REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 07-04-2011	**2. REPORT TYPE** Technical Paper	**3. DATES COVERED** *(From - To)* MAR 2011 - APR 2011

4. TITLE AND SUBTITLE Cyber Situational Awareness through Operational Streaming Analysis	**5a. CONTRACT NUMBER** FA8720-05-C-0002
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) William W. Streilein, John Truelove, Chad R. Meiners, Gregory Eakman	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) MIT Lincoln Laboratory 244 Wood Street Lexington, MA 02420	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) NSA 9800 Savage Rd Ft. Meade, MD 20755	**10. SPONSOR/MONITOR'S ACRONYM(S)** NSA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

As the scope and scale of Internet traffic continue to increase the task of maintaining cyber situational awareness about this traffic becomes ever more difficult. There is strong need for real-time on-line algorithms that characterize high-speed / high-volume data to support relevant situational awareness. Recently, much work has been done to create and improve analysis algorithms that operate in a streaming fashion (minimal CPU and memory utilization) in order to calculate important summary statistics (moments) of this network data for the purpose of characterization. While the research literature contains improvements to streaming algorithms in terms of efficiency and accuracy (i.e. approximation with error bounds), the literature lacks research results that demonstrate streaming algorithms in operational situations. The focus of our work is the development of a live network situational awareness system that relies upon streaming algorithms for the determination of important stream characterizations and also for the detection of anomalous behavior. We present our system and discuss its applicability to situational awareness of high-speed networks. We present refinements and enhancements that we have made to a well-known streaming algorithm and improve its performance as applied within our system. We also present performance and detection results of the system when it is applied to a live high-speed mid-scale enterprise network.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF: U			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Zach Sweet
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	SAR	6	**19b. TELEPHONE NUMBER** *(include area code)* 781-981-5997

MS-55040

Cyber Situational Awareness through Operational Streaming Analysis[†]

William W. Streilein John Truelove Chad R. Meiners ~~Gregory Eakman~~
MIT Lincoln Laboratory
244 Wood Street, Lexington, MA 02420
{wws, jtruelove, chad.meiners, greg.eakman}@ll.mit.edu

***Abstract*—As the scope and scale of Internet traffic continue to increase the task of maintaining cyber situational awareness about this traffic becomes ever more difficult. There is strong need for real-time on-line algorithms that characterize high-speed / high-volume data to support relevant situational awareness. Recently, much work has been done to create and improve analysis algorithms that operate in a streaming fashion (minimal CPU and memory utilization) in order to calculate important summary statistics (moments) of this network data for the purpose of characterization. While the research literature contains improvements to streaming algorithms in terms of efficiency and accuracy (i.e. approximation with error bounds), the literature lacks research results that demonstrate streaming algorithms in operational situations.**

The focus of our work is the development of a live network situational awareness system that relies upon streaming algorithms for the determination of important stream characterizations and also for the detection of anomalous behavior. We present our system and discuss its applicability to situational awareness of high-speed networks. We present refinements and enhancements that we have made to a well-known streaming algorithm and improve its performance as applied within our system. We also present performance and detection results of the system when it is applied to a live high-speed mid-scale enterprise network.

I. Introduction

Situational awareness refers to the ability to observe, assimilate and make predictions about relevant elements and attributes of one's environment for the purpose of robust survival [4]. Within the domain of cyber security, effective situational awareness requires knowledge of historical and current cyber (i.e. network or host) activity in order to recognize and respond to threatening behaviors. As the variety of systems and networks increases and available bandwidth and data usage rates on enterprise networks continue to rise, security analysts and administrators striving to create and maintain cyber situational awareness face unique challenges of scope, scale and speed.[5], [10], [11] Traditional network situational awareness systems that rely upon standard analysis techniques have difficulty scaling to meet the increase.

To address this challenge, we have developed a real-time characterization and analysis system that uses stream processing algorithms to provide relevant situational awareness and anomaly detection. We have implemented our system using an existing stream processing platform that provides dynamic instantiation of lightweight processing elements on a set of distributed computation nodes. Our system has been deployed on an operational network, and stands as an end-to-end characterization environment, that passively monitors and characterizes high-speed / high-volume network traffic in order to provide indications of anomalous behavior for network and system analysts and administrators.

Our system consists of three main stages: a filtering stage, which finds and selects the Top-k producers of traffic on the network; a trending and anomaly detection stage, which tracks and monitors selected traffic for anomalous behavior; and a correlation stage, which correlates the individual anomalies from the second stage by time and source address in order to both reduce false alarms and to recognize truly anomalous behavior that is indicative of cyber attacks. Our system makes use of two specific data sources from network traffic: raw packet data and NetFlow connection summary records (described below). Our feature extraction and anomaly detection algorithms have been designed to be general-purpose and can be readily applied to other data sources.

The contributions of our paper are the following. First, we present an architecture for an on-line cyber situational awareness system that can handle high-speed / high-volume traffic environments. Second, we present enhanced versions of well-known streaming algorithms that have been improved based upon our operational experience. Finally, we demonstrate that our system performs well in operational environments and under heavy traffic load to provide situational awareness. Our work is unique in that it is aimed at applying streaming algorithms from academic research to real-world environments of an enterprise network.

Our paper presents the results of our research in the following way. Section II presents an overview of our situational awareness system with details about its architecture, data sources and various traffic processing pipelines. Section III presents the modified stream characterization algorithm that we use to filter and process the data in the operational environment. We present results from the application of our characterization and anomaly detection system using a labeled data set in Section V and we discuss the system's performance in a live operational environment. Section VI presents a summary of our cyber situational awareness research and discusses future directions for investigation.

[†]This work is sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government

II. System Description

We have implemented our system as an application within an existing stream processing platform that provides an interface for importing and processing high-speed data streams. The core building block within the platform is the ***processing element*** (PE), which serially reads from incoming streams of data tuples, performs some amount of processing, and generates outgoing tuple streams. The existing streams processing environment allows users to compose processing pipelines of arbitrary length and complexity by chaining individual PEs together.

Our general approach to developing analysis applications that provide cyber situational awareness for network traffic places PEs that perform lightweight tasks dedicated to summary and aggregation early in the chain where the highest rate of tuples are received. PEs that perform more CPU-intensive tasks such as statistical modeling and anomaly detection are placed later in the processing chain where the rate of incoming tuples is lower because of filtering.

A. High-Level Architecture

Figure 1 presents a block diagram for a typical chain of PEs in our system. As stated above, each data stream imported by the system is processed so that the high rate of incoming data is gradually filtered and aggregated into usable situational awareness data that is presented to the network operator. Each PE in the system can be classified into one of the following categories:

Filters, described in Section III, create meaningful and accurate characteristics of the data stream. ***Anomaly detectors***, described in Section IV, use statistical techniques to generate historical models of data streams, predict future trends, and identify anomalies in data streams. ***Anomaly correlators***, also described in Section IV, coalesce anomalies from various detectors, filter potential false positives, and present alert messages to an network operator. Each processing pipeline contains at least one of each of these types of PEs, and may contain more, depending on the nature of the processing and the needs of the network operator.

B. Data Sources

Although our PEs are configurable and designed to be deployed on various network data sources, we have implemented an operational prototype system using the following two data feeds.

a) NetFlow Data: Our system processes the NetFlow records of all Internet gateway traffic for a large enterprise network. It uses the standard Cisco NetFlow version 5 protocol, which defines a flow as a unidirectional network transaction identified by a five-tuple: source and destination host addresses, source and destination ports, and transport layer protocol. Each flow record also includes meta-data such as byte and packet counts, tcp session length, and tcp flags. Raw packet content is not included in the data feed.

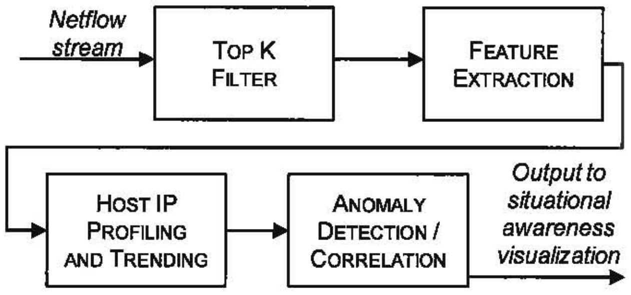
Netflow stream
Top K Filter
Feature Extraction
Host IP Profiling and Trending
Anomaly Detection / Correlation
Output to situational awareness visualization

Fig. 1. System block diagram.

b) Raw packet content: In addition to NetFlow data, our system also processes raw network packet data for situational awareness. This data is not sessionized or reassembled into a higher level protocol and depending on traffic rates and system load this traffic may be sampled.

C. Processing Pipeline

Figure 1 illustrates the processing pipeline. Once the NetFlow records are imported into the stream processing platform, they are passed to a filtering PE, which uses techniques described in section III to detect "heavy hitter" hosts based on various network features such as byte count, flow count and session length. This PE filters the incoming data stream to find the k heavy hitters for further processing; k is determined by available system resources within the stream processing environment.

The resulting stream is received by the feature extraction PE, which extracts the features enumerated in Table I and produces a stream of these features for each time interval monitored (e.g. 10 minutes). Some features, such as raw byte and packet counts are aggregated over the time window being monitored while others, such as unique host count, utilize a bloom-filter-based counter to record their value. In the latter case, a hash function is applied to each arriving key-value pair to determine the correct bit location to set in a storage hash to record unique occurrences. The feature extraction PE is extensible and allows for the addition of new feature extraction analytics as they are developed.

The host IP profiling and trending PE receives the stream of extracted features and constructs statistical models to characterize host behavior during the time interval monitored. These statistical models and the original feature stream are sent to the anomaly detection PE which generates internal anomaly messages when real-time values deviate from expected values. The anomaly detector is described in greater detail in section IV

Due to the noisy nature of network traffic, a number of uncorrelated (e.g. random) anomalies are expected to be seen during each time interval. These anomalies do not represent truly anomalous activity and should be removed from the processing stream. To remove this noise, individual feature anomalies are sent to the anomaly correlator PE which uses historical context to correlate internal anomaly messages over

Inbound/outbound	
Byte count	Unique destination country count
Packet count	Avg. bytes/packet
Flow count	Avg. packets/flow
Unique host count	Avg. bytes/flow
Unique source port count	Avg. TCP session length
Unique destination port count	Bytes in/out ratio
Unique destination AS count	Flows in/out ratio
Packets in/out ratio	

TABLE I
FEATURES MONITORED BY HOST PROFILER

the observation time windows. When correlated anomaly activity exceeds a predetermined threshold, an alert containing the address of the anomalous host and associated historical data is sent to an external visualization tool for review by an operator. The anomaly correlator PE is described in greater detail in section IV.

III. STREAM CHARACTERIZATION ALGORITHMS

In this section, we describe the customization necessary for stream algorithms to perform data characterization. While the PE is derived from algorithms found in literature, our system constraints do not match those considered important in the literature. We have therefore designed our PE's algorithm to take advantage of available resources to better address the needs of the operational environment. We have developed the Top-k algorithm, instead of the frequent-item algorithm predominant in the literature, because it requires no prior knowledge of the network deployment environment in order to specify operational parameters. Below we describe our customized Top-k algorithm, which exemplifies our data characterization algorithms.

A. Top-k Operator

The Top-k algorithm identifies from a stream of $(key, value)$ pairs the k pairs with the highest value over an interval of time. The keys of these pairs are often used to filter traffic downstream. During a time interval, when a series of pairs with the same key, k, $(k, v_1), \cdots, (k, v_n)$ occurs, the set of values $V = \{v_1, \cdots, v_n\}$ are aggregated according to an aggregation function, ag. In our system, we use two common aggregation functions: summation, $ag(V) = \sum_{v \in V} v$, and average, $ag(V) = \frac{\sum_{v \in V} v}{|V|}$.

1) Solutions within the Literature: The Top-k algorithm solves a variant of the frequent item identification problem[8]. This problem is often stated as one of the two following problems: the ***frequent item detection problem***, or the ***k most frequent item detection problem***.

Definition 1 (Frequent item detection): Given a set X of aggregated $(key, value)$ pairs and a frequency threshold $\phi \in [0, 1]$, find all pairs such that $value > \phi A$ where A is the sum of all the values.

Definition 2 (k most frequent item detection): Given a set X of aggregated $(key, value)$ pairs and $k \in \mathbb{Z}^+$, find the k pairs with the largest values.

Solutions that solve either of these problems also deal with the problem of creating the aggregated set X. The key characteristic of each problem is the type of data found; as such, solutions can be classified by the type of data they can provide from a stream (e.g., Top-k solves the second problem).

Originating from a data mining and database background, prior solutions have mainly focused on the frequent item detection problem. The large number of possible keys and the large number of total pairs within streams from this domain have lead to the development of algorithms that have tight bounds in terms of writable memory and the number of passes through the data stream. These bounds have lead to the development of approximation algorithms for both problems. When available memory is greater than the number of unique keys both problems are considered to be trivially solved by sorting the aggregate key-value pairs.

In our environment, the number of unique keys is large but within the bounds of available memory. For example, when finding the top twenty IP addresses by the number of bytes received, the number of keys may be large; yet, it is feasible to have enough memory to store a counter for each observed IP address. Thus, when the necessary amount memory is available, the trivial solution is unsatisfactory within our environment.

To illustrate this point, we examine the behavior of the space saving algorithm [9], which acts as an efficient implementation of the trivial solution when given a sufficiently large memory supply, and is the base algorithm for our customization.

B. The Space Saving Algorithm

The space saving algorithm is a deterministic algorithm that may be use to find approximate solutions for both the frequent item detection problem and the k most frequent item detection problem.

```
let $m$ be a map of $Key \to Value$ ;
for each pair $(k, v)$ in the stream do
    if $v'$ is $m[k]$ then
        let $m = (m - (k, v)) + (k, aggregate(v', v))$
    else
        if $|m| < memorybound$ then
            let m = m + (k,v)
        else
            let $k'$ be the key associate with the minimum value within $m$ ;
            let $m = (m - (k, v)) + (k', aggregate(v', v))$
end
let $r$ be the list of $m$ pairs sorted in descending order of value ;
return the first $k$ elements of $r$ ;
```
Algorithm 1: The space saving algorithm

Algorithm 1 shows an abstracted algorithm for space saving that solves the k most frequent item detection problem. Key-value pairs are adding to a map with aggregation until the map reaches a pre-specified size. At which point, the smallest valued key is replaced with the new value, and the smallest value is aggregated with the new key's value. While the key replacement policy of space saving is important for approximating an answer, with sufficient memory it acts as the trivial solution. Therefore, a Top-k PE that is implemented as space saving would have the same performance characteristics as the trivial solution until it reaches its memory bound.

In our environment, the PE receives pairs continuously, and generates results after each time interval elapses. With the trivial implementation, storing each pair received has an efficient implementation, the maps can be implemented as either a hash table or a tree data structure, and the minimum valued key value can be maintained with a heap. However, once a time interval elapses, sorting the key-value pairs and clearing the maps are time expensive operations, and have experimentally resulted in the overflow of PE buffers when the number of unique keys is large and the rate of traffic is high.

```
let $m$ be a map of $Key \rightarrow Value$ ;
let $topk$ be a min-max heap ;
for each pair $(k, v)$ in the stream do
    if $v'$ is $m[k]$ then
        if $v'$ is old then
            let $m = (m - (k, v)) + (k, v')$
        else
            let $m = (m - (k, v)) + (k, aggregate(v', v))$
        end
    else
        if $|m| < memorybound$ then
            let m = m + (k,v)
        else
            let $k'$ be the key associate with the minimum
            value within $m$ ;
            let $m = (m - (k, v)) + (k', aggregate(v', v))$
end
let $(k, v'') = m[k]$;
if $(k, v'')$ is in $topk$ then
    update $(k, v'')$'s position in $topk$
else if $|topk| < k$ then
    add $(k, v'')$ to $topk$
else if $v'' > \min topk$ then
    replace $topk$'s root with $(k, v'')$ and adjust the root's
    position

return the pairs in $topk$ ;
```

Algorithm 2: The stream oriented space saving algorithm

C. Stream Oriented Space Saving Algorithm

In Algorithm 2, we customize the space saving algorithm with two modifications: we add a time stamp to map entries, and we add a min–max heap to keep track of the current highest valued k pairs. These modifications enable the quick production of the Top-k pairs on each time interval since we only need to output the pairs within the min–max heap and then clear the min-max heap. The timestamp eliminates the need to reinitialize the map because the timestamp allows stale map entries to be detected and replaced.

1) Implementation specifics: Adding a timestamp is a straightforward modification to the map. Since our operator must keep track of time, we add the timestamp of the last update to each map element. The algorithm can detect a stale map element by checking if its timestamp occurred during a time that is not within the current time interval. By using this technique, we can eliminate the need to clear the map when each time interval elapses.

Maintaining efficient heap operations can be achieved by using a binary heap. Each heap stores a pointer to the map element, and each map element stores a pointer back to it's respective heap element. This optimization enables the value for each key to be updated at a central location and minimizes the amount of copying required for each heap adjustment operation.

2) Performance characteristics: With regards to performance, it would appear that performing more work on each pair update would decrease the total throughput of the system. However, in practice the heap operations are very efficient because heap items rarely percolate more than one position in the heap per update. For example, when using the summation aggregation function and the stream values are fixed to the value of one (*i.e.*, we are counting stream frequency), each heap operation cannot percolate more than one position on each update.

IV. Anomaly Detection and Correlation

A. Anomaly Detection

The overall goal of our situational awareness system is the characterization of network traffic such that it supports the recognition and identification of anomalous behavior, which is often indicative of malicious activity. The ability to detect anomalous or abnormal behavior therefore begins with the ability to model normal network behavior. Although aggregate network traffic on the Internet is often modeled with long-tailed distributions, such as the Pareto [2], [3], it has also been found that when traffic is normalized for variations due to the time of day and day of week, a Gaussian probability distribution is an appropriate model for the traffic[1]. With this in mind, our anomaly detection mechanism calculates summary statistics for the normal probability distribution (Equation 1) for relevant network features while accounting for time of day and day of week. To account for time of day variations, network features are sampled at small increments of time (e.g. 10 minutes) throughout the day, as shown in Figure 2 a depiction of the windowed sampling scheme used by our ***time baseline*** anomaly detector. Models of network behavior are further separated into those corresponding to workdays (Monday-Friday) and weekend days (Saturday and Sunday).

The time baseline anomaly detector records historical data for various features described in Section II and uses this data to create a statistical profile for each host being monitored. This profile is then used to recognize deviations from expected behavior. Thus, our time baseline anomaly detection mechanism consists of a two-phase process. In the first phase, models are trained in an on-line fashion; descriptive statistics for an assumed Gaussian distribution are calculated from observed network variables.

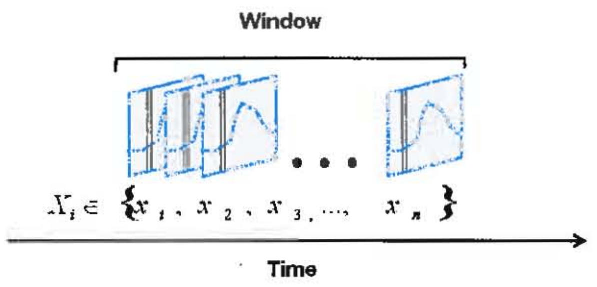

Fig. 2. Illustration of the windowing scheme used by the time baseline anomaly detector used to monitor different intervals of network variables

$$x_i \rightarrow N(\mu, \sigma) \tag{1}$$

During the second phase, variables observed during operation are compared to stored model parameters to determine the likelihood of occurrence. If the likelihood is below the experimentally determined threshold τ an indication of anomaly is issued within the system.

$$f(x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{2}$$

B. Anomaly Correlation

Due to the bursty nature of network traffic, we expect the behavior of each host to generate some number of false positive across many time intervals. However, we have observed in our experimental results (described in detail in section V and illustrated in Figure 3), that a true positive anomaly is more likely to be detected when either there are multiple independent anomalies across various features or there is sustained anomalous activite within a single feature.

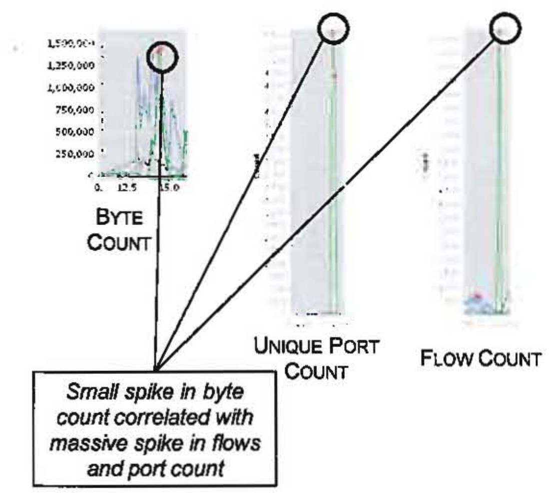

Fig. 3. Example of anomaly correlation.

In the anomaly correlation phase individual anomalies are correlated across source IP addresses and time intervals. The correlator receives anomaly messages with associated confidence intervals from anomaly detectors for each feature. The correlator does not have prior knowledge of the relationship between anomaly detectors, and therefore, each anomaly is considered to be an independent event. Thus, the overall likelihood of the occurrence of multiple individual anomalies is determined by calculating the product of the individual anomaly confidence scores.

$$L(\Theta/X_i) = 1 - \Pi(1 - f(x_i)) \tag{3}$$

$$\forall x_i \in X_i : f(x_i) > \tau \tag{4}$$

In equation 3 $L(\Theta/X_i)$ represents the likelihood of the model Θ, represented by the various Gaussian distributions occurring given the data samples X_i . This likelihood is calculated as 1 minus the product of the individual anomaly scores for each x_i above threshold τ.

This likelihood value is used as the *anomaly score*, and is compared against a tunable threshold to determine if an alert message should be issued. This score is aged over time, to allow for detection of both a spike in high-confidence anomalies, and a low intensity, sustained anomaly over time. Figure 4 illustrates the anomaly score of a sample "victim" host during the attack from the DARPA dataset illustrated in Figure 3. Note that the anomaly score is determined by unique anomaly detector alerts received by the anomaly correlator, and is compared against a tunable parameter in order to decide when to generate an alert message.

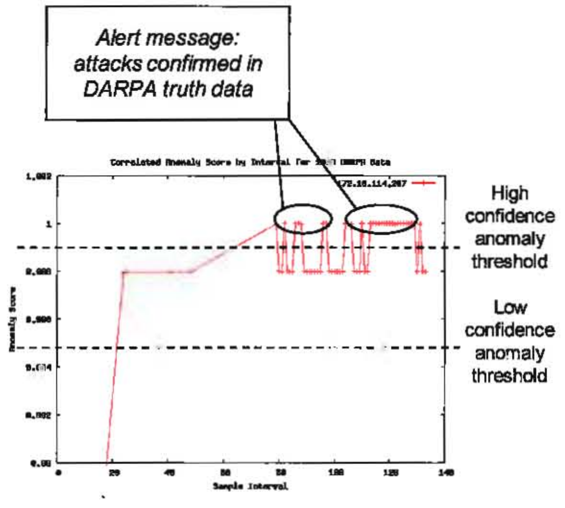

Fig. 4. Example of anomaly score over time for attack in DARPA dataset.

V. Experimental Results

A. Accuracy

We evaluated the detection accuracy of our system using the MIT Lincoln Laboratory 1999 DARPA Intrusion Detection Dataset.[7], [6] The DARPA dataset was designed to support the evaluation of various intrusion detection systems. It consists of a number of different known cyber attacks, on top of continual background traffic. To begin our evaluation we trained our system on one week's (week 1) of clean network traffic. This allowed the system to create internal baseline models of network behavior. Next, we presented the system with a different week (week 2) of network traffic known to contain attacks. During this second phase of evaluation we monitored the systems output to determine true and false positives.

We structured our evaluation by first classifying each attack as either "flow detectable" or "not flow detectable" and only concern ourselves with flow detectable attacks since our host profiler pipeline receives only NetFlow records, and not the packet capture data. Therefore, attacks that require the detection of a signature within packet data, such as the byte code of a known piece of malware, are out of scope for our detection system.

To determine the rate of true and false positives, we began by dividing each day of the attack set into 10 minute intervals to match the sampling length of our host profiler. The total number of potential alerts that the system can generate is calculated as the total number of intervals in a week times the total number of hosts being monitored or 2880. The pathological case occurs if every host generates an alert for every interval. We define a true positive to be an alert generated for a host in an interval in which there is a labeled attack, and a false positive to be an alert generated for a host in which there is no labeled attack. Similarly, we define a true negative as each host–interval in which there is neither an alert or an attack, and a false negative as each host-interval for which there is a labeled attack but no alert.

We replayed the week of labeled attack data through our trained system multiple times while varying the anomaly detector's threshold, σ, and recorded the detection rate of true and false positives. We use this data to produce the ROC curve shown in figure 5. Our system detected 8 of the 9 relevant attacks from the dataset for a probability of detection (PD) of 88% with a probability of false alarms (PFA) of 0.01%. Our system demonstrates comparable performance to other intrusion detection systems evaluated over the 1999 DARPA dataset for the detection of flow detectable attacks.[6]

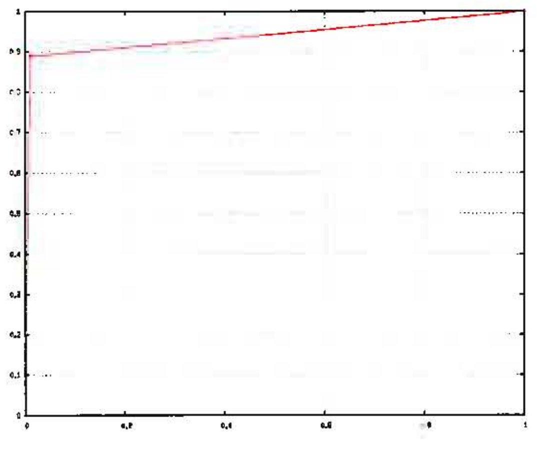

Fig. 5. A receiver operating characteristic curve showing detection performance on the DARPA 1999 data set.

B. Throughput

In addition to evaluating our system for accuracy using truth data from the DARPA dataset, we evaluated the throughput of our system by running it operationally on a very large enterprise network. Our system was able to handle data rates of up to 500,000 raw packets and 2.5 million flow records per second. Although we did not have truth data on this network to calculate accuracy values, we were able to demonstrate that our algorithms perform at very high data rates.

VI. Conclusion

We have presented a stream processing based system which performs characterization and analysis of high-speed / high-volume data for the purpose of cyber situational awareness. Our system makes use of an existing streams processing environment that enables the creation of data analysis pipelines which are composed of individual processing elements. We have employed analysis algorithms for stream characterization and shown how their deployment in an operational environments has led to several improvements in performance. Our system performs well on a labeled data set and in an operational environment where high speed network data is encountered. Future research directions include the exploration of additional data sources and features that can be used for cyber situational awareness, such as server and proxy logs, and the investigation of other mechanisms for the detection of anomalies and anomaly correlation.

References

[1] K. M. Carter, S. W. Boyer, R. P. Lippmann, and R. K. Cunningham, "Baselineing the nipmet through flow analysis," Centaur/Trickler Technical Exchange Meeting, Tech. Rep., November 2009.

[2] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup, "Algorithms and estimators for accurate summarization of internet traffic," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 265–278. [Online]. Available: http://doi.acm.org/10.1145/1298306.1298344

[3] A. B. Downey, "Evidence for long-tailed distributions in the internet," in *In Proceedings of ACM SIGCOMM Internet Measurment Workshop*. ACM Press, 2001, pp. 229–241.

[4] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 37, pp. 32–64(33), March 1995.

[5] C. Labovitz, D. McPherson, S. Iekel Johnson, and M. Hollyman. (2008, June) Internet traffic trends — a view from 67 ISPs. [Online]. Available: www.nanog.org/meetings/nanog43/abstracts.php?pt=NjgmbmFub2c0Mw==&nm=nanog43

[6] R. Lippman, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "Analysis and results of the 1999 DARPA off-line intrusion detection evaluation," in *RAID*, 2000, pp. 162–182.

[7] R. P. Lippmann, R. K. Cunningham, D. J. Fried, I. Graf, K. R., Kendall, S. W. Webster, and M. Zissman, "Results of the 1999 darpa off-line intrusion detection evaluation," in *Second International Workshop on Recent Advances in Intrusion Detection*, West Lafayette, Indiana, 1999.

[8] N. Manerikar and T. Palpanas, "Frequent items in streaming data: An experimental evaluation of the state-of-the-art," *Data & Knowledge Engineering*, vol. 68, no. 4, pp. 415 – 430, 2009.

[9] A. Metwally, D. Agrawal, and A. E. Abbadi, "Efficient computation of frequent and top-k elements in data streams," in *In International Conference on Database Theory*, 2005, pp. 398–412.

[10] K. Mittal, "Internet traffic growth analysis of trends and predictions." [Online]. Available: www.kunalmittal.com/includes/Papers/PredictingInternetTrafficGrowth.pdf

[11] L. G. Roberts. (1999, December) Internet growth trends. [Online]. Available: www.nanog.org/meetings/nanog43/abstracts.php?pt=NjgmbmFub2c0Mw==&nm=nanog43