of approximately one alert for every 16,000 packets.

3.2.4. Collegiate Cyber Defense Competition 2010 (CCDC 2010)

Paul Asadoorian describes his experiences as a Red Team captain in the Mid-Atlantic Regional Collegiate Cyber Defense Competition which pitted blue teams representing Universities against red teams composed of security experts.[16] We were able to obtain the packet capture data from CCDC 2010. This dataset contains 264,973,151 packets and about 32 Gigabytes of data. Snort detected 84,913 alerts for a ratio of about one alert for every 3,120 packets.

4. Results

Each dataset was abridged dropping packets from 0% to 100% at 5% intervals using each of the dropping algorithms. The abridged datasets were analyzed the with Snort version 2.6.8. Figure 5 depicts the packet loss rate as the x axis against the alert loss rate as the y axis. The alert loss rate was plotted to normalize the data allowing us to compare the different datasets. Thus we have the results from six dropping algorithms and four datasets (480 data points) plotted on the graph.

For the DARPA datasets the deterministic, random, sine and state algorithms produce a very linear relationship right along the middle line with a slight amount of variance. For the CDX 2009 dataset this relation begins to curve slightly above the line and the variance is increased. The CDX dataset is the smallest of the four, and we have seen similar variance in other small datasets. For the CCDC 2010 dataset the deterministic and random algorithms produce a relationship that is more curved above the middle but with very little variation. The sine and state algorithms produce a similar curve, but show significant deviation from the deterministic relation and from each other.

We did see a clear differentiation between algorithms that independent and dependent of traffic volume. Those algorithms that are independent of the traffic volume (e.g. deterministic, random, sine and state) all produced similar results that were very near linear for the older datasets and slightly curved above for the newer datasets. Looking at the results of the deterministic and the random algorithm, we find that they are almost identical; therefore, we can conclude that changing the seed of the random algorithm will not significantly change the results. The results abridging the algorithms with the deterministic algorithm and the sine algorithm are also similar enough for to conclude that there is not a significant difference. The algorithms that depended upon the traffic volume (e.g. capped-pkt and capped-bit) consistently produced relationships that are lower compared to the independent algorithms in almost a $y = x^2$ relationship. The exception to this is the CCDC 2010 dataset where the capped by bit algorithm produced a curve looking more like $y = \sqrt{x}$. There is sufficient consistent difference between the results for abridging data with that dependent algorithms to conclude that the way the data is abridged is significant.

For the independent algorithms the graphs for the DARPA datasets are very nearly linear; whereas, a curve may be seen for the CDX 2009 and CCDC 2010 datasets. The results are not consistent enough across the datasets to allow us to conclude that the packet loss to alert loss relationship is independent of the composition of the network traffic.

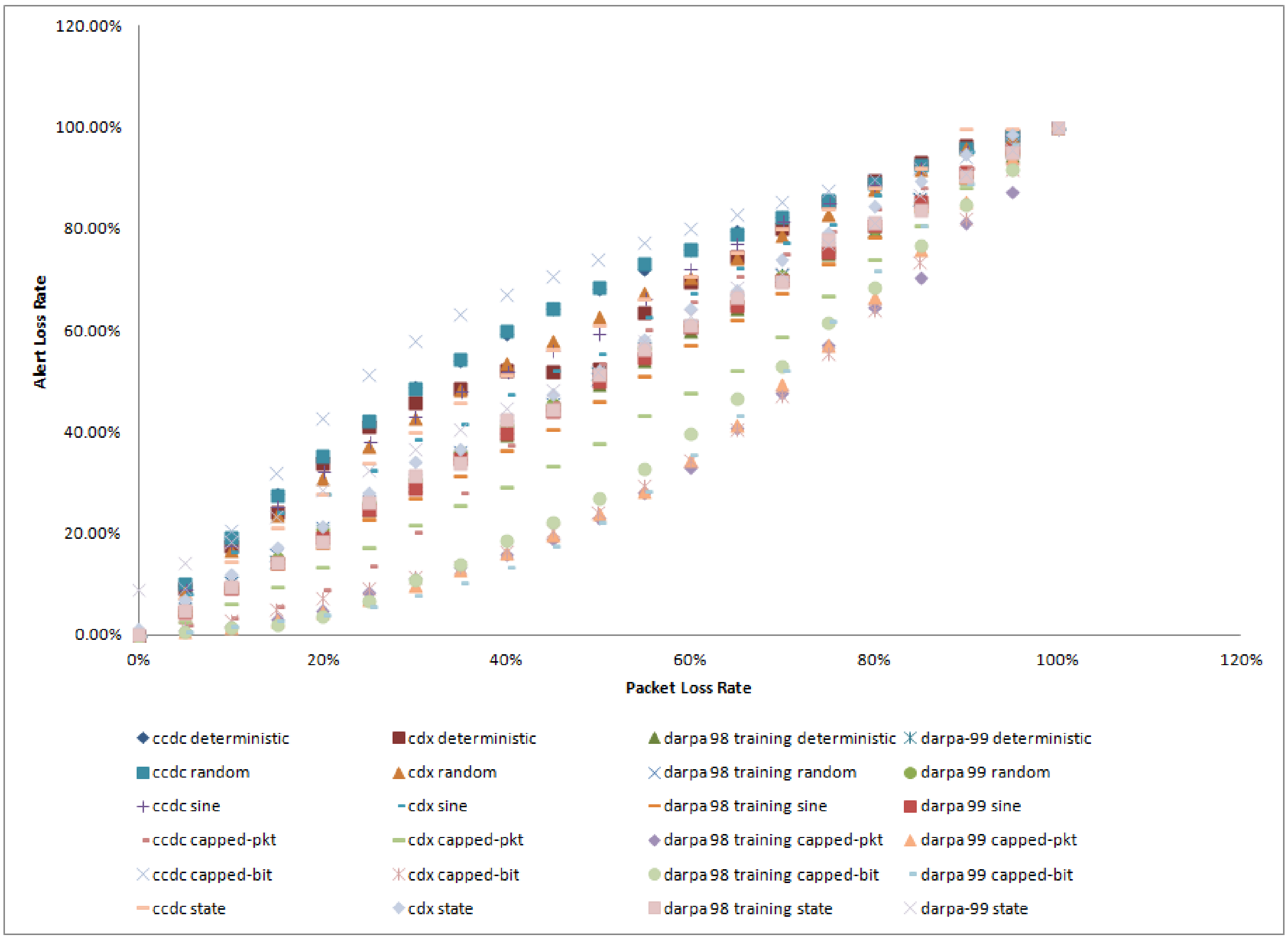

Fig. 5 Alert Loss Rate vs. Packet Loss Rate

5. Conclusion and Future Work

In this paper we reviewed the literature, examined the problem, and posited theories regarding packet loss as it pertains to network intrusion detection (NID). We developed the Packet Dropper application and used it to abridge four datasets, analyzed the results with Snort and plotted our findings. We were able to determine that although independent algorithms produced similar results, there is a significant difference between the results of the dependent algorithms allowing us to answer our first research question and conclude that the same rate of packet loss will not induce the same loss of sensor alerts regardless of the algorithm used to induce the packet loss. We were also able to determine that different network datasets from about the same time behave similarly; however, datasets from different times behave differently; therefore, we may answer our second research question concluding that the results are dependent on the composition of the network traffic.

We must limit our conclusions to addressing these two research questions because we have not validated any of our theories. This is the reason that we did not apply sophisticated regression analysis tools to our results. The contribution of this work is to lay the ground work for future efforts to build a model that can be used to predict the impact of packet loss on the performance of the network intrusion detection system.

We plan to experimentally study how packets are dropped in a laboratory environment and use this information to refine our theories and validate our algorithms. This additional data will allow us to improve both the Packet Dropper application and our predictive formulas. Once we have validated the theories and the algorithms, we should be able to repeat these tests against other datasets to generalize the work.

A closer look into the data may also be warranted. If the number of alerts is higher during times of low bandwidth, then we can expect the capped algorithms will behave very differently from algorithms that are bandwidth independent. In these experiments, the same Snort rules were used for all four datasets. If our experimental results indicate a bandwidth dependent algorithm, then tailoring the rules to the datasets may better reveal the natural distribution of alerts providing greater fidelity.

Once we have validated and generalized our theories, we may be able to discover, isolate, and model the currently unknown variables that are creating the variance between the behaviors of our algorithms against different datasets.

References

1. A. T. Mzrak, S. Savage and K. Marzullo, "Detecting Malicious Packet Losses," *Parallel and Distributed Systems, IEEE Transactions on,* pp. 191-206, 2009.
2. T. O'Neill, "SPAN Port or TAP? CSO Beware," 23 August 2007. [Online]. Available: http://www.lovemytool.com/blog/2007/08/span-ports-or-t.html. [Accessed 21 February 2012].
3. K. Salah and A. Kahtani, "Improving Snort performance under Linux," *IET Communications,* pp. 1883-1895, 2009.
4. L. Schaelicke and J. C. Freeland, "Characterizing sources and remedies for packet loss in network intrusion detection systems," in *Workload Characterization Symposium, 2005. Proceedings of the IEEE International*, Austin, Texas, 2005.
5. G. Vasiliadis, M. Polychronakis and S. Ioannidis, "MIDeA: a multi-parallel intrusion detection architecture," in *Proceedings of the 18th ACM conference on Computer and communications security*, New York, 2011.
6. Z. Yueai and C. Junjie, "Application of Unbalanced Data Approach to Network Intrusion Detection," in *First International Workshop on Database Technology and Applications*, 2009.
7. B.-H. Chung, J.-N. Kim, S.-W. Sohn and C.-h. Park, " Kernel-level intrusion detection system for minimum packet loss," in *Advanced Communication Technology, 2004. The 6th International Conference on*, 2004.
8. N.-U. Kim, M.-W. Park, S.-H. Park, S.-M. Jung, J.-H. Eom and T.-M. Chung, "A study on effective hash-based load balancing scheme for parallel NIDS," in *Advanced Communication Technology (ICACT), 2011 13th International Conference on* , 2011.
9. B. Song, W. Yang, M. Chen, X. Zhao and J. Fan, "Achieving Flow-Level Controllability in Network Intrusion Detection System," in *SNPD '10 Proceedings of the 2010 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing* , Washington, DC, 2010.
10. C. Wei, Z. Fang, W. Li, X. Liu and H. Yang, "The IDS Model Adapt to Load Characteristic under," in *CORD Conference Proceedings*, 2008.
11. E. N. Gilbert, "Capacity of a Burst-Noise Channel," *The Bell System Technical Journal,* pp. 1253-1265, 1960.
12. P. Asadoorian, "The Mid-Atlantic Regional CCDC 2010 Event - Part I," 18 March 2010. [Online]. Available: http://www.tenable.com/blog/the-mid-atlantic-regional-ccdc-2010-event-part-i. [Accessed 22 March 2013].
13. R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham and M. Zissman, "Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation," in *DARPA Information Survivability Conference and Exposition*, 2000.
14. J. W. Haines, R. P. Lippman and R. K. Cunningham, "Extending the DARPA off-line intrusion detection evaluations," in *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01. Proceedings , vol.1*, 2001.
15. "West Point Takes the NSA Cyber Defense Trophy for the Third Straight Year," 28 April 2009. [Online]. Available: http://www.nsa.gov/public_info/press_room/2009/cyber_defense_trophy.shtml. [Accessed 22 March 2013].
16. B. Sangster, T. J. O'Connor, T. Cook, R. Fanelli, E. Dean, J. Adams, C. Morrell and G. Conti, "Toward Instrumenting Network Warfare Comptetions to Generate Labeled Datasets," in *USENIX Security's Workshop on Cyber Security Experimentation and Test (CST)*, 2009.